10Gb/s packet Processing on Hybrid SoC/FPGA Platform

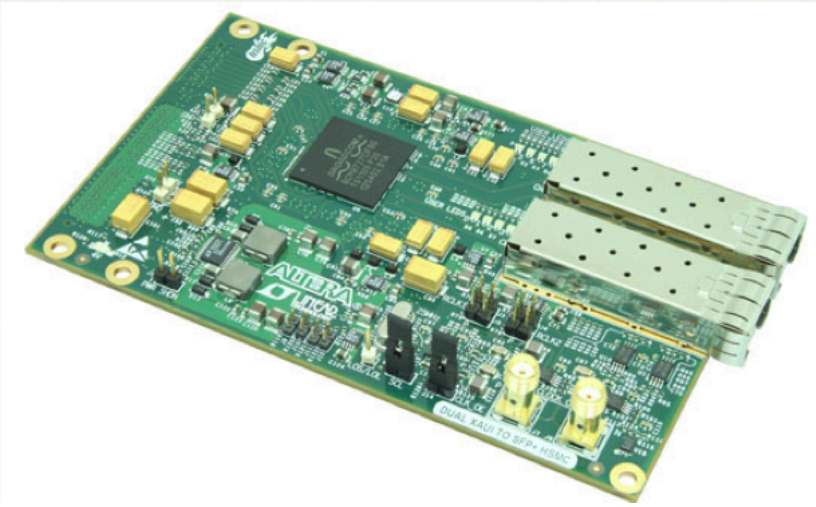
[CSEE 4840] Embedded System Design Final Report

John Chaiyasarikul (jc4104)
Yumeng Xu (yx2239)
Shuguan Yang (sy2518)
Jian Zhong (jz2523)

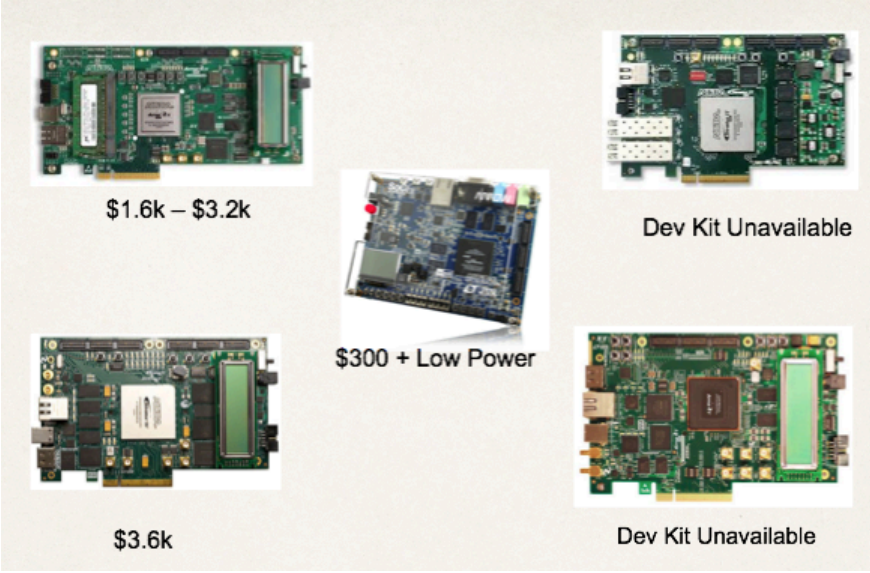
Contents

1. Introduction

Ethernet is a set computer networking technologies that were standardized in 1983 by IEEE for local area networks (LAN). Ethernet is one of the most common and important interfaces found in current technology dealing with wired networking. Commercially, interfaces that support up to 1Gb/s has been the most adopted and is seen in most Ethernet cards in PCs used to access the internet. In some cases, such as in financial applications, a faster interface is important and sometimes critical.

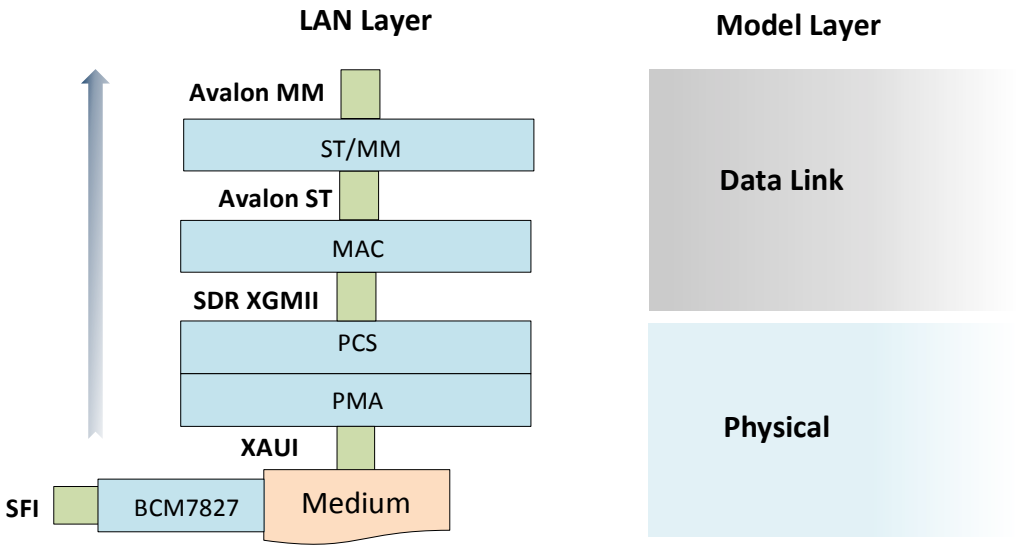


In FPGA applications, interfaces that support up to 10Gb/s or 10GbE interfaces usually require high-end or state-of-the-art boards. These boards are extremely expensive and cost many thousands of dollars. However, with the new Cyclone V FPGA, there is a low-power and low-cost FPGA that has the capability of supporting 10GbE. Our project is to interface the Dual XAUI to SFP+ HSMC board with a Cyclone V FPGA creating a 10GbE platform that may cost only a fraction of traditional platforms that use high-power boards.



2.Interfaces

Figure 1-1: XAUI PHY Link Layer Model Diagram



I. SFI

To match the implementation of XAUI PHY later on, we use 15 meter long copper wire to connect SFP+ direct attachment. This is strictly refer to IEEE 802.3 standard. The advantages of SFP+ are low cost, low latency and low power. In table x-x, specifications are list and compared between SFP+ Direct Attach and some other interconnections.

Table 1-1: Comparison between different 10GbE Interconnects

Interconnect	Year	Connector	Medium	Media Type	Max range	Notes
10GBASE-SR	2002	XENPAK,X2 XFP,SFP+	fiber	Serial multi-mode	400m	-
10GBASE-LX4	2002	XENPAK,X2 SFP+	fiber	WDM multi-mode or single-mode	300m (Multi), 10 km (single)	Costly and complex, Replaced by 10GBASE-LR
10GBASE-CX4	2004	XENPAK,X2	copper	InfiniBand 4X twinaxial 8-pair	15m	4 lanes, each at 3.125 Gbit/s. Larger, bulkier
SFP+ Direct Attach	2006	SFP+	copper	Twinxial 2-pair	15m	Cheaper, low latency, low power

Each of the two identical SFP+ ports are combined with a SFP+ optical module (which in this project we use copper wires) from the 10GbE optical interface. From the block diagram we can see that SFP+ modules communicate with the BCM7827 via the serial SFI protocol. The SFP+ optical modules will contain status and configuration registers accessible through and I2C port. Power for the SFP+ modules is provided from 12V and 3.3V available on the HSMC connecters.

II. XAUI

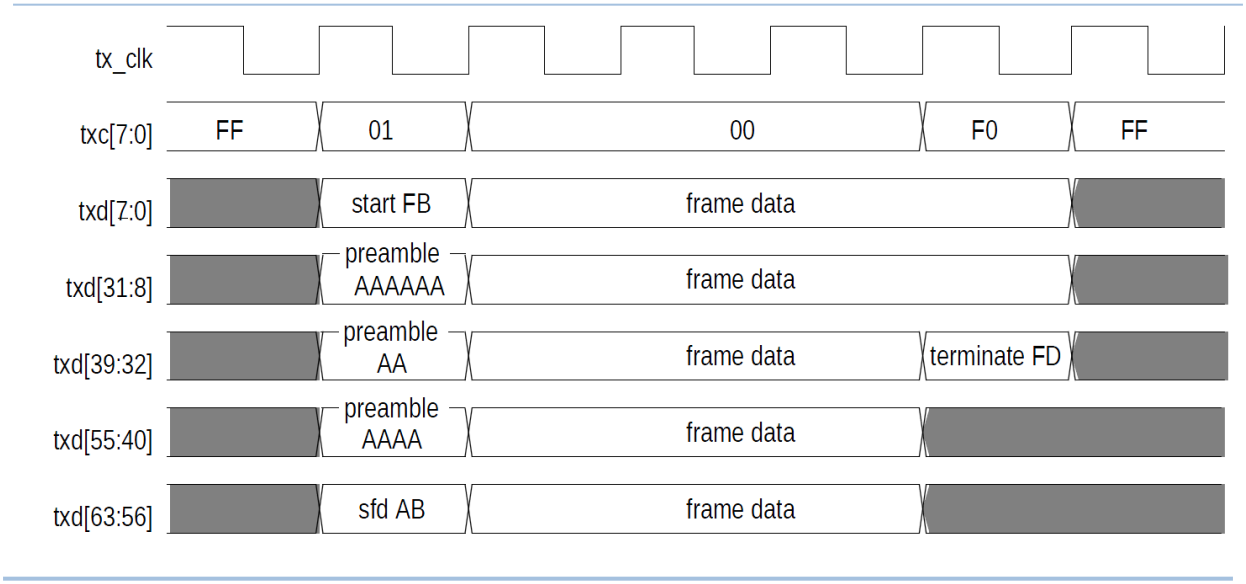
XAUI is abbreviation for 10 Gigabit Attachment Unit Interface, a standard for extending the XGMII. XAUI has four lanes of independent receive and transmit data paths. It can be considered as a simple and straight forward mapping to the XGMII. Four lanes carries the XGMII 32-bit data and control signals. One of the significant merit implementing XAUI is that it can share both technology and functionality with other 10 Gbit/s interfaces or Ethernet blocks.

In this project, data packet is separated into 4 serial data packets. Each of four receive and transmit lanes operates at a rate of 3.125 Gbit/s. The Dual XAUI to SFP+ HSMC board contains a HSMC connecter with three banks of connectors. In this project, not all of the signals are used via HSMC connecter. More details will be explained in later chapters along with hardware configuration set up.

III. SDR XGMII

SDR stands for Single Data Rate. XGMII is the abbreviation for 10 Gigabit Media Independent Interface, a standard defined in IEEE 802.3 for connecting full duplex 10 Gigabit Ethernet (10GbE) ports to each other and to other electronic devices on a printed circuit board. It is composed from 64-bits data paths along with 8-bit control flows, operating at 156.25MHz.

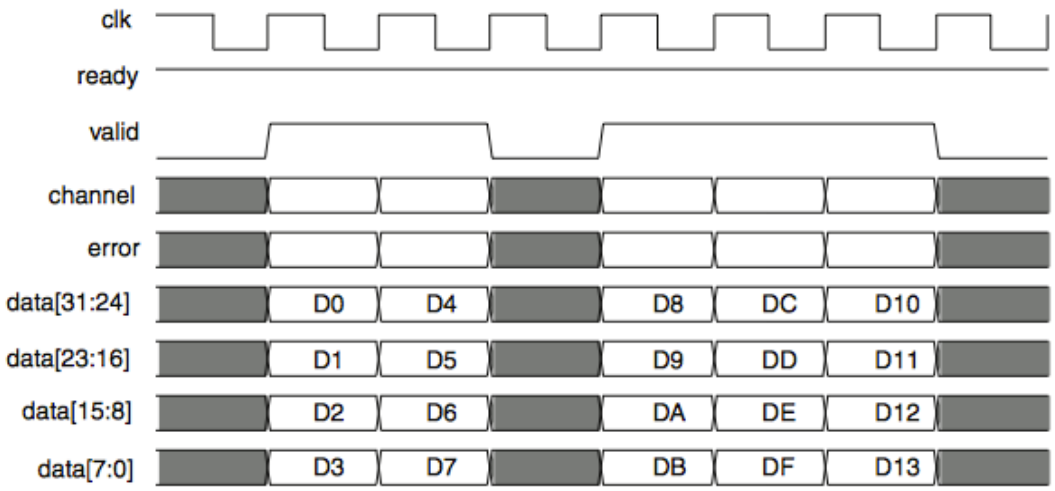
Figure 1-2: SDR XGMII DataTiming Diagram



The network side interface of the 10GbE MAC implements the SDR version of the XGMII protocol. Data bus carries the MAC frame and the most significant byte occupies the least significant lane.

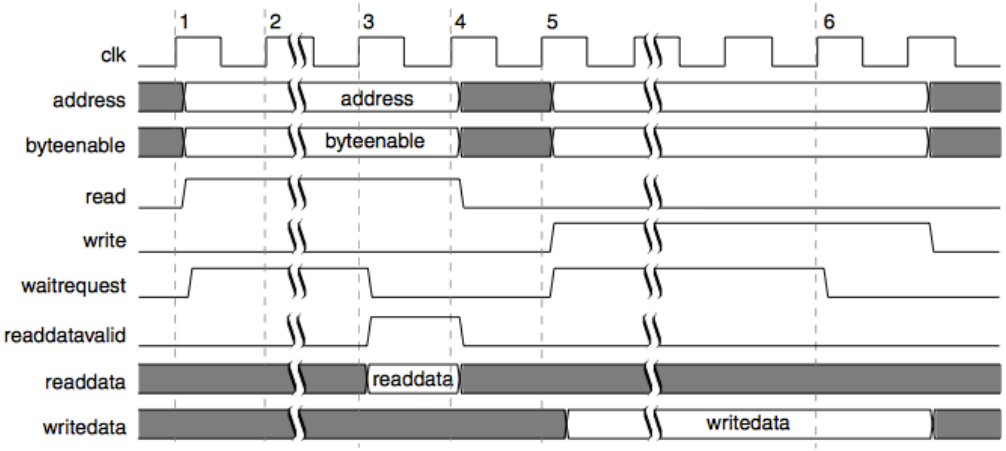
IV. Avalon ST

Avalon Streaming Interface supports the unidirectional flow of data, including multiplexed streams, packets, and DSP data. A 64-bit Avalon-ST interface is used to connect the 10GbE MAC IP core to the traffic controller through FIFO buffers. This client interface runs at 156.25MHz.



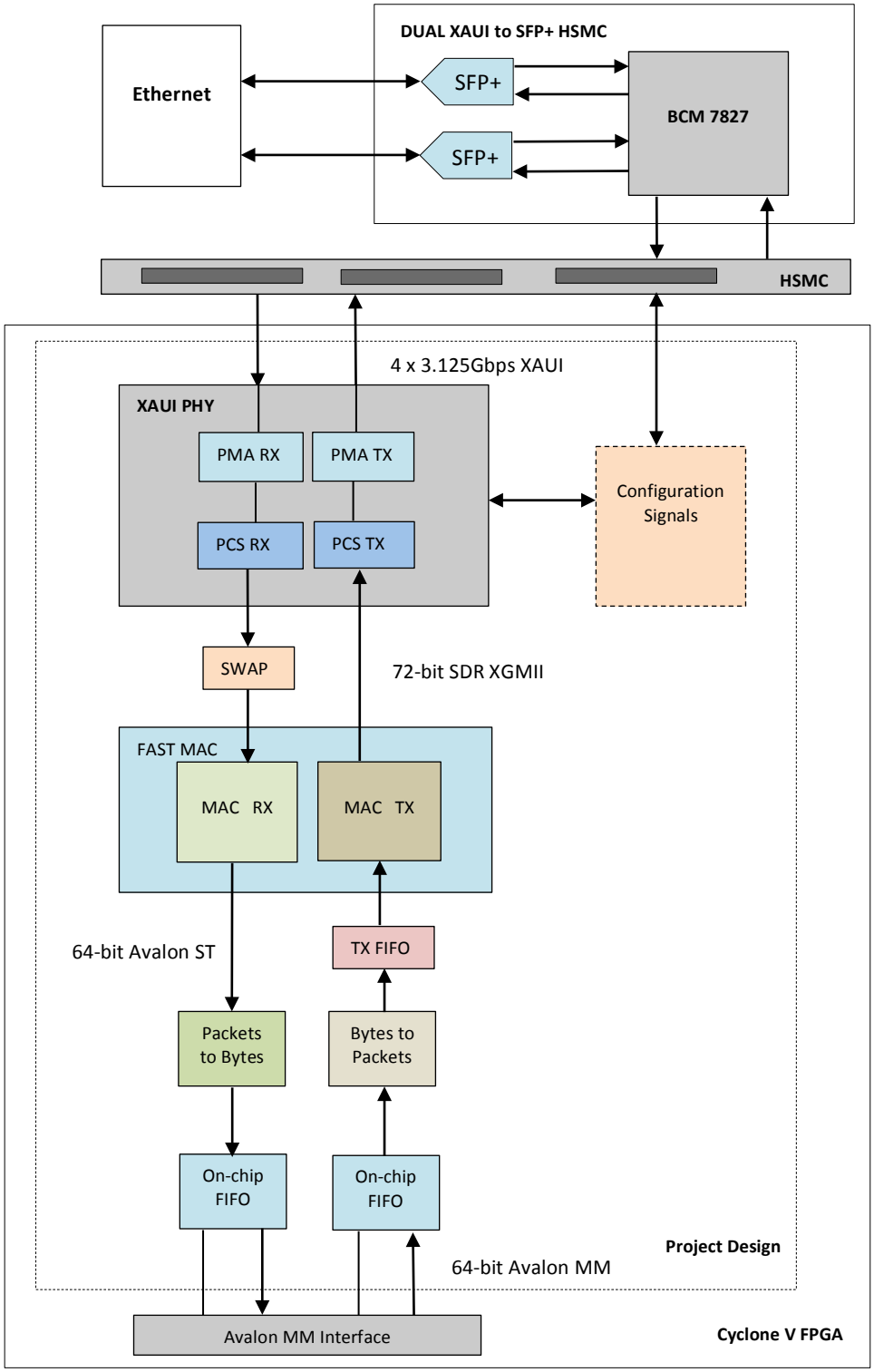
V. Avalon MM

A standard address-based read/write interface typical of master-slave connections. A 32-bit Avalon-MM interface is used to form a bus that connects to all the IP cores in the design and to JTAG where the system console and client application can access the data for control or debugging purposes.



3. Design

10GbE Hybrid (Dual XAUI + SoCKit) Project Block Diagram



4. Modules

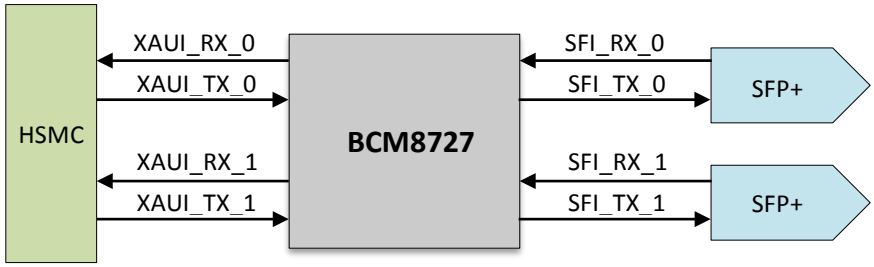
I. DUAL XAUI to SFP+ HSMC

Broadcom BCM7827

The BCM 8727 is a dual-channel 10GbE SFI-to-XAUI transceiver that incorporates and Electronic Dispersion Compensation (EDC) equalizer supporting SFP+ line-card applications. The BCM 8727 is fully compliant to the 10GbE IEEE 802.3ap standard. This chip is developed using an all-DSP high-speed front-end providing the highest performance and most flexibility for line-card designers.

On our daughter card, as shown in Figure 2-1, XAUI interfaces are attached to the HSMC side of the card and the SFI side of the interface will be attached to the SFP+ optical modules on the opposite side of the board.

Figure 2-1: SFP to XAUI signal block diagram with BCM8727 on DUAL XAUI daughter card

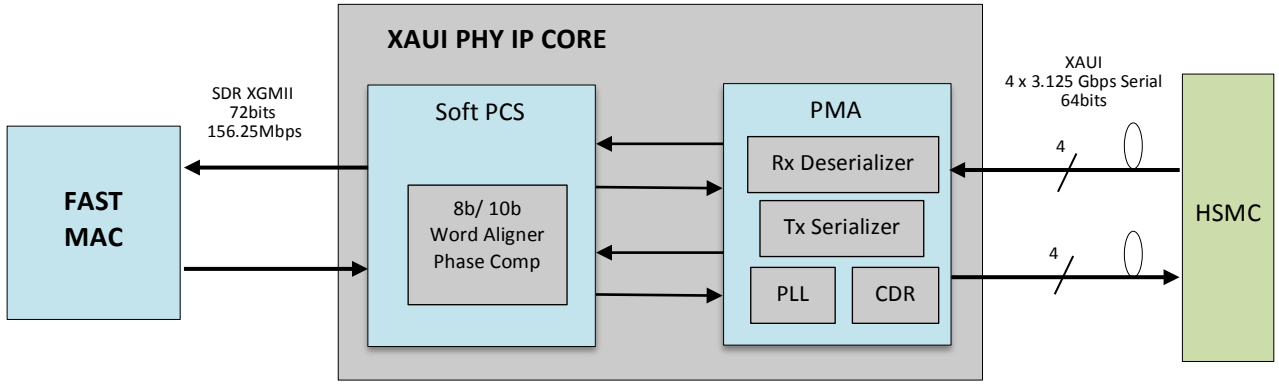


II. SoCKit/ Cyclone V FPGA

A. XAUI PHY

In this project, we use Altera XAUI PHY IP Core to receive the packet data from our daughter and transform from XAUI (10G Attached Unit I/F) to XGMII (10G Media Independent I/F). The Altera XAUI OHY IP Core implements the IEEE 802.3ap Clause 48 specification to extend the operational distance of the XGMII interface and reduce the number of interface signals.

Figure 2-2: XAUI to XGMII signal block diagram with Soft XAUI PHY IP Core



As shown in Figure 2-2, four lanes of serial packet data comes in at the receiving stage between XUAI PHY IP Core and HSMC connector. Each of the four lanes carries 64 bits data at 3.125 Gbps. Once the 64 bits packet enter XAUI PHY IP Core, we should make sure these packet data along with configuration signals can be detected by XAUI PHY and the data is ready to be transformed into XGMII. After XAUI PHY, each byte from the original packets add one bit control signal which makes the data comes out of XAUI PHY becomes 72bits at the recovery clock of 156.25Mbps.

Inside this XAUI PHY IP Core, there two main component: PMA (Physical Medium Attachment) and PCS (Physical Coding Sub-layer). As shown in Figure 4, Physical layer include PCA and PMA as well as data comes from HSMC connector.

1. PMA (Physical Medium Attachment)

The PMA includes the transmitter and receiver data paths, clock multiplier unit (CMU) PLL and the clock divider. Each transmitter channel has a clock divider. Details about the clock divider and PLL can be found at altera.com

In general, PMA performs the following functions:

- Bit level multiplexing with $x^7 + x^6 + 1$ scrambling/ descrambling
- Clock recovery, clock generation and data drivers
- oopback and test pattern generation and detection

Receiver PMA Data path

There are three blocks in the receiver PMA data path – the receiver buffer, channel PLL configured for clock data recovery (CDR) Operation, and deserializer.

Receiver Buffer

Receiver buffer is in charge of receiving the serial data stream outside the XAUI PHY and feeds the stream to the channel PLL. By implementing the buffer, it boosts the high-frequency components of the received signal, which may be attenuated when propagating through the transmission medium. The physical transmission medium, which is everything before HSMC including daughter board in this project, can be represented as a low-pass filter in the frequency domain. Variation of this kind in the signal frequency response may cause incorrect sampling on the input data at the receiver. Therefore implementation of receiver buffer guaranteed the improvement of signal integrity so that the high-frequency boost required at the receiver is able to overcome signal attenuation under different circumstances losing characteristics of the physical medium. Other two features of receiver buffer are saving board space and reduction of power consumption.

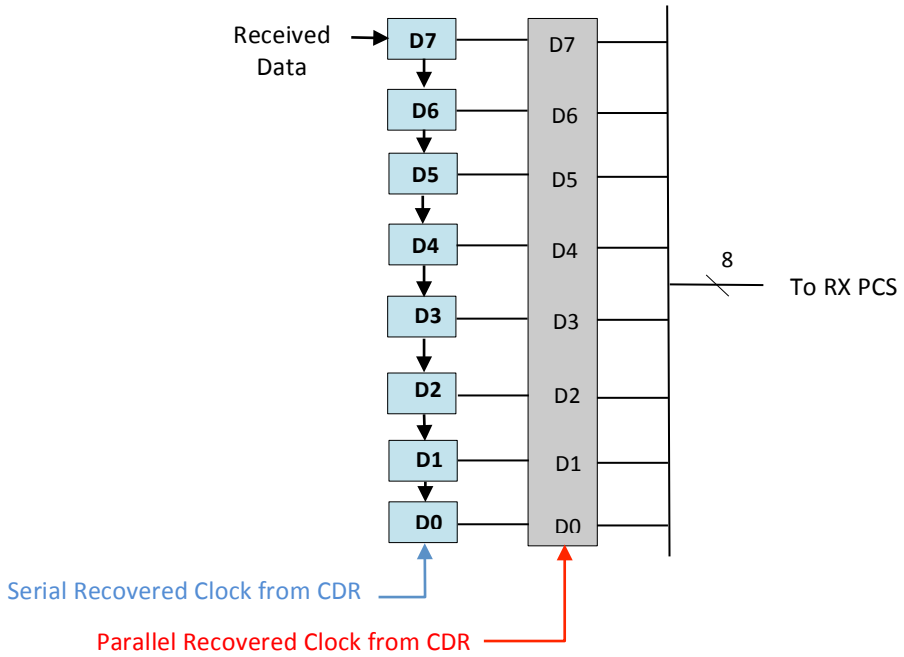
Channel PLL

One of the main function for PLL is to recover the clock and serial data stream. To activate this function, channel PLL should be configured as CDR. Offset cancellation is also required to correct the analog offset voltages

Deserializer

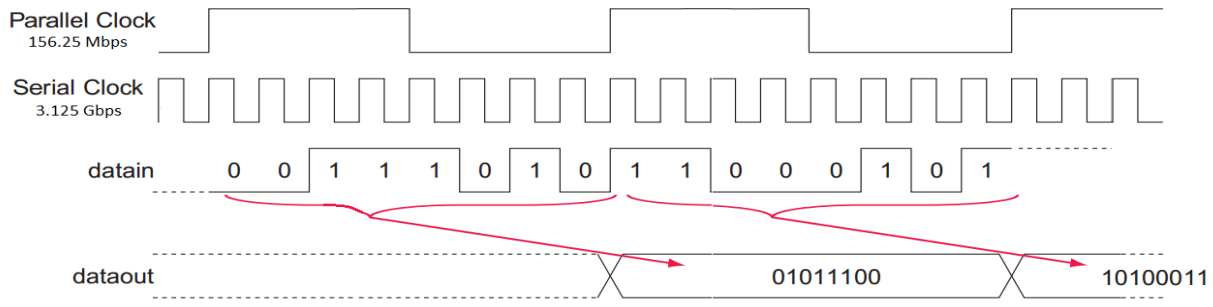
The deserializer block clocks in serial input data from the receiver buffer using the high-speed serial recovered clock and deserializes the data using the low-speed recovered clock. The deserializer forwards the deserialized data to the receiver PCS shown in Figure 2-4.

Figure 2-4: Deserializer Operation with an 8-bit Deserialization Factor



In figure 2-5, we can see in the previous clock cycle serial packet are received and detained under serial clock. In the next clock circle, 8 bits of packet accumulated previously is given to the parallel data sink. In this example, the serial steam (01011100) is deserialized to an 8'h5C value. Serial data is received lowest significant bit to highest. By far, four lanes of 3.125Gbps XAUI serial packet data transferred to a parallel XGMII data stream under 156.25Mbps. In order to simplify the demonstration, 8-bit is implied in this example shown in Figure x-x. In fact the project has 4X20 serial clock cycles in one parallel clock cycle.

Figure 2-5: Deserializer Bit order with 8-bit Deserialization Factor



Transmitter PMA Datapath

Serializer

The serializer converts the incoming low-speed parallel data from the transceiver PCS to High-speed serial data and sends the data to the transmitter buffer. Like the deserializer bit order introduced previously, bit order in serializer is from MSB to LSB when bit reversal option is enabled.

Transmitter Buffer

To improve signal integrity, transmitter buffers support the programmable analog setting which are: programmable differential output voltage, programmable pre-emphasis and programmable slew rate. In addition, like receiver buffers, transmitter buffers also save board space and cost as well as reducing power consumption.

One significant difference here from the receiver buffer is that a protocol-specific function is added in transmitter buffer. First, the transmitter output tri-state enables the transmitter differential pair voltages to be hold constant at the same value determined by the common mode level in the high impedance state. Second, Receiver detection provides link partner detection capability at the transmitter end using an analog mechanism for the receiver detection sequence.

2. PCS (Physical Coding Sublayer)

Receiver PCS Data path

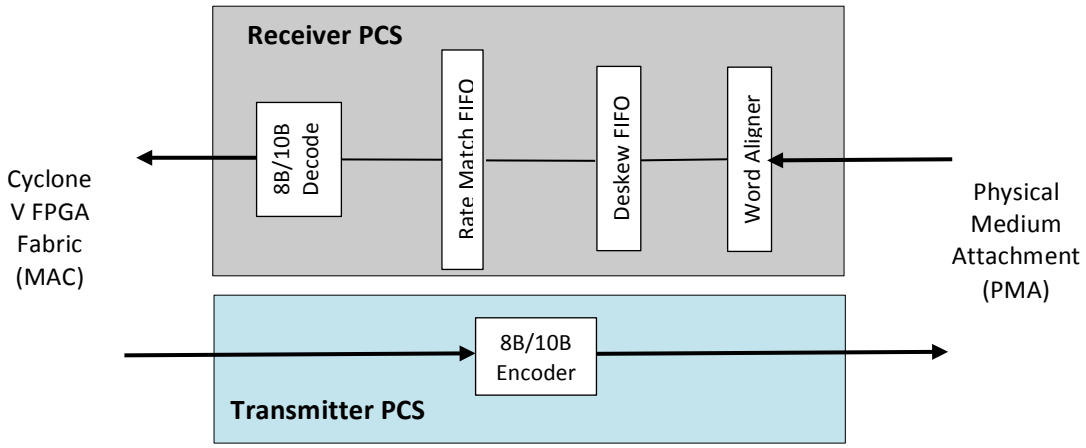
Word Aligner

The word aligner searches for a pre-defined alignment pattern in the deserialized data to identify the correct boundary and restores the word boundary during link synchronization.

In addition to restoring the word boundary, the word aligner implements the following features:

- Synchronization state machine
- Programmable run length violation detection
- Receiver polarity inversion
- Receiver bit/byte reversal
- The word aligner operates in one of the following three modes:
- Manual alignment
- Automatic synchronization state machine
- Bit-slip
- Deterministic latency state machine

Table 2-6: PCS Block Diagram of a Transceiver Channel in XAUI PHY IP Core



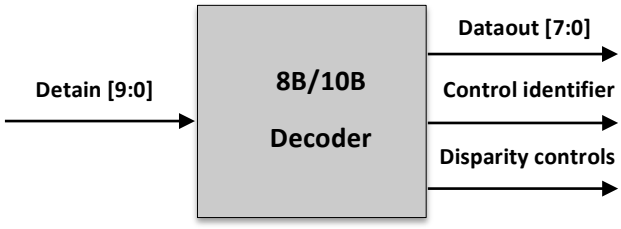
Rate Match FIFO

The rate match FIFO is 20 words deep, which compensates for the small clock frequency differences of up to ±300 ppm (600 ppm total) by performing symbol insertion or deletion, depending on the ppm difference on the clocks. It compensates for the small clock frequency differences between the upstream transmitter and the local receiver clocks.

8B/10B Decoder

The receiver channel PCS data path implements the 8B/10B decoder after rate match FIFO. The 8B/10B Decoder supports both single- and double-width Mode. As shown in Figure x-x, the decoder decodes the received 10-bit code groups into an 8-bit data with a 1-bit control identifier, based on Clause 36 in the IEEE 802.3 specification. The control identifier indicate whether the decoded 8-bit code is a valid data or special control code. The decoded data is fed to the byte deserializer or the receiver phase compensation FIFO (if the byte deserializer is disabled).

Figure2-7: 8B/10B Decoder



Byte Deserializer

Byte deserializer operates in single-width and double-width Mode. Either it is the 8-bit data comes from 8B/10B decoder or 10-bit wide data from the word aligner, the basic idea for the Byte Deserializer is to break these data into wider data at half the speed. Data path width for both single width and double width are listed in Table 2-1.

Table 2-1: Comparison between different 10GbE Interconnects

Mode	Byte DeserializerInput Datapath Width	Receiver Output Datapath Width
Single Width	8	16
	10	20
Double Width	16	32
	20	40

Byte Ordering

Byte ordering operates by inserting a predefined pad pattern to the byte deserialized data if the predefined byte ordering pattern found is not in the lowest significant byte (LSB) position.

Byte ordering requires the following:

- A receiver with the byte deserializer enabled
- A predefined byte ordering pattern that must be ordered at the LSB position of data
- A predefined pad pattern

Receiver Phase Compensation FIFO

The low-speed parallel clock feeds the write clock, while the FPGA fabric interface clock feeds the read clock. The clocks must have 0 ppm difference in frequency or a receiver phase compensation FIFO underrun or overflow condition may result.

The FIFO supports the following operations:

- Phase compensation mode with various clocking modes on the read clock and write clock
- Registered mode with only one clock cycle of data path latency

Transmitter PCS Data path

Transmitter Phase Compensation FIFO

The transmitter Phase compensation FIFO is four words deep and interfaces with the transmitter channel PCS and the FPGA fabric or hard IP block. Transmitter Phase compensation FIFO makes up for the phase difference between the low-speed parallel clock and the FPGA fabric interface clock.

Byte Serializer

The Byte serializer divides the input data path by two to run the transceiver channel at higher data rates while keeping the FPGA fabric interface frequency within the maximum limit.

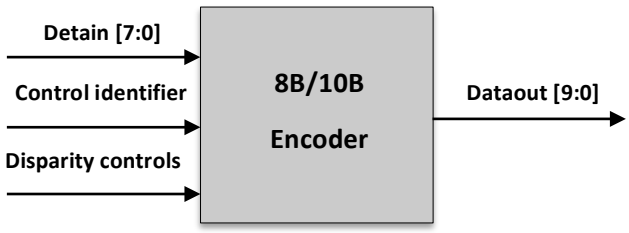
There are single and double width modes which byte serializer supports to operate on. The data path clock rate at the output of the byte serializer is twice the FPGA fabric-transmitter interface clock frequency. The byte serializer forwards the least significant word first followed by the most significant word.

8B/10B Encoder

8B/10B is a line code that maps 8-bit symbols to 10-bit symbols to achieve DC balance, bounded disparity and provide enough state changes to allow reasonable clock recovery in telecommunications. The principle is that the difference between the count of ones and zeros in a string of, at least 20 bits, is no more and two. And there should be no more than five ones or zeros in a row. This enables the communication to reduce the demand for the lower bandwidth limit of the channel necessary to transfer the signal.

In this project, the 8B/10B encoder generates 10-bit code groups from 8-bit code groups from 8-bit data and 1-bit control identifier with proper disparity according to the PCS reference diagram in the Clause 36 of the IEEE 802.3 specification. The 8B/10B encoder translates the 8-bit data to a 10-bit code group (control word or data word) with proper disparity. As shown in Figure x-x, if the tx_datak input is high, the encoder translates the input to a 10-bit control word and otherwise I will be a 10-bit data word.

Figure 2-8: 8B/10B Encoder

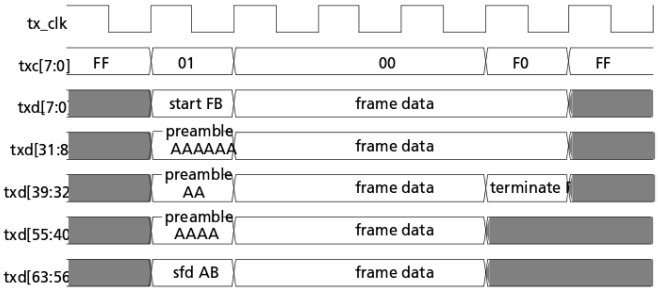


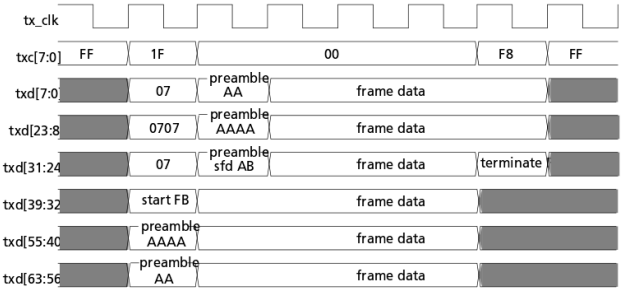
Transmitter Bit-slip

The transmitter bit-slip allows you to compensate for the channel-tochannel skew between multiple transmitter channels by slipping the data sent to the PMA. The maximum number of bits slipped equal to the width of the PMA-PCS minus 1.

B. SWAP

The output from the XAUI PHY arrives in only 2 cases. The initial sequence either starts at the first byte or the 5th byte of the 8 byte interface. To reduce complexity in the MAC, we decided to create a module that would buffer 4 bytes of data so that the initial sequence always starts at the first byte before going into the MAC.





The figure on the right shows the timing diagrams of when the initial sequence arrives at the first byte where as the figure on the right shows when the signal arrives at the fifth byte. After the SWAP module all signals to the MAC will be in the scenerio shown in the left.

C. FAST MAC

The media access controller (MAC) is the hardware that implements the media access control layer that interfaces the physical layer and the logical link control sub-layer and performs many functions such as appending and discarding pre-amples and checksums, calculating and checking checksums, filter malformed frames, and basic monitoring of input and output frames. For our project, we decided to design our own Fast MAC in order to increase throughput and decrease latency. Our MAC module acts as a bridge between the XAUI PHY and the memory module that interfaces with the Hard Processor System (HPS). Since the goal of the project is to build a platform for other projects, our MAC is made to be transparent as possible. The input of the MAC uses XGMII protocol which is the output of the XAUI PHY and output using a 72-bit Avalon ST interface.

In our project we decided to split the MAC into two modules. One for the receiving and the other for the transmitting. In the receiving module, we convert the data between interfaces, getting rid of the pre- and post-amples as well as compute the checksum to ensure that the frame is not corrupted. If it is the checksum error signal will turn high on the same cycle the end of packet is transmitted. Our receiving module only adds 3 clock cycle of latency for each packet.

For the transmitting module, the module converts the data from the Avalon ST interface into XGMII protocol for the XAUI. It also adds the pre and post-amples and appends the calculated checksum to the end of the packet.

Frame Checking Sequence

For Ethernet, the standard frame checking sequence or error-detection code for IEEE 802.3 is calculated using a 32-bit cyclic redundancy check (CRC32) algorithm. The CRC checksum is basically the remainder of a binary division between the data divided by a polynomial based on the CRC length. In our case, we have a 32 bit checksum thus our polynomial (also referred to as generator) is:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Or in binary:

100110000010001110110110111

For simplicity, we will be giving an example using CRC3 in which the generator is 1101 using 010111001 as the transmitted frame.

Figure 2-9: Frame Checking Sequence 1

```
1101 | 01011101 000   ← Append zeros equal to the size of the checksum
       1101
       ------
       1000           ← Perform XOR until more bits are needed
       1101
       ------
       01011          ← Omit leading zeros and bring down next bit
        1101
        ------
        01101
         1101         ← Repeat steps until run out of bits
         -------------
         0000 0100
              1101
              ------
              1001
              1101
              ------
              01000
               1101
               -----
                0101  ← Remainder is the checksum
```

When the CRC is appended to the data instead of the zeros, the algorithm will always give zeros as the resulting checksum letting us know that the checksum is indeed correct for the given data.

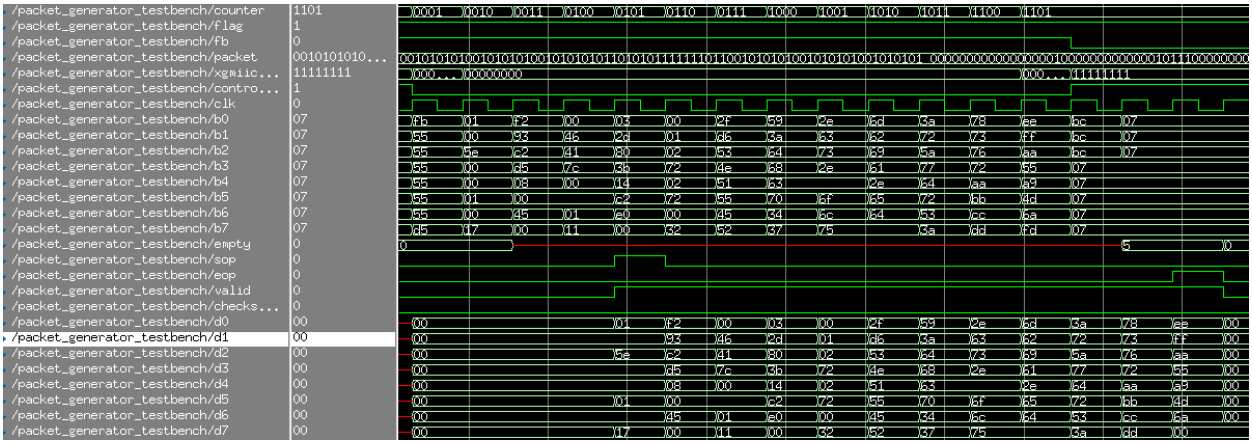
To increase performance, it is exponentially faster to use parallel computation especially when the input data is 64 bits. If we look at how a bit from the data is pulled and used for the XOR division each time, you can see it is similar to a shift. Also, since the algorithm is linear, CRC(X)

XOR CRC(Y) = CRC(X XOR Y), it is easy to see that there is a way to perform parallel calculations by using multiple input XORs. For more information and reference, we referred to the pa-per found at www.cypress.com/?docID=31573 and code from www.easics.com for the parallel computation method.

Figure 2-10: Frame Checking Sequence 1

```
1101 | 01011101 101   ← Append checksum to data
       1101
       ------
       1000            ← Compute as before
       1101
       ------
       01011
        1101
        ------
        01101
         1101
         ------------
         0000 0110
              1101
              ------
              1011
              1101
              ------
              01101
               1101
               ------
               0000   ← Checksum will be zero
```

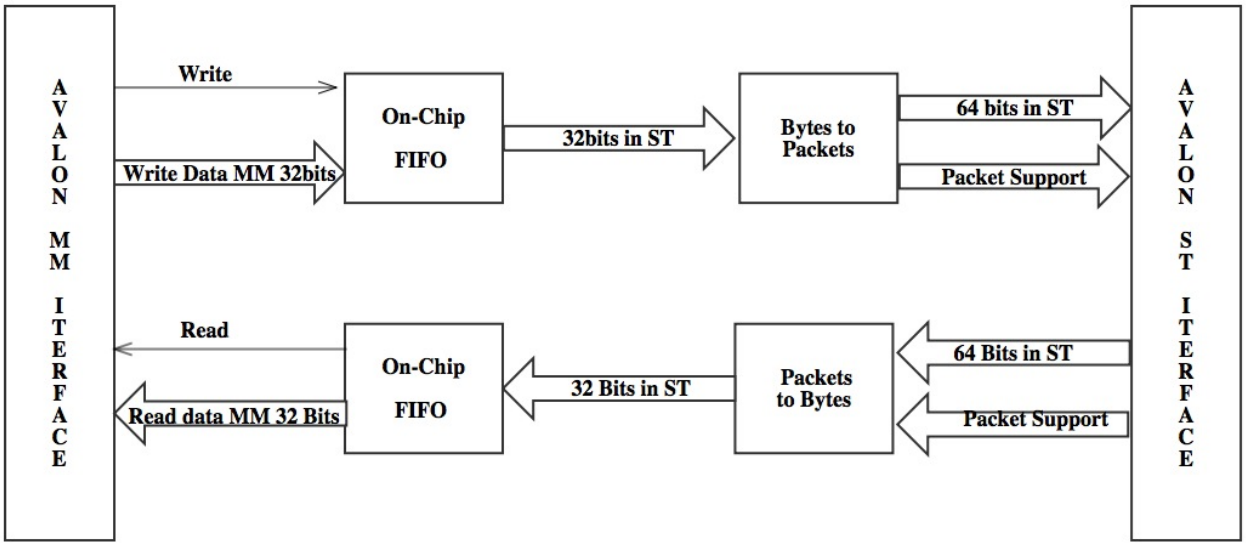
Figure 1-2: FAST MAC Simulations in ModSim



D. TX FIFO

When software transmitting data to PHY, it cannot transmit at each clock cycle because the higher latency compared to hardware. Also, the MM to ST converter would take some clock cycles to organize the 64 bits Stream data when the input is 32bits. As a result, each output data settles on wire for several clock cycles and ends up with a pulse valid signal. Unfortunately, this scenario could not be passed by XAUI PHY. In order to transmit the 64bits parallel data one time per clock cycle, we instantiate a FIFO to hold the data output from dual clock FIFO, and "squeeze" the data to 8 bytes per clock cycle with a continuously valid signal.

E. Avalon ST to Avalon MM



Packets/ Bytes

One of the big challenge in last scenario is the byte to packet converter only supports 8 bits(1 bytes). While the 64-bits DFA could only generate a valid sop when the first 8 bits is the first byte in the packet. However, the first byte in the packet could be inserted into the middle of the 64 bits Avalon ST interface due to the last packet's tail. In order to solve this dilemma, a customized byte to packet converter based on self-designed protocol could be utilized.

Now 7a is extended to 32 bits 7a 7a 7a 7a, when this sequence is detected, an sop is generated. Same, 7b 7b 7b 7b is used to indicate end of packet.

Only if in the same clock cycle, a data sequence is 7a 7a 7a 7a or 7b 7b 7b 7b, we need to pad 7d 7d 7d 7d with 5a 5a 5a 5a or 5b 5b 5b 5b to indicate these are not the control codes but the valid data. By XORing with 20202020 we could restore the original valid data. At same time, a empty signal is generated by software after the end of packet to indicate how many bits is valid in the last clock cycle. For instance, if the empty is 32'h2, only the first two bytes in the last

clock cycle is valid. When converting to ST, a 64 bits empty is generated, if the packet need even clock cycles to transmit in MM interface, it just need to pad 4 bytes zero to the empty in ST, i.e. 64'h6. If the packet need odd number of clock cycles, it has to pad 4 bytes valid in front of the original 32-bits mask, i.e. 64'h2 because the last odd cycle fills the first 32-bits in the last 64-bit clock cycle in ST.

F. Avalon ST to Linux

To connect our hardware module with Linux operating system, we need to build our own Ethernet driver, which can transmit and receive packets through software via Avalon-MM. For this project, we developed a unique protocol that can handle all scenarios when transmitting and receiving packets between hardware and software, we developed and compared several different protocols that might be suitable for our module, and finally we found that the following one is of the most efficiency when transmitting data between software and hardware:

The transmission is 32-bits wide (4-bytes), and we occupied four unique 32-bit patterns as control signals:

- Start of packet: 0x7a7a7a7a
- End of packet: 0x7b7b7b7b
- Change of channel: 0x7c7c7c7c
- Escape: 0x7d7d7d7d
- IDLE for transmission: 0x7e7e7e7e

The reason for using these special patterns is because there is no control signal when transmitting data via Avalon-MM and the data in packet can be any value, so it is necessary to use some data pattern as control signal and replace them with other patterns when we need to transmit these patterns as actual data.

Software Write

When software is sending a packet to hardware, it will first send 32-bits start of packet control signal(0x7a7a7a7a), followed by all the content of the packet, when the 32-bits to be send next cycle is in conflict with one of those control signals, then an escape command is transmitted, followed by the actual content XOR with 0x20202020, for example, when the software is transmitting 0x7b7b7b7b to the hardware, first 32-bits of escape signal 0x7d7d7d7d is transmitted, then 0x5b5b5b5b(0x7b7b7b7b XOR with 0x20202020) is transmitted. At the end of each packet, a signal of 0x7b7b7b7b is transmitted, then followed by a mask indicating the number of bytes that is valid in the last cycle, 0xFF means this byte in the following cycle is part of the packet, 0x00 means this byte in the following cycle is not part of the packet. For example, if there are only three bytes left to be sent in the last cycle, then 0xFFFFFF00 is transmitted, followed by the last three bytes with 0x00 appending in the end, such as 0xaabbcc00, where 0xaabbcc is the ending of the packet.

After software has sent one packet to the proper address, the software will check on another hardware address to read the 32-bit Count-of-Output signal, which is added by one to inform

software when hardware has successfully recognized one packet, so the software will know if the packet has been delivered to hardware completely and whether it can start sending the next one.

Software Read

The same protocol applies when software is reading a packet from hardware module. Whenever the software recognized a start of packet signal, the reading cycle will begin, 32-bits are read and write to input file each cycle except it receives one of the control signals mentioned above, in which case special action will be took to handle scenarios like end of packet(including mask for the last 32-bits of packet), escape reverse as well as the idle signal(0x7e7e7e7e), which means that there is no valid data from hardware this cycle, and software will not write anything to file in this cycle.

By using the protocol described above, we can successfully transmit data from software to hardware as well as hardware to software in a bandwidth of 32-bits. In the hardware module, the 32-bits data will first been transformed into byte wise data and then into 64 bit Avalon-ST signal, see “Avalon MM to ST/ ST to MM convertor” for more details.

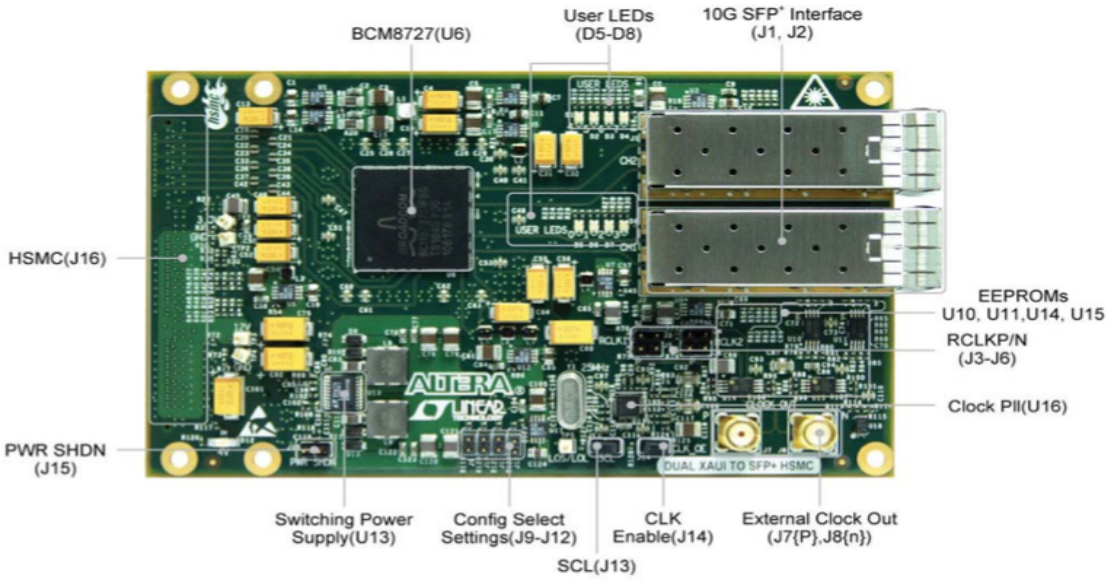
Efficiency analysis and other alternative protocols

Bytewise escape: this is also a reasonable way to generate and recognize control signals, however, since the possibility of running into a confliction with bytewise control signals when transmitting a single byte of actual data is much greater than that with 32-bits control signals, i.e. 0x7a as start of packet comparing with 0x7a7a7a7a, the operation of escape will be more frequent when using this protocol which will results into the decreasing of the actual data transmitting rate.

Occupying more bits as control signal: this is actually an efficient way for hardware to process parallel data, however, software is not capable of running multiple command in one cycle, which means the software will spend more time when it needs to process control bits first and then actual data in each round. So although this method is good for hardware to work more efficiently in parallel works, since the software will run slower in this way, the overall efficiency is not improved comparing to the protocol we use.

Direct memory access(DMA): With DMA, software and hardware can access and write data on memory separately, there is no need to consider SOP/EOP or escape, the writer need to tell the reader where does the packet start in memory as well as what is the length of the packet, and reader can access the data directly and asynchronously. However, it is necessary to handle memory allocation in both software and hardware, which is not trivial especially when we have limited time developing this project, we thus decide to save this method for future improvement.

4. Tutorial



Tutorial for XAUI daughter card Implementation

The XAUI daughter card is to provide a platform to transmit packages between SFP+ interface (Cable side) and HSMC interface (XAUI PHY side). By this board, we could design 10G Ethernet systems based on transceiver host boards that support XAUI interfaces. The transceiver could be implemented on Cyclone V by XAUI PHY from Altera.

A. On-Board Configuration

IMPORTANT: Attach the Heat Sink on the top of BCM8727 chip. The chip would generate a great deal of heat when working. Missing of heat sink may cause the chip on daughter card to overheat and sustain damage making the board defunct. Do not touch anywhere on the board except for the edge.

Jumpers: Before powering on the host board on, make sure to install a shunt on J13 and J14. Then plug the HSMC board into the host board.

Figure 3-1: J13 and J14 Jumper Functions

Board Reference	Signal Name	IO Standard	Function
J13	Si5338_SCL	3.3V	When using the Si5334C (default) this pin is a ground pin. It must be pulled to GND. When using the Si5338 device this pin is SCL.
J14	CLK_OE	3.3V	When using the Si5334C (default) this pin is output enable (OEB, active low). When pulled low all programmed outputs are active. When using the Si5338 device this pin is SDA.

B. PHY Configurations

Altera's STRATIX device family is famous for high frequency Ethernet embedded system. So almost all the resources we found about the implementation of this 10Gb/s XAUI daughter card with FPGA board are on STRATIX IV board, which is the common host board for high frequency trading. As a result, it is highly uncertain that whether Cyclone V could also support this 10Gb/s daughter card. In order to connect the daughter card on host board, we referred the configurations from a reference design provided by TerasIC.

1. Import Assignment Settings

First, we got the STRATIX IV pin locations of these HSMC control signals from the STRATIX IV top level signals, and then used these pin locations to trace the HSMC port number in the STRATIX IV schematic layout. The HSMC port number is a universal standard. As a result, we could match all these HSMC configuration (control) signals through the schematic layout of SoCKit board to the project's top level. The configuration signals are shown in Table 3-2:

Figure 3-2: Configuration Signals for Pin Assignment I

HSMC Number	SoCKit(Cyclone V) Schematic Pin Name	Schematic Name	DUAL XAUI Signal Name	Signal Function
14	HSMC_GXB_RX_P[4]	HSM_RX_P4	XAUI_RX_2P0	XAUI Parallel Receive
10	HSMC_GXB_RX_P[5]	HSM_RX_P5	XAUI_RX_2P1	
6	HSMC_GXB_RX_P[6]	HSM_RX_P6	XAUI_RX_2P2	
2	HSMC_GXB_RX_P[7]	HSM_RX_P7	XAUI_RX_2P3	
13	HSMC_GXB_TX_P[4]	HSM_TX_P4	XAUI_TX_2P0	XAUI Parallel Transmit
9	HSMC_GXB_TX_P[5]	HSM_TX_P5	XAUI_TX_2P1	
5	HSMC_GXB_TX_P[6]	HSM_TX_P6	XAUI_TX_2P2	
1	HSMC_GXB_TX_P[7]	HSM_TX_P7	XAUI_TX_2P3	
72	HSMC_RX_P4	PHYRESET	PHYRESET	PHY Reset, Active low.
56	HSMC_RX_N1	MDC1	MDC1	Master Input/Slave Output Channel 1/2. Management Data Clock channel 2 for dual MDIO device. Management Data Clock for single device (default)
54	HSMC_RX_P1	MDIO1	MDIO1	
116	HSMC_RX_N10	MDC2	MDC2	
114	HSMC_RX_P10	MDIO2	MDIO2	
65	HSMC_TX_P3	PRTAD4	PRTAD4	PHY Address bit 4/3/2/1; Channel 1/2 PHY Address LSB.
67	HSMC_TX_N3	PRTAD3	PRTAD3	
71	HSMC_TX_P4	PRTAD2	PRTAD2	
73	HSMC_TX_N4	PRTAD1	PRTAD1	

77	HSMC_TX_P5	PRTAD01	PRTAD01	
79	HSMC_TX_N5	PRTAD02	PRTAD02	
125	HSMC_TX_P12	CONFIG0_1	CONFIG0_1	Configuration mode channel 1, bit 0. Internally pulled down. Configuration mode channel 2, bit 0. Internally pulled down. Configuration mode channel 1, bit 1. Internally pulled down. Configuration mode channel 2, bit 1. Internally pulled down.
127	HSMC_TX_N12	CONFIG0_2	CONFIG0_2	
131	HSMC_TX_P13	CONFIG1_1	CONFIG1_1	
133	HSMC_TX_N13	CONFIG1_2	CONFIG1_2	
95	HSMC_CLKOUT_P1	SS338_CLKIN	SS338_CLKIN	CLKIN to Si5334 device
121	HSMC_TX_N11	OPINLVL	NVMA1SEL	Non-volatile Memory Select
132	HSMC_RX_P13	NVMPROT	NVMPROT	Non-volatile Memory Protect
137	HSMC_TX_P14	PRTAD01	GPIO0_1	Programmable general purpose I/O.
139	HSMC_TX_N14	PRTAD02	GPIO1_1	
110	HSMC_RX_N9	SS_N1	GPIO0_2	
140	HSMC_RX_N14	GFIO1_2	GPIO1_2	
50	HSMC_RX_N0	SER_BOOT	SER_BOOT	SPI ROM Boot Enable active high.
48	HSMC_RX_P0	SMB8PDSEL1	SMBSPDSEL1	2-wire Speed Select channel 1/2.
108	HSMC_RX_P9	SMB8PDSEL2	SMBSPDSEL2	
119	HSMC_TX_P11	OPOUTLVL	SMBWEN	2-wire Write Enable
74	HSMC_RX_N4	TXCNOFF1	TXONOFF1	Transmit Driver On or Off channel 1/2.
134	HSMC_RX_N13	TXCNOFF2	TXONOFF2	
59	HSMC_TX_P2	OPOUTLVL	OPOUTLVL	Optical Control Output Level.
61	HSMC_TX_N2	OPINLVL	OPINLVL	Optical Control Output Level.
102	HSMC_RX_P8	OPRXLOB2	OPRXLOS2	Optical Receiver Loss of Signal Channel 2.
104	HSMC_RX_N8	OPTXPLT2	OPTXFLT2	Optical Transmitter Fault Indicator Channel 2
113	HSMC_TX_P10	SFP_TXD031	SFP_TXDIS2	Optical Transmitter Enable channel 2.
115	HSMC_TX_N10	SFP_TXR310	SFP_TXRS20	Rate Select 0 for SFP+ module receiver channel 2, pulled high via R3. This sets the input rate > 4.25 GBd, pull low for rates ≤ 4.25 GBd.

2. Set control signals as below

```
assign CONFIG0_1 = 1'b1;
assign CONFIG1_1 = 1'b0;
assign CONFIG0_2 = 1'b1;
assign CONFIG1_2 = 1'b0;
assign SS338_CLKIN = 1'b0;

assign SER_BOOT  = 1'b0;
assign SMBSPDSEL1 = 1'b0;
assign SMBSPDSEL2 = 1'b0;
assign SMBWEN    = 1'b1;
assign GPIO0_1   = 1'b0;
assign GPIO1_1   = 1'b0;
assign GPIO0_2   = 1'b0;
assign GPIO1_2   = 1'b0;
assign NVMA1SEL = 1'b1;
assign NVMPROT  = 1'b0;

assign MDIO1 = !MDOEN1? MDO1 : 1'bz;
assign MDIN1 = MDIO1;
assign MDC2   = 1'bz;
assign MDIO2 = 1'bz;
assign PHYRESET =  KEY[1];
assign STOPMON  = ~KEY[2];

{PRTAD4,PRTAD3,PRTAD2,PRTAD1,PRTAD01} = 5'b00000;
assign  PRTAD02 = 1'b0;
assign TXONOFF1 = 1'b1;
assign TXONOFF2 = 1'b1;
assign OPOUTLVL = 1'b0;
assign OPINLVL  = 1'b1;
assign USER_LED_R = 8'b00001111;
assign USER_LED_G = 8'b11110000;
```

C. Reference PLL Generation

Generate 156.25MHz clock from an on-chip 50MHz oscillator.
Feed this clock to the reference PLL clock input in XAUI IP.

D. XAUI IP Implementation

Select Altera XAUI IP core from Mega Wizard, and chose Soft XAUI in interface type and leave the base data rate and PLL type as default at 3125Mps and CMU type.

E. MAC Implementation

Import mac_rx.sv and mac_tx.sv from the project files. The MAC runs on the clock produced by the XAUI PHY, xgmii_clk_rx. Make the appropriate connects as shown in the design or reference the top level module, “SoCKit_top.sv”.

F. HPS implemenation

To implement the HPS, you must make sure that the top level has all the required assignments for the FPGA. For our project, we used the implementation used for Lab 3 as a reference for the assignments. Open lab3.qys in Qsys and generate to incorporate the module into your project. Make the appropriate connects as shown in the design or reference the top level module, “SoCKit_top.sv”.

F. Timing Constraints

You MUST ensure that timing constraints are set for the clocks. In order to add a constraint, open up TimeQuest and enter the constraints for the signal. For our project, the PLL reference clock, clockbuff0 MUST be implemented with the 50Mhz oscillator (OSC_50_B4A) with duty cycle set at 3.2ns and 6.4ns for period.

5. References

http://www.terasic.com.tw/cgibin/page/archive_download.pl?Language=English&No=597&FID=ef795de1ac586e31317766f74e7ffedd

http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=167&No=816

www.cypress.com/?docID=31573

www.easics.com

www.outputlogic.com

http://www.altera.com/literature/ug/xcvr_user_guide.pdf

http://www.altera.com/literature/an/AN638.pdf

http://www.altera.com/technology/high_speed/protocols/10gb_ethernet/pro-10gb_ethernet.html

http://www.altera.com/support/refdesigns/ip/interface/ref-10gbe-hardware-demo.html

http://www.altera.com/literature/manual/mnl_avalon_spec.pdf

6. Contribution

John Chaiyasarikul:MAC, checksum, SWAP, XAUI, Testbench, Configurations*

Yumeng Xu: 10GB Example implementation, XAUI, MAC, TX FIFO, Testbench, Configurations*

Shuguan Yang: Linux Kernel and Software Code (not implemented)

Jian Zhong: Documentation, Pin Assignments

Quishi Ding (TA): Avalon MM and ST Conversion

*Configurations: Pin Assignments/Mapping, PHY Control Signals, RESET, Clocking

Appendix I. Pin Assignment Mapping

Stratix VI Stratix VI- Signal Name	Stratix VI Pin Locattion	Stratix Pin Standard	Stratix VI Schematic Pin Name	HSMC Number	SoCKit(Cyclone V) Schematic Pin Name	SoCKit Signal pin Location	DUAL XAUI Schematic Nam	DUAL XAUI Signal Name
	AU2		HSMA_RX_P0	30	HSMC_GXB_RX_P[0]	AE2	HSM_RX_P0	XAUI_RX_1P0
	AR2		HSMA_RX_P1	26	HSMC_GXB_RX_P[1]	AC2	HSM_RX_P1	XAUI_RX_1P1
	AJ2		HSMA_RX_P2	22	HSMC_GXB_RX_P[2]	AA2	HSM_RX_P2	XAUI_RX_1P2
	AG2		HSMA_RX_P3	18	HSMC_GXB_RX_P[3]	W2	HSM_RX_P3	XAUI_RX_1P3
XAUI_RX[0]	AE2	1.4-V PCML	HSMA_RX_P4	14	HSMC_GXB_RX_P[4]	U2	HSM_RX_P4	XAUI_RX_2P0
XAUI_RX[1]	AC2	1.4-V PCML	HSMA_RX_P5	10	HSMC_GXB_RX_P[5]	R2	HSM_RX_P5	XAUI_RX_2P1
XAUI_RX[2]	U2	1.4-V PCML	HSMA_RX_P6	6	HSMC_GXB_RX_P[6]	N2	HSM_RX_P6	XAUI_RX_2P2
XAUI_RX[3]	R2	1.4-V PCML	HSMA_RX_P7	2	HSMC_GXB_RX_P[7]	J2	HSM_RX_P7	XAUI_RX_2P3
	AU1		HSMA_RX_N0	32	HSMC_GXB_RX_N[0]	AE1	HSM_RX_N0	XAUI_RX_1N0
	AR1		HSMA_RX_N1	28	HSMC_GXB_RX_N[1]	AC1	HSM_RX_N1	XAUI_RX_1N1
	AJ1		HSMA_RX_N2	24	HSMC_GXB_RX_N[2]	AA1	HSM_RX_N2	XAUI_RX_1N2
	AG1		HSMA_RX_N3	20	HSMC_GXB_RX_N[3]	W1	HSM_RX_N3	XAUI_RX_1N3
	AE1		HSMA_RX_N4	16	HSMC_GXB_RX_N[4]	U1	HSM_RX_N4	XAUI_RX_2N0
	AC1		HSMA_RX_N5	12	HSMC_GXB_RX_N[5]	R1	HSM_RX_N5	XAUI_RX_2N1
	U1		HSMA_RX_N6	8	HSMC_GXB_RX_N[6]	N1	HSM_RX_N6	XAUI_RX_2N2
	R1		HSMA_RX_N7	4	HSMC_GXB_RX_N[7]	J1	HSM_RX_N7	XAUI_RX_2N3
	AT4		HSMA_TX_P0	29	HSMC_GXB_TX_P[0]	AD4	HSM_TX_P0	XAUI_TX_1P0
	AP4		HSMA_TX_P1	25	HSMC_GXB_TX_P[1]	AB4	HSM_TX_P1	XAUI_TX_1P1
	AH4		HSMA_TX_P2	21	HSMC_GXB_TX_P[2]	Y4	HSM_TX_P2	XAUI_TX_1P2
	AF4		HSMA_TX_P3	17	HSMC_GXB_TX_P[3]	V4	HSM_TX_P3	XAUI_TX_1P3
XAUI_TX[0]	AD4	1.4-V PCML	HSMA_TX_P4	13	HSMC_GXB_TX_P[4]	T4	HSM_TX_P4	XAUI_TX_2P0
XAUI_TX[1]	AB4	1.4-V PCML	HSMA_TX_P5	9	HSMC_GXB_TX_P[5]	P4	HSM_TX_P5	XAUI_TX_2P1
XAUI_TX[2]	T4	1.4-V PCML	HSMA_TX_P6	5	HSMC_GXB_TX_P[6]	M4	HSM_TX_P6	XAUI_TX_2P2
XAUI_TX[3]	P4	1.4-V PCML	HSMA_TX_P7	1	HSMC_GXB_TX_P[7]	H4	HSM_TX_P7	XAUI_TX_2P3
	AT3		HSMA_TX_N0	31	HSMC_GXB_TX_N[0]	AD3	HSM_TX_N0	XAUI_RX_1N0
	AP3		HSMA_TX_N1	27	HSMC_GXB_TX_N[1]	AB3	HSM_TX_N1	XAUI_RX_1N1
	AH3		HSMA_TX_N2	23	HSMC_GXB_TX_N[2]	Y3	HSM_TX_N2	XAUI_RX_1N2
	AF3		HSMA_TX_N3	19	HSMC_GXB_TX_N[3]	V3	HSM_TX_N3	XAUI_RX_1N3
	AD3		HSMA_TX_N4	15	HSMC_GXB_TX_N[4]	T3	HSM_TX_N4	XAUI_RX_2N0
	AB3		HSMA_TX_N5	11	HSMC_GXB_TX_N[5]	P3	HSM_TX_N5	XAUI_RX_2N1
	T3		HSMA_TX_N6	7	HSMC_GXB_TX_N[6]	M3	HSM_TX_N6	XAUI_RX_2N2
	P3		HSMA_TX_N7	3	HSMC_GXB_TX_N[7]	H3	HSM_TX_N7	XAUI_RX_2N3
LEDHSMC_GRN[7]	PIN_AG6		HSMA_RX_D_P15	144	HSMC_RX_P15		USER_LED_G7	USER_LED_G7
LEDHSMC_GRN[6]	PIN_AG5		HSMA_RX_D_N15	146	HSMC_RX_N15		USER_LED_G6	USER_LED_G6
LEDHSMC_GRN[5]	PIN_AB9		HSMA_RX_D_P16	150	HSMC_RX_P16		USER_LED_G5	USER_LED_G5
LEDHSMC_GRN[4]	PIN_AC8		HSMA_RX_D_N16	152	HSMC_RX_N16		USER_LED_G4	USER_LED_G4
LEDHSMC_GRN[3]	PIN_AT6		HSMA_RX_D_P6	84	HSMC_RX_P6		USER_LED_G3	USER_LED_G3
LEDHSMC_GRN[2]	PIN_AU6		HSMA_RX_D_N6	86	HSMC_RX_N6		USER_LED_G2	USER_LED_G2
LEDHSMC_GRN[1]	PIN_AR5		HSMA_RX_D_P7	90	HSMC_RX_P7		USER_LED_G1	USER_LED_G1
LEDHSMC_GRN[0]	PIN_AT5		HSMA_RX_D_N7	92	HSMC_RX_N7		USER_LED_G0	USER_LED_G0
LEDHSMC_RED[7]	PIN_AB11		HSMA_TX_D_P15	143	HSMC_TX_P15		USER_LED_R7	USER_LED_R7
LEDHSMC_RED[6]	PIN_AB10		HSMA_TX_D_N15	145	HSMC_TX_N15		USER_LED_R6	USER_LED_R6
LEDHSMC_RED[5]	PIN_AC11		HSMA_TX_D_P16	149	HSMC_TX_P16		USER_LED_R5	USER_LED_R5
LEDHSMC_RED[4]	PIN_AC10		HSMA_TX_D_N16	151	HSMC_TX_N16		USER_LED_R4	USER_LED_R4
LEDHSMC_RED[3]	PIN_AH10		HSMA_TX_D_P6	83	HSMC_TX_P6		USER_LED_R3	USER_LED_R3
LEDHSMC_RED[2]	PIN_AJ10		HSMA_TX_D_N6	85	HSMC_TX_N6		USER_LED_R2	USER_LED_R2
LEDHSMC_RED[1]	PIN_AH9		HSMA_TX_D_P7	89	HSMC_TX_P7		USER_LED_R1	USER_LED_R1
LEDHSMC_RED[0]	PIN_AH8		HSMA_TX_D_N7	91	HSMC_TX_N7		USER_LED_R0	USER_LED_R0
PHYRESET	PIN_AV5		HSMA_RX_D_P4	72	HSMC_RX_P4		PHYRESET	PHYRESET
MDC1	PIN_AU8		HSMA_RX_D_N1	56	HSMC_RX_N1		MDC1	MDC1
MDIO1	PIN_AT8		HSMA_RX_D_P1	54	HSMC_RX_P1		MDIO1	MDIO1
MDC2	PIN_AM5		HSMA_RX_D_N10	116	HSMC_RX_N10		MDC2	MDC2
MDIO2	PIN_AM6		HSMA_RX_D_P10	114	HSMC_RX_P10		MDIO2	MDIO2
PRTAD4	PIN_AL8		HSMA_TX_D_P3	65	HSMC_TX_P3		PRTAD4	PRTAD4
PRTAD3	PIN_AM8		HSMA_TX_D_N3	67	HSMC_TX_N3		PRTAD3	PRTAD3
PRTAD2	PIN_AK9		HSMA_TX_D_P4	71	HSMC_TX_P4		PRTAD2	PRTAD2
PRTAD1	PIN_AL9		HSMA_TX_D_N4	73	HSMC_TX_N4		PRTAD1	PRTAD1
PRTAD1	PIN_AL9		HSMA_TX_D_N4	73	HSMC_TX_N4		PRTAD1	PRTAD1
PRTAD01	PIN_AK8		HSMA_TX_D_P5	77	HSMC_TX_P5		PRTAD01	PRTAD01
PRTAD02	PIN_AK7		HSMA_TX_D_N5	79	HSMC_TX_N5		PRTAD02	PRTAD02
CONFIG0_1	PIN_AE11		HSMA_TX_D_P12	125	HSMC_TX_P12		CONFIG0_1	CONFIG0_1
CONFIG0_2	PIN_AE10		HSMA_TX_D_N12	127	HSMC_TX_N12		CONFIG0_2	CONFIG0_2
CONFIG1_1	PIN_AD13		HSMA_TX_D_P13	131	HSMC_TX_P13		CONFIG1_1	CONFIG1_1
CONFIG1_2	PIN_AD12		HSMA_TX_D_N13	133	HSMC_TX_N13		CONFIG1_2	CONFIG1_2
si5338_CLKIN	PIN_AL10		HSMA_CLK_OUT_P1	95	HSMC_CLKOUT_P1		SS338_CLKIN	SS338_CLKIN
NVMA1SEL	PIN_AD9		HSMA_TX_D_N11	121	HSMC_TX_N11		OPINLVL	NVMA1SEL
NVMPROT	PIN_AJ6		HSMA_RX_D_P13	132	HSMC_RX_P13		NVMPROT	NVMPROT
GPIO0_1	PIN_AB13		HSMA_TX_D_P14	137	HSMC_TX_P14		PRTAD01	GPIO0_1
GPIO1_1	PIN_AB12		HSMA_TX_D_N14	139	HSMC_TX_N14		PRTAD02	GPIO1_1
GPIO0_2	PIN_AN5		HSMA_RX_D_N9	110	HSMC_RX_N9		SS_N1	GPIO0_2
GPIO1_2	PIN_AH5		HSMA_RX_D_N14	140	HSMC_RX_N14		GPIO1_2	GPIO1_2
SER_BOOT	PIN_AU9		HSMA_RX_D_N0	50	HSMC_RX_N0		SER_BOOT	SER_BOOT
SMBPDSEL1	PIN_AT9		HSMA_RX_D_P0	48	HSMC_RX_P0		SMB8PDSEL1	SMBSPDSEL1
SMBPDSEL2	PIN_AN6		HSMA_RX_D_P9	108	HSMC_RX_P9		SMB8PDSEL2	SMBSPDSEL2
SMBWEN	PIN_AD10		HSMA_TX_D_P11	119	HSMC_TX_P11		OPOUTLVL	SMBWEN
TXONOFF1	PIN_AW4		HSMA_RX_D_N4	74	HSMC_RX_N4		TXCNOFF1	TXONOFF1
TXONOFF2	PIN_AJ5		HSMA_RX_D_N13	134	HSMC_RX_N13		TXCNOFF2	TXONOFF2
OPOUTLVL	PIN_AE13		HSMA_TX_D_P2	59	HSMC_TX_P2		OPOUTLVL	OPOUTLVL
OPINLVL	PIN_AE12		HSMA_TX_D_N2	61	HSMC_TX_N2		OPINLVL	OPINLVL
OPRXLOS2	PIN_AP6		HSMA_RX_D_P8	102	HSMC_RX_P8		OPRXLOB2	OPRXLOS2
OPTXFLT2	PIN_AP5		HSMA_RX_D_N8	104	HSMC_RX_N8		OPTXPLT2	OPTXFLT2
OPTXENB2	PIN_AF11		HSMA_TX_D_P10	113	HSMC_TX_P10		SFP_TXD031	SFP_TXDIS2
OPTXRST2	PIN_AF10		HSMA_TX_D_N10	115	HSMC_TX_N10		SFP_TXR310	SFP_TXRS20
LASI2	PIN_AK5		HSMA_RX_D_N12	128	HSMC_RX_N12		LA812	LASI2

II. Source Code

Top Level :

```
// ============================================================================
// Copyright (c) 2013 by Terasic Technologies Inc.
// ============================================================================
//
// Permission:
//
//   Terasic grants permission to use and modify this code for use
//   in synthesis for all Terasic Development Boards and Altera Development
//   Kits made by Terasic.  Other use of this code, including the selling
//   ,duplication, or modification of any portion is strictly prohibited.
//
// Disclaimer:
//
//   This VHDL/Verilog or C/C++ source code is intended as a design reference
//   which illustrates how these types of functions can be implemented.
//   It is the user's responsibility to verify their design for
//   consistency and functionality through the use of formal
//   verification methods.  Terasic provides no warranty regarding the use
//   or functionality of this code.
//
// ============================================================================
//
//  Terasic Technologies Inc
//  9F., No.176, Sec.2, Gongdao 5th Rd, East Dist, Hsinchu City, 30070. Taiwan
//
//
//                     web: http://www.terasic.com/
//                     email: support@terasic.com
```

```
//
// =====================================================================
// =====================================================================
//
// Major Functions:          SoCKit_Default
//
// =====================================================================
// Revision History :
// =====================================================================
//   Ver  :| Author           :| Mod. Date :| Changes Made:
//   V1.0 :| xinxian          :| 04/02/13  :| Initial Revision
// =====================================================================

//`define ENABLE_DDR3
//`define ENABLE_HPS
//`define ENABLE_HSMC_XCVR

`define ENABLE_MDIO

module SoCKit_Top(

                        ///////////AUD//////////////
                        AUD_ADCDAT,
                        AUD_ADCLRCK,
                        AUD_BCLK,
                        AUD_DACDAT,
                        AUD_DACLRCK,
                        AUD_I2C_SCLK,
                        AUD_I2C_SDAT,
                        AUD_MUTE,
                        AUD_XCK,
```

```
`ifdef ENABLE_DDR3
                ////////DDR3////////
                DDR3_A,
                DDR3_BA,
                DDR3_CAS_n,
                DDR3_CKE,
                DDR3_CK_n,
                DDR3_CK_p,
                DDR3_CS_n,
                DDR3_DM,
                DDR3_DQ,
                DDR3_DQS_n,
                DDR3_DQS_p,
                DDR3_ODT,
                DDR3_RAS_n,
                DDR3_RESET_n,
                DDR3_RZQ,
                DDR3_WE_n,
`endif /*ENABLE_DDR3*/

                ////////FAN////////
                FAN_CTRL,

`ifdef ENABLE_HPS
                ////////HPS////////
                HPS_CLOCK_25,
                HPS_CLOCK_50,
                HPS_CONV_USB_n,
                HPS_DDR3_A,
                HPS_DDR3_BA,
                HPS_DDR3_CAS_n,
                HPS_DDR3_CKE,
```

HPS_DDR3_CK_n,

HPS_DDR3_CK_p,

HPS_DDR3_CS_n,

HPS_DDR3_DM,

HPS_DDR3_DQ,

HPS_DDR3_DQS_n,

HPS_DDR3_DQS_p,

HPS_DDR3_ODT,

HPS_DDR3_RAS_n,

HPS_DDR3_RESET_n,

HPS_DDR3_RZQ,

HPS_DDR3_WE_n,

HPS_ENET_GTX_CLK,

HPS_ENET_INT_n,

HPS_ENET_MDC,

HPS_ENET_MDIO,

HPS_ENET_RESET_n,

HPS_ENET_RX_CLK,

HPS_ENET_RX_DATA,

HPS_ENET_RX_DV,

HPS_ENET_TX_DATA,

HPS_ENET_TX_EN,

HPS_FLASH_DATA,

HPS_FLASH_DCLK,

HPS_FLASH_NCSO,

HPS_GSENSOR_INT,

HPS_I2C_CLK,

HPS_I2C_SDA,

HPS_KEY,

HPS_LCM_D_C,

HPS_LCM_RST_N,

HPS_LCM_SPIM_CLK,

```
                HPS_LCM_SPIM_MISO,
                HPS_LCM_SPIM_MOSI,
                HPS_LCM_SPIM_SS,
                HPS_LED,
                HPS_LTC_GPIO,
                HPS_RESET_n,
                HPS_SD_CLK,
                HPS_SD_CMD,
                HPS_SD_DATA,
                HPS_SPIM_CLK,
                HPS_SPIM_MISO,
                HPS_SPIM_MOSI,
                HPS_SPIM_SS,
                HPS_SW,
                HPS_UART_RX,
                HPS_UART_TX,
                HPS_USB_CLKOUT,
                HPS_USB_DATA,
                HPS_USB_DIR,
                HPS_USB_NXT,
                HPS_USB_RESET_PHY,
                HPS_USB_STP,
                HPS_WARM_RST_n,
`endif /*ENABLE_HPS*/


                ////////IRDA////////
                IRDA_RXD,

                ////////KEY////////
                KEY,
```

```
/////////LED/////////
LED,

/////////OSC/////////
OSC_50_B3B,
OSC_50_B4A,
OSC_50_B5B,
OSC_50_B8A,

/////////PCIE/////////
PCIE_PERST_n,
PCIE_WAKE_n,

/////////RESET/////////
RESET_n,

/////////SI5338/////////
SI5338_SCL,
SI5338_SDA,

/////////SW/////////
SW,

/////////TEMP/////////
TEMP_CS_n,
TEMP_DIN,
TEMP_DOUT,
TEMP_SCLK,

/////////USB/////////
USB_B2_CLK,
USB_B2_DATA,
```

```
USB_EMPTY,
USB_FULL,
USB_OE_n,
USB_RD_n,
USB_RESET_n,
USB_SCL,
USB_SDA,
USB_WR_n,

////////VGA////////
VGA_B,
VGA_BLANK_n,
VGA_CLK,
VGA_G,
VGA_HS,
VGA_R,
VGA_SYNC_n,
VGA_VS,
//////////hps/////////
memory_mem_a,
memory_mem_ba,
memory_mem_ck,
memory_mem_ck_n,
memory_mem_cke,
memory_mem_cs_n,
memory_mem_ras_n,
memory_mem_cas_n,
memory_mem_we_n,
memory_mem_reset_n,
memory_mem_dq,
memory_mem_dqs,
memory_mem_dqs_n,
```

memory_mem_odt,

memory_mem_dm,

memory_oct_rzqin,

hps_io_hps_io_emac1_inst_TX_CLK,

hps_io_hps_io_emac1_inst_TXD0,

hps_io_hps_io_emac1_inst_TXD1,

hps_io_hps_io_emac1_inst_TXD2,

hps_io_hps_io_emac1_inst_TXD3,

hps_io_hps_io_emac1_inst_RXD0,

hps_io_hps_io_emac1_inst_MDIO,

hps_io_hps_io_emac1_inst_MDC,

hps_io_hps_io_emac1_inst_RX_CTL,

hps_io_hps_io_emac1_inst_TX_CTL,

hps_io_hps_io_emac1_inst_RX_CLK,

hps_io_hps_io_emac1_inst_RXD1,

hps_io_hps_io_emac1_inst_RXD2,

hps_io_hps_io_emac1_inst_RXD3,

hps_io_hps_io_qspi_inst_IO0,

hps_io_hps_io_qspi_inst_IO1,

hps_io_hps_io_qspi_inst_IO2,

hps_io_hps_io_qspi_inst_IO3,

hps_io_hps_io_qspi_inst_SS0,

hps_io_hps_io_qspi_inst_CLK,

hps_io_hps_io_sdio_inst_CMD,

hps_io_hps_io_sdio_inst_D0,

hps_io_hps_io_sdio_inst_D1,

hps_io_hps_io_sdio_inst_CLK,

hps_io_hps_io_sdio_inst_D2,

hps_io_hps_io_sdio_inst_D3,

hps_io_hps_io_usb1_inst_D0,

hps_io_hps_io_usb1_inst_D1,

hps_io_hps_io_usb1_inst_D2,

hps_io_hps_io_usb1_inst_D3,

hps_io_hps_io_usb1_inst_D4,

hps_io_hps_io_usb1_inst_D5,

hps_io_hps_io_usb1_inst_D6,

hps_io_hps_io_usb1_inst_D7,

hps_io_hps_io_usb1_inst_CLK,

hps_io_hps_io_usb1_inst_STP,

hps_io_hps_io_usb1_inst_DIR,

hps_io_hps_io_usb1_inst_NXT,

hps_io_hps_io_spim0_inst_CLK,

hps_io_hps_io_spim0_inst_MOSI,

hps_io_hps_io_spim0_inst_MISO,

hps_io_hps_io_spim0_inst_SS0,

hps_io_hps_io_spim1_inst_CLK,

hps_io_hps_io_spim1_inst_MOSI,

hps_io_hps_io_spim1_inst_MISO,

hps_io_hps_io_spim1_inst_SS0,

hps_io_hps_io_uart0_inst_RX,

hps_io_hps_io_uart0_inst_TX,

hps_io_hps_io_i2c1_inst_SDA,

hps_io_hps_io_i2c1_inst_SCL,

hps_io_hps_io_gpio_inst_GPIO00,

//////////HMSC//////////////////////

HSMC_CLKIN_p,

HSMC_XAUI_RX_p0,

HSMC_XAUI_TX_p0,

MDC2,

MDIO2,

MDC1,

MDIO1,

PRTAD02,

PRTAD01,

PRTAD4,

PRTAD3,

PRTAD2,

PRTAD1,

TXONOFF1,

TXONOFF2,

OPINLVL,

OPOUTLVL,

PHYRESET,

USER_LED_G,

USER_LED_R,

CONFIG1_1,

CONFIG0_1,

CONFIG1_2,

CONFIG0_2,

SS338_CLKIN,

GPIO0_1,

GPIO1_1,

GPIO0_2,

GPIO1_2,

SER_BOOT,

SMBSPDSEL1,

SMBSPDSEL2,

SMBWEN,

NVMA1SEL,

NVMPROT,

OPRXLOS2,

OPTXFLT2,

SFP_TXDIS2,

SFP_TXRS20,

LASI2);

```
//=======================================================
//  PORT declarations
//=======================================================

///////// AUD /////////
input                       AUD_ADCDAT;
inout                       AUD_ADCLRCK;
inout                       AUD_BCLK;
output                      AUD_DACDAT;
inout                       AUD_DACLRCK;
output                      AUD_I2C_SCLK;
inout                       AUD_I2C_SDAT;
output                      AUD_MUTE;
output                      AUD_XCK;

`ifdef ENABLE_DDR3
///////// DDR3 /////////
output [14:0]                       DDR3_A;
output [2:0]                        DDR3_BA;
output                      DDR3_CAS_n;
output                      DDR3_CKE;
output                      DDR3_CK_n;
output                      DDR3_CK_p;
output                      DDR3_CS_n;
output [3:0]                        DDR3_DM;
inout [31:0]                        DDR3_DQ;
inout [3:0]                             DDR3_DQS_n;
inout [3:0]                             DDR3_DQS_p;
output                      DDR3_ODT;
output                      DDR3_RAS_n;
output                      DDR3_RESET_n;
input                       DDR3_RZQ;
```

```
  output              DDR3_WE_n;
`endif /*ENABLE_DDR3*/

  ///////// FAN /////////
  output              FAN_CTRL;

`ifdef ENABLE_HPS
  ///////// HPS /////////
  input               HPS_CLOCK_25;
  input               HPS_CLOCK_50;
  input               HPS_CONV_USB_n;
  output [14:0]                 HPS_DDR3_A;
  output [2:0]                  HPS_DDR3_BA;
  output              HPS_DDR3_CAS_n;
  output              HPS_DDR3_CKE;
  output              HPS_DDR3_CK_n;
  output              HPS_DDR3_CK_p;
  output              HPS_DDR3_CS_n;
  output [3:0]                  HPS_DDR3_DM;
  inout [31:0]                  HPS_DDR3_DQ;
  inout [3:0]                       HPS_DDR3_DQS_n;
  inout [3:0]                       HPS_DDR3_DQS_p;
  output              HPS_DDR3_ODT;
  output              HPS_DDR3_RAS_n;
  output              HPS_DDR3_RESET_n;
  input               HPS_DDR3_RZQ;
  output              HPS_DDR3_WE_n;
  input               HPS_ENET_GTX_CLK;
  input               HPS_ENET_INT_n;
  output              HPS_ENET_MDC;
  inout               HPS_ENET_MDIO;
  output              HPS_ENET_RESET_n;
```

```
input                HPS_ENET_RX_CLK;
input [3:0]                          HPS_ENET_RX_DATA;
input                HPS_ENET_RX_DV;
output [3:0]                 HPS_ENET_TX_DATA;
output               HPS_ENET_TX_EN;
inout [3:0]                          HPS_FLASH_DATA;
output               HPS_FLASH_DCLK;
output               HPS_FLASH_NCSO;
input                HPS_GSENSOR_INT;
inout                HPS_I2C_CLK;
inout                HPS_I2C_SDA;
inout [3:0]                          HPS_KEY;
output               HPS_LCM_D_C;
output               HPS_LCM_RST_N;
input                HPS_LCM_SPIM_CLK;
inout                HPS_LCM_SPIM_MISO;
output               HPS_LCM_SPIM_MOSI;
output               HPS_LCM_SPIM_SS;
output [3:0]                 HPS_LED;
inout                HPS_LTC_GPIO;
input                HPS_RESET_n;
output               HPS_SD_CLK;
inout                HPS_SD_CMD;
inout [3:0]                          HPS_SD_DATA;
output               HPS_SPIM_CLK;
input                HPS_SPIM_MISO;
output               HPS_SPIM_MOSI;
output               HPS_SPIM_SS;
input [3:0]                          HPS_SW;
input                HPS_UART_RX;
output               HPS_UART_TX;
input                HPS_USB_CLKOUT;
```

```
  inout [7:0]                                   HPS_USB_DATA;
  input                         HPS_USB_DIR;
  input                         HPS_USB_NXT;
  output                         HPS_USB_RESET_PHY;
  output                         HPS_USB_STP;
  input                         HPS_WARM_RST_n;
`endif /*ENABLE_HPS*/

  ///////// IRDA /////////
  input                         IRDA_RXD;

  ///////// KEY /////////
  input [3:0]                                   KEY;

  ///////// LED /////////
  output [3:0]                              LED;

  ///////// OSC /////////
  input                         OSC_50_B3B;
  input                         OSC_50_B4A;
  input                         OSC_50_B5B;
  input                         OSC_50_B8A;

  ///////// PCIE /////////
  input                         PCIE_PERST_n;
  input                         PCIE_WAKE_n;

  ///////// RESET /////////
  input                         RESET_n;

  ///////// SI5338 /////////
  inout                         SI5338_SCL;
```

```
inout                       SI5338_SDA;

//////// SW ////////
input [3:0]                             SW;

//////// TEMP ////////
output                      TEMP_CS_n;
output                      TEMP_DIN;
input                       TEMP_DOUT;
output                      TEMP_SCLK;

//////// USB ////////
input                       USB_B2_CLK;
inout [7:0]
        USB_B2_DATA;
output                      USB_EMPTY;
output                      USB_FULL;
input                       USB_OE_n;
input                       USB_RD_n;
input                       USB_RESET_n;
inout                       USB_SCL;
inout                       USB_SDA;
input                       USB_WR_n;

//////// VGA ////////
output [7:0]                                                    VGA_B;
output                      VGA_BLANK_n;
output                      VGA_CLK;
output [7:0]                                                    VGA_G;
output                      VGA_HS;
output [7:0]                                                    VGA_R;
output                      VGA_SYNC_n;
```

```
output                          VGA_VS;


/////////hps pin///////
output wire [14:0]                      memory_mem_a;
output wire [2:0]                       memory_mem_ba;
output wire                             memory_mem_ck;
output wire                             memory_mem_ck_n;
output wire                             memory_mem_cke;
output wire                             memory_mem_cs_n;
output wire                             memory_mem_ras_n;
output wire                             memory_mem_cas_n;
output wire                             memory_mem_we_n;
output wire                             memory_mem_reset_n;
inout  wire [31:0]                      memory_mem_dq;
inout  wire [3:0]                       memory_mem_dqs;
inout  wire [3:0]                       memory_mem_dqs_n;
output wire                             memory_mem_odt;
output wire [3:0]                       memory_mem_dm;
input  wire                             memory_oct_rzqin;
output wire                             hps_io_hps_io_emac1_inst_TX_CLK;
output wire                             hps_io_hps_io_emac1_inst_TXD0;
output wire                             hps_io_hps_io_emac1_inst_TXD1;
output wire                             hps_io_hps_io_emac1_inst_TXD2;
output wire                             hps_io_hps_io_emac1_inst_TXD3;
input  wire                             hps_io_hps_io_emac1_inst_RXD0;
inout  wire                             hps_io_hps_io_emac1_inst_MDIO;
output wire                             hps_io_hps_io_emac1_inst_MDC;
input  wire                             hps_io_hps_io_emac1_inst_RX_CTL;
output wire                             hps_io_hps_io_emac1_inst_TX_CTL;
input  wire                             hps_io_hps_io_emac1_inst_RX_CLK;
input  wire                             hps_io_hps_io_emac1_inst_RXD1;
input  wire                             hps_io_hps_io_emac1_inst_RXD2;
```

```
input  wire        hps_io_hps_io_emac1_inst_RXD3;
inout  wire        hps_io_hps_io_qspi_inst_IO0;
inout  wire        hps_io_hps_io_qspi_inst_IO1;
inout  wire        hps_io_hps_io_qspi_inst_IO2;
inout  wire        hps_io_hps_io_qspi_inst_IO3;
output wire        hps_io_hps_io_qspi_inst_SS0;
output wire        hps_io_hps_io_qspi_inst_CLK;
inout  wire        hps_io_hps_io_sdio_inst_CMD;
inout  wire        hps_io_hps_io_sdio_inst_D0;
inout  wire        hps_io_hps_io_sdio_inst_D1;
output wire        hps_io_hps_io_sdio_inst_CLK;
inout  wire        hps_io_hps_io_sdio_inst_D2;
inout  wire        hps_io_hps_io_sdio_inst_D3;
inout  wire        hps_io_hps_io_usb1_inst_D0;
inout  wire        hps_io_hps_io_usb1_inst_D1;
inout  wire        hps_io_hps_io_usb1_inst_D2;
inout  wire        hps_io_hps_io_usb1_inst_D3;
inout  wire        hps_io_hps_io_usb1_inst_D4;
inout  wire        hps_io_hps_io_usb1_inst_D5;
inout  wire        hps_io_hps_io_usb1_inst_D6;
inout  wire        hps_io_hps_io_usb1_inst_D7;
input  wire        hps_io_hps_io_usb1_inst_CLK;
output wire        hps_io_hps_io_usb1_inst_STP;
input  wire        hps_io_hps_io_usb1_inst_DIR;
input  wire        hps_io_hps_io_usb1_inst_NXT;
output wire        hps_io_hps_io_spim0_inst_CLK;
output wire        hps_io_hps_io_spim0_inst_MOSI;
input  wire        hps_io_hps_io_spim0_inst_MISO;
output wire        hps_io_hps_io_spim0_inst_SS0;
output wire        hps_io_hps_io_spim1_inst_CLK;
output wire        hps_io_hps_io_spim1_inst_MOSI;
input  wire        hps_io_hps_io_spim1_inst_MISO;
```

```
output wire                                 hps_io_hps_io_spim1_inst_SS0;
input  wire                                 hps_io_hps_io_uart0_inst_RX;
output wire                                 hps_io_hps_io_uart0_inst_TX;
inout  wire                                 hps_io_hps_io_i2c1_inst_SDA;
inout  wire                                 hps_io_hps_io_i2c1_inst_SCL;
inout  wire                                 hps_io_hps_io_gpio_inst_GPIO00;


        ////////HSMC//////////////////////
                    input HSMC_CLKIN_p;

  input    logic   [3:0]  HSMC_XAUI_RX_p0;
  output logic     [3:0]  HSMC_XAUI_TX_p0;



  output logic          MDC2;
                  inout    logic          MDIO2;
                  output logic          MDC1;
                  inout    logic          MDIO1;
                  output logic          PRTAD02;
                  output logic          PRTAD01;
                  output logic          PRTAD4;
                  output logic          PRTAD3;
                  output logic          PRTAD2;
                  output logic          PRTAD1;
                  output logic          TXONOFF1;
                  output logic          TXONOFF2;
                  output logic          OPINLVL;
                  output logic          OPOUTLVL;
                  output logic          PHYRESET;
```

```
                output logic        [7:0] USER_LED_G;
                output logic        [7:0] USER_LED_R;


                output logic          CONFIG1_1;
                output logic          CONFIG0_1;
                output logic          CONFIG1_2;
                output logic          CONFIG0_2;
                output logic          SS338_CLKIN;

                inout   logic         GPIO0_1;
                inout   logic         GPIO1_1;
                inout   logic         GPIO0_2;
                inout   logic         GPIO1_2;
                output logic          SER_BOOT;
                output logic          SMBSPDSEL1;
                output logic          SMBSPDSEL2;
                inout   logic         SMBWEN;

                inout   logic         NVMA1SEL;
                inout   logic         NVMPROT;


                input   logic         OPRXLOS2;
                input   logic         OPTXFLT2;
                input   logic         SFP_TXDIS2;
                input   logic                                 SFP_TXRS20;
                input   logic         LASI2;
//=========================================================
//  REG/WIRE declarations
```

```
//=======================================================

//      For Audio CODEC
wire                                AUD_CTRL_CLK;       //      For Audio Controller

reg [31:0]                          Cont;
wire                                VGA_CTRL_CLK;
wire [9:0]                          mVGA_R;
wire [9:0]                          mVGA_G;
wire [9:0]                          mVGA_B;
wire [19:0]                         mVGA_ADDR;
wire                                DLY_RST;

//      For VGA Controller
wire                                        mVGA_CLK;
wire [9:0]                          mRed;
wire [9:0]                          mGreen;
wire [9:0]                          mBlue;
wire                                        VGA_Read;       //      VGA data request

wire [9:0]                          recon_VGA_R;
wire [9:0]                          recon_VGA_G;
wire [9:0]                          recon_VGA_B;

//      For Down Sample
wire [3:0]                          Remain;
wire [9:0]                          Quotient;

wire                                        AUD_MUTE;


// Make the FPGA reset cause an HPS reset
```

```
reg [19:0]                                  hps_reset_counter = 20'h0;
reg                                         hps_fpga_reset_n = 0;

always @(posedge OSC_50_B4A) begin
  if (hps_reset_counter == 20'h ffffff) hps_fpga_reset_n <= 1;
  hps_reset_counter <= hps_reset_counter + 1;
end



        wire        MDIN1;
wire        MDOEN1;
wire        MDO1;
logic                       RESET_N; // S5 active low
wire                        clk_buff0;
wire                        clk_buff1;
wire                        clk_buff2;
wire [71:0] link;
wire                        link_clk;

        logic  [71:0]    xgmii_buff ;
        logic  [71:0]    xgmii_rx_dc;
        logic  [71:0]    xgmii_tx_dc;
        logic  [71:0]    xgmiialigned;
logic [31:0] crclink;
logic eop;
logic sop;
logic valid;
logic ready;
logic checksum_err;
logic  [2:0] empty;
logic  [63:0] data;
logic [1:0] state;
```

```
                    logic eoptx;
  logic soptx;
  logic readytx;
  logic  [2:0] emptytx;
  logic  [63:0] stdata;
          logic validtx;
          logic [31:0] crclinktx;
          logic [31:0] cctx;
          logic [63:0] xgmiirevtx;
  logic [71:0] xgmiitx;
  logic [71:0] tx_buff;

          logic [63:0] databuff;
          logic validbuff;
          logic readybuff;
          logic sopbuff;
          logic eopbuff;
          logic [2:0] emptybuff;

  parameter idle = 0, trig =1, lock =2;


//=========================================================
//  Structural coding
//=========================================================


 // +-----------------------------------------------------------------
 // +
   assign CONFIG0_1 = 1'b1;            // Configure BCM8727 from EEPROM
   assign CONFIG1_1 = 1'b0;
```

```
  assign CONFIG0_2 = 1'b1;
  assign CONFIG1_2 = 1'b0;
  assign SS338_CLKIN       = 1'b0;

// 11.02.2010 changes
// BCM8272C allow spi-rom to be removed
// 1 = Boot microcode from spi proms
  assign SER_BOOT  = 1'b0;
  assign SMBSPDSEL1 = 1'b0;
  assign SMBSPDSEL2 = 1'b0;
  assign SMBWEN    = 1'b1;
  assign GPIO0_1   = 1'b0;
  assign GPIO1_1   = 1'b0;
  assign GPIO0_2   = 1'b0;
  assign GPIO1_2   = 1'b0;

// 11.02.2010 changes done

// ____________________________________________________________________
// 1: EEPROM Slave Addr 52, 0: 50 addr:
// During deassertion of BCM8727 reset,
// latched into bit 10 of register 1.8002h
  assign NVMA1SEL = 1'b1;

// when high protect non volatile memory
  assign NVMPROT  = 1'b0;

// MDIO ports connection
`ifdef ENABLE_MDIO
  assign MDIO1 = !MDOEN1? MDO1 : 1'bz;
  assign MDIN1 = MDIO1;
`else
```

```
  assign MDC1     = 1'bz;
  assign MDIO1 = 1'bz;
`endif
  assign MDC2     = 1'bz;
  assign MDIO2 = 1'bz;

// +----------------------------------------------------------------
// +TXONOFF2

  assign PHYRESET =  KEY[1]; // S4 active low
 // assign STOPMON  = ~KEY[2]; // S3 active high

  assign {PRTAD4,PRTAD3,PRTAD2,PRTAD1,PRTAD01} = 5'b00000;
  assign  PRTAD02 = 1'b0;
// +
  assign TXONOFF1 = 1'b1;
  assign TXONOFF2 = 1'b1;
  assign OPOUTLVL = 1'b0;                                    // 0 for active low OPTXENB/OPTXRST
  assign OPINLVL  = 1'b1;                                  // 1 for active high OPRXLOS/TXONOFF

          assign USER_LED_R = 8'b00001111;
          assign USER_LED_G = 8'b11110000;

          assign LED[0] =SFP_TXRS20;
          assign LED[1] =SFP_TXDIS2;


always @(state)
                  begin
                                  case (state)
                                  idle:
                                                                  RESET_N <= 1;
```

```
                trig:
                        RESET_N <= 0;
                lock:
                        RESET_N <= 1;

                endcase
        end

always @(posedge OSC_50_B4A)
                begin
                        case(state)
                        idle:
                                if (KEY[0]==0)
                                        state <= trig;
                                else
                                        state <= idle;
                        trig:
                                if (KEY[0]==0)
                                        state <= lock;
                                else
                                        state <= idle;
                        lock:
                                if (KEY[0]==0)
                                        state <= lock;
                                else
                                        state <= idle;
                        endcase
                end

 pll p1(.refclk(OSC_50_B4A), .rst(0), .outclk_0(clk_buff0), .locked());
```

```
xaui test(
        .pll_ref_clk                    (clk_buff0),        //      pll_ref_clk.clk
        .xgmii_tx_clk           (link_clk),     //      xgmii_tx_clk.clk
        .xgmii_rx_clk           (link_clk),     //      xgmii_rx_clk.clk
        .xgmii_rx_dc                    (xgmii_rx_dc),      //      xgmii_rx_dc.data
        .xgmii_tx_dc                    (tx_buff),      //      xgmii_tx_dc.data
        .xaui_rx_serial_data                    (HSMC_XAUI_RX_p0),  // xaui_rx_serial_data.export
        .xaui_tx_serial_data                    (HSMC_XAUI_TX_p0),  // xaui_tx_serial_data.export
        .rx_ready                       (LED[2]),       //      rx_ready.export
        .tx_ready                       (LED[3]),       //      tx_ready.export
        .phy_mgmt_clk           (OSC_50_B4A),       //      phy_mgmt_clk.clk
        .phy_mgmt_clk_reset             (!RESET_N),   //  phy_mgmt_clk_reset.reset
        .phy_mgmt_address       (3'h000),    //      phy_mgmt.address
        .phy_mgmt_read (0),      //             .read
        .phy_mgmt_readdata(8'hFFFFFFFF),    //             .readdata
        .phy_mgmt_write (0),     //             .write
        .phy_mgmt_writedata (8'h00000000),   //             .writedata
        .phy_mgmt_waitrequest(0), //             .waitrequest
        .reconfig_from_xcvr (),   //  reconfig_from_xcvr.data
        .reconfig_to_xcvr  ()    //    reconfig_to_xcvr.data
);
```

```
xgmii swap (
.clk(link_clk),
.xgmiidata(xgmii_rx_dc),// input data
.reset(!RESET_N),
.flag(),
.xgmii(xgmiialigned)//output data
);
```

```
mac_rx rx(
                                        .clk(link_clk),
                                        .xgmiidata(xgmiialigned),// input data
                                        .reset(!RESET_N),
                                        .c(crclink),

                                            .data(data),//output data
                                            .empty(empty),
                                            .sop(sop),
                                            .eop(eop),
                                            .valid(valid),
                                            .ready(ready),
                                            .newcrc(crclink),
                                            .checksum_err(checksum_err)
);



 mac_tx tx (
                                                .clk(link_clk),
                                                .stdata(databuff),// input data
                                                .reset(!RESET_N),
                                                .c(crclinktx),
                                                .sopin(sopbuff),
                                                .eopin(eopbuff),
                                                .valid(validbuff),
                                                .ready(readybuff),
                                                .empty(emptybuff),

                                                .xgmiirev(xgmiirevtx),
                                                .xgmii(tx_buff),
```

```
                                                    .newcrc(crclinktx),
                                                    .cc(cctx)
);

  lab3 u0 (
        .clk_clk                       (OSC_50_B4A),                 //            clk.clk
        .reset_reset_n                 (hps_fpga_reset_n),           //          reset.reset_n
        .memory_mem_a                  (memory_mem_a),               //         memory.mem_a
        .memory_mem_ba                 (memory_mem_ba),              //           .mem_ba
        .memory_mem_ck                 (memory_mem_ck),              //           .mem_ck
        .memory_mem_ck_n               (memory_mem_ck_n),            //           .mem_ck_n
        .memory_mem_cke                (memory_mem_cke),             //           .mem_cke
        .memory_mem_cs_n               (memory_mem_cs_n),            //           .mem_cs_n
        .memory_mem_ras_n              (memory_mem_ras_n),           //           .mem_ras_n
        .memory_mem_cas_n              (memory_mem_cas_n),           //           .mem_cas_n
        .memory_mem_we_n               (memory_mem_we_n),            //           .mem_we_n
        .memory_mem_reset_n            (memory_mem_reset_n),         //           .mem_reset_n
        .memory_mem_dq                 (memory_mem_dq),              //           .mem_dq
        .memory_mem_dqs                (memory_mem_dqs),             //           .mem_dqs
        .memory_mem_dqs_n              (memory_mem_dqs_n),           //           .mem_dqs_n
        .memory_mem_odt                (memory_mem_odt),             //           .mem_odt
        .memory_mem_dm                 (memory_mem_dm),              //           .mem_dm
        .memory_oct_rzqin              (memory_oct_rzqin),           //           .oct_rzqin
        .hps_io_hps_io_emac1_inst_TX_CLK              (hps_io_hps_io_emac1_inst_TX_CLK), //
           .hps_0_hps_io.hps_io_emac1_inst_TX_CLK
        .hps_io_hps_io_emac1_inst_TXD0                (hps_io_hps_io_emac1_inst_TXD0),
//            .hps_io_emac1_inst_TXD0
```

```
        .hps_io_hps_io_emac1_inst_TXD1          (hps_io_hps_io_emac1_inst_TXD1),
//          .hps_io_emac1_inst_TXD1
        .hps_io_hps_io_emac1_inst_TXD2          (hps_io_hps_io_emac1_inst_TXD2),
//          .hps_io_emac1_inst_TXD2
        .hps_io_hps_io_emac1_inst_TXD3          (hps_io_hps_io_emac1_inst_TXD3),
//          .hps_io_emac1_inst_TXD3
        .hps_io_hps_io_emac1_inst_RXD0          (hps_io_hps_io_emac1_inst_RXD0),
//          .hps_io_emac1_inst_RXD0
        .hps_io_hps_io_emac1_inst_MDIO          (hps_io_hps_io_emac1_inst_MDIO),
//          .hps_io_emac1_inst_MDIO
        .hps_io_hps_io_emac1_inst_MDC           (hps_io_hps_io_emac1_inst_MDC),
//          .hps_io_emac1_inst_MDC
        .hps_io_hps_io_emac1_inst_RX_CTL        (hps_io_hps_io_emac1_inst_RX_CTL),
//          .hps_io_emac1_inst_RX_CTL
        .hps_io_hps_io_emac1_inst_TX_CTL        (hps_io_hps_io_emac1_inst_TX_CTL),
//          .hps_io_emac1_inst_TX_CTL
        .hps_io_hps_io_emac1_inst_RX_CLK        (hps_io_hps_io_emac1_inst_RX_CLK),
//          .hps_io_emac1_inst_RX_CLK
        .hps_io_hps_io_emac1_inst_RXD1          (hps_io_hps_io_emac1_inst_RXD1),
//          .hps_io_emac1_inst_RXD1
        .hps_io_hps_io_emac1_inst_RXD2          (hps_io_hps_io_emac1_inst_RXD2),
//          .hps_io_emac1_inst_RXD2
        .hps_io_hps_io_emac1_inst_RXD3          (hps_io_hps_io_emac1_inst_RXD3),
//          .hps_io_emac1_inst_RXD3
        .hps_io_hps_io_qspi_inst_IO0            (hps_io_hps_io_qspi_inst_IO0),
//          .hps_io_qspi_inst_IO0
        .hps_io_hps_io_qspi_inst_IO1            (hps_io_hps_io_qspi_inst_IO1),
//          .hps_io_qspi_inst_IO1
        .hps_io_hps_io_qspi_inst_IO2            (hps_io_hps_io_qspi_inst_IO2),
//          .hps_io_qspi_inst_IO2
        .hps_io_hps_io_qspi_inst_IO3            (hps_io_hps_io_qspi_inst_IO3),
//          .hps_io_qspi_inst_IO3
        .hps_io_hps_io_qspi_inst_SS0            (hps_io_hps_io_qspi_inst_SS0),
//          .hps_io_qspi_inst_SS0
        .hps_io_hps_io_qspi_inst_CLK            (hps_io_hps_io_qspi_inst_CLK),
//          .hps_io_qspi_inst_CLK
        .hps_io_hps_io_sdio_inst_CMD            (hps_io_hps_io_sdio_inst_CMD),
//          .hps_io_sdio_inst_CMD
        .hps_io_hps_io_sdio_inst_D0             (hps_io_hps_io_sdio_inst_D0),
//          .hps_io_sdio_inst_D0
```

```
        .hps_io_hps_io_sdio_inst_D1                 (hps_io_hps_io_sdio_inst_D1),
//          .hps_io_sdio_inst_D1
        .hps_io_hps_io_sdio_inst_CLK                (hps_io_hps_io_sdio_inst_CLK),
//          .hps_io_sdio_inst_CLK
        .hps_io_hps_io_sdio_inst_D2                 (hps_io_hps_io_sdio_inst_D2),
//          .hps_io_sdio_inst_D2
        .hps_io_hps_io_sdio_inst_D3                 (hps_io_hps_io_sdio_inst_D3),
//          .hps_io_sdio_inst_D3
        .hps_io_hps_io_usb1_inst_D0                 (hps_io_hps_io_usb1_inst_D0),
//          .hps_io_usb1_inst_D0
        .hps_io_hps_io_usb1_inst_D1                 (hps_io_hps_io_usb1_inst_D1),
//          .hps_io_usb1_inst_D1
        .hps_io_hps_io_usb1_inst_D2                 (hps_io_hps_io_usb1_inst_D2),
//          .hps_io_usb1_inst_D2
        .hps_io_hps_io_usb1_inst_D3                 (hps_io_hps_io_usb1_inst_D3),
//          .hps_io_usb1_inst_D3
        .hps_io_hps_io_usb1_inst_D4                 (hps_io_hps_io_usb1_inst_D4),
//          .hps_io_usb1_inst_D4
        .hps_io_hps_io_usb1_inst_D5                 (hps_io_hps_io_usb1_inst_D5),
//          .hps_io_usb1_inst_D5
        .hps_io_hps_io_usb1_inst_D6                 (hps_io_hps_io_usb1_inst_D6),
//          .hps_io_usb1_inst_D6
        .hps_io_hps_io_usb1_inst_D7                 (hps_io_hps_io_usb1_inst_D7),
//          .hps_io_usb1_inst_D7
        .hps_io_hps_io_usb1_inst_CLK                (hps_io_hps_io_usb1_inst_CLK),
//          .hps_io_usb1_inst_CLK
        .hps_io_hps_io_usb1_inst_STP                (hps_io_hps_io_usb1_inst_STP),
//          .hps_io_usb1_inst_STP
        .hps_io_hps_io_usb1_inst_DIR                (hps_io_hps_io_usb1_inst_DIR),
//          .hps_io_usb1_inst_DIR
        .hps_io_hps_io_usb1_inst_NXT                (hps_io_hps_io_usb1_inst_NXT),
//          .hps_io_usb1_inst_NXT
        .hps_io_hps_io_spim0_inst_CLK               (hps_io_hps_io_spim0_inst_CLK),
//          .hps_io_spim0_inst_CLK
        .hps_io_hps_io_spim0_inst_MOSI              (hps_io_hps_io_spim0_inst_MOSI),
//          .hps_io_spim0_inst_MOSI
        .hps_io_hps_io_spim0_inst_MISO              (hps_io_hps_io_spim0_inst_MISO),
//          .hps_io_spim0_inst_MISO
        .hps_io_hps_io_spim0_inst_SS0               (hps_io_hps_io_spim0_inst_SS0),
//          .hps_io_spim0_inst_SS0
```

```
        .hps_io_hps_io_spim1_inst_CLK               (hps_io_hps_io_spim1_inst_CLK),
//          .hps_io_spim1_inst_CLK
        .hps_io_hps_io_spim1_inst_MOSI              (hps_io_hps_io_spim1_inst_MOSI),
//          .hps_io_spim1_inst_MOSI
        .hps_io_hps_io_spim1_inst_MISO              (hps_io_hps_io_spim1_inst_MISO),
//          .hps_io_spim1_inst_MISO
        .hps_io_hps_io_spim1_inst_SS0               (hps_io_hps_io_spim1_inst_SS0),
//          .hps_io_spim1_inst_SS0
        .hps_io_hps_io_uart0_inst_RX                (hps_io_hps_io_uart0_inst_RX),
//          .hps_io_uart0_inst_RX
        .hps_io_hps_io_uart0_inst_TX                (hps_io_hps_io_uart0_inst_TX),
//          .hps_io_uart0_inst_TX
        .hps_io_hps_io_i2c1_inst_SDA                (hps_io_hps_io_i2c1_inst_SDA),
//          .hps_io_i2c1_inst_SDA
        .hps_io_hps_io_i2c1_inst_SCL                (hps_io_hps_io_i2c1_inst_SCL),
//          .hps_io_i2c1_inst_SCL
                                    .xaui_clk_clk               (link_clk),             //   xaui_clk.clk
                                    .xaui_reset_reset_n         (RESET_N),          //  xaui_reset.reset_n
                                    .stream_src_data            (stdata),           //  stream_src.data
                                    .stream_src_valid           (validtx),          //          .valid
                                    .stream_src_ready           (readytx),          //          .ready
                                    .stream_src_startofpacket   (soptx),    //          .startofpacket
                                    .stream_src_endofpacket     (eoptx),    //          .endofpacket
                                    .stream_src_empty           (emptytx),          //          .empty
                                    .stream_sink_data           (data),         // stream_sink.data
                                    .stream_sink_valid          (valid),            //          .valid
                                    .stream_sink_ready          (ready),            //          .ready
                                    .stream_sink_startofpacket  (sop),      //          .startofpacket
                                    .stream_sink_endofpacket    (eop),      //          .endofpacket
                                    .stream_sink_empty          (empty)
        );



txbuffer buffer(
```

```
.clk(link_clk),
.reset(!RESET_N),
.stdata(stdata),              //  stream_src.data
.valid(validtx),              //          .valid
.sopin(soptx),       //          .startofpacket
.eopin(eoptx),       //          .endofpacket
.emptyin(emptytx),            //          .empty

.databuff(databuff),
.validout(validbuff),
.readybuff(readybuff),
.sopout(sopbuff),
.eopout(eopbuff),
.emptybuff(emptybuff),
.readytx(readytx)             //          .ready
);

endmodule
```

Swap:

```
module xgmii(
 input logic clk,
        input logic [71:0] xgmiidata,// input data
        input logic reset,
        output logic flag,

        output logic [71:0] xgmii//output data
        );

        logic flag2;
        logic [35:0] buff;
        logic[7:0] xgmiicontrol;
```

```
assign xgmiicontrol = {xgmiidata[44], xgmiidata[53],xgmiidata[62], xgmiidata[71],xgmiidata[8], xgmiidata[17], xgmiidata[26], xgmiidata[35]};


initial begin
        flag<=0;
        flag2<=0;
        buff<=0;
end
always@(posedge clk) begin
        if(reset) begin
                flag<=0;
                buff<=0;
        end
        else begin
         if(flag==0 && xgmiicontrol !=8'h1F) begin
                xgmii<=xgmiidata;

         end
         if(xgmiicontrol == 8'h1F)begin
                xgmii[34:27]    <=      8'h07;
                xgmii[25:18]    <=      8'h07;
                xgmii[16:9]   <=   8'h07;
                xgmii[7:0]      <=   8'h07;
                xgmii[70:63]    <=      8'h07;
                xgmii[61:54]    <=   8'h07;
                xgmii[52:45]    <=      8'h07;
                xgmii[43:36]    <=      8'h07;
                {xgmii[44], xgmii[53],xgmii[62], xgmii[71],xgmii[8], xgmii[17], xgmii[26], xgmii[35]} <=8'hFF;
                buff<=xgmiidata[71:36];
                flag<=1;
                flag2<=1;
         end
```

```
                if(flag2)begin
                                        {xgmii[44], xgmii[53],xgmii[62], xgmii[71],xgmii[8], xgmii[17],
xgmii[26], xgmii[35]} <=8'h01;
                                        flag2<=0;
                end
                if(flag==1)begin
                        xgmii[35:0]<=buff;
                        xgmii[71:36]<=xgmiidata[35:0];
                        buff<=xgmiidata[71:36];
                end
                if(flag==1 && xgmiicontrol == 8'hFF)begin
                        flag<=0;
                end
               end
        end
endmodule
```

MAC RX:

```
module mac_rx(
                                        input logic clk,
                                        input logic [71:0] xgmiidata,// input data
                                        input logic reset,
                                        input logic [31:0]c,

                                        output logic [63:0] data,//output data
                                        output logic [2:0] empty,
                                        output logic sop,
                                        output logic eop,
                                        output logic valid,
                                        output logic ready,
```

```
                                output logic [63:0] xgmii,//temp
                                output logic [63:0] xgmiibuff,//temp
                                output logic [31:0] newcrc,
                                output logic [31:0] newcrc8,//temp
                                output logic [63:0]xgmiirev,
                                output logic [63:0]swapxgmiirev,//combo xgmiirev
                                output logic checksum_err,
                                output logic [31:0] cc);

logic [7:0] xgmiicontrol;
logic [7:0] controlbuff;
//logic [63:0] xgmii;// output buffer
logic controland, controlor;
logic valid_flag;
logic valid_flag_buff;
logic sop_flag;
logic sop_flag_buff;
logic sop_flag_buff2;
logic eop_flag=0;
logic eop_flag_buff =0;
logic crc_flag;
logic checksum_buff =0;
logic checksum_flag=0;
logic [2:0] empty_flag;
//logic [31:0] c;

//logic [63:0]xgmiirev;
logic [63:0]swapxgmiidata;
//logic [63:0]swapxgmiirev;
logic[7:0] xgmiirev8;
logic[15:0] xgmiirev16;
logic[23:0] xgmiirev24;
```

```
        logic[31:0] xgmiirev32;
        logic[39:0] xgmiirev40;
        logic[47:0] xgmiirev48;
        logic[55:0] xgmiirev56;

        assign ready = 1;

        assign xgmiicontrol = {xgmiidata[44], xgmiidata[53],xgmiidata[62], xgmiidata[71],xgmiidata[8],
xgmiidata[17], xgmiidata[26], xgmiidata[35]};

        assign xgmiirev8=swapxgmiirev[63:56];
        assign xgmiirev16=swapxgmiirev[63:48];
        assign xgmiirev24=swapxgmiirev[63:40];
        assign xgmiirev32=swapxgmiirev[63:32];

        assign xgmiirev40=xgmiirev[63:24];
        assign xgmiirev48=xgmiirev[63:16];
        assign xgmiirev56=xgmiirev[63:8];

        assign swapxgmiidata[7:0] = xgmiidata[43:36];
        assign swapxgmiidata[15:8] =  xgmiidata[52:45];
        assign swapxgmiidata[23:16] = xgmiidata[61:54];
        assign swapxgmiidata[31:24] = xgmiidata[70:63];
        assign swapxgmiidata[39:32] = xgmiidata[7:0];
        assign swapxgmiidata[47:40] = xgmiidata[16:9];
        assign swapxgmiidata[55:48] = xgmiidata[25:18];
        assign swapxgmiidata[63:56] = xgmiidata[34:27];

    integer i;
    integer g;
    integer k;
```

```
always@* begin
   for (i=0;i<8;i++)begin
    for(g=0;g<8;g++)begin
      xgmiirev[g+i*8]=swapxgmiidata[(7-g)+i*8];
                                swapxgmiirev[g+i*8]=swapxgmiidata[(7-g)+i*8];
    end
   end


   for (k=0;k<32;k++)begin
     cc[k] = ~newcrc[31-k];
   end
 end




                always @(posedge clk) begin

                                                if (reset == 1)begin
                                                                xgmii[63:0]<= 64'b0;
                                                                sop_flag <= 0;
                                                                sop_flag_buff <= 0;
                                                                eop <= 0;
                                                                sop <=0;
                                                                valid_flag <= 0;
                                                                crc_flag <=0;
                                                                valid <= 0;
                                                                newcrc<=32'hFFFFFFFF;
                                                                valid_flag_buff <= 0;
```

```
                checksum_err <=0;
                empty <=3'b0;
            end
            else begin
                empty <= empty_flag;
                eop_flag <= eop_flag_buff;
                ///detect sop///
                if(xgmiicontrol == 1) begin
                                sop_flag<=1;
                                crc_flag <= 1;

                end
                if(sop_flag) begin
                                sop_flag_buff<=
1;
                                sop_flag<=0;
                end
                if(sop_flag_buff) begin
                                sop_flag_buff<=
0;
        sop_flag_buff2<=1;
                end
                if(sop_flag_buff2)begin
                                sop <=1;
                                valid <= 1;
                                sop_flag_buff2
<=0;
                end
                else begin
                                sop <= 0;
                end
                ////////////////////
```

```
///when in middle of packet///
data[63:0] <= xgmiibuff[63:0];
xgmiibuff[63:0]<=xgmii[63:0];

if(crc_flag)begin
        newcrc[0]   <= xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[37] ^ xgmiirev[34] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[16] ^ xgmiirev[12] ^ xgmiirev[10] ^ xgmiirev[9] ^ xgmiirev[6] ^ xgmiirev[0] ^ c[0] ^ c[2] ^ c[5] ^ c[12] ^ c[13] ^ c[15] ^ c[16] ^ c[18] ^ c[21] ^ c[22] ^ c[23] ^ c[26] ^ c[28] ^ c[29] ^ c[31];
        newcrc[1]   <= xgmiirev[63] ^ xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[53] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[24] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[13] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[9] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[1] ^ c[2] ^ c[3] ^ c[5] ^ c[6] ^ c[12] ^ c[14] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[24] ^ c[26] ^ c[27] ^ c[28] ^ c[30] ^ c[31];
        newcrc[2]   <= xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[44] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[14] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[2] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[0] ^ c[3] ^ c[4] ^ c[5] ^ c[6] ^ c[7] ^ c[12] ^ c[19] ^ c[20] ^ c[21] ^ c[23] ^ c[25] ^ c[26] ^ c[27];
        newcrc[3]   <= xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[45] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[27] ^ xgmiirev[25] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[15] ^ xgmiirev[14] ^ xgmiirev[10] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[3] ^ xgmiirev[2] ^ xgmiirev[1] ^ c[0] ^ c[1] ^ c[4] ^ c[5] ^ c[6] ^ c[7] ^ c[8] ^ c[13] ^ c[20] ^ c[21] ^ c[22] ^ c[24] ^ c[26] ^ c[27] ^ c[28];
        newcrc[4]   <= xgmiirev[63] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[33] ^ xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[15] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[6] ^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[2] ^ xgmiirev[0] ^ c[1] ^ c[6] ^ c[7] ^ c[8] ^ c[9] ^ c[12] ^ c[13] ^ c[14] ^ c[15] ^ c[16] ^ c[18] ^ c[25] ^ c[26] ^ c[27] ^ c[31];
        newcrc[5]   <= xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[37] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[24] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[13] ^ xgmiirev[10] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[5] ^ c[7] ^ c[8] ^ c[9] ^ c[10] ^ c[12] ^ c[14] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[22] ^ c[23] ^ c[27] ^ c[29] ^ c[31];
```

```
newcrc[6]  <= xgmiirev[62] ^
xgmiirev[60] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^
xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[38] ^
xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[25] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[14] ^
xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[2] ^ xgmiirev[1] ^ c[6]
^ c[8] ^ c[9] ^ c[10] ^ c[11] ^ c[13] ^ c[15] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[23] ^ c[24] ^ c[28] ^ c[30];

newcrc[7]  <= xgmiirev[60] ^
xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[54] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^
xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[39] ^
xgmiirev[37] ^ xgmiirev[34] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[25] ^ xgmiirev[24] ^
xgmiirev[23] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[10] ^ xgmiirev[8] ^ xgmiirev[7]
^ xgmiirev[5] ^ xgmiirev[3] ^ xgmiirev[2] ^ xgmiirev[0] ^ c[0] ^ c[2] ^ c[5] ^ c[7] ^ c[9] ^ c[10] ^ c[11] ^ c[13] ^ c[14]
^ c[15] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[24] ^ c[25] ^ c[26] ^ c[28];

newcrc[8]  <= xgmiirev[63] ^
xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[54] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^
xgmiirev[46] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[40] ^ xgmiirev[38] ^ xgmiirev[37] ^
xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[23] ^
xgmiirev[22] ^ xgmiirev[17] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[10] ^ xgmiirev[8] ^ xgmiirev[4] ^ xgmiirev[3] ^
xgmiirev[1] ^ xgmiirev[0] ^ c[0] ^ c[1] ^ c[2] ^ c[3] ^ c[5] ^ c[6] ^ c[8] ^ c[10] ^ c[11] ^ c[13] ^ c[14] ^ c[18] ^ c[19] ^
c[20] ^ c[22] ^ c[25] ^ c[27] ^ c[28] ^ c[31];

newcrc[9]  <= xgmiirev[61] ^
xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[55] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[47] ^
xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[41] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[36] ^
xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[24] ^ xgmiirev[23] ^
xgmiirev[18] ^ xgmiirev[13] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[9] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[2] ^
xgmiirev[1] ^ c[0] ^ c[1] ^ c[2] ^ c[3] ^ c[4] ^ c[6] ^ c[7] ^ c[9] ^ c[11] ^ c[12] ^ c[14] ^ c[15] ^ c[19] ^ c[20] ^ c[21] ^
c[23] ^ c[26] ^ c[28] ^ c[29];

newcrc[10] <= xgmiirev[63] ^
xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[52] ^
xgmiirev[50] ^ xgmiirev[42] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[33] ^
xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[19] ^ xgmiirev[16] ^
xgmiirev[14] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[5] ^ xgmiirev[3] ^ xgmiirev[2] ^ xgmiirev[0] ^ c[0] ^ c[1] ^ c[3]
^ c[4] ^ c[7] ^ c[8] ^ c[10] ^ c[18] ^ c[20] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[28] ^ c[30] ^ c[31];

newcrc[11] <= xgmiirev[59] ^
xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[51] ^ xgmiirev[50] ^
xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[41] ^ xgmiirev[40] ^
xgmiirev[36] ^ xgmiirev[33] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[25] ^
xgmiirev[24] ^ xgmiirev[20] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[14] ^ xgmiirev[12] ^ xgmiirev[9]
^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[1] ^ c[4] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[13] ^ c[15] ^ c[16]
^ c[18] ^ c[19] ^ c[22] ^ c[23] ^ c[24] ^ c[25] ^ c[26] ^ c[27];

newcrc[12] <= xgmiirev[63] ^
xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^
xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[42] ^ xgmiirev[41] ^
xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[27] ^ xgmiirev[24] ^ xgmiirev[21] ^ xgmiirev[18] ^ xgmiirev[17] ^
xgmiirev[15] ^ xgmiirev[13] ^ xgmiirev[12] ^ xgmiirev[9] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[2] ^
xgmiirev[1] ^ xgmiirev[0] ^ c[9] ^ c[10] ^ c[14] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^ c[20] ^ c[21] ^ c[22] ^ c[24] ^ c[25] ^
c[27] ^ c[29] ^ c[31];
```

newcrc[13] <= xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[25] ^ xgmiirev[22] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[16] ^ xgmiirev[14] ^ xgmiirev[13] ^ xgmiirev[10] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[3] ^ xgmiirev[2] ^ xgmiirev[1] ^ c[0] ^ c[10] ^ c[11] ^ c[15] ^ c[16] ^ c[18] ^ c[19] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[25] ^ c[26] ^ c[28] ^ c[30];

newcrc[14] <= xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[26] ^ xgmiirev[23] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[17] ^ xgmiirev[15] ^ xgmiirev[14] ^ xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[2] ^ c[0] ^ c[1] ^ c[11] ^ c[12] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[29] ^ c[31];

newcrc[15] <= xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[30] ^ xgmiirev[27] ^ xgmiirev[24] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[18] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[12] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[3] ^ c[1] ^ c[2] ^ c[12] ^ c[13] ^ c[17] ^ c[18] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[24] ^ c[25] ^ c[27] ^ c[28] ^ c[30];

newcrc[16] <= xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[51] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[37] ^ xgmiirev[35] ^ xgmiirev[32] ^ xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[19] ^ xgmiirev[17] ^ xgmiirev[13] ^ xgmiirev[12] ^ xgmiirev[8] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[0] ^ c[0] ^ c[3] ^ c[5] ^ c[12] ^ c[14] ^ c[15] ^ c[16] ^ c[19] ^ c[24] ^ c[25];

newcrc[17] <= xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[52] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[38] ^ xgmiirev[36] ^ xgmiirev[33] ^ xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[27] ^ xgmiirev[25] ^ xgmiirev[23] ^ xgmiirev[22] ^ xgmiirev[20] ^ xgmiirev[18] ^ xgmiirev[14] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[1] ^ c[1] ^ c[4] ^ c[6] ^ c[13] ^ c[15] ^ c[16] ^ c[17] ^ c[20] ^ c[25] ^ c[26];

newcrc[18] <= xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[53] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[46] ^ xgmiirev[39] ^ xgmiirev[37] ^ xgmiirev[34] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[21] ^ xgmiirev[19] ^ xgmiirev[15] ^ xgmiirev[14] ^ xgmiirev[10] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[2] ^ c[0] ^ c[2] ^ c[5] ^ c[7] ^ c[14] ^ c[16] ^ c[17] ^ c[18] ^ c[21] ^ c[26] ^ c[27];

newcrc[19] <= xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[54] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[47] ^ xgmiirev[40] ^ xgmiirev[38] ^ xgmiirev[35] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[22] ^ xgmiirev[20] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[3] ^ c[0] ^ c[1] ^ c[3] ^ c[6] ^ c[8] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^ c[22] ^ c[27] ^ c[28];

newcrc[20] <= xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[55] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[41] ^ xgmiirev[39] ^ xgmiirev[36] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[30] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[25] ^ xgmiirev[23] ^ xgmiirev[21] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[12] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[4] ^ c[1] ^ c[2] ^ c[4] ^ c[7] ^ c[9] ^ c[16] ^ c[18] ^ c[19] ^ c[20] ^ c[23] ^ c[28] ^ c[29];

newcrc[21] <= xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[56] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[49] ^ xgmiirev[42] ^

xgmiirev[40] ^ xgmiirev[37] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[22] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[13] ^ xgmiirev[10] ^ xgmiirev[9] ^ xgmiirev[5] ^ c[2] ^ c[3] ^ c[5] ^ c[8] ^ c[10] ^ c[17] ^ c[19] ^ c[20] ^ c[21] ^ c[24] ^ c[29] ^ c[30];

newcrc[22] <= xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[52] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[41] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[16] ^ xgmiirev[14] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[9] ^ xgmiirev[0] ^ c[2] ^ c[3] ^ c[4] ^ c[5] ^ c[6] ^ c[9] ^ c[11] ^ c[12] ^ c[13] ^ c[15] ^ c[16] ^ c[20] ^ c[23] ^ c[25] ^ c[26] ^ c[28] ^ c[29] ^ c[30];

newcrc[23] <= xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[42] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[6] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[2] ^ c[3] ^ c[4] ^ c[6] ^ c[7] ^ c[10] ^ c[14] ^ c[15] ^ c[17] ^ c[18] ^ c[22] ^ c[23] ^ c[24] ^ c[27] ^ c[28] ^ c[30];

newcrc[24] <= xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[43] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[32] ^ xgmiirev[30] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[14] ^ xgmiirev[10] ^ xgmiirev[7] ^ xgmiirev[2] ^ xgmiirev[1] ^ c[0] ^ c[3] ^ c[4] ^ c[5] ^ c[7] ^ c[8] ^ c[11] ^ c[15] ^ c[16] ^ c[18] ^ c[19] ^ c[23] ^ c[24] ^ c[25] ^ c[28] ^ c[29] ^ c[31];

newcrc[25] <= xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[44] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[33] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[15] ^ xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[3] ^ xgmiirev[2] ^ c[1] ^ c[4] ^ c[5] ^ c[6] ^ c[8] ^ c[9] ^ c[12] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[24] ^ c[25] ^ c[26] ^ c[29] ^ c[30];

newcrc[26] <= xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[52] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[44] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[22] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[10] ^ xgmiirev[6] ^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[0] ^ c[6] ^ c[7] ^ c[9] ^ c[10] ^ c[12] ^ c[15] ^ c[16] ^ c[17] ^ c[20] ^ c[22] ^ c[23] ^ c[25] ^ c[27] ^ c[28] ^ c[29] ^ c[30];

newcrc[27] <= xgmiirev[63] ^ xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[53] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[11] ^ xgmiirev[7] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[1] ^ c[0] ^ c[7] ^ c[8] ^ c[10] ^ c[11] ^ c[13] ^ c[16] ^ c[17] ^ c[18] ^ c[21] ^ c[23] ^ c[24] ^ c[26] ^ c[28] ^ c[29] ^ c[30] ^ c[31];

newcrc[28] <= xgmiirev[63] ^ xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[54] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[33] ^ xgmiirev[30] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[25] ^ xgmiirev[24] ^

```
xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[12] ^ xgmiirev[8] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[2] ^
c[1] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[14] ^ c[17] ^ c[18] ^ c[19] ^ c[22] ^ c[24] ^ c[25] ^ c[27] ^ c[29] ^ c[30] ^ c[31];

                                                    newcrc[29] <= xgmiirev[63] ^
xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[52] ^ xgmiirev[51] ^
xgmiirev[50] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[34] ^
xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[25] ^ xgmiirev[23] ^
xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[3] ^ c[2] ^ c[9] ^
c[10] ^ c[12] ^ c[13] ^ c[15] ^ c[18] ^ c[19] ^ c[20] ^ c[23] ^ c[25] ^ c[26] ^ c[28] ^ c[30] ^ c[31];

                                                    newcrc[30] <= xgmiirev[63] ^
xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^
xgmiirev[48] ^ xgmiirev[46] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[35] ^ xgmiirev[32] ^
xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[23] ^
xgmiirev[22] ^ xgmiirev[14] ^ xgmiirev[10] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[4] ^ c[0] ^ c[3] ^ c[10] ^ c[11] ^
c[13] ^ c[14] ^ c[16] ^ c[19] ^ c[20] ^ c[21] ^ c[24] ^ c[26] ^ c[27] ^ c[29] ^ c[31];

                                                      newcrc[31] <= xgmiirev[62] ^ xgmiirev[60]
^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[49] ^ xgmiirev[47] ^
xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[36] ^ xgmiirev[33] ^ xgmiirev[31] ^ xgmiirev[30] ^
xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[15] ^
xgmiirev[11] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[5] ^ c[1] ^ c[4] ^ c[11] ^ c[12] ^ c[14] ^ c[15] ^ c[17] ^ c[20] ^
c[21] ^ c[22] ^ c[25] ^ c[27] ^ c[28] ^ c[30];

                                                    end



                                                    ///
                                                    if(xgmiicontrol==8'h00)begin
                                                            controlbuff [7:0]<=
xgmiicontrol[7:0];

                                                            xgmii[7:0] <= xgmiidata[43:36];
                                                            xgmii[15:8] <=  xgmiidata[52:45];
                                                            xgmii[23:16] <= xgmiidata[61:54];
                                                            xgmii[31:24] <= xgmiidata[70:63];
                                                            xgmii[39:32] <= xgmiidata[7:0];
                                                            xgmii[47:40] <= xgmiidata[16:9];
                                                            xgmii[55:48] <= xgmiidata[25:18];
                                                            xgmii[63:56] <= xgmiidata[34:27];
                                                            xgmiibuff[63:0]<=xgmii[63:0];
                                                    end
```

```
                                                    ///end of packet detect through terminating
FD
                                                    //Buff data is eop
                                                    else if (xgmiidata[34:27]==8'hFD) begin
                                                                xgmii[63:0] <= 0;
                                                                eop_flag <= 1;
                                                                empty<=4;
                                                                empty_flag<=4;
                                                                checksum_flag
<=1;
                                                                newcrc8<=c;
                                                                crc_flag<=0;
                                                    end
                                                    else if (xgmiidata[25:18]==8'hFD) begin
        xgmii[63:56]<=xgmiidata[34:27];
                                                                xgmii[55:0]<=0;
                                                                empty<=3;
                                                                empty_flag<=3;
                                                                eop_flag <= 1;
                                                                newcrc8[0]
        <= xgmiirev8[6] ^ xgmiirev8[0] ^ c[24] ^ c[30];
                                                                newcrc8[1]
        <= xgmiirev8[7] ^ xgmiirev8[6] ^ xgmiirev8[1] ^ xgmiirev8[0] ^ c[24] ^ c[25] ^ c[30] ^ c[31];
                                                                newcrc8[2]
        <= xgmiirev8[7] ^ xgmiirev8[6] ^ xgmiirev8[2] ^ xgmiirev8[1] ^ xgmiirev8[0] ^ c[24] ^ c[25] ^ c[26] ^ c[30] ^
c[31];
                                                                newcrc8[3]
        <= xgmiirev8[7] ^ xgmiirev8[3] ^ xgmiirev8[2] ^ xgmiirev8[1] ^ c[25] ^ c[26] ^ c[27] ^ c[31];
                                                                newcrc8[4]
        <= xgmiirev8[6] ^ xgmiirev8[4] ^ xgmiirev8[3] ^ xgmiirev8[2] ^ xgmiirev8[0] ^ c[24] ^ c[26] ^ c[27] ^ c[28] ^
c[30];
                                                                newcrc8[5]
        <= xgmiirev8[7] ^ xgmiirev8[6] ^ xgmiirev8[5] ^ xgmiirev8[4] ^ xgmiirev8[3] ^ xgmiirev8[1] ^ xgmiirev8[0] ^
c[24] ^ c[25] ^ c[27] ^ c[28] ^ c[29] ^ c[30] ^ c[31];
                                                                newcrc8[6]
        <= xgmiirev8[7] ^ xgmiirev8[6] ^ xgmiirev8[5] ^ xgmiirev8[4] ^ xgmiirev8[2] ^ xgmiirev8[1] ^ c[25] ^ c[26] ^
c[28] ^ c[29] ^ c[30] ^ c[31];
```

```
                                                                    newcrc8[7]
        <= xgmiirev8[7] ^ xgmiirev8[5] ^ xgmiirev8[3] ^ xgmiirev8[2] ^ xgmiirev8[0] ^ c[24] ^ c[26] ^ c[27] ^ c[29] ^
c[31];

                                                                    newcrc8[8]
        <= xgmiirev8[4] ^ xgmiirev8[3] ^ xgmiirev8[1] ^ xgmiirev8[0] ^ c[0] ^ c[24] ^ c[25] ^ c[27] ^ c[28];

                                                                    newcrc8[9]
        <= xgmiirev8[5] ^ xgmiirev8[4] ^ xgmiirev8[2] ^ xgmiirev8[1] ^ c[1] ^ c[25] ^ c[26] ^ c[28] ^ c[29];

                                                                    newcrc8[10] <=
xgmiirev8[5] ^ xgmiirev8[3] ^ xgmiirev8[2] ^ xgmiirev8[0] ^ c[2] ^ c[24] ^ c[26] ^ c[27] ^ c[29];

                                                                    newcrc8[11] <=
xgmiirev8[4] ^ xgmiirev8[3] ^ xgmiirev8[1] ^ xgmiirev8[0] ^ c[3] ^ c[24] ^ c[25] ^ c[27] ^ c[28];

                                                                    newcrc8[12] <=
xgmiirev8[6] ^ xgmiirev8[5] ^ xgmiirev8[4] ^ xgmiirev8[2] ^ xgmiirev8[1] ^ xgmiirev8[0] ^ c[4] ^ c[24] ^ c[25] ^ c[26]
^ c[28] ^ c[29] ^ c[30];

                                                                    newcrc8[13] <=
xgmiirev8[7] ^ xgmiirev8[6] ^ xgmiirev8[5] ^ xgmiirev8[3] ^ xgmiirev8[2] ^ xgmiirev8[1] ^ c[5] ^ c[25] ^ c[26] ^ c[27]
^ c[29] ^ c[30] ^ c[31];

                                                                    newcrc8[14] <=
xgmiirev8[7] ^ xgmiirev8[6] ^ xgmiirev8[4] ^ xgmiirev8[3] ^ xgmiirev8[2] ^ c[6] ^ c[26] ^ c[27] ^ c[28] ^ c[30] ^ c[31];

                                                                    newcrc8[15] <=
xgmiirev8[7] ^ xgmiirev8[5] ^ xgmiirev8[4] ^ xgmiirev8[3] ^ c[7] ^ c[27] ^ c[28] ^ c[29] ^ c[31];

                                                                    newcrc8[16] <=
xgmiirev8[5] ^ xgmiirev8[4] ^ xgmiirev8[0] ^ c[8] ^ c[24] ^ c[28] ^ c[29];

                                                                    newcrc8[17] <=
xgmiirev8[6] ^ xgmiirev8[5] ^ xgmiirev8[1] ^ c[9] ^ c[25] ^ c[29] ^ c[30];

                                                                    newcrc8[18] <=
xgmiirev8[7] ^ xgmiirev8[6] ^ xgmiirev8[2] ^ c[10] ^ c[26] ^ c[30] ^ c[31];

                                                                    newcrc8[19] <=
xgmiirev8[7] ^ xgmiirev8[3] ^ c[11] ^ c[27] ^ c[31];

                                                                    newcrc8[20] <=
xgmiirev8[4] ^ c[12] ^ c[28];

                                                                    newcrc8[21] <=
xgmiirev8[5] ^ c[13] ^ c[29];

                                                                    newcrc8[22] <=
xgmiirev8[0] ^ c[14] ^ c[24];

                                                                    newcrc8[23] <=
xgmiirev8[6] ^ xgmiirev8[1] ^ xgmiirev8[0] ^ c[15] ^ c[24] ^ c[25] ^ c[30];

                                                                    newcrc8[24] <=
xgmiirev8[7] ^ xgmiirev8[2] ^ xgmiirev8[1] ^ c[16] ^ c[25] ^ c[26] ^ c[31];

                                                                    newcrc8[25] <=
xgmiirev8[3] ^ xgmiirev8[2] ^ c[17] ^ c[26] ^ c[27];
```

```
                                                                    newcrc8[26] <=
xgmiirev8[6] ^ xgmiirev8[4] ^ xgmiirev8[3] ^ xgmiirev8[0] ^ c[18] ^ c[24] ^ c[27] ^ c[28] ^ c[30];
                                                                    newcrc8[27] <=
xgmiirev8[7] ^ xgmiirev8[5] ^ xgmiirev8[4] ^ xgmiirev8[1] ^ c[19] ^ c[25] ^ c[28] ^ c[29] ^ c[31];
                                                                    newcrc8[28] <=
xgmiirev8[6] ^ xgmiirev8[5] ^ xgmiirev8[2] ^ c[20] ^ c[26] ^ c[29] ^ c[30];
                                                                    newcrc8[29] <=
xgmiirev8[7] ^ xgmiirev8[6] ^ xgmiirev8[3] ^ c[21] ^ c[27] ^ c[30] ^ c[31];
                                                                    newcrc8[30] <=
xgmiirev8[7] ^ xgmiirev8[4] ^ c[22] ^ c[28] ^ c[31];
                                                                    newcrc8[31] <=
xgmiirev8[5] ^ c[23] ^ c[29];

        checksum_flag<=1;
                                                                    crc_flag<=0;
                                                            end
                                                            else if (xgmiidata[16:9]==8'hFD) begin
                                                                    xgmii[55:48] <=
xgmiidata[25:18];
                                                                    xgmii[63:56] <=
xgmiidata[34:27];
                                                                    xgmii[47:0]<=0;
                                                                    empty<=2;
                                                                    empty_flag <= 2;
                                                                    eop_flag <= 1;
                                                                    newcrc8[0]  <=
xgmiirev16[12] ^ xgmiirev16[10] ^ xgmiirev16[9] ^ xgmiirev16[6] ^ xgmiirev16[0] ^ c[16] ^ c[22] ^ c[25] ^ c[26] ^
c[28];
                                                                    newcrc8[1]  <=
xgmiirev16[13] ^ xgmiirev16[12] ^ xgmiirev16[11] ^ xgmiirev16[9] ^ xgmiirev16[7] ^ xgmiirev16[6] ^ xgmiirev16[1]
^ xgmiirev16[0] ^ c[16] ^ c[17] ^ c[22] ^ c[23] ^ c[25] ^ c[27] ^ c[28] ^ c[29];
                                                                    newcrc8[2]  <=
xgmiirev16[14] ^ xgmiirev16[13] ^ xgmiirev16[9] ^ xgmiirev16[8] ^ xgmiirev16[7] ^ xgmiirev16[6] ^ xgmiirev16[2] ^
xgmiirev16[1] ^ xgmiirev16[0] ^ c[16] ^ c[17] ^ c[18] ^ c[22] ^ c[23] ^ c[24] ^ c[25] ^ c[29] ^ c[30];
                                                                    newcrc8[3]  <=
xgmiirev16[15] ^ xgmiirev16[14] ^ xgmiirev16[10] ^ xgmiirev16[9] ^ xgmiirev16[8] ^ xgmiirev16[7] ^ xgmiirev16[3]
^ xgmiirev16[2] ^ xgmiirev16[1] ^ c[17] ^ c[18] ^ c[19] ^ c[23] ^ c[24] ^ c[25] ^ c[26] ^ c[30] ^ c[31];
```

newcrc8[4] <= xgmiirev16[15] ^ xgmiirev16[12] ^ xgmiirev16[11] ^ xgmiirev16[8] ^ xgmiirev16[6] ^ xgmiirev16[4] ^ xgmiirev16[3] ^ xgmiirev16[2] ^ xgmiirev16[0] ^ c[16] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[24] ^ c[27] ^ c[28] ^ c[31];

newcrc8[5] <= xgmiirev16[13] ^ xgmiirev16[10] ^ xgmiirev16[7] ^ xgmiirev16[6] ^ xgmiirev16[5] ^ xgmiirev16[4] ^ xgmiirev16[3] ^ xgmiirev16[1] ^ xgmiirev16[0] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[26] ^ c[29];

newcrc8[6] <= xgmiirev16[14] ^ xgmiirev16[11] ^ xgmiirev16[8] ^ xgmiirev16[7] ^ xgmiirev16[6] ^ xgmiirev16[5] ^ xgmiirev16[4] ^ xgmiirev16[2] ^ xgmiirev16[1] ^ c[17] ^ c[18] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[24] ^ c[27] ^ c[30];

newcrc8[7] <= xgmiirev16[15] ^ xgmiirev16[10] ^ xgmiirev16[8] ^ xgmiirev16[7] ^ xgmiirev16[5] ^ xgmiirev16[3] ^ xgmiirev16[2] ^ xgmiirev16[0] ^ c[16] ^ c[18] ^ c[19] ^ c[21] ^ c[23] ^ c[24] ^ c[26] ^ c[31];

newcrc8[8] <= xgmiirev16[12] ^ xgmiirev16[11] ^ xgmiirev16[10] ^ xgmiirev16[8] ^ xgmiirev16[4] ^ xgmiirev16[3] ^ xgmiirev16[1] ^ xgmiirev16[0] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[24] ^ c[26] ^ c[27] ^ c[28];

newcrc8[9] <= xgmiirev16[13] ^ xgmiirev16[12] ^ xgmiirev16[11] ^ xgmiirev16[9] ^ xgmiirev16[5] ^ xgmiirev16[4] ^ xgmiirev16[2] ^ xgmiirev16[1] ^ c[17] ^ c[18] ^ c[20] ^ c[21] ^ c[25] ^ c[27] ^ c[28] ^ c[29];

newcrc8[10] <= xgmiirev16[14] ^ xgmiirev16[13] ^ xgmiirev16[9] ^ xgmiirev16[5] ^ xgmiirev16[3] ^ xgmiirev16[2] ^ xgmiirev16[0] ^ c[16] ^ c[18] ^ c[19] ^ c[21] ^ c[25] ^ c[29] ^ c[30];

newcrc8[11] <= xgmiirev16[15] ^ xgmiirev16[14] ^ xgmiirev16[12] ^ xgmiirev16[9] ^ xgmiirev16[4] ^ xgmiirev16[3] ^ xgmiirev16[1] ^ xgmiirev16[0] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[25] ^ c[28] ^ c[30] ^ c[31];

newcrc8[12] <= xgmiirev16[15] ^ xgmiirev16[13] ^ xgmiirev16[12] ^ xgmiirev16[9] ^ xgmiirev16[6] ^ xgmiirev16[5] ^ xgmiirev16[4] ^ xgmiirev16[2] ^ xgmiirev16[1] ^ xgmiirev16[0] ^ c[16] ^ c[17] ^ c[18] ^ c[20] ^ c[21] ^ c[22] ^ c[25] ^ c[28] ^ c[29] ^ c[31];

newcrc8[13] <= xgmiirev16[14] ^ xgmiirev16[13] ^ xgmiirev16[10] ^ xgmiirev16[7] ^ xgmiirev16[6] ^ xgmiirev16[5] ^ xgmiirev16[3] ^ xgmiirev16[2] ^ xgmiirev16[1] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[22] ^ c[23] ^ c[26] ^ c[29] ^ c[30];

newcrc8[14] <= xgmiirev16[15] ^ xgmiirev16[14] ^ xgmiirev16[11] ^ xgmiirev16[8] ^ xgmiirev16[7] ^ xgmiirev16[6] ^ xgmiirev16[4] ^ xgmiirev16[3] ^ xgmiirev16[2] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[23] ^ c[24] ^ c[27] ^ c[30] ^ c[31];

newcrc8[15] <= xgmiirev16[15] ^ xgmiirev16[12] ^ xgmiirev16[9] ^ xgmiirev16[8] ^ xgmiirev16[7] ^ xgmiirev16[5] ^ xgmiirev16[4] ^ xgmiirev16[3] ^ c[19] ^ c[20] ^ c[21] ^ c[23] ^ c[24] ^ c[25] ^ c[28] ^ c[31];

newcrc8[16] <= xgmiirev16[13] ^ xgmiirev16[12] ^ xgmiirev16[8] ^ xgmiirev16[5] ^ xgmiirev16[4] ^ xgmiirev16[0] ^ c[0] ^ c[16] ^ c[20] ^ c[21] ^ c[24] ^ c[28] ^ c[29];

newcrc8[17] <= xgmiirev16[14] ^ xgmiirev16[13] ^ xgmiirev16[9] ^ xgmiirev16[6] ^ xgmiirev16[5] ^ xgmiirev16[1] ^ c[1] ^ c[17] ^ c[21] ^ c[22] ^ c[25] ^ c[29] ^ c[30];

newcrc8[18] <= xgmiirev16[15] ^ xgmiirev16[14] ^ xgmiirev16[10] ^ xgmiirev16[7] ^ xgmiirev16[6] ^ xgmiirev16[2] ^ c[2] ^ c[18] ^ c[22] ^ c[23] ^ c[26] ^ c[30] ^ c[31];

newcrc8[19] <= xgmiirev16[15] ^ xgmiirev16[11] ^ xgmiirev16[8] ^ xgmiirev16[7] ^ xgmiirev16[3] ^ c[3] ^ c[19] ^ c[23] ^ c[24] ^ c[27] ^ c[31];

newcrc8[20] <= xgmiirev16[12] ^ xgmiirev16[9] ^ xgmiirev16[8] ^ xgmiirev16[4] ^ c[4] ^ c[20] ^ c[24] ^ c[25] ^ c[28];

newcrc8[21] <= xgmiirev16[13] ^ xgmiirev16[10] ^ xgmiirev16[9] ^ xgmiirev16[5] ^ c[5] ^ c[21] ^ c[25] ^ c[26] ^ c[29];

newcrc8[22] <= xgmiirev16[14] ^ xgmiirev16[12] ^ xgmiirev16[11] ^ xgmiirev16[9] ^ xgmiirev16[0] ^ c[6] ^ c[16] ^ c[25] ^ c[27] ^ c[28] ^ c[30];

newcrc8[23] <= xgmiirev16[15] ^ xgmiirev16[13] ^ xgmiirev16[9] ^ xgmiirev16[6] ^ xgmiirev16[1] ^ xgmiirev16[0] ^ c[7] ^ c[16] ^ c[17] ^ c[22] ^ c[25] ^ c[29] ^ c[31];

newcrc8[24] <= xgmiirev16[14] ^ xgmiirev16[10] ^ xgmiirev16[7] ^ xgmiirev16[2] ^ xgmiirev16[1] ^ c[8] ^ c[17] ^ c[18] ^ c[23] ^ c[26] ^ c[30];

newcrc8[25] <= xgmiirev16[15] ^ xgmiirev16[11] ^ xgmiirev16[8] ^ xgmiirev16[3] ^ xgmiirev16[2] ^ c[9] ^ c[18] ^ c[19] ^ c[24] ^ c[27] ^ c[31];

newcrc8[26] <= xgmiirev16[10] ^ xgmiirev16[6] ^ xgmiirev16[4] ^ xgmiirev16[3] ^ xgmiirev16[0] ^ c[10] ^ c[16] ^ c[19] ^ c[20] ^ c[22] ^ c[26];

newcrc8[27] <= xgmiirev16[11] ^ xgmiirev16[7] ^ xgmiirev16[5] ^ xgmiirev16[4] ^ xgmiirev16[1] ^ c[11] ^ c[17] ^ c[20] ^ c[21] ^ c[23] ^ c[27];

newcrc8[28] <= xgmiirev16[12] ^ xgmiirev16[8] ^ xgmiirev16[6] ^ xgmiirev16[5] ^ xgmiirev16[2] ^ c[12] ^ c[18] ^ c[21] ^ c[22] ^ c[24] ^ c[28];

newcrc8[29] <= xgmiirev16[13] ^ xgmiirev16[9] ^ xgmiirev16[7] ^ xgmiirev16[6] ^ xgmiirev16[3] ^ c[13] ^ c[19] ^ c[22] ^ c[23] ^ c[25] ^ c[29];

newcrc8[30] <= xgmiirev16[14] ^ xgmiirev16[10] ^ xgmiirev16[8] ^ xgmiirev16[7] ^ xgmiirev16[4] ^ c[14] ^ c[20] ^ c[23] ^ c[24] ^ c[26] ^ c[30];

newcrc8[31] <= xgmiirev16[15] ^ xgmiirev16[11] ^ xgmiirev16[9] ^ xgmiirev16[8] ^ xgmiirev16[5] ^ c[15] ^ c[21] ^ c[24] ^ c[25] ^ c[27] ^ c[31];

checksum_flag<=1;

crc_flag<=0;

```
                end
                else if (xgmiidata[7:0]==8'hFD) begin
                    xgmii[47:40] <= xgmiidata[16:9];
                    xgmii[55:48] <= xgmiidata[25:18];
                    xgmii[63:56] <= xgmiidata[34:27];
                    xgmii[39:0]<=0;
                    empty<=1;
                    empty_flag<=1;
                    eop_flag <= 1;
                    newcrc8[0] <= xgmiirev24[16] ^ xgmiirev24[12] ^ xgmiirev24[10] ^ xgmiirev24[9] ^ xgmiirev24[6] ^ xgmiirev24[0] ^ c[8] ^ c[14] ^ c[17] ^ c[18] ^ c[20] ^ c[24];
                    newcrc8[1] <= xgmiirev24[17] ^ xgmiirev24[16] ^ xgmiirev24[13] ^ xgmiirev24[12] ^ xgmiirev24[11] ^ xgmiirev24[9] ^ xgmiirev24[7] ^ xgmiirev24[6] ^ xgmiirev24[1] ^ xgmiirev24[0] ^ c[8] ^ c[9] ^ c[14] ^ c[15] ^ c[17] ^ c[19] ^ c[20] ^ c[21] ^ c[24] ^ c[25];
                    newcrc8[2] <= xgmiirev24[18] ^ xgmiirev24[17] ^ xgmiirev24[16] ^ xgmiirev24[14] ^ xgmiirev24[13] ^ xgmiirev24[9] ^ xgmiirev24[8] ^ xgmiirev24[7] ^ xgmiirev24[6] ^ xgmiirev24[2] ^ xgmiirev24[1] ^ xgmiirev24[0] ^ c[8] ^ c[9] ^ c[10] ^ c[14] ^ c[15] ^ c[16] ^ c[17] ^ c[21] ^ c[22] ^ c[24] ^ c[25] ^ c[26];
                    newcrc8[3] <= xgmiirev24[19] ^ xgmiirev24[18] ^ xgmiirev24[17] ^ xgmiirev24[15] ^ xgmiirev24[14] ^ xgmiirev24[10] ^ xgmiirev24[9] ^ xgmiirev24[8] ^ xgmiirev24[7] ^ xgmiirev24[3] ^ xgmiirev24[2] ^ xgmiirev24[1] ^ c[9] ^ c[10] ^ c[11] ^ c[15] ^ c[16] ^ c[17] ^ c[18] ^ c[22] ^ c[23] ^ c[25] ^ c[26] ^ c[27];
                    newcrc8[4] <= xgmiirev24[20] ^ xgmiirev24[19] ^ xgmiirev24[18] ^ xgmiirev24[15] ^ xgmiirev24[12] ^ xgmiirev24[11] ^ xgmiirev24[8] ^ xgmiirev24[6] ^ xgmiirev24[4] ^ xgmiirev24[3] ^ xgmiirev24[2] ^ xgmiirev24[0] ^ c[8] ^ c[10] ^ c[11] ^ c[12] ^ c[14] ^ c[16] ^ c[19] ^ c[20] ^ c[23] ^ c[26] ^ c[27] ^ c[28];
                    newcrc8[5] <= xgmiirev24[21] ^ xgmiirev24[20] ^ xgmiirev24[19] ^ xgmiirev24[13] ^ xgmiirev24[10] ^ xgmiirev24[7] ^ xgmiirev24[6] ^ xgmiirev24[5] ^ xgmiirev24[4] ^ xgmiirev24[3] ^ xgmiirev24[1] ^ xgmiirev24[0] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[13] ^ c[14] ^ c[15] ^ c[18] ^ c[21] ^ c[27] ^ c[28] ^ c[29];
                    newcrc8[6] <= xgmiirev24[22] ^ xgmiirev24[21] ^ xgmiirev24[20] ^ xgmiirev24[14] ^ xgmiirev24[11] ^ xgmiirev24[8] ^ xgmiirev24[7] ^ xgmiirev24[6] ^ xgmiirev24[5] ^ xgmiirev24[4] ^ xgmiirev24[2] ^ xgmiirev24[1] ^ c[9] ^ c[10] ^ c[12] ^ c[13] ^ c[14] ^ c[15] ^ c[16] ^ c[19] ^ c[22] ^ c[28] ^ c[29] ^ c[30];
                    newcrc8[7] <= xgmiirev24[23] ^ xgmiirev24[22] ^ xgmiirev24[21] ^ xgmiirev24[16] ^ xgmiirev24[15] ^ xgmiirev24[10] ^
```

xgmiirev24[8] ^ xgmiirev24[7] ^ xgmiirev24[5] ^ xgmiirev24[3] ^ xgmiirev24[2] ^ xgmiirev24[0] ^ c[8] ^ c[10] ^ c[11] ^ c[13] ^ c[15] ^ c[16] ^ c[18] ^ c[23] ^ c[24] ^ c[29] ^ c[30] ^ c[31];

newcrc8[8] <= xgmiirev24[23] ^ xgmiirev24[22] ^ xgmiirev24[17] ^ xgmiirev24[12] ^ xgmiirev24[11] ^ xgmiirev24[10] ^ xgmiirev24[8] ^ xgmiirev24[4] ^ xgmiirev24[3] ^ xgmiirev24[1] ^ xgmiirev24[0] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[16] ^ c[18] ^ c[19] ^ c[20] ^ c[25] ^ c[30] ^ c[31];

newcrc8[9] <= xgmiirev24[23] ^ xgmiirev24[18] ^ xgmiirev24[13] ^ xgmiirev24[12] ^ xgmiirev24[11] ^ xgmiirev24[9] ^ xgmiirev24[5] ^ xgmiirev24[4] ^ xgmiirev24[2] ^ xgmiirev24[1] ^ c[9] ^ c[10] ^ c[12] ^ c[13] ^ c[17] ^ c[19] ^ c[20] ^ c[21] ^ c[26] ^ c[31];

newcrc8[10] <= xgmiirev24[19] ^ xgmiirev24[16] ^ xgmiirev24[14] ^ xgmiirev24[13] ^ xgmiirev24[9] ^ xgmiirev24[5] ^ xgmiirev24[3] ^ xgmiirev24[2] ^ xgmiirev24[0] ^ c[8] ^ c[10] ^ c[11] ^ c[13] ^ c[17] ^ c[21] ^ c[22] ^ c[24] ^ c[27];

newcrc8[11] <= xgmiirev24[20] ^ xgmiirev24[17] ^ xgmiirev24[16] ^ xgmiirev24[15] ^ xgmiirev24[14] ^ xgmiirev24[12] ^ xgmiirev24[9] ^ xgmiirev24[4] ^ xgmiirev24[3] ^ xgmiirev24[1] ^ xgmiirev24[0] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[17] ^ c[20] ^ c[22] ^ c[23] ^ c[24] ^ c[25] ^ c[28];

newcrc8[12] <= xgmiirev24[21] ^ xgmiirev24[18] ^ xgmiirev24[17] ^ xgmiirev24[15] ^ xgmiirev24[13] ^ xgmiirev24[12] ^ xgmiirev24[9] ^ xgmiirev24[6] ^ xgmiirev24[5] ^ xgmiirev24[4] ^ xgmiirev24[2] ^ xgmiirev24[1] ^ xgmiirev24[0] ^ c[8] ^ c[9] ^ c[10] ^ c[12] ^ c[13] ^ c[14] ^ c[17] ^ c[20] ^ c[21] ^ c[23] ^ c[25] ^ c[26] ^ c[29];

newcrc8[13] <= xgmiirev24[22] ^ xgmiirev24[19] ^ xgmiirev24[18] ^ xgmiirev24[16] ^ xgmiirev24[14] ^ xgmiirev24[13] ^ xgmiirev24[10] ^ xgmiirev24[7] ^ xgmiirev24[6] ^ xgmiirev24[5] ^ xgmiirev24[3] ^ xgmiirev24[2] ^ xgmiirev24[1] ^ c[9] ^ c[10] ^ c[11] ^ c[13] ^ c[14] ^ c[15] ^ c[18] ^ c[21] ^ c[22] ^ c[24] ^ c[26] ^ c[27] ^ c[30];

newcrc8[14] <= xgmiirev24[23] ^ xgmiirev24[20] ^ xgmiirev24[19] ^ xgmiirev24[17] ^ xgmiirev24[15] ^ xgmiirev24[14] ^ xgmiirev24[11] ^ xgmiirev24[8] ^ xgmiirev24[7] ^ xgmiirev24[6] ^ xgmiirev24[4] ^ xgmiirev24[3] ^ xgmiirev24[2] ^ c[10] ^ c[11] ^ c[12] ^ c[14] ^ c[15] ^ c[16] ^ c[19] ^ c[22] ^ c[23] ^ c[25] ^ c[27] ^ c[28] ^ c[31];

newcrc8[15] <= xgmiirev24[21] ^ xgmiirev24[20] ^ xgmiirev24[18] ^ xgmiirev24[16] ^ xgmiirev24[15] ^ xgmiirev24[12] ^ xgmiirev24[9] ^ xgmiirev24[8] ^ xgmiirev24[7] ^ xgmiirev24[5] ^ xgmiirev24[4] ^ xgmiirev24[3] ^ c[11] ^ c[12] ^ c[13] ^ c[15] ^ c[16] ^ c[17] ^ c[20] ^ c[23] ^ c[24] ^ c[26] ^ c[28] ^ c[29];

newcrc8[16] <= xgmiirev24[22] ^ xgmiirev24[21] ^ xgmiirev24[19] ^ xgmiirev24[17] ^ xgmiirev24[13] ^ xgmiirev24[12] ^ xgmiirev24[8] ^ xgmiirev24[5] ^ xgmiirev24[4] ^ xgmiirev24[0] ^ c[8] ^ c[12] ^ c[13] ^ c[16] ^ c[20] ^ c[21] ^ c[25] ^ c[27] ^ c[29] ^ c[30];

newcrc8[17] <= xgmiirev24[23] ^ xgmiirev24[22] ^ xgmiirev24[20] ^ xgmiirev24[18] ^ xgmiirev24[14] ^ xgmiirev24[13] ^ xgmiirev24[9] ^ xgmiirev24[6] ^ xgmiirev24[5] ^ xgmiirev24[1] ^ c[9] ^ c[13] ^ c[14] ^ c[17] ^ c[21] ^ c[22] ^ c[26] ^ c[28] ^ c[30] ^ c[31];

newcrc8[18] <= xgmiirev24[23] ^ xgmiirev24[21] ^ xgmiirev24[19] ^ xgmiirev24[15] ^ xgmiirev24[14] ^ xgmiirev24[10] ^ xgmiirev24[7] ^ xgmiirev24[6] ^ xgmiirev24[2] ^ c[10] ^ c[14] ^ c[15] ^ c[18] ^ c[22] ^ c[23] ^ c[27] ^ c[29] ^ c[31];

newcrc8[19] <= xgmiirev24[22] ^ xgmiirev24[20] ^ xgmiirev24[16] ^ xgmiirev24[15] ^ xgmiirev24[11] ^ xgmiirev24[8] ^ xgmiirev24[7] ^ xgmiirev24[3] ^ c[11] ^ c[15] ^ c[16] ^ c[19] ^ c[23] ^ c[24] ^ c[28] ^ c[30];

newcrc8[20] <= xgmiirev24[23] ^ xgmiirev24[21] ^ xgmiirev24[17] ^ xgmiirev24[16] ^ xgmiirev24[12] ^ xgmiirev24[9] ^ xgmiirev24[8] ^ xgmiirev24[4] ^ c[12] ^ c[16] ^ c[17] ^ c[20] ^ c[24] ^ c[25] ^ c[29] ^ c[31];

newcrc8[21] <= xgmiirev24[22] ^ xgmiirev24[18] ^ xgmiirev24[17] ^ xgmiirev24[13] ^ xgmiirev24[10] ^ xgmiirev24[9] ^ xgmiirev24[5] ^ c[13] ^ c[17] ^ c[18] ^ c[21] ^ c[25] ^ c[26] ^ c[30];

newcrc8[22] <= xgmiirev24[23] ^ xgmiirev24[19] ^ xgmiirev24[18] ^ xgmiirev24[16] ^ xgmiirev24[14] ^ xgmiirev24[12] ^ xgmiirev24[11] ^ xgmiirev24[9] ^ xgmiirev24[0] ^ c[8] ^ c[17] ^ c[19] ^ c[20] ^ c[22] ^ c[24] ^ c[26] ^ c[27] ^ c[31];

newcrc8[23] <= xgmiirev24[20] ^ xgmiirev24[19] ^ xgmiirev24[17] ^ xgmiirev24[16] ^ xgmiirev24[15] ^ xgmiirev24[13] ^ xgmiirev24[9] ^ xgmiirev24[6] ^ xgmiirev24[1] ^ xgmiirev24[0] ^ c[8] ^ c[9] ^ c[14] ^ c[17] ^ c[21] ^ c[23] ^ c[24] ^ c[25] ^ c[27] ^ c[28];

newcrc8[24] <= xgmiirev24[21] ^ xgmiirev24[20] ^ xgmiirev24[18] ^ xgmiirev24[17] ^ xgmiirev24[16] ^ xgmiirev24[14] ^ xgmiirev24[10] ^ xgmiirev24[7] ^ xgmiirev24[2] ^ xgmiirev24[1] ^ c[0] ^ c[9] ^ c[10] ^ c[15] ^ c[18] ^ c[22] ^ c[24] ^ c[25] ^ c[26] ^ c[28] ^ c[29];

newcrc8[25] <= xgmiirev24[22] ^ xgmiirev24[21] ^ xgmiirev24[19] ^ xgmiirev24[18] ^ xgmiirev24[17] ^ xgmiirev24[15] ^ xgmiirev24[11] ^ xgmiirev24[8] ^ xgmiirev24[3] ^ xgmiirev24[2] ^ c[1] ^ c[10] ^ c[11] ^ c[16] ^ c[19] ^ c[23] ^ c[25] ^ c[26] ^ c[27] ^ c[29] ^ c[30];

newcrc8[26] <= xgmiirev24[23] ^ xgmiirev24[22] ^ xgmiirev24[20] ^ xgmiirev24[19] ^ xgmiirev24[18] ^ xgmiirev24[10] ^ xgmiirev24[6] ^ xgmiirev24[4] ^ xgmiirev24[3] ^ xgmiirev24[0] ^ c[2] ^ c[8] ^ c[11] ^ c[12] ^ c[14] ^ c[18] ^ c[26] ^ c[27] ^ c[28] ^ c[30] ^ c[31];

newcrc8[27] <= xgmiirev24[23] ^ xgmiirev24[21] ^ xgmiirev24[20] ^ xgmiirev24[19] ^ xgmiirev24[11] ^ xgmiirev24[7] ^ xgmiirev24[5] ^ xgmiirev24[4] ^ xgmiirev24[1] ^ c[3] ^ c[9] ^ c[12] ^ c[13] ^ c[15] ^ c[19] ^ c[27] ^ c[28] ^ c[29] ^ c[31];

newcrc8[28] <= xgmiirev24[22] ^ xgmiirev24[21] ^ xgmiirev24[20] ^ xgmiirev24[12] ^ xgmiirev24[8] ^ xgmiirev24[6] ^ xgmiirev24[5] ^ xgmiirev24[2] ^ c[4] ^ c[10] ^ c[13] ^ c[14] ^ c[16] ^ c[20] ^ c[28] ^ c[29] ^ c[30];

newcrc8[29] <= xgmiirev24[23] ^ xgmiirev24[22] ^ xgmiirev24[21] ^ xgmiirev24[13] ^ xgmiirev24[9] ^ xgmiirev24[7] ^ xgmiirev24[6] ^ xgmiirev24[3] ^ c[5] ^ c[11] ^ c[14] ^ c[15] ^ c[17] ^ c[21] ^ c[29] ^ c[30] ^ c[31];

newcrc8[30] <= xgmiirev24[23] ^ xgmiirev24[22] ^ xgmiirev24[14] ^ xgmiirev24[10] ^ xgmiirev24[8] ^ xgmiirev24[7] ^ xgmiirev24[4] ^ c[6] ^ c[12] ^ c[15] ^ c[16] ^ c[18] ^ c[22] ^ c[30] ^ c[31];

newcrc8[31] <= xgmiirev24[23] ^ xgmiirev24[15] ^ xgmiirev24[11] ^ xgmiirev24[9] ^ xgmiirev24[8] ^ xgmiirev24[5] ^ c[7] ^ c[13] ^ c[16] ^ c[17] ^ c[19] ^ c[23] ^ c[31];

```
checksum_flag<=1;
                                                                    crc_flag<=0;
                                                                end
                                                                else if (xgmiidata[70:63]==8'hFD) begin
                                                                    xgmii[39:32] <=
xgmiidata[7:0];
                                                                    xgmii[47:40] <=
xgmiidata[16:9];
                                                                    xgmii[55:48] <=
xgmiidata[25:18];
                                                                    xgmii[63:56] <=
xgmiidata[34:27];
                                                                    xgmii[32:0]<=0;
                                                                    empty<=0;
                                                                    empty_flag<=0;
                                                                    eop_flag <= 1;
                                                                    newcrc8[0]  <=
xgmiirev32[31] ^ xgmiirev32[30] ^ xgmiirev32[29] ^ xgmiirev32[28] ^ xgmiirev32[26] ^ xgmiirev32[25] ^
xgmiirev32[24] ^ xgmiirev32[16] ^ xgmiirev32[12] ^ xgmiirev32[10] ^ xgmiirev32[9] ^ xgmiirev32[6] ^ xgmiirev32[0]
^ c[0] ^ c[6] ^ c[9] ^ c[10] ^ c[12] ^ c[16] ^ c[24] ^ c[25] ^ c[26] ^ c[28] ^ c[29] ^ c[30] ^ c[31];
                                                                    newcrc8[1]  <=
xgmiirev32[28] ^ xgmiirev32[27] ^ xgmiirev32[24] ^ xgmiirev32[17] ^ xgmiirev32[16] ^ xgmiirev32[13] ^
xgmiirev32[12] ^ xgmiirev32[11] ^ xgmiirev32[9] ^ xgmiirev32[7] ^ xgmiirev32[6] ^ xgmiirev32[1] ^ xgmiirev32[0] ^
c[0] ^ c[1] ^ c[6] ^ c[7] ^ c[9] ^ c[11] ^ c[12] ^ c[13] ^ c[16] ^ c[17] ^ c[24] ^ c[27] ^ c[28];
                                                                    newcrc8[2]  <=
xgmiirev32[31] ^ xgmiirev32[30] ^ xgmiirev32[26] ^ xgmiirev32[24] ^ xgmiirev32[18] ^ xgmiirev32[17] ^
xgmiirev32[16] ^ xgmiirev32[14] ^ xgmiirev32[13] ^ xgmiirev32[9] ^ xgmiirev32[8] ^ xgmiirev32[7] ^ xgmiirev32[6]
^ xgmiirev32[2] ^ xgmiirev32[1] ^ xgmiirev32[0] ^ c[0] ^ c[1] ^ c[2] ^ c[6] ^ c[7] ^ c[8] ^ c[9] ^ c[13] ^ c[14] ^ c[16] ^
c[17] ^ c[18] ^ c[24] ^ c[26] ^ c[30] ^ c[31];
                                                                    newcrc8[3]  <=
xgmiirev32[31] ^ xgmiirev32[27] ^ xgmiirev32[25] ^ xgmiirev32[19] ^ xgmiirev32[18] ^ xgmiirev32[17] ^
xgmiirev32[15] ^ xgmiirev32[14] ^ xgmiirev32[10] ^ xgmiirev32[9] ^ xgmiirev32[8] ^ xgmiirev32[7] ^ xgmiirev32[3]
^ xgmiirev32[2] ^ xgmiirev32[1] ^ c[1] ^ c[2] ^ c[3] ^ c[7] ^ c[8] ^ c[9] ^ c[10] ^ c[14] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^
c[25] ^ c[27] ^ c[31];
                                                                    newcrc8[4]  <=
xgmiirev32[31] ^ xgmiirev32[30] ^ xgmiirev32[29] ^ xgmiirev32[25] ^ xgmiirev32[24] ^ xgmiirev32[20] ^
xgmiirev32[19] ^ xgmiirev32[18] ^ xgmiirev32[15] ^ xgmiirev32[12] ^ xgmiirev32[11] ^ xgmiirev32[8] ^
xgmiirev32[6] ^ xgmiirev32[4] ^ xgmiirev32[3] ^ xgmiirev32[2] ^ xgmiirev32[0] ^ c[0] ^ c[2] ^ c[3] ^ c[4] ^ c[6] ^ c[8]
^ c[11] ^ c[12] ^ c[15] ^ c[18] ^ c[19] ^ c[20] ^ c[24] ^ c[25] ^ c[29] ^ c[30] ^ c[31];
```

newcrc8[5] <=
xgmiirev32[29] ^ xgmiirev32[28] ^ xgmiirev32[24] ^ xgmiirev32[21] ^ xgmiirev32[20] ^ xgmiirev32[19] ^
xgmiirev32[13] ^ xgmiirev32[10] ^ xgmiirev32[7] ^ xgmiirev32[6] ^ xgmiirev32[5] ^ xgmiirev32[4] ^ xgmiirev32[3] ^
xgmiirev32[1] ^ xgmiirev32[0] ^ c[0] ^ c[1] ^ c[3] ^ c[4] ^ c[5] ^ c[6] ^ c[7] ^ c[10] ^ c[13] ^ c[19] ^ c[20] ^ c[21] ^
c[24] ^ c[28] ^ c[29];

newcrc8[6] <=
xgmiirev32[30] ^ xgmiirev32[29] ^ xgmiirev32[25] ^ xgmiirev32[22] ^ xgmiirev32[21] ^ xgmiirev32[20] ^
xgmiirev32[14] ^ xgmiirev32[11] ^ xgmiirev32[8] ^ xgmiirev32[7] ^ xgmiirev32[6] ^ xgmiirev32[5] ^ xgmiirev32[4] ^
xgmiirev32[2] ^ xgmiirev32[1] ^ c[1] ^ c[2] ^ c[4] ^ c[5] ^ c[6] ^ c[7] ^ c[8] ^ c[11] ^ c[14] ^ c[20] ^ c[21] ^ c[22] ^
c[25] ^ c[29] ^ c[30];

newcrc8[7] <=
xgmiirev32[29] ^ xgmiirev32[28] ^ xgmiirev32[25] ^ xgmiirev32[24] ^ xgmiirev32[23] ^ xgmiirev32[22] ^
xgmiirev32[21] ^ xgmiirev32[16] ^ xgmiirev32[15] ^ xgmiirev32[10] ^ xgmiirev32[8] ^ xgmiirev32[7] ^ xgmiirev32[5]
^ xgmiirev32[3] ^ xgmiirev32[2] ^ xgmiirev32[0] ^ c[0] ^ c[2] ^ c[3] ^ c[5] ^ c[7] ^ c[8] ^ c[10] ^ c[15] ^ c[16] ^ c[21]
^ c[22] ^ c[23] ^ c[24] ^ c[25] ^ c[28] ^ c[29];

newcrc8[8] <=
xgmiirev32[31] ^ xgmiirev32[28] ^ xgmiirev32[23] ^ xgmiirev32[22] ^ xgmiirev32[17] ^ xgmiirev32[12] ^
xgmiirev32[11] ^ xgmiirev32[10] ^ xgmiirev32[8] ^ xgmiirev32[4] ^ xgmiirev32[3] ^ xgmiirev32[1] ^ xgmiirev32[0] ^
c[0] ^ c[1] ^ c[3] ^ c[4] ^ c[8] ^ c[10] ^ c[11] ^ c[12] ^ c[17] ^ c[22] ^ c[23] ^ c[28] ^ c[31];

newcrc8[9] <=
xgmiirev32[29] ^ xgmiirev32[24] ^ xgmiirev32[23] ^ xgmiirev32[18] ^ xgmiirev32[13] ^ xgmiirev32[12] ^
xgmiirev32[11] ^ xgmiirev32[9] ^ xgmiirev32[5] ^ xgmiirev32[4] ^ xgmiirev32[2] ^ xgmiirev32[1] ^ c[1] ^ c[2] ^ c[4]
^ c[5] ^ c[9] ^ c[11] ^ c[12] ^ c[13] ^ c[18] ^ c[23] ^ c[24] ^ c[29];

newcrc8[10] <=
xgmiirev32[31] ^ xgmiirev32[29] ^ xgmiirev32[28] ^ xgmiirev32[26] ^ xgmiirev32[19] ^ xgmiirev32[16] ^
xgmiirev32[14] ^ xgmiirev32[13] ^ xgmiirev32[9] ^ xgmiirev32[5] ^ xgmiirev32[3] ^ xgmiirev32[2] ^ xgmiirev32[0] ^
c[0] ^ c[2] ^ c[3] ^ c[5] ^ c[9] ^ c[13] ^ c[14] ^ c[16] ^ c[19] ^ c[26] ^ c[28] ^ c[29] ^ c[31];

newcrc8[11] <=
xgmiirev32[31] ^ xgmiirev32[28] ^ xgmiirev32[27] ^ xgmiirev32[26] ^ xgmiirev32[25] ^ xgmiirev32[24] ^
xgmiirev32[20] ^ xgmiirev32[17] ^ xgmiirev32[16] ^ xgmiirev32[15] ^ xgmiirev32[14] ^ xgmiirev32[12] ^
xgmiirev32[9] ^ xgmiirev32[4] ^ xgmiirev32[3] ^ xgmiirev32[1] ^ xgmiirev32[0] ^ c[0] ^ c[1] ^ c[3] ^ c[4] ^ c[9] ^
c[12] ^ c[14] ^ c[15] ^ c[16] ^ c[17] ^ c[20] ^ c[24] ^ c[25] ^ c[26] ^ c[27] ^ c[28] ^ c[31];

newcrc8[12] <=
xgmiirev32[31] ^ xgmiirev32[30] ^ xgmiirev32[27] ^ xgmiirev32[24] ^ xgmiirev32[21] ^ xgmiirev32[18] ^
xgmiirev32[17] ^ xgmiirev32[15] ^ xgmiirev32[13] ^ xgmiirev32[12] ^ xgmiirev32[9] ^ xgmiirev32[6] ^ xgmiirev32[5]
^ xgmiirev32[4] ^ xgmiirev32[2] ^ xgmiirev32[1] ^ xgmiirev32[0] ^ c[0] ^ c[1] ^ c[2] ^ c[4] ^ c[5] ^ c[6] ^ c[9] ^ c[12]
^ c[13] ^ c[15] ^ c[17] ^ c[18] ^ c[21] ^ c[24] ^ c[27] ^ c[30] ^ c[31];

newcrc8[13] <=
xgmiirev32[31] ^ xgmiirev32[28] ^ xgmiirev32[25] ^ xgmiirev32[22] ^ xgmiirev32[19] ^ xgmiirev32[18] ^
xgmiirev32[16] ^ xgmiirev32[14] ^ xgmiirev32[13] ^ xgmiirev32[10] ^ xgmiirev32[7] ^ xgmiirev32[6] ^ xgmiirev32[5]
^ xgmiirev32[3] ^ xgmiirev32[2] ^ xgmiirev32[1] ^ c[1] ^ c[2] ^ c[3] ^ c[5] ^ c[6] ^ c[7] ^ c[10] ^ c[13] ^ c[14] ^ c[16]
^ c[18] ^ c[19] ^ c[22] ^ c[25] ^ c[28] ^ c[31];

newcrc8[14] <=
xgmiirev32[29] ^ xgmiirev32[26] ^ xgmiirev32[23] ^ xgmiirev32[20] ^ xgmiirev32[19] ^ xgmiirev32[17] ^
xgmiirev32[15] ^ xgmiirev32[14] ^ xgmiirev32[11] ^ xgmiirev32[8] ^ xgmiirev32[7] ^ xgmiirev32[6] ^ xgmiirev32[4]
^ xgmiirev32[3] ^ xgmiirev32[2] ^ c[2] ^ c[3] ^ c[4] ^ c[6] ^ c[7] ^ c[8] ^ c[11] ^ c[14] ^ c[15] ^ c[17] ^ c[19] ^ c[20] ^
c[23] ^ c[26] ^ c[29];

newcrc8[15] <= xgmiirev32[30] ^ xgmiirev32[27] ^ xgmiirev32[24] ^ xgmiirev32[21] ^ xgmiirev32[20] ^ xgmiirev32[18] ^ xgmiirev32[16] ^ xgmiirev32[15] ^ xgmiirev32[12] ^ xgmiirev32[9] ^ xgmiirev32[8] ^ xgmiirev32[7] ^ xgmiirev32[5] ^ xgmiirev32[4] ^ xgmiirev32[3] ^ c[3] ^ c[4] ^ c[5] ^ c[7] ^ c[8] ^ c[9] ^ c[12] ^ c[15] ^ c[16] ^ c[18] ^ c[20] ^ c[21] ^ c[24] ^ c[27] ^ c[30];

newcrc8[16] <= xgmiirev32[30] ^ xgmiirev32[29] ^ xgmiirev32[26] ^ xgmiirev32[24] ^ xgmiirev32[22] ^ xgmiirev32[21] ^ xgmiirev32[19] ^ xgmiirev32[17] ^ xgmiirev32[13] ^ xgmiirev32[12] ^ xgmiirev32[8] ^ xgmiirev32[5] ^ xgmiirev32[4] ^ xgmiirev32[0] ^ c[0] ^ c[4] ^ c[5] ^ c[8] ^ c[12] ^ c[13] ^ c[17] ^ c[19] ^ c[21] ^ c[22] ^ c[24] ^ c[26] ^ c[29] ^ c[30];

newcrc8[17] <= xgmiirev32[31] ^ xgmiirev32[30] ^ xgmiirev32[27] ^ xgmiirev32[25] ^ xgmiirev32[23] ^ xgmiirev32[22] ^ xgmiirev32[20] ^ xgmiirev32[18] ^ xgmiirev32[14] ^ xgmiirev32[13] ^ xgmiirev32[9] ^ xgmiirev32[6] ^ xgmiirev32[5] ^ xgmiirev32[1] ^ c[1] ^ c[5] ^ c[6] ^ c[9] ^ c[13] ^ c[14] ^ c[18] ^ c[20] ^ c[22] ^ c[23] ^ c[25] ^ c[27] ^ c[30] ^ c[31];

newcrc8[18] <= xgmiirev32[31] ^ xgmiirev32[28] ^ xgmiirev32[26] ^ xgmiirev32[24] ^ xgmiirev32[23] ^ xgmiirev32[21] ^ xgmiirev32[19] ^ xgmiirev32[15] ^ xgmiirev32[14] ^ xgmiirev32[10] ^ xgmiirev32[7] ^ xgmiirev32[6] ^ xgmiirev32[2] ^ c[2] ^ c[6] ^ c[7] ^ c[10] ^ c[14] ^ c[15] ^ c[19] ^ c[21] ^ c[23] ^ c[24] ^ c[26] ^ c[28] ^ c[31];

newcrc8[19] <= xgmiirev32[29] ^ xgmiirev32[27] ^ xgmiirev32[25] ^ xgmiirev32[24] ^ xgmiirev32[22] ^ xgmiirev32[20] ^ xgmiirev32[16] ^ xgmiirev32[15] ^ xgmiirev32[11] ^ xgmiirev32[8] ^ xgmiirev32[7] ^ xgmiirev32[3] ^ c[3] ^ c[7] ^ c[8] ^ c[11] ^ c[15] ^ c[16] ^ c[20] ^ c[22] ^ c[24] ^ c[25] ^ c[27] ^ c[29];

newcrc8[20] <= xgmiirev32[30] ^ xgmiirev32[28] ^ xgmiirev32[26] ^ xgmiirev32[25] ^ xgmiirev32[23] ^ xgmiirev32[21] ^ xgmiirev32[17] ^ xgmiirev32[16] ^ xgmiirev32[12] ^ xgmiirev32[9] ^ xgmiirev32[8] ^ xgmiirev32[4] ^ c[4] ^ c[8] ^ c[9] ^ c[12] ^ c[16] ^ c[17] ^ c[21] ^ c[23] ^ c[25] ^ c[26] ^ c[28] ^ c[30];

newcrc8[21] <= xgmiirev32[31] ^ xgmiirev32[29] ^ xgmiirev32[27] ^ xgmiirev32[26] ^ xgmiirev32[24] ^ xgmiirev32[22] ^ xgmiirev32[18] ^ xgmiirev32[17] ^ xgmiirev32[13] ^ xgmiirev32[10] ^ xgmiirev32[9] ^ xgmiirev32[5] ^ c[5] ^ c[9] ^ c[10] ^ c[13] ^ c[17] ^ c[18] ^ c[22] ^ c[24] ^ c[26] ^ c[27] ^ c[29] ^ c[31];

newcrc8[22] <= xgmiirev32[31] ^ xgmiirev32[29] ^ xgmiirev32[27] ^ xgmiirev32[26] ^ xgmiirev32[24] ^ xgmiirev32[23] ^ xgmiirev32[19] ^ xgmiirev32[18] ^ xgmiirev32[16] ^ xgmiirev32[14] ^ xgmiirev32[12] ^ xgmiirev32[11] ^ xgmiirev32[9] ^ xgmiirev32[0] ^ c[0] ^ c[9] ^ c[11] ^ c[12] ^ c[14] ^ c[16] ^ c[18] ^ c[19] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[29] ^ c[31];

newcrc8[23] <= xgmiirev32[31] ^ xgmiirev32[29] ^ xgmiirev32[27] ^ xgmiirev32[26] ^ xgmiirev32[20] ^ xgmiirev32[19] ^ xgmiirev32[17] ^ xgmiirev32[16] ^ xgmiirev32[15] ^ xgmiirev32[13] ^ xgmiirev32[9] ^ xgmiirev32[6] ^ xgmiirev32[1] ^ xgmiirev32[0] ^ c[0] ^ c[1] ^ c[6] ^ c[9] ^ c[13] ^ c[15] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[26] ^ c[27] ^ c[29] ^ c[31];

newcrc8[24] <= xgmiirev32[30] ^ xgmiirev32[28] ^ xgmiirev32[27] ^ xgmiirev32[21] ^ xgmiirev32[20] ^ xgmiirev32[18] ^ xgmiirev32[17] ^ xgmiirev32[16] ^ xgmiirev32[14] ^ xgmiirev32[10] ^ xgmiirev32[7] ^ xgmiirev32[2] ^ xgmiirev32[1] ^ c[1] ^ c[2] ^ c[7] ^ c[10] ^ c[14] ^ c[16] ^ c[17] ^ c[18] ^ c[20] ^ c[21] ^ c[27] ^ c[28] ^ c[30];

newcrc8[25] <= xgmiirev32[31] ^ xgmiirev32[29] ^ xgmiirev32[28] ^ xgmiirev32[22] ^ xgmiirev32[21] ^ xgmiirev32[19] ^ xgmiirev32[18] ^ xgmiirev32[17] ^ xgmiirev32[15] ^ xgmiirev32[11] ^ xgmiirev32[8] ^ xgmiirev32[3] ^ xgmiirev32[2] ^ c[2] ^ c[3] ^ c[8] ^ c[11] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[22] ^ c[28] ^ c[29] ^ c[31];

```
                                                                    newcrc8[26] <=
xgmiirev32[31] ^ xgmiirev32[28] ^ xgmiirev32[26] ^ xgmiirev32[25] ^ xgmiirev32[24] ^ xgmiirev32[23] ^
xgmiirev32[22] ^ xgmiirev32[20] ^ xgmiirev32[19] ^ xgmiirev32[18] ^ xgmiirev32[10] ^ xgmiirev32[6] ^
xgmiirev32[4] ^ xgmiirev32[3] ^ xgmiirev32[0] ^ c[0] ^ c[3] ^ c[4] ^ c[6] ^ c[10] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[23]
^ c[24] ^ c[25] ^ c[26] ^ c[28] ^ c[31];

                                                                    newcrc8[27] <=
xgmiirev32[29] ^ xgmiirev32[27] ^ xgmiirev32[26] ^ xgmiirev32[25] ^ xgmiirev32[24] ^ xgmiirev32[23] ^
xgmiirev32[21] ^ xgmiirev32[20] ^ xgmiirev32[19] ^ xgmiirev32[11] ^ xgmiirev32[7] ^ xgmiirev32[5] ^ xgmiirev32[4]
^ xgmiirev32[1] ^ c[1] ^ c[4] ^ c[5] ^ c[7] ^ c[11] ^ c[19] ^ c[20] ^ c[21] ^ c[23] ^ c[24] ^ c[25] ^ c[26] ^ c[27] ^ c[29];

                                                                    newcrc8[28] <=
xgmiirev32[30] ^ xgmiirev32[28] ^ xgmiirev32[27] ^ xgmiirev32[26] ^ xgmiirev32[25] ^ xgmiirev32[24] ^
xgmiirev32[22] ^ xgmiirev32[21] ^ xgmiirev32[20] ^ xgmiirev32[12] ^ xgmiirev32[8] ^ xgmiirev32[6] ^ xgmiirev32[5]
^ xgmiirev32[2] ^ c[2] ^ c[5] ^ c[6] ^ c[8] ^ c[12] ^ c[20] ^ c[21] ^ c[22] ^ c[24] ^ c[25] ^ c[26] ^ c[27] ^ c[28] ^ c[30];

                                                                    newcrc8[29] <=
xgmiirev32[31] ^ xgmiirev32[29] ^ xgmiirev32[28] ^ xgmiirev32[27] ^ xgmiirev32[26] ^ xgmiirev32[25] ^
xgmiirev32[23] ^ xgmiirev32[22] ^ xgmiirev32[21] ^ xgmiirev32[13] ^ xgmiirev32[9] ^ xgmiirev32[7] ^ xgmiirev32[6]
^ xgmiirev32[3] ^ c[3] ^ c[6] ^ c[7] ^ c[9] ^ c[13] ^ c[21] ^ c[22] ^ c[23] ^ c[25] ^ c[26] ^ c[27] ^ c[28] ^ c[29] ^ c[31];

                                                                    newcrc8[30] <=
xgmiirev32[30] ^ xgmiirev32[29] ^ xgmiirev32[28] ^ xgmiirev32[27] ^ xgmiirev32[26] ^ xgmiirev32[24] ^
xgmiirev32[23] ^ xgmiirev32[22] ^ xgmiirev32[14] ^ xgmiirev32[10] ^ xgmiirev32[8] ^ xgmiirev32[7] ^ xgmiirev32[4]
^ c[4] ^ c[7] ^ c[8] ^ c[10] ^ c[14] ^ c[22] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[28] ^ c[29] ^ c[30];

                                                                    newcrc8[31] <=
xgmiirev32[31] ^ xgmiirev32[30] ^ xgmiirev32[29] ^ xgmiirev32[28] ^ xgmiirev32[27] ^ xgmiirev32[25] ^
xgmiirev32[24] ^ xgmiirev32[23] ^ xgmiirev32[15] ^ xgmiirev32[11] ^ xgmiirev32[9] ^ xgmiirev32[8] ^ xgmiirev32[5]
^ c[5] ^ c[8] ^ c[9] ^ c[11] ^ c[15] ^ c[23] ^ c[24] ^ c[25] ^ c[27] ^ c[28] ^ c[29] ^ c[30] ^ c[31];


        checksum_flag<=1;

                                                                    crc_flag<=0;

                                                            end

                                                            //

                                                            //input data is eop

                                                            else if (xgmiidata[61:54]==8'hFD) begin

                                                                    xgmii[31:24] <=
xgmiidata[70:63];

                                                                    xgmii[39:32] <=
xgmiidata[7:0];

                                                                    xgmii[47:40] <=
xgmiidata[16:9];

                                                                    xgmii[55:48] <=
xgmiidata[25:18];

                                                                    xgmii[63:56] <=
xgmiidata[34:27];
```

xgmii[24:0]<=0;

empty_flag<=7;

eop_flag_buff <=
1;

newcrc8[0] <=
xgmiirev40[37] ^ xgmiirev40[34] ^ xgmiirev40[32] ^ xgmiirev40[31] ^ xgmiirev40[30] ^ xgmiirev40[29] ^
xgmiirev40[28] ^ xgmiirev40[26] ^ xgmiirev40[25] ^ xgmiirev40[24] ^ xgmiirev40[16] ^ xgmiirev40[12] ^
xgmiirev40[10] ^ xgmiirev40[9] ^ xgmiirev40[6] ^ xgmiirev40[0] ^ c[1] ^ c[2] ^ c[4] ^ c[8] ^ c[16] ^ c[17] ^ c[18] ^
c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[24] ^ c[26] ^ c[29];

newcrc8[1] <=
xgmiirev40[38] ^ xgmiirev40[37] ^ xgmiirev40[35] ^ xgmiirev40[34] ^ xgmiirev40[33] ^ xgmiirev40[28] ^
xgmiirev40[27] ^ xgmiirev40[24] ^ xgmiirev40[17] ^ xgmiirev40[16] ^ xgmiirev40[13] ^ xgmiirev40[12] ^
xgmiirev40[11] ^ xgmiirev40[9] ^ xgmiirev40[7] ^ xgmiirev40[6] ^ xgmiirev40[1] ^ xgmiirev40[0] ^ c[1] ^ c[3] ^ c[4]
^ c[5] ^ c[8] ^ c[9] ^ c[16] ^ c[19] ^ c[20] ^ c[25] ^ c[26] ^ c[27] ^ c[29] ^ c[30];

newcrc8[2] <=
xgmiirev40[39] ^ xgmiirev40[38] ^ xgmiirev40[37] ^ xgmiirev40[36] ^ xgmiirev40[35] ^ xgmiirev40[32] ^
xgmiirev40[31] ^ xgmiirev40[30] ^ xgmiirev40[26] ^ xgmiirev40[24] ^ xgmiirev40[18] ^ xgmiirev40[17] ^
xgmiirev40[16] ^ xgmiirev40[14] ^ xgmiirev40[13] ^ xgmiirev40[9] ^ xgmiirev40[8] ^ xgmiirev40[7] ^ xgmiirev40[6]
^ xgmiirev40[2] ^ xgmiirev40[1] ^ xgmiirev40[0] ^ c[0] ^ c[1] ^ c[5] ^ c[6] ^ c[8] ^ c[9] ^ c[10] ^ c[16] ^ c[18] ^ c[22]
^ c[23] ^ c[24] ^ c[27] ^ c[28] ^ c[29] ^ c[30] ^ c[31];

newcrc8[3] <=
xgmiirev40[39] ^ xgmiirev40[38] ^ xgmiirev40[37] ^ xgmiirev40[36] ^ xgmiirev40[33] ^ xgmiirev40[32] ^
xgmiirev40[31] ^ xgmiirev40[27] ^ xgmiirev40[25] ^ xgmiirev40[19] ^ xgmiirev40[18] ^ xgmiirev40[17] ^
xgmiirev40[15] ^ xgmiirev40[14] ^ xgmiirev40[10] ^ xgmiirev40[9] ^ xgmiirev40[8] ^ xgmiirev40[7] ^ xgmiirev40[3]
^ xgmiirev40[2] ^ xgmiirev40[1] ^ c[0] ^ c[1] ^ c[2] ^ c[6] ^ c[7] ^ c[9] ^ c[10] ^ c[11] ^ c[17] ^ c[19] ^ c[23] ^ c[24] ^
c[25] ^ c[28] ^ c[29] ^ c[30] ^ c[31];

newcrc8[4] <=
xgmiirev40[39] ^ xgmiirev40[38] ^ xgmiirev40[33] ^ xgmiirev40[31] ^ xgmiirev40[30] ^ xgmiirev40[29] ^
xgmiirev40[25] ^ xgmiirev40[24] ^ xgmiirev40[20] ^ xgmiirev40[19] ^ xgmiirev40[18] ^ xgmiirev40[15] ^
xgmiirev40[12] ^ xgmiirev40[11] ^ xgmiirev40[8] ^ xgmiirev40[6] ^ xgmiirev40[4] ^ xgmiirev40[3] ^ xgmiirev40[2] ^
xgmiirev40[0] ^ c[0] ^ c[3] ^ c[4] ^ c[7] ^ c[10] ^ c[11] ^ c[12] ^ c[16] ^ c[17] ^ c[21] ^ c[22] ^ c[23] ^ c[25] ^ c[30] ^
c[31];

newcrc8[5] <=
xgmiirev40[39] ^ xgmiirev40[37] ^ xgmiirev40[29] ^ xgmiirev40[28] ^ xgmiirev40[24] ^ xgmiirev40[21] ^
xgmiirev40[20] ^ xgmiirev40[19] ^ xgmiirev40[13] ^ xgmiirev40[10] ^ xgmiirev40[7] ^ xgmiirev40[6] ^ xgmiirev40[5]
^ xgmiirev40[4] ^ xgmiirev40[3] ^ xgmiirev40[1] ^ xgmiirev40[0] ^ c[2] ^ c[5] ^ c[11] ^ c[12] ^ c[13] ^ c[16] ^ c[20] ^
c[21] ^ c[29] ^ c[31];

newcrc8[6] <=
xgmiirev40[38] ^ xgmiirev40[30] ^ xgmiirev40[29] ^ xgmiirev40[25] ^ xgmiirev40[22] ^ xgmiirev40[21] ^
xgmiirev40[20] ^ xgmiirev40[14] ^ xgmiirev40[11] ^ xgmiirev40[8] ^ xgmiirev40[7] ^ xgmiirev40[6] ^ xgmiirev40[5]
^ xgmiirev40[4] ^ xgmiirev40[2] ^ xgmiirev40[1] ^ c[0] ^ c[3] ^ c[6] ^ c[12] ^ c[13] ^ c[14] ^ c[17] ^ c[21] ^ c[22] ^
c[30];

newcrc8[7] <=
xgmiirev40[39] ^ xgmiirev40[37] ^ xgmiirev40[34] ^ xgmiirev40[32] ^ xgmiirev40[29] ^ xgmiirev40[28] ^

xgmiirev40[25] ^ xgmiirev40[24] ^ xgmiirev40[23] ^ xgmiirev40[22] ^ xgmiirev40[21] ^ xgmiirev40[16] ^ xgmiirev40[15] ^ xgmiirev40[10] ^ xgmiirev40[8] ^ xgmiirev40[7] ^ xgmiirev40[5] ^ xgmiirev40[3] ^ xgmiirev40[2] ^ xgmiirev40[0] ^ c[0] ^ c[2] ^ c[7] ^ c[8] ^ c[13] ^ c[14] ^ c[15] ^ c[16] ^ c[17] ^ c[20] ^ c[21] ^ c[24] ^ c[26] ^ c[29] ^ c[31];

newcrc8[8] <= xgmiirev40[38] ^ xgmiirev40[37] ^ xgmiirev40[35] ^ xgmiirev40[34] ^ xgmiirev40[33] ^ xgmiirev40[32] ^ xgmiirev40[31] ^ xgmiirev40[28] ^ xgmiirev40[23] ^ xgmiirev40[22] ^ xgmiirev40[17] ^ xgmiirev40[12] ^ xgmiirev40[11] ^ xgmiirev40[10] ^ xgmiirev40[8] ^ xgmiirev40[4] ^ xgmiirev40[3] ^ xgmiirev40[1] ^ xgmiirev40[0] ^ c[0] ^ c[2] ^ c[3] ^ c[4] ^ c[9] ^ c[14] ^ c[15] ^ c[20] ^ c[23] ^ c[24] ^ c[25] ^ c[26] ^ c[27] ^ c[29] ^ c[30];

newcrc8[9] <= xgmiirev40[39] ^ xgmiirev40[38] ^ xgmiirev40[36] ^ xgmiirev40[35] ^ xgmiirev40[34] ^ xgmiirev40[33] ^ xgmiirev40[32] ^ xgmiirev40[29] ^ xgmiirev40[24] ^ xgmiirev40[23] ^ xgmiirev40[18] ^ xgmiirev40[13] ^ xgmiirev40[12] ^ xgmiirev40[11] ^ xgmiirev40[9] ^ xgmiirev40[5] ^ xgmiirev40[4] ^ xgmiirev40[2] ^ xgmiirev40[1] ^ c[1] ^ c[3] ^ c[4] ^ c[5] ^ c[10] ^ c[15] ^ c[16] ^ c[21] ^ c[24] ^ c[25] ^ c[26] ^ c[27] ^ c[28] ^ c[30] ^ c[31];

newcrc8[10] <= xgmiirev40[39] ^ xgmiirev40[36] ^ xgmiirev40[35] ^ xgmiirev40[33] ^ xgmiirev40[32] ^ xgmiirev40[31] ^ xgmiirev40[29] ^ xgmiirev40[28] ^ xgmiirev40[26] ^ xgmiirev40[19] ^ xgmiirev40[16] ^ xgmiirev40[14] ^ xgmiirev40[13] ^ xgmiirev40[9] ^ xgmiirev40[5] ^ xgmiirev40[3] ^ xgmiirev40[2] ^ xgmiirev40[0] ^ c[1] ^ c[5] ^ c[6] ^ c[8] ^ c[11] ^ c[18] ^ c[20] ^ c[21] ^ c[23] ^ c[24] ^ c[25] ^ c[27] ^ c[28] ^ c[31];

newcrc8[11] <= xgmiirev40[36] ^ xgmiirev40[33] ^ xgmiirev40[31] ^ xgmiirev40[28] ^ xgmiirev40[27] ^ xgmiirev40[26] ^ xgmiirev40[25] ^ xgmiirev40[24] ^ xgmiirev40[20] ^ xgmiirev40[17] ^ xgmiirev40[16] ^ xgmiirev40[15] ^ xgmiirev40[14] ^ xgmiirev40[12] ^ xgmiirev40[9] ^ xgmiirev40[4] ^ xgmiirev40[3] ^ xgmiirev40[1] ^ xgmiirev40[0] ^ c[1] ^ c[4] ^ c[6] ^ c[7] ^ c[8] ^ c[9] ^ c[12] ^ c[16] ^ c[17] ^ c[18] ^ c[19] ^ c[20] ^ c[23] ^ c[25] ^ c[28];

newcrc8[12] <= xgmiirev40[31] ^ xgmiirev40[30] ^ xgmiirev40[27] ^ xgmiirev40[24] ^ xgmiirev40[21] ^ xgmiirev40[18] ^ xgmiirev40[17] ^ xgmiirev40[15] ^ xgmiirev40[13] ^ xgmiirev40[12] ^ xgmiirev40[9] ^ xgmiirev40[6] ^ xgmiirev40[5] ^ xgmiirev40[4] ^ xgmiirev40[2] ^ xgmiirev40[1] ^ xgmiirev40[0] ^ c[1] ^ c[4] ^ c[5] ^ c[7] ^ c[9] ^ c[10] ^ c[13] ^ c[16] ^ c[19] ^ c[22] ^ c[23];

newcrc8[13] <= xgmiirev40[32] ^ xgmiirev40[31] ^ xgmiirev40[28] ^ xgmiirev40[25] ^ xgmiirev40[22] ^ xgmiirev40[19] ^ xgmiirev40[18] ^ xgmiirev40[16] ^ xgmiirev40[14] ^ xgmiirev40[13] ^ xgmiirev40[10] ^ xgmiirev40[7] ^ xgmiirev40[6] ^ xgmiirev40[5] ^ xgmiirev40[3] ^ xgmiirev40[2] ^ xgmiirev40[1] ^ c[2] ^ c[5] ^ c[6] ^ c[8] ^ c[10] ^ c[11] ^ c[14] ^ c[17] ^ c[20] ^ c[23] ^ c[24];

newcrc8[14] <= xgmiirev40[33] ^ xgmiirev40[32] ^ xgmiirev40[29] ^ xgmiirev40[26] ^ xgmiirev40[23] ^ xgmiirev40[20] ^ xgmiirev40[19] ^ xgmiirev40[17] ^ xgmiirev40[15] ^ xgmiirev40[14] ^ xgmiirev40[11] ^ xgmiirev40[8] ^ xgmiirev40[7] ^ xgmiirev40[6] ^ xgmiirev40[4] ^ xgmiirev40[3] ^ xgmiirev40[2] ^ c[0] ^ c[3] ^ c[6] ^ c[7] ^ c[9] ^ c[11] ^ c[12] ^ c[15] ^ c[18] ^ c[21] ^ c[24] ^ c[25];

newcrc8[15] <= xgmiirev40[34] ^ xgmiirev40[33] ^ xgmiirev40[30] ^ xgmiirev40[27] ^ xgmiirev40[24] ^ xgmiirev40[21] ^ xgmiirev40[20] ^ xgmiirev40[18] ^ xgmiirev40[16] ^ xgmiirev40[15] ^ xgmiirev40[12] ^ xgmiirev40[9] ^ xgmiirev40[8] ^ xgmiirev40[7] ^ xgmiirev40[5] ^ xgmiirev40[4] ^ xgmiirev40[3] ^ c[0] ^ c[1] ^ c[4] ^ c[7] ^ c[8] ^ c[10] ^ c[12] ^ c[13] ^ c[16] ^ c[19] ^ c[22] ^ c[25] ^ c[26];

newcrc8[16] <= xgmiirev40[37] ^ xgmiirev40[35] ^ xgmiirev40[32] ^ xgmiirev40[30] ^ xgmiirev40[29] ^ xgmiirev40[26] ^ xgmiirev40[24] ^ xgmiirev40[22] ^ xgmiirev40[21] ^ xgmiirev40[19] ^ xgmiirev40[17] ^ xgmiirev40[13] ^

xgmiirev40[12] ^ xgmiirev40[8] ^ xgmiirev40[5] ^ xgmiirev40[4] ^ xgmiirev40[0] ^ c[0] ^ c[4] ^ c[5] ^ c[9] ^ c[11] ^ c[13] ^ c[14] ^ c[16] ^ c[18] ^ c[21] ^ c[22] ^ c[24] ^ c[27] ^ c[29];

newcrc8[17] <= xgmiirev40[38] ^ xgmiirev40[36] ^ xgmiirev40[33] ^ xgmiirev40[31] ^ xgmiirev40[30] ^ xgmiirev40[27] ^ xgmiirev40[25] ^ xgmiirev40[23] ^ xgmiirev40[22] ^ xgmiirev40[20] ^ xgmiirev40[18] ^ xgmiirev40[14] ^ xgmiirev40[13] ^ xgmiirev40[9] ^ xgmiirev40[6] ^ xgmiirev40[5] ^ xgmiirev40[1] ^ c[1] ^ c[5] ^ c[6] ^ c[10] ^ c[12] ^ c[14] ^ c[15] ^ c[17] ^ c[19] ^ c[22] ^ c[23] ^ c[25] ^ c[28] ^ c[30];

newcrc8[18] <= xgmiirev40[39] ^ xgmiirev40[37] ^ xgmiirev40[34] ^ xgmiirev40[32] ^ xgmiirev40[31] ^ xgmiirev40[28] ^ xgmiirev40[26] ^ xgmiirev40[24] ^ xgmiirev40[23] ^ xgmiirev40[21] ^ xgmiirev40[19] ^ xgmiirev40[15] ^ xgmiirev40[14] ^ xgmiirev40[10] ^ xgmiirev40[7] ^ xgmiirev40[6] ^ xgmiirev40[2] ^ c[2] ^ c[6] ^ c[7] ^ c[11] ^ c[13] ^ c[15] ^ c[16] ^ c[18] ^ c[20] ^ c[23] ^ c[24] ^ c[26] ^ c[29] ^ c[31];

newcrc8[19] <= xgmiirev40[38] ^ xgmiirev40[35] ^ xgmiirev40[33] ^ xgmiirev40[32] ^ xgmiirev40[29] ^ xgmiirev40[27] ^ xgmiirev40[25] ^ xgmiirev40[24] ^ xgmiirev40[22] ^ xgmiirev40[20] ^ xgmiirev40[16] ^ xgmiirev40[15] ^ xgmiirev40[11] ^ xgmiirev40[8] ^ xgmiirev40[7] ^ xgmiirev40[3] ^ c[0] ^ c[3] ^ c[7] ^ c[8] ^ c[12] ^ c[14] ^ c[16] ^ c[17] ^ c[19] ^ c[21] ^ c[24] ^ c[25] ^ c[27] ^ c[30];

newcrc8[20] <= xgmiirev40[39] ^ xgmiirev40[36] ^ xgmiirev40[34] ^ xgmiirev40[33] ^ xgmiirev40[30] ^ xgmiirev40[28] ^ xgmiirev40[26] ^ xgmiirev40[25] ^ xgmiirev40[23] ^ xgmiirev40[21] ^ xgmiirev40[17] ^ xgmiirev40[16] ^ xgmiirev40[12] ^ xgmiirev40[9] ^ xgmiirev40[8] ^ xgmiirev40[4] ^ c[0] ^ c[1] ^ c[4] ^ c[8] ^ c[9] ^ c[13] ^ c[15] ^ c[17] ^ c[18] ^ c[20] ^ c[22] ^ c[25] ^ c[26] ^ c[28] ^ c[31];

newcrc8[21] <= xgmiirev40[37] ^ xgmiirev40[35] ^ xgmiirev40[34] ^ xgmiirev40[31] ^ xgmiirev40[29] ^ xgmiirev40[27] ^ xgmiirev40[26] ^ xgmiirev40[24] ^ xgmiirev40[22] ^ xgmiirev40[18] ^ xgmiirev40[17] ^ xgmiirev40[13] ^ xgmiirev40[10] ^ xgmiirev40[9] ^ xgmiirev40[5] ^ c[1] ^ c[2] ^ c[5] ^ c[9] ^ c[10] ^ c[14] ^ c[16] ^ c[18] ^ c[19] ^ c[21] ^ c[23] ^ c[26] ^ c[27] ^ c[29];

newcrc8[22] <= xgmiirev40[38] ^ xgmiirev40[37] ^ xgmiirev40[36] ^ xgmiirev40[35] ^ xgmiirev40[34] ^ xgmiirev40[31] ^ xgmiirev40[29] ^ xgmiirev40[27] ^ xgmiirev40[26] ^ xgmiirev40[24] ^ xgmiirev40[23] ^ xgmiirev40[19] ^ xgmiirev40[18] ^ xgmiirev40[16] ^ xgmiirev40[14] ^ xgmiirev40[12] ^ xgmiirev40[11] ^ xgmiirev40[9] ^ xgmiirev40[0] ^ c[1] ^ c[3] ^ c[4] ^ c[6] ^ c[8] ^ c[10] ^ c[11] ^ c[15] ^ c[16] ^ c[18] ^ c[19] ^ c[21] ^ c[23] ^ c[26] ^ c[27] ^ c[28] ^ c[29] ^ c[30];

newcrc8[23] <= xgmiirev40[39] ^ xgmiirev40[38] ^ xgmiirev40[36] ^ xgmiirev40[35] ^ xgmiirev40[34] ^ xgmiirev40[31] ^ xgmiirev40[29] ^ xgmiirev40[27] ^ xgmiirev40[26] ^ xgmiirev40[20] ^ xgmiirev40[19] ^ xgmiirev40[17] ^ xgmiirev40[16] ^ xgmiirev40[15] ^ xgmiirev40[13] ^ xgmiirev40[9] ^ xgmiirev40[6] ^ xgmiirev40[1] ^ xgmiirev40[0] ^ c[1] ^ c[5] ^ c[7] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[18] ^ c[19] ^ c[21] ^ c[23] ^ c[26] ^ c[27] ^ c[28] ^ c[30] ^ c[31];

newcrc8[24] <= xgmiirev40[39] ^ xgmiirev40[37] ^ xgmiirev40[36] ^ xgmiirev40[35] ^ xgmiirev40[32] ^ xgmiirev40[30] ^ xgmiirev40[28] ^ xgmiirev40[27] ^ xgmiirev40[21] ^ xgmiirev40[20] ^ xgmiirev40[18] ^ xgmiirev40[17] ^ xgmiirev40[16] ^ xgmiirev40[14] ^ xgmiirev40[10] ^ xgmiirev40[7] ^ xgmiirev40[2] ^ xgmiirev40[1] ^ c[2] ^ c[6] ^ c[8] ^ c[9] ^ c[10] ^ c[12] ^ c[13] ^ c[19] ^ c[20] ^ c[22] ^ c[24] ^ c[27] ^ c[28] ^ c[29] ^ c[31];

newcrc8[25] <= xgmiirev40[38] ^ xgmiirev40[37] ^ xgmiirev40[36] ^ xgmiirev40[33] ^ xgmiirev40[31] ^ xgmiirev40[29] ^ xgmiirev40[28] ^ xgmiirev40[22] ^ xgmiirev40[21] ^ xgmiirev40[19] ^ xgmiirev40[18] ^ xgmiirev40[17] ^

```
xgmiirev40[15] ^ xgmiirev40[11] ^ xgmiirev40[8] ^ xgmiirev40[3] ^ xgmiirev40[2] ^ c[0] ^ c[3] ^ c[7] ^ c[9] ^ c[10] ^
c[11] ^ c[13] ^ c[14] ^ c[20] ^ c[21] ^ c[23] ^ c[25] ^ c[28] ^ c[29] ^ c[30];

                                                                    newcrc8[26] <=
xgmiirev40[39] ^ xgmiirev40[38] ^ xgmiirev40[31] ^ xgmiirev40[28] ^ xgmiirev40[26] ^ xgmiirev40[25] ^
xgmiirev40[24] ^ xgmiirev40[23] ^ xgmiirev40[22] ^ xgmiirev40[20] ^ xgmiirev40[19] ^ xgmiirev40[18] ^
xgmiirev40[10] ^ xgmiirev40[6] ^ xgmiirev40[4] ^ xgmiirev40[3] ^ xgmiirev40[0] ^ c[2] ^ c[10] ^ c[11] ^ c[12] ^ c[14]
^ c[15] ^ c[16] ^ c[17] ^ c[18] ^ c[20] ^ c[23] ^ c[30] ^ c[31];

                                                                    newcrc8[27] <=
xgmiirev40[39] ^ xgmiirev40[32] ^ xgmiirev40[29] ^ xgmiirev40[27] ^ xgmiirev40[26] ^ xgmiirev40[25] ^
xgmiirev40[24] ^ xgmiirev40[23] ^ xgmiirev40[21] ^ xgmiirev40[20] ^ xgmiirev40[19] ^ xgmiirev40[11] ^
xgmiirev40[7] ^ xgmiirev40[5] ^ xgmiirev40[4] ^ xgmiirev40[1] ^ c[3] ^ c[11] ^ c[12] ^ c[13] ^ c[15] ^ c[16] ^ c[17] ^
c[18] ^ c[19] ^ c[21] ^ c[24] ^ c[31];

                                                                    newcrc8[28] <=
xgmiirev40[33] ^ xgmiirev40[30] ^ xgmiirev40[28] ^ xgmiirev40[27] ^ xgmiirev40[26] ^ xgmiirev40[25] ^
xgmiirev40[24] ^ xgmiirev40[22] ^ xgmiirev40[21] ^ xgmiirev40[20] ^ xgmiirev40[12] ^ xgmiirev40[8] ^
xgmiirev40[6] ^ xgmiirev40[5] ^ xgmiirev40[2] ^ c[0] ^ c[4] ^ c[12] ^ c[13] ^ c[14] ^ c[16] ^ c[17] ^ c[18] ^ c[19] ^
c[20] ^ c[22] ^ c[25];

                                                                    newcrc8[29] <=
xgmiirev40[34] ^ xgmiirev40[31] ^ xgmiirev40[29] ^ xgmiirev40[28] ^ xgmiirev40[27] ^ xgmiirev40[26] ^
xgmiirev40[25] ^ xgmiirev40[23] ^ xgmiirev40[22] ^ xgmiirev40[21] ^ xgmiirev40[13] ^ xgmiirev40[9] ^
xgmiirev40[7] ^ xgmiirev40[6] ^ xgmiirev40[3] ^ c[1] ^ c[5] ^ c[13] ^ c[14] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^ c[20] ^
c[21] ^ c[23] ^ c[26];

                                                                    newcrc8[30] <=
xgmiirev40[35] ^ xgmiirev40[32] ^ xgmiirev40[30] ^ xgmiirev40[29] ^ xgmiirev40[28] ^ xgmiirev40[27] ^
xgmiirev40[26] ^ xgmiirev40[24] ^ xgmiirev40[23] ^ xgmiirev40[22] ^ xgmiirev40[14] ^ xgmiirev40[10] ^
xgmiirev40[8] ^ xgmiirev40[7] ^ xgmiirev40[4] ^ c[0] ^ c[2] ^ c[6] ^ c[14] ^ c[15] ^ c[16] ^ c[18] ^ c[19] ^ c[20] ^ c[21]
^ c[22] ^ c[24] ^ c[27];

                                                                    newcrc8[31] <=
xgmiirev40[36] ^ xgmiirev40[33] ^ xgmiirev40[31] ^ xgmiirev40[30] ^ xgmiirev40[29] ^ xgmiirev40[28] ^
xgmiirev40[27] ^ xgmiirev40[25] ^ xgmiirev40[24] ^ xgmiirev40[23] ^ xgmiirev40[15] ^ xgmiirev40[11] ^
xgmiirev40[9] ^ xgmiirev40[8] ^ xgmiirev40[5] ^ c[0] ^ c[1] ^ c[3] ^ c[7] ^ c[15] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[21]
^ c[22] ^ c[23] ^ c[25] ^ c[28];



        checksum_buff<=1;

                                                                end

                                                                else if (xgmiidata[52:45]==8'hFD) begin

                                                                    xgmii[23:16] <=
xgmiidata[61:54];

                                                                    xgmii[31:24] <=
xgmiidata[70:63];

                                                                    xgmii[39:32] <=
xgmiidata[7:0];
```

xgmii[47:40] <= xgmiidata[16:9];

xgmii[55:48] <= xgmiidata[25:18];

xgmii[63:56] <= xgmiidata[34:27];

xgmii[16:0]<=0;

empty_flag<=6;

eop_flag_buff <= 1;

newcrc8[0] <= xgmiirev48[47] ^ xgmiirev48[45] ^ xgmiirev48[44] ^ xgmiirev48[37] ^ xgmiirev48[34] ^ xgmiirev48[32] ^ xgmiirev48[31] ^ xgmiirev48[30] ^ xgmiirev48[29] ^ xgmiirev48[28] ^ xgmiirev48[26] ^ xgmiirev48[25] ^ xgmiirev48[24] ^ xgmiirev48[16] ^ xgmiirev48[12] ^ xgmiirev48[10] ^ xgmiirev48[9] ^ xgmiirev48[6] ^ xgmiirev48[0] ^ c[0] ^ c[8] ^ c[9] ^ c[10] ^ c[12] ^ c[13] ^ c[14] ^ c[15] ^ c[16] ^ c[18] ^ c[21] ^ c[28] ^ c[29] ^ c[31];

newcrc8[1] <= xgmiirev48[47] ^ xgmiirev48[46] ^ xgmiirev48[44] ^ xgmiirev48[38] ^ xgmiirev48[37] ^ xgmiirev48[35] ^ xgmiirev48[34] ^ xgmiirev48[33] ^ xgmiirev48[28] ^ xgmiirev48[27] ^ xgmiirev48[24] ^ xgmiirev48[17] ^ xgmiirev48[16] ^ xgmiirev48[13] ^ xgmiirev48[12] ^ xgmiirev48[11] ^ xgmiirev48[9] ^ xgmiirev48[7] ^ xgmiirev48[6] ^ xgmiirev48[1] ^ xgmiirev48[0] ^ c[0] ^ c[1] ^ c[8] ^ c[11] ^ c[12] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[22] ^ c[28] ^ c[30] ^ c[31];

newcrc8[2] <= xgmiirev48[44] ^ xgmiirev48[39] ^ xgmiirev48[38] ^ xgmiirev48[37] ^ xgmiirev48[36] ^ xgmiirev48[35] ^ xgmiirev48[32] ^ xgmiirev48[31] ^ xgmiirev48[30] ^ xgmiirev48[26] ^ xgmiirev48[24] ^ xgmiirev48[18] ^ xgmiirev48[17] ^ xgmiirev48[16] ^ xgmiirev48[14] ^ xgmiirev48[13] ^ xgmiirev48[9] ^ xgmiirev48[8] ^ xgmiirev48[7] ^ xgmiirev48[6] ^ xgmiirev48[2] ^ xgmiirev48[1] ^ xgmiirev48[0] ^ c[0] ^ c[1] ^ c[2] ^ c[8] ^ c[10] ^ c[14] ^ c[15] ^ c[16] ^ c[19] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[28];

newcrc8[3] <= xgmiirev48[45] ^ xgmiirev48[40] ^ xgmiirev48[39] ^ xgmiirev48[38] ^ xgmiirev48[37] ^ xgmiirev48[36] ^ xgmiirev48[33] ^ xgmiirev48[32] ^ xgmiirev48[31] ^ xgmiirev48[27] ^ xgmiirev48[25] ^ xgmiirev48[19] ^ xgmiirev48[18] ^ xgmiirev48[17] ^ xgmiirev48[15] ^ xgmiirev48[14] ^ xgmiirev48[10] ^ xgmiirev48[9] ^ xgmiirev48[8] ^ xgmiirev48[7] ^ xgmiirev48[3] ^ xgmiirev48[2] ^ xgmiirev48[1] ^ c[1] ^ c[2] ^ c[3] ^ c[9] ^ c[11] ^ c[15] ^ c[16] ^ c[17] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[24] ^ c[29];

newcrc8[4] <= xgmiirev48[47] ^ xgmiirev48[46] ^ xgmiirev48[45] ^ xgmiirev48[44] ^ xgmiirev48[41] ^ xgmiirev48[40] ^ xgmiirev48[39] ^ xgmiirev48[38] ^ xgmiirev48[33] ^ xgmiirev48[31] ^ xgmiirev48[30] ^ xgmiirev48[29] ^ xgmiirev48[25] ^ xgmiirev48[24] ^ xgmiirev48[20] ^ xgmiirev48[19] ^ xgmiirev48[18] ^ xgmiirev48[15] ^ xgmiirev48[12] ^ xgmiirev48[11] ^ xgmiirev48[8] ^ xgmiirev48[6] ^ xgmiirev48[4] ^ xgmiirev48[3] ^ xgmiirev48[2] ^ xgmiirev48[0] ^ c[2] ^ c[3] ^ c[4] ^ c[8] ^ c[9] ^ c[13] ^ c[14] ^ c[15] ^ c[17] ^ c[22] ^ c[23] ^ c[24] ^ c[25] ^ c[28] ^ c[29] ^ c[30] ^ c[31];

newcrc8[5] <= xgmiirev48[46] ^ xgmiirev48[44] ^ xgmiirev48[42] ^ xgmiirev48[41] ^ xgmiirev48[40] ^ xgmiirev48[39] ^ xgmiirev48[37] ^ xgmiirev48[29] ^ xgmiirev48[28] ^ xgmiirev48[24] ^ xgmiirev48[21] ^ xgmiirev48[20] ^ xgmiirev48[19] ^ xgmiirev48[13] ^ xgmiirev48[10] ^ xgmiirev48[7] ^ xgmiirev48[6] ^ xgmiirev48[5] ^ xgmiirev48[4]

^ xgmiirev48[3] ^ xgmiirev48[1] ^ xgmiirev48[0] ^ c[3] ^ c[4] ^ c[5] ^ c[8] ^ c[12] ^ c[13] ^ c[21] ^ c[23] ^ c[24] ^ c[25] ^ c[26] ^ c[28] ^ c[30];

newcrc8[6] <= xgmiirev48[47] ^ xgmiirev48[45] ^ xgmiirev48[43] ^ xgmiirev48[42] ^ xgmiirev48[41] ^ xgmiirev48[40] ^ xgmiirev48[38] ^ xgmiirev48[30] ^ xgmiirev48[29] ^ xgmiirev48[25] ^ xgmiirev48[22] ^ xgmiirev48[21] ^ xgmiirev48[20] ^ xgmiirev48[14] ^ xgmiirev48[11] ^ xgmiirev48[8] ^ xgmiirev48[7] ^ xgmiirev48[6] ^ xgmiirev48[5] ^ xgmiirev48[4] ^ xgmiirev48[2] ^ xgmiirev48[1] ^ c[4] ^ c[5] ^ c[6] ^ c[9] ^ c[13] ^ c[14] ^ c[22] ^ c[24] ^ c[25] ^ c[26] ^ c[27] ^ c[29] ^ c[31];

newcrc8[7] <= xgmiirev48[47] ^ xgmiirev48[46] ^ xgmiirev48[45] ^ xgmiirev48[43] ^ xgmiirev48[42] ^ xgmiirev48[41] ^ xgmiirev48[39] ^ xgmiirev48[37] ^ xgmiirev48[34] ^ xgmiirev48[32] ^ xgmiirev48[29] ^ xgmiirev48[28] ^ xgmiirev48[25] ^ xgmiirev48[24] ^ xgmiirev48[23] ^ xgmiirev48[22] ^ xgmiirev48[21] ^ xgmiirev48[16] ^ xgmiirev48[15] ^ xgmiirev48[10] ^ xgmiirev48[8] ^ xgmiirev48[7] ^ xgmiirev48[5] ^ xgmiirev48[3] ^ xgmiirev48[2] ^ xgmiirev48[0] ^ c[0] ^ c[5] ^ c[6] ^ c[7] ^ c[8] ^ c[9] ^ c[12] ^ c[13] ^ c[16] ^ c[18] ^ c[21] ^ c[23] ^ c[25] ^ c[26] ^ c[27] ^ c[29] ^ c[30] ^ c[31];

newcrc8[8] <= xgmiirev48[46] ^ xgmiirev48[45] ^ xgmiirev48[43] ^ xgmiirev48[42] ^ xgmiirev48[40] ^ xgmiirev48[38] ^ xgmiirev48[37] ^ xgmiirev48[35] ^ xgmiirev48[34] ^ xgmiirev48[33] ^ xgmiirev48[32] ^ xgmiirev48[31] ^ xgmiirev48[28] ^ xgmiirev48[23] ^ xgmiirev48[22] ^ xgmiirev48[17] ^ xgmiirev48[12] ^ xgmiirev48[11] ^ xgmiirev48[10] ^ xgmiirev48[8] ^ xgmiirev48[4] ^ xgmiirev48[3] ^ xgmiirev48[1] ^ xgmiirev48[0] ^ c[1] ^ c[6] ^ c[7] ^ c[12] ^ c[15] ^ c[16] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[22] ^ c[24] ^ c[26] ^ c[27] ^ c[29] ^ c[30];

newcrc8[9] <= xgmiirev48[47] ^ xgmiirev48[46] ^ xgmiirev48[44] ^ xgmiirev48[43] ^ xgmiirev48[41] ^ xgmiirev48[39] ^ xgmiirev48[38] ^ xgmiirev48[36] ^ xgmiirev48[35] ^ xgmiirev48[34] ^ xgmiirev48[33] ^ xgmiirev48[32] ^ xgmiirev48[29] ^ xgmiirev48[24] ^ xgmiirev48[23] ^ xgmiirev48[18] ^ xgmiirev48[13] ^ xgmiirev48[12] ^ xgmiirev48[11] ^ xgmiirev48[9] ^ xgmiirev48[5] ^ xgmiirev48[4] ^ xgmiirev48[2] ^ xgmiirev48[1] ^ c[2] ^ c[7] ^ c[8] ^ c[13] ^ c[16] ^ c[17] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[23] ^ c[25] ^ c[27] ^ c[28] ^ c[30] ^ c[31];

newcrc8[10] <= xgmiirev48[42] ^ xgmiirev48[40] ^ xgmiirev48[39] ^ xgmiirev48[36] ^ xgmiirev48[35] ^ xgmiirev48[33] ^ xgmiirev48[32] ^ xgmiirev48[31] ^ xgmiirev48[29] ^ xgmiirev48[28] ^ xgmiirev48[26] ^ xgmiirev48[19] ^ xgmiirev48[16] ^ xgmiirev48[14] ^ xgmiirev48[13] ^ xgmiirev48[9] ^ xgmiirev48[5] ^ xgmiirev48[3] ^ xgmiirev48[2] ^ xgmiirev48[0] ^ c[0] ^ c[3] ^ c[10] ^ c[12] ^ c[13] ^ c[15] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[23] ^ c[24] ^ c[26];

newcrc8[11] <= xgmiirev48[47] ^ xgmiirev48[45] ^ xgmiirev48[44] ^ xgmiirev48[43] ^ xgmiirev48[41] ^ xgmiirev48[40] ^ xgmiirev48[36] ^ xgmiirev48[33] ^ xgmiirev48[31] ^ xgmiirev48[28] ^ xgmiirev48[27] ^ xgmiirev48[26] ^ xgmiirev48[25] ^ xgmiirev48[24] ^ xgmiirev48[20] ^ xgmiirev48[17] ^ xgmiirev48[16] ^ xgmiirev48[15] ^ xgmiirev48[14] ^ xgmiirev48[12] ^ xgmiirev48[9] ^ xgmiirev48[4] ^ xgmiirev48[3] ^ xgmiirev48[1] ^ xgmiirev48[0] ^ c[0] ^ c[1] ^ c[4] ^ c[8] ^ c[9] ^ c[10] ^ c[11] ^ c[12] ^ c[15] ^ c[17] ^ c[20] ^ c[24] ^ c[25] ^ c[27] ^ c[28] ^ c[29] ^ c[31];

newcrc8[12] <= xgmiirev48[47] ^ xgmiirev48[46] ^ xgmiirev48[42] ^ xgmiirev48[41] ^ xgmiirev48[31] ^ xgmiirev48[30] ^ xgmiirev48[27] ^ xgmiirev48[24] ^ xgmiirev48[21] ^ xgmiirev48[18] ^ xgmiirev48[17] ^ xgmiirev48[15] ^ xgmiirev48[13] ^ xgmiirev48[12] ^ xgmiirev48[9] ^ xgmiirev48[6] ^ xgmiirev48[5] ^ xgmiirev48[4] ^ xgmiirev48[2] ^ xgmiirev48[1] ^ xgmiirev48[0] ^ c[1] ^ c[2] ^ c[5] ^ c[8] ^ c[11] ^ c[14] ^ c[15] ^ c[25] ^ c[26] ^ c[30] ^ c[31];

newcrc8[13] <= xgmiirev48[47] ^ xgmiirev48[43] ^ xgmiirev48[42] ^ xgmiirev48[32] ^ xgmiirev48[31] ^ xgmiirev48[28] ^ xgmiirev48[25] ^ xgmiirev48[22] ^ xgmiirev48[19] ^ xgmiirev48[18] ^ xgmiirev48[16] ^ xgmiirev48[14] ^

xgmiirev48[13] ^ xgmiirev48[10] ^ xgmiirev48[7] ^ xgmiirev48[6] ^ xgmiirev48[5] ^ xgmiirev48[3] ^ xgmiirev48[2] ^ xgmiirev48[1] ^ c[0] ^ c[2] ^ c[3] ^ c[6] ^ c[9] ^ c[12] ^ c[15] ^ c[16] ^ c[26] ^ c[27] ^ c[31];

newcrc8[14] <= xgmiirev48[44] ^ xgmiirev48[43] ^ xgmiirev48[33] ^ xgmiirev48[32] ^ xgmiirev48[29] ^ xgmiirev48[26] ^ xgmiirev48[23] ^ xgmiirev48[20] ^ xgmiirev48[19] ^ xgmiirev48[17] ^ xgmiirev48[15] ^ xgmiirev48[14] ^ xgmiirev48[11] ^ xgmiirev48[8] ^ xgmiirev48[7] ^ xgmiirev48[6] ^ xgmiirev48[4] ^ xgmiirev48[3] ^ xgmiirev48[2] ^ c[1] ^ c[3] ^ c[4] ^ c[7] ^ c[10] ^ c[13] ^ c[16] ^ c[17] ^ c[27] ^ c[28];

newcrc8[15] <= xgmiirev48[45] ^ xgmiirev48[44] ^ xgmiirev48[34] ^ xgmiirev48[33] ^ xgmiirev48[30] ^ xgmiirev48[27] ^ xgmiirev48[24] ^ xgmiirev48[21] ^ xgmiirev48[20] ^ xgmiirev48[18] ^ xgmiirev48[16] ^ xgmiirev48[15] ^ xgmiirev48[12] ^ xgmiirev48[9] ^ xgmiirev48[8] ^ xgmiirev48[7] ^ xgmiirev48[5] ^ xgmiirev48[4] ^ xgmiirev48[3] ^ c[0] ^ c[2] ^ c[4] ^ c[5] ^ c[8] ^ c[11] ^ c[14] ^ c[17] ^ c[18] ^ c[28] ^ c[29];

newcrc8[16] <= xgmiirev48[47] ^ xgmiirev48[46] ^ xgmiirev48[44] ^ xgmiirev48[37] ^ xgmiirev48[35] ^ xgmiirev48[32] ^ xgmiirev48[30] ^ xgmiirev48[29] ^ xgmiirev48[26] ^ xgmiirev48[24] ^ xgmiirev48[22] ^ xgmiirev48[21] ^ xgmiirev48[19] ^ xgmiirev48[17] ^ xgmiirev48[13] ^ xgmiirev48[12] ^ xgmiirev48[8] ^ xgmiirev48[5] ^ xgmiirev48[4] ^ xgmiirev48[0] ^ c[1] ^ c[3] ^ c[5] ^ c[6] ^ c[8] ^ c[10] ^ c[13] ^ c[14] ^ c[16] ^ c[19] ^ c[21] ^ c[28] ^ c[30] ^ c[31];

newcrc8[17] <= xgmiirev48[47] ^ xgmiirev48[45] ^ xgmiirev48[38] ^ xgmiirev48[36] ^ xgmiirev48[33] ^ xgmiirev48[31] ^ xgmiirev48[30] ^ xgmiirev48[27] ^ xgmiirev48[25] ^ xgmiirev48[23] ^ xgmiirev48[22] ^ xgmiirev48[20] ^ xgmiirev48[18] ^ xgmiirev48[14] ^ xgmiirev48[13] ^ xgmiirev48[9] ^ xgmiirev48[6] ^ xgmiirev48[5] ^ xgmiirev48[1] ^ c[2] ^ c[4] ^ c[6] ^ c[7] ^ c[9] ^ c[11] ^ c[14] ^ c[15] ^ c[17] ^ c[20] ^ c[22] ^ c[29] ^ c[31];

newcrc8[18] <= xgmiirev48[46] ^ xgmiirev48[39] ^ xgmiirev48[37] ^ xgmiirev48[34] ^ xgmiirev48[32] ^ xgmiirev48[31] ^ xgmiirev48[28] ^ xgmiirev48[26] ^ xgmiirev48[24] ^ xgmiirev48[23] ^ xgmiirev48[21] ^ xgmiirev48[19] ^ xgmiirev48[15] ^ xgmiirev48[14] ^ xgmiirev48[10] ^ xgmiirev48[7] ^ xgmiirev48[6] ^ xgmiirev48[2] ^ c[3] ^ c[5] ^ c[7] ^ c[8] ^ c[10] ^ c[12] ^ c[15] ^ c[16] ^ c[18] ^ c[21] ^ c[23] ^ c[30];

newcrc8[19] <= xgmiirev48[47] ^ xgmiirev48[40] ^ xgmiirev48[38] ^ xgmiirev48[35] ^ xgmiirev48[33] ^ xgmiirev48[32] ^ xgmiirev48[29] ^ xgmiirev48[27] ^ xgmiirev48[25] ^ xgmiirev48[24] ^ xgmiirev48[22] ^ xgmiirev48[20] ^ xgmiirev48[16] ^ xgmiirev48[15] ^ xgmiirev48[11] ^ xgmiirev48[8] ^ xgmiirev48[7] ^ xgmiirev48[3] ^ c[0] ^ c[4] ^ c[6] ^ c[8] ^ c[9] ^ c[11] ^ c[13] ^ c[16] ^ c[17] ^ c[19] ^ c[22] ^ c[24] ^ c[31];

newcrc8[20] <= xgmiirev48[41] ^ xgmiirev48[39] ^ xgmiirev48[36] ^ xgmiirev48[34] ^ xgmiirev48[33] ^ xgmiirev48[30] ^ xgmiirev48[28] ^ xgmiirev48[26] ^ xgmiirev48[25] ^ xgmiirev48[23] ^ xgmiirev48[21] ^ xgmiirev48[17] ^ xgmiirev48[16] ^ xgmiirev48[12] ^ xgmiirev48[9] ^ xgmiirev48[8] ^ xgmiirev48[4] ^ c[0] ^ c[1] ^ c[5] ^ c[7] ^ c[9] ^ c[10] ^ c[12] ^ c[14] ^ c[17] ^ c[18] ^ c[20] ^ c[23] ^ c[25];

newcrc8[21] <= xgmiirev48[42] ^ xgmiirev48[40] ^ xgmiirev48[37] ^ xgmiirev48[35] ^ xgmiirev48[34] ^ xgmiirev48[31] ^ xgmiirev48[29] ^ xgmiirev48[27] ^ xgmiirev48[26] ^ xgmiirev48[24] ^ xgmiirev48[22] ^ xgmiirev48[18] ^ xgmiirev48[17] ^ xgmiirev48[13] ^ xgmiirev48[10] ^ xgmiirev48[9] ^ xgmiirev48[5] ^ c[1] ^ c[2] ^ c[6] ^ c[8] ^ c[10] ^ c[11] ^ c[13] ^ c[15] ^ c[18] ^ c[19] ^ c[21] ^ c[24] ^ c[26];

newcrc8[22] <= xgmiirev48[47] ^ xgmiirev48[45] ^ xgmiirev48[44] ^ xgmiirev48[43] ^ xgmiirev48[41] ^ xgmiirev48[38] ^ xgmiirev48[37] ^ xgmiirev48[36] ^ xgmiirev48[35] ^ xgmiirev48[34] ^ xgmiirev48[31] ^ xgmiirev48[29] ^ xgmiirev48[27] ^ xgmiirev48[26] ^ xgmiirev48[24] ^ xgmiirev48[23] ^ xgmiirev48[19] ^ xgmiirev48[18] ^ xgmiirev48[16] ^ xgmiirev48[14] ^ xgmiirev48[12] ^ xgmiirev48[11] ^ xgmiirev48[9] ^ xgmiirev48[0] ^ c[0] ^ c[2] ^

c[3] ^ c[7] ^ c[8] ^ c[10] ^ c[11] ^ c[13] ^ c[15] ^ c[18] ^ c[19] ^ c[20] ^ c[21] ^ c[22] ^ c[25] ^ c[27] ^ c[28] ^ c[29] ^ c[31];

newcrc8[23] <= xgmiirev48[47] ^ xgmiirev48[46] ^ xgmiirev48[42] ^ xgmiirev48[39] ^ xgmiirev48[38] ^ xgmiirev48[36] ^ xgmiirev48[35] ^ xgmiirev48[34] ^ xgmiirev48[31] ^ xgmiirev48[29] ^ xgmiirev48[27] ^ xgmiirev48[26] ^ xgmiirev48[20] ^ xgmiirev48[19] ^ xgmiirev48[17] ^ xgmiirev48[16] ^ xgmiirev48[15] ^ xgmiirev48[13] ^ xgmiirev48[9] ^ xgmiirev48[6] ^ xgmiirev48[1] ^ xgmiirev48[0] ^ c[0] ^ c[1] ^ c[3] ^ c[4] ^ c[10] ^ c[11] ^ c[13] ^ c[15] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[23] ^ c[26] ^ c[30] ^ c[31];

newcrc8[24] <= xgmiirev48[47] ^ xgmiirev48[43] ^ xgmiirev48[40] ^ xgmiirev48[39] ^ xgmiirev48[37] ^ xgmiirev48[36] ^ xgmiirev48[35] ^ xgmiirev48[32] ^ xgmiirev48[30] ^ xgmiirev48[28] ^ xgmiirev48[27] ^ xgmiirev48[21] ^ xgmiirev48[20] ^ xgmiirev48[18] ^ xgmiirev48[17] ^ xgmiirev48[16] ^ xgmiirev48[14] ^ xgmiirev48[10] ^ xgmiirev48[7] ^ xgmiirev48[2] ^ xgmiirev48[1] ^ c[0] ^ c[1] ^ c[2] ^ c[4] ^ c[5] ^ c[11] ^ c[12] ^ c[14] ^ c[16] ^ c[19] ^ c[20] ^ c[21] ^ c[23] ^ c[24] ^ c[27] ^ c[31];

newcrc8[25] <= xgmiirev48[44] ^ xgmiirev48[41] ^ xgmiirev48[40] ^ xgmiirev48[38] ^ xgmiirev48[37] ^ xgmiirev48[36] ^ xgmiirev48[33] ^ xgmiirev48[31] ^ xgmiirev48[29] ^ xgmiirev48[28] ^ xgmiirev48[22] ^ xgmiirev48[21] ^ xgmiirev48[19] ^ xgmiirev48[18] ^ xgmiirev48[17] ^ xgmiirev48[15] ^ xgmiirev48[11] ^ xgmiirev48[8] ^ xgmiirev48[3] ^ xgmiirev48[2] ^ c[1] ^ c[2] ^ c[3] ^ c[5] ^ c[6] ^ c[12] ^ c[13] ^ c[15] ^ c[17] ^ c[20] ^ c[21] ^ c[22] ^ c[24] ^ c[25] ^ c[28];

newcrc8[26] <= xgmiirev48[47] ^ xgmiirev48[44] ^ xgmiirev48[42] ^ xgmiirev48[41] ^ xgmiirev48[39] ^ xgmiirev48[38] ^ xgmiirev48[31] ^ xgmiirev48[28] ^ xgmiirev48[26] ^ xgmiirev48[25] ^ xgmiirev48[24] ^ xgmiirev48[23] ^ xgmiirev48[22] ^ xgmiirev48[20] ^ xgmiirev48[19] ^ xgmiirev48[18] ^ xgmiirev48[10] ^ xgmiirev48[6] ^ xgmiirev48[4] ^ xgmiirev48[3] ^ xgmiirev48[0] ^ c[2] ^ c[3] ^ c[4] ^ c[6] ^ c[7] ^ c[8] ^ c[9] ^ c[10] ^ c[12] ^ c[15] ^ c[22] ^ c[23] ^ c[25] ^ c[26] ^ c[28] ^ c[31];

newcrc8[27] <= xgmiirev48[45] ^ xgmiirev48[43] ^ xgmiirev48[42] ^ xgmiirev48[40] ^ xgmiirev48[39] ^ xgmiirev48[32] ^ xgmiirev48[29] ^ xgmiirev48[27] ^ xgmiirev48[26] ^ xgmiirev48[25] ^ xgmiirev48[24] ^ xgmiirev48[23] ^ xgmiirev48[21] ^ xgmiirev48[20] ^ xgmiirev48[19] ^ xgmiirev48[11] ^ xgmiirev48[7] ^ xgmiirev48[5] ^ xgmiirev48[4] ^ xgmiirev48[1] ^ c[3] ^ c[4] ^ c[5] ^ c[7] ^ c[8] ^ c[9] ^ c[10] ^ c[11] ^ c[13] ^ c[16] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[29];

newcrc8[28] <= xgmiirev48[46] ^ xgmiirev48[44] ^ xgmiirev48[43] ^ xgmiirev48[41] ^ xgmiirev48[40] ^ xgmiirev48[33] ^ xgmiirev48[30] ^ xgmiirev48[28] ^ xgmiirev48[27] ^ xgmiirev48[26] ^ xgmiirev48[25] ^ xgmiirev48[24] ^ xgmiirev48[22] ^ xgmiirev48[21] ^ xgmiirev48[20] ^ xgmiirev48[12] ^ xgmiirev48[8] ^ xgmiirev48[6] ^ xgmiirev48[5] ^ xgmiirev48[2] ^ c[4] ^ c[5] ^ c[6] ^ c[8] ^ c[9] ^ c[10] ^ c[11] ^ c[12] ^ c[14] ^ c[17] ^ c[24] ^ c[25] ^ c[27] ^ c[28] ^ c[30];

newcrc8[29] <= xgmiirev48[47] ^ xgmiirev48[45] ^ xgmiirev48[44] ^ xgmiirev48[42] ^ xgmiirev48[41] ^ xgmiirev48[34] ^ xgmiirev48[31] ^ xgmiirev48[29] ^ xgmiirev48[28] ^ xgmiirev48[27] ^ xgmiirev48[26] ^ xgmiirev48[25] ^ xgmiirev48[23] ^ xgmiirev48[22] ^ xgmiirev48[21] ^ xgmiirev48[13] ^ xgmiirev48[9] ^ xgmiirev48[7] ^ xgmiirev48[6] ^ xgmiirev48[3] ^ c[5] ^ c[6] ^ c[7] ^ c[9] ^ c[10] ^ c[11] ^ c[12] ^ c[13] ^ c[15] ^ c[18] ^ c[25] ^ c[26] ^ c[28] ^ c[29] ^ c[31];

newcrc8[30] <= xgmiirev48[46] ^ xgmiirev48[45] ^ xgmiirev48[43] ^ xgmiirev48[42] ^ xgmiirev48[35] ^ xgmiirev48[32] ^ xgmiirev48[30] ^ xgmiirev48[29] ^ xgmiirev48[28] ^ xgmiirev48[27] ^ xgmiirev48[26] ^ xgmiirev48[24] ^

```
xgmiirev48[23] ^ xgmiirev48[22] ^ xgmiirev48[14] ^ xgmiirev48[10] ^ xgmiirev48[8] ^ xgmiirev48[7] ^ xgmiirev48[4]
^ c[6] ^ c[7] ^ c[8] ^ c[10] ^ c[11] ^ c[12] ^ c[13] ^ c[14] ^ c[16] ^ c[19] ^ c[26] ^ c[27] ^ c[29] ^ c[30];

                                                                    newcrc8[31] <=
xgmiirev48[47] ^ xgmiirev48[46] ^ xgmiirev48[44] ^ xgmiirev48[43] ^ xgmiirev48[36] ^ xgmiirev48[33] ^
xgmiirev48[31] ^ xgmiirev48[30] ^ xgmiirev48[29] ^ xgmiirev48[28] ^ xgmiirev48[27] ^ xgmiirev48[25] ^
xgmiirev48[24] ^ xgmiirev48[23] ^ xgmiirev48[15] ^ xgmiirev48[11] ^ xgmiirev48[9] ^ xgmiirev48[8] ^ xgmiirev48[5]
^ c[7] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[13] ^ c[14] ^ c[15] ^ c[17] ^ c[20] ^ c[27] ^ c[28] ^ c[30] ^ c[31];



        checksum_buff<=1;

                                                                end

                                                                else if (xgmiidata[43:36]==8'hFD) begin

                                                                    xgmii[15:8] <=
xgmiidata[52:45];

                                                                    xgmii[23:16] <=
xgmiidata[61:54];

                                                                    xgmii[31:24] <=
xgmiidata[70:63];

                                                                    xgmii[39:32] <=
xgmiidata[7:0];

                                                                    xgmii[47:40] <=
xgmiidata[16:9];

                                                                    xgmii[55:48] <=
xgmiidata[25:18];

                                                                    xgmii[63:56] <=
xgmiidata[34:27];

                                                                    xgmii[7:0]<=0;

                                                                    empty_flag<=5;

                                                                    eop_flag_buff <=
1;



                                                                    newcrc8[0]  <=
xgmiirev56[55] ^ xgmiirev56[54] ^ xgmiirev56[53] ^ xgmiirev56[50] ^ xgmiirev56[48] ^ xgmiirev56[47] ^
xgmiirev56[45] ^ xgmiirev56[44] ^ xgmiirev56[37] ^ xgmiirev56[34] ^ xgmiirev56[32] ^ xgmiirev56[31] ^
xgmiirev56[30] ^ xgmiirev56[29] ^ xgmiirev56[28] ^ xgmiirev56[26] ^ xgmiirev56[25] ^ xgmiirev56[24] ^
xgmiirev56[16] ^ xgmiirev56[12] ^ xgmiirev56[10] ^ xgmiirev56[9] ^ xgmiirev56[6] ^ xgmiirev56[0] ^ c[0] ^ c[1] ^
c[2] ^ c[4] ^ c[5] ^ c[6] ^ c[7] ^ c[8] ^ c[10] ^ c[13] ^ c[20] ^ c[21] ^ c[23] ^ c[24] ^ c[26] ^ c[29] ^ c[30] ^ c[31];

                                                                    newcrc8[1]  <=
xgmiirev56[53] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[49] ^ xgmiirev56[47] ^ xgmiirev56[46] ^
xgmiirev56[44] ^ xgmiirev56[38] ^ xgmiirev56[37] ^ xgmiirev56[35] ^ xgmiirev56[34] ^ xgmiirev56[33] ^
xgmiirev56[28] ^ xgmiirev56[27] ^ xgmiirev56[24] ^ xgmiirev56[17] ^ xgmiirev56[16] ^ xgmiirev56[13] ^
```

xgmiirev56[12] ^ xgmiirev56[11] ^ xgmiirev56[9] ^ xgmiirev56[7] ^ xgmiirev56[6] ^ xgmiirev56[1] ^ xgmiirev56[0] ^ c[0] ^ c[3] ^ c[4] ^ c[9] ^ c[10] ^ c[11] ^ c[13] ^ c[14] ^ c[20] ^ c[22] ^ c[23] ^ c[25] ^ c[26] ^ c[27] ^ c[29];

newcrc8[2] <= xgmiirev56[55] ^ xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[44] ^ xgmiirev56[39] ^ xgmiirev56[38] ^ xgmiirev56[37] ^ xgmiirev56[36] ^ xgmiirev56[35] ^ xgmiirev56[32] ^ xgmiirev56[31] ^ xgmiirev56[30] ^ xgmiirev56[26] ^ xgmiirev56[24] ^ xgmiirev56[18] ^ xgmiirev56[17] ^ xgmiirev56[16] ^ xgmiirev56[14] ^ xgmiirev56[13] ^ xgmiirev56[9] ^ xgmiirev56[8] ^ xgmiirev56[7] ^ xgmiirev56[6] ^ xgmiirev56[2] ^ xgmiirev56[1] ^ xgmiirev56[0] ^ c[0] ^ c[2] ^ c[6] ^ c[7] ^ c[8] ^ c[11] ^ c[12] ^ c[13] ^ c[14] ^ c[15] ^ c[20] ^ c[27] ^ c[28] ^ c[29] ^ c[31];

newcrc8[3] <= xgmiirev56[54] ^ xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[45] ^ xgmiirev56[40] ^ xgmiirev56[39] ^ xgmiirev56[38] ^ xgmiirev56[37] ^ xgmiirev56[36] ^ xgmiirev56[33] ^ xgmiirev56[32] ^ xgmiirev56[31] ^ xgmiirev56[27] ^ xgmiirev56[25] ^ xgmiirev56[19] ^ xgmiirev56[18] ^ xgmiirev56[17] ^ xgmiirev56[15] ^ xgmiirev56[14] ^ xgmiirev56[10] ^ xgmiirev56[9] ^ xgmiirev56[8] ^ xgmiirev56[7] ^ xgmiirev56[3] ^ xgmiirev56[2] ^ xgmiirev56[1] ^ c[1] ^ c[3] ^ c[7] ^ c[8] ^ c[9] ^ c[12] ^ c[13] ^ c[14] ^ c[15] ^ c[16] ^ c[21] ^ c[28] ^ c[29] ^ c[30];

newcrc8[4] <= xgmiirev56[50] ^ xgmiirev56[48] ^ xgmiirev56[47] ^ xgmiirev56[46] ^ xgmiirev56[45] ^ xgmiirev56[44] ^ xgmiirev56[41] ^ xgmiirev56[40] ^ xgmiirev56[39] ^ xgmiirev56[38] ^ xgmiirev56[33] ^ xgmiirev56[31] ^ xgmiirev56[30] ^ xgmiirev56[29] ^ xgmiirev56[25] ^ xgmiirev56[24] ^ xgmiirev56[20] ^ xgmiirev56[19] ^ xgmiirev56[18] ^ xgmiirev56[15] ^ xgmiirev56[12] ^ xgmiirev56[11] ^ xgmiirev56[8] ^ xgmiirev56[6] ^ xgmiirev56[4] ^ xgmiirev56[3] ^ xgmiirev56[2] ^ xgmiirev56[0] ^ c[0] ^ c[1] ^ c[5] ^ c[6] ^ c[7] ^ c[9] ^ c[14] ^ c[15] ^ c[16] ^ c[17] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[24] ^ c[26];

newcrc8[5] <= xgmiirev56[55] ^ xgmiirev56[54] ^ xgmiirev56[53] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[49] ^ xgmiirev56[46] ^ xgmiirev56[44] ^ xgmiirev56[42] ^ xgmiirev56[41] ^ xgmiirev56[40] ^ xgmiirev56[39] ^ xgmiirev56[37] ^ xgmiirev56[29] ^ xgmiirev56[28] ^ xgmiirev56[24] ^ xgmiirev56[21] ^ xgmiirev56[20] ^ xgmiirev56[19] ^ xgmiirev56[13] ^ xgmiirev56[10] ^ xgmiirev56[7] ^ xgmiirev56[6] ^ xgmiirev56[5] ^ xgmiirev56[4] ^ xgmiirev56[3] ^ xgmiirev56[1] ^ xgmiirev56[0] ^ c[0] ^ c[4] ^ c[5] ^ c[13] ^ c[15] ^ c[16] ^ c[17] ^ c[18] ^ c[20] ^ c[22] ^ c[25] ^ c[26] ^ c[27] ^ c[29] ^ c[30] ^ c[31];

newcrc8[6] <= xgmiirev56[55] ^ xgmiirev56[54] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[47] ^ xgmiirev56[45] ^ xgmiirev56[43] ^ xgmiirev56[42] ^ xgmiirev56[41] ^ xgmiirev56[40] ^ xgmiirev56[38] ^ xgmiirev56[30] ^ xgmiirev56[29] ^ xgmiirev56[25] ^ xgmiirev56[22] ^ xgmiirev56[21] ^ xgmiirev56[20] ^ xgmiirev56[14] ^ xgmiirev56[11] ^ xgmiirev56[8] ^ xgmiirev56[7] ^ xgmiirev56[6] ^ xgmiirev56[5] ^ xgmiirev56[4] ^ xgmiirev56[2] ^ xgmiirev56[1] ^ c[1] ^ c[5] ^ c[6] ^ c[14] ^ c[16] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[23] ^ c[26] ^ c[27] ^ c[28] ^ c[30] ^ c[31];

newcrc8[7] <= xgmiirev56[54] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[47] ^ xgmiirev56[46] ^ xgmiirev56[45] ^ xgmiirev56[43] ^ xgmiirev56[42] ^ xgmiirev56[41] ^ xgmiirev56[39] ^ xgmiirev56[37] ^ xgmiirev56[34] ^ xgmiirev56[32] ^ xgmiirev56[29] ^ xgmiirev56[28] ^ xgmiirev56[25] ^ xgmiirev56[24] ^ xgmiirev56[23] ^ xgmiirev56[22] ^ xgmiirev56[21] ^ xgmiirev56[16] ^ xgmiirev56[15] ^ xgmiirev56[10] ^ xgmiirev56[8] ^ xgmiirev56[7] ^ xgmiirev56[5] ^ xgmiirev56[3] ^ xgmiirev56[2] ^ xgmiirev56[0] ^ c[0] ^ c[1] ^ c[4] ^ c[5] ^ c[8] ^ c[10] ^ c[13] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[22] ^ c[23] ^ c[26] ^ c[27] ^ c[28] ^ c[30];

newcrc8[8] <= xgmiirev56[54] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[46] ^ xgmiirev56[45] ^ xgmiirev56[43] ^ xgmiirev56[42] ^ xgmiirev56[40] ^ xgmiirev56[38] ^ xgmiirev56[37] ^ xgmiirev56[35] ^ xgmiirev56[34] ^ xgmiirev56[33] ^ xgmiirev56[32] ^ xgmiirev56[31] ^ xgmiirev56[28] ^ xgmiirev56[23] ^ xgmiirev56[22] ^ xgmiirev56[17] ^ xgmiirev56[12] ^ xgmiirev56[11] ^ xgmiirev56[10] ^ xgmiirev56[8] ^

xgmiirev56[4] ^ xgmiirev56[3] ^ xgmiirev56[1] ^ xgmiirev56[0] ^ c[4] ^ c[7] ^ c[8] ^ c[9] ^ c[10] ^ c[11] ^ c[13] ^ c[14] ^ c[16] ^ c[18] ^ c[19] ^ c[21] ^ c[22] ^ c[26] ^ c[27] ^ c[28] ^ c[30];

newcrc8[9] <= xgmiirev56[55] ^ xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[47] ^ xgmiirev56[46] ^ xgmiirev56[44] ^ xgmiirev56[43] ^ xgmiirev56[41] ^ xgmiirev56[39] ^ xgmiirev56[38] ^ xgmiirev56[36] ^ xgmiirev56[35] ^ xgmiirev56[34] ^ xgmiirev56[33] ^ xgmiirev56[32] ^ xgmiirev56[29] ^ xgmiirev56[24] ^ xgmiirev56[23] ^ xgmiirev56[18] ^ xgmiirev56[13] ^ xgmiirev56[12] ^ xgmiirev56[11] ^ xgmiirev56[9] ^ xgmiirev56[5] ^ xgmiirev56[4] ^ xgmiirev56[2] ^ xgmiirev56[1] ^ c[0] ^ c[5] ^ c[8] ^ c[9] ^ c[10] ^ c[11] ^ c[12] ^ c[14] ^ c[15] ^ c[17] ^ c[19] ^ c[20] ^ c[22] ^ c[23] ^ c[27] ^ c[28] ^ c[29] ^ c[31];

newcrc8[10] <= xgmiirev56[55] ^ xgmiirev56[52] ^ xgmiirev56[50] ^ xgmiirev56[42] ^ xgmiirev56[40] ^ xgmiirev56[39] ^ xgmiirev56[36] ^ xgmiirev56[35] ^ xgmiirev56[33] ^ xgmiirev56[32] ^ xgmiirev56[31] ^ xgmiirev56[29] ^ xgmiirev56[28] ^ xgmiirev56[26] ^ xgmiirev56[19] ^ xgmiirev56[16] ^ xgmiirev56[14] ^ xgmiirev56[13] ^ xgmiirev56[9] ^ xgmiirev56[5] ^ xgmiirev56[3] ^ xgmiirev56[2] ^ xgmiirev56[0] ^ c[2] ^ c[4] ^ c[5] ^ c[7] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[15] ^ c[16] ^ c[18] ^ c[26] ^ c[28] ^ c[31];

newcrc8[11] <= xgmiirev56[55] ^ xgmiirev56[54] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[48] ^ xgmiirev56[47] ^ xgmiirev56[45] ^ xgmiirev56[44] ^ xgmiirev56[43] ^ xgmiirev56[41] ^ xgmiirev56[40] ^ xgmiirev56[36] ^ xgmiirev56[33] ^ xgmiirev56[31] ^ xgmiirev56[28] ^ xgmiirev56[27] ^ xgmiirev56[26] ^ xgmiirev56[25] ^ xgmiirev56[24] ^ xgmiirev56[20] ^ xgmiirev56[17] ^ xgmiirev56[16] ^ xgmiirev56[15] ^ xgmiirev56[14] ^ xgmiirev56[12] ^ xgmiirev56[9] ^ xgmiirev56[4] ^ xgmiirev56[3] ^ xgmiirev56[1] ^ xgmiirev56[0] ^ c[0] ^ c[1] ^ c[2] ^ c[3] ^ c[4] ^ c[7] ^ c[9] ^ c[12] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[21] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[30] ^ c[31];

newcrc8[12] <= xgmiirev56[54] ^ xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[49] ^ xgmiirev56[47] ^ xgmiirev56[46] ^ xgmiirev56[42] ^ xgmiirev56[41] ^ xgmiirev56[31] ^ xgmiirev56[30] ^ xgmiirev56[27] ^ xgmiirev56[24] ^ xgmiirev56[21] ^ xgmiirev56[18] ^ xgmiirev56[17] ^ xgmiirev56[15] ^ xgmiirev56[13] ^ xgmiirev56[12] ^ xgmiirev56[9] ^ xgmiirev56[6] ^ xgmiirev56[5] ^ xgmiirev56[4] ^ xgmiirev56[2] ^ xgmiirev56[1] ^ xgmiirev56[0] ^ c[0] ^ c[3] ^ c[6] ^ c[7] ^ c[17] ^ c[18] ^ c[22] ^ c[23] ^ c[25] ^ c[26] ^ c[27] ^ c[28] ^ c[29] ^ c[30];

newcrc8[13] <= xgmiirev56[55] ^ xgmiirev56[54] ^ xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[48] ^ xgmiirev56[47] ^ xgmiirev56[43] ^ xgmiirev56[42] ^ xgmiirev56[32] ^ xgmiirev56[31] ^ xgmiirev56[28] ^ xgmiirev56[25] ^ xgmiirev56[22] ^ xgmiirev56[19] ^ xgmiirev56[18] ^ xgmiirev56[16] ^ xgmiirev56[14] ^ xgmiirev56[13] ^ xgmiirev56[10] ^ xgmiirev56[7] ^ xgmiirev56[6] ^ xgmiirev56[5] ^ xgmiirev56[3] ^ xgmiirev56[2] ^ xgmiirev56[1] ^ c[1] ^ c[4] ^ c[7] ^ c[8] ^ c[18] ^ c[19] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[28] ^ c[29] ^ c[30] ^ c[31];

newcrc8[14] <= xgmiirev56[55] ^ xgmiirev56[54] ^ xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[49] ^ xgmiirev56[48] ^ xgmiirev56[44] ^ xgmiirev56[43] ^ xgmiirev56[33] ^ xgmiirev56[32] ^ xgmiirev56[29] ^ xgmiirev56[26] ^ xgmiirev56[23] ^ xgmiirev56[20] ^ xgmiirev56[19] ^ xgmiirev56[17] ^ xgmiirev56[15] ^ xgmiirev56[14] ^ xgmiirev56[11] ^ xgmiirev56[8] ^ xgmiirev56[7] ^ xgmiirev56[6] ^ xgmiirev56[4] ^ xgmiirev56[3] ^ xgmiirev56[2] ^ c[2] ^ c[5] ^ c[8] ^ c[9] ^ c[19] ^ c[20] ^ c[24] ^ c[25] ^ c[27] ^ c[28] ^ c[29] ^ c[30] ^ c[31];

newcrc8[15] <= xgmiirev56[55] ^ xgmiirev56[54] ^ xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[50] ^ xgmiirev56[49] ^ xgmiirev56[45] ^ xgmiirev56[44] ^ xgmiirev56[34] ^ xgmiirev56[33] ^ xgmiirev56[30] ^ xgmiirev56[27] ^ xgmiirev56[24] ^ xgmiirev56[21] ^ xgmiirev56[20] ^ xgmiirev56[18] ^ xgmiirev56[16] ^ xgmiirev56[15] ^ xgmiirev56[12] ^ xgmiirev56[9] ^ xgmiirev56[8] ^ xgmiirev56[7] ^ xgmiirev56[5] ^ xgmiirev56[4] ^ xgmiirev56[3] ^ c[0] ^ c[3] ^ c[6] ^ c[9] ^ c[10] ^ c[20] ^ c[21] ^ c[25] ^ c[26] ^ c[28] ^ c[29] ^ c[30] ^ c[31];

newcrc8[16] <=
xgmiirev56[51] ^ xgmiirev56[48] ^ xgmiirev56[47] ^ xgmiirev56[46] ^ xgmiirev56[44] ^ xgmiirev56[37] ^
xgmiirev56[35] ^ xgmiirev56[32] ^ xgmiirev56[30] ^ xgmiirev56[29] ^ xgmiirev56[26] ^ xgmiirev56[24] ^
xgmiirev56[22] ^ xgmiirev56[21] ^ xgmiirev56[19] ^ xgmiirev56[17] ^ xgmiirev56[13] ^ xgmiirev56[12] ^
xgmiirev56[8] ^ xgmiirev56[5] ^ xgmiirev56[4] ^ xgmiirev56[0] ^ c[0] ^ c[2] ^ c[5] ^ c[6] ^ c[8] ^ c[11] ^ c[13] ^ c[20]
^ c[22] ^ c[23] ^ c[24] ^ c[27];

newcrc8[17] <=
xgmiirev56[52] ^ xgmiirev56[49] ^ xgmiirev56[48] ^ xgmiirev56[47] ^ xgmiirev56[45] ^ xgmiirev56[38] ^
xgmiirev56[36] ^ xgmiirev56[33] ^ xgmiirev56[31] ^ xgmiirev56[30] ^ xgmiirev56[27] ^ xgmiirev56[25] ^
xgmiirev56[23] ^ xgmiirev56[22] ^ xgmiirev56[20] ^ xgmiirev56[18] ^ xgmiirev56[14] ^ xgmiirev56[13] ^
xgmiirev56[9] ^ xgmiirev56[6] ^ xgmiirev56[5] ^ xgmiirev56[1] ^ c[1] ^ c[3] ^ c[6] ^ c[7] ^ c[9] ^ c[12] ^ c[14] ^ c[21]
^ c[23] ^ c[24] ^ c[25] ^ c[28];

newcrc8[18] <=
xgmiirev56[53] ^ xgmiirev56[50] ^ xgmiirev56[49] ^ xgmiirev56[48] ^ xgmiirev56[46] ^ xgmiirev56[39] ^
xgmiirev56[37] ^ xgmiirev56[34] ^ xgmiirev56[32] ^ xgmiirev56[31] ^ xgmiirev56[28] ^ xgmiirev56[26] ^
xgmiirev56[24] ^ xgmiirev56[23] ^ xgmiirev56[21] ^ xgmiirev56[19] ^ xgmiirev56[15] ^ xgmiirev56[14] ^
xgmiirev56[10] ^ xgmiirev56[7] ^ xgmiirev56[6] ^ xgmiirev56[2] ^ c[0] ^ c[2] ^ c[4] ^ c[7] ^ c[8] ^ c[10] ^ c[13] ^ c[15]
^ c[22] ^ c[24] ^ c[25] ^ c[26] ^ c[29];

newcrc8[19] <=
xgmiirev56[54] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[49] ^ xgmiirev56[47] ^ xgmiirev56[40] ^
xgmiirev56[38] ^ xgmiirev56[35] ^ xgmiirev56[33] ^ xgmiirev56[32] ^ xgmiirev56[29] ^ xgmiirev56[27] ^
xgmiirev56[25] ^ xgmiirev56[24] ^ xgmiirev56[22] ^ xgmiirev56[20] ^ xgmiirev56[16] ^ xgmiirev56[15] ^
xgmiirev56[11] ^ xgmiirev56[8] ^ xgmiirev56[7] ^ xgmiirev56[3] ^ c[0] ^ c[1] ^ c[3] ^ c[5] ^ c[8] ^ c[9] ^ c[11] ^ c[14]
^ c[16] ^ c[23] ^ c[25] ^ c[26] ^ c[27] ^ c[30];

newcrc8[20] <=
xgmiirev56[55] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[48] ^ xgmiirev56[41] ^
xgmiirev56[39] ^ xgmiirev56[36] ^ xgmiirev56[34] ^ xgmiirev56[33] ^ xgmiirev56[30] ^ xgmiirev56[28] ^
xgmiirev56[26] ^ xgmiirev56[25] ^ xgmiirev56[23] ^ xgmiirev56[21] ^ xgmiirev56[17] ^ xgmiirev56[16] ^
xgmiirev56[12] ^ xgmiirev56[9] ^ xgmiirev56[8] ^ xgmiirev56[4] ^ c[1] ^ c[2] ^ c[4] ^ c[6] ^ c[9] ^ c[10] ^ c[12] ^ c[15]
^ c[17] ^ c[24] ^ c[26] ^ c[27] ^ c[28] ^ c[31];

newcrc8[21] <=
xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[49] ^ xgmiirev56[42] ^ xgmiirev56[40] ^
xgmiirev56[37] ^ xgmiirev56[35] ^ xgmiirev56[34] ^ xgmiirev56[31] ^ xgmiirev56[29] ^ xgmiirev56[27] ^
xgmiirev56[26] ^ xgmiirev56[24] ^ xgmiirev56[22] ^ xgmiirev56[18] ^ xgmiirev56[17] ^ xgmiirev56[13] ^
xgmiirev56[10] ^ xgmiirev56[9] ^ xgmiirev56[5] ^ c[0] ^ c[2] ^ c[3] ^ c[5] ^ c[7] ^ c[10] ^ c[11] ^ c[13] ^ c[16] ^ c[18]
^ c[25] ^ c[27] ^ c[28] ^ c[29];

newcrc8[22] <=
xgmiirev56[55] ^ xgmiirev56[52] ^ xgmiirev56[48] ^ xgmiirev56[47] ^ xgmiirev56[45] ^ xgmiirev56[44] ^
xgmiirev56[43] ^ xgmiirev56[41] ^ xgmiirev56[38] ^ xgmiirev56[37] ^ xgmiirev56[36] ^ xgmiirev56[35] ^
xgmiirev56[34] ^ xgmiirev56[31] ^ xgmiirev56[29] ^ xgmiirev56[27] ^ xgmiirev56[26] ^ xgmiirev56[24] ^
xgmiirev56[23] ^ xgmiirev56[19] ^ xgmiirev56[18] ^ xgmiirev56[16] ^ xgmiirev56[14] ^ xgmiirev56[12] ^
xgmiirev56[11] ^ xgmiirev56[9] ^ xgmiirev56[0] ^ c[0] ^ c[2] ^ c[3] ^ c[5] ^ c[7] ^ c[10] ^ c[11] ^ c[12] ^ c[13] ^ c[14]
^ c[17] ^ c[19] ^ c[20] ^ c[21] ^ c[23] ^ c[24] ^ c[28] ^ c[31];

newcrc8[23] <=
xgmiirev56[55] ^ xgmiirev56[54] ^ xgmiirev56[50] ^ xgmiirev56[49] ^ xgmiirev56[47] ^ xgmiirev56[46] ^
xgmiirev56[42] ^ xgmiirev56[39] ^ xgmiirev56[38] ^ xgmiirev56[36] ^ xgmiirev56[35] ^ xgmiirev56[34] ^
xgmiirev56[31] ^ xgmiirev56[29] ^ xgmiirev56[27] ^ xgmiirev56[26] ^ xgmiirev56[20] ^ xgmiirev56[19] ^
xgmiirev56[17] ^ xgmiirev56[16] ^ xgmiirev56[15] ^ xgmiirev56[13] ^ xgmiirev56[9] ^ xgmiirev56[6] ^ xgmiirev56[1]

^ xgmiirev56[0] ^ c[2] ^ c[3] ^ c[5] ^ c[7] ^ c[10] ^ c[11] ^ c[12] ^ c[14] ^ c[15] ^ c[18] ^ c[22] ^ c[23] ^ c[25] ^ c[26] ^ c[30] ^ c[31];

newcrc8[24] <= xgmiirev56[55] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[48] ^ xgmiirev56[47] ^ xgmiirev56[43] ^ xgmiirev56[40] ^ xgmiirev56[39] ^ xgmiirev56[37] ^ xgmiirev56[36] ^ xgmiirev56[35] ^ xgmiirev56[32] ^ xgmiirev56[30] ^ xgmiirev56[28] ^ xgmiirev56[27] ^ xgmiirev56[21] ^ xgmiirev56[20] ^ xgmiirev56[18] ^ xgmiirev56[17] ^ xgmiirev56[16] ^ xgmiirev56[14] ^ xgmiirev56[10] ^ xgmiirev56[7] ^ xgmiirev56[2] ^ xgmiirev56[1] ^ c[3] ^ c[4] ^ c[6] ^ c[8] ^ c[11] ^ c[12] ^ c[13] ^ c[15] ^ c[16] ^ c[19] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[31];

newcrc8[25] <= xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[49] ^ xgmiirev56[48] ^ xgmiirev56[44] ^ xgmiirev56[41] ^ xgmiirev56[40] ^ xgmiirev56[38] ^ xgmiirev56[37] ^ xgmiirev56[36] ^ xgmiirev56[33] ^ xgmiirev56[31] ^ xgmiirev56[29] ^ xgmiirev56[28] ^ xgmiirev56[22] ^ xgmiirev56[21] ^ xgmiirev56[19] ^ xgmiirev56[18] ^ xgmiirev56[17] ^ xgmiirev56[15] ^ xgmiirev56[11] ^ xgmiirev56[8] ^ xgmiirev56[3] ^ xgmiirev56[2] ^ c[4] ^ c[5] ^ c[7] ^ c[9] ^ c[12] ^ c[13] ^ c[14] ^ c[16] ^ c[17] ^ c[20] ^ c[24] ^ c[25] ^ c[27] ^ c[28];

newcrc8[26] <= xgmiirev56[55] ^ xgmiirev56[54] ^ xgmiirev56[52] ^ xgmiirev56[49] ^ xgmiirev56[48] ^ xgmiirev56[47] ^ xgmiirev56[44] ^ xgmiirev56[42] ^ xgmiirev56[41] ^ xgmiirev56[39] ^ xgmiirev56[38] ^ xgmiirev56[31] ^ xgmiirev56[28] ^ xgmiirev56[26] ^ xgmiirev56[25] ^ xgmiirev56[24] ^ xgmiirev56[23] ^ xgmiirev56[22] ^ xgmiirev56[20] ^ xgmiirev56[19] ^ xgmiirev56[18] ^ xgmiirev56[10] ^ xgmiirev56[6] ^ xgmiirev56[4] ^ xgmiirev56[3] ^ xgmiirev56[0] ^ c[0] ^ c[1] ^ c[2] ^ c[4] ^ c[7] ^ c[14] ^ c[15] ^ c[17] ^ c[18] ^ c[20] ^ c[23] ^ c[24] ^ c[25] ^ c[28] ^ c[30] ^ c[31];

newcrc8[27] <= xgmiirev56[55] ^ xgmiirev56[53] ^ xgmiirev56[50] ^ xgmiirev56[49] ^ xgmiirev56[48] ^ xgmiirev56[45] ^ xgmiirev56[43] ^ xgmiirev56[42] ^ xgmiirev56[40] ^ xgmiirev56[39] ^ xgmiirev56[32] ^ xgmiirev56[29] ^ xgmiirev56[27] ^ xgmiirev56[26] ^ xgmiirev56[25] ^ xgmiirev56[24] ^ xgmiirev56[23] ^ xgmiirev56[21] ^ xgmiirev56[20] ^ xgmiirev56[19] ^ xgmiirev56[11] ^ xgmiirev56[7] ^ xgmiirev56[5] ^ xgmiirev56[4] ^ xgmiirev56[1] ^ c[0] ^ c[1] ^ c[2] ^ c[3] ^ c[5] ^ c[8] ^ c[15] ^ c[16] ^ c[18] ^ c[19] ^ c[21] ^ c[24] ^ c[25] ^ c[26] ^ c[29] ^ c[31];

newcrc8[28] <= xgmiirev56[54] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[49] ^ xgmiirev56[46] ^ xgmiirev56[44] ^ xgmiirev56[43] ^ xgmiirev56[41] ^ xgmiirev56[40] ^ xgmiirev56[33] ^ xgmiirev56[30] ^ xgmiirev56[28] ^ xgmiirev56[27] ^ xgmiirev56[26] ^ xgmiirev56[25] ^ xgmiirev56[24] ^ xgmiirev56[22] ^ xgmiirev56[21] ^ xgmiirev56[20] ^ xgmiirev56[12] ^ xgmiirev56[8] ^ xgmiirev56[6] ^ xgmiirev56[5] ^ xgmiirev56[2] ^ c[0] ^ c[1] ^ c[2] ^ c[3] ^ c[4] ^ c[6] ^ c[9] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[22] ^ c[25] ^ c[26] ^ c[27] ^ c[30];

newcrc8[29] <= xgmiirev56[55] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[50] ^ xgmiirev56[47] ^ xgmiirev56[45] ^ xgmiirev56[44] ^ xgmiirev56[42] ^ xgmiirev56[41] ^ xgmiirev56[34] ^ xgmiirev56[31] ^ xgmiirev56[29] ^ xgmiirev56[28] ^ xgmiirev56[27] ^ xgmiirev56[26] ^ xgmiirev56[25] ^ xgmiirev56[23] ^ xgmiirev56[22] ^ xgmiirev56[21] ^ xgmiirev56[13] ^ xgmiirev56[9] ^ xgmiirev56[7] ^ xgmiirev56[6] ^ xgmiirev56[3] ^ c[1] ^ c[2] ^ c[3] ^ c[4] ^ c[5] ^ c[7] ^ c[10] ^ c[17] ^ c[18] ^ c[20] ^ c[21] ^ c[23] ^ c[26] ^ c[27] ^ c[28] ^ c[31];

newcrc8[30] <= xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[51] ^ xgmiirev56[48] ^ xgmiirev56[46] ^ xgmiirev56[45] ^ xgmiirev56[43] ^ xgmiirev56[42] ^ xgmiirev56[35] ^ xgmiirev56[32] ^ xgmiirev56[30] ^ xgmiirev56[29] ^ xgmiirev56[28] ^ xgmiirev56[27] ^ xgmiirev56[26] ^ xgmiirev56[24] ^ xgmiirev56[23] ^ xgmiirev56[22] ^ xgmiirev56[14] ^ xgmiirev56[10] ^ xgmiirev56[8] ^ xgmiirev56[7] ^ xgmiirev56[4] ^ c[0] ^ c[2] ^ c[3] ^ c[4] ^ c[5] ^ c[6] ^ c[8] ^ c[11] ^ c[18] ^ c[19] ^ c[21] ^ c[22] ^ c[24] ^ c[27] ^ c[28] ^ c[29];

newcrc8[31] <= xgmiirev56[54] ^ xgmiirev56[53] ^ xgmiirev56[52] ^ xgmiirev56[49] ^ xgmiirev56[47] ^ xgmiirev56[46] ^

xgmiirev56[44] ^ xgmiirev56[43] ^ xgmiirev56[36] ^ xgmiirev56[33] ^ xgmiirev56[31] ^ xgmiirev56[30] ^ xgmiirev56[29] ^ xgmiirev56[28] ^ xgmiirev56[27] ^ xgmiirev56[25] ^ xgmiirev56[24] ^ xgmiirev56[23] ^ xgmiirev56[15] ^ xgmiirev56[11] ^ xgmiirev56[9] ^ xgmiirev56[8] ^ xgmiirev56[5] ^ c[0] ^ c[1] ^ c[3] ^ c[4] ^ c[5] ^ c[6] ^ c[7] ^ c[9] ^ c[12] ^ c[19] ^ c[20] ^ c[22] ^ c[23] ^ c[25] ^ c[28] ^ c[29] ^ c[30];

checksum_buff<=1;

end

//

else if(eop_flag==0 && eop_flag_buff==0) begin

eop <= 0;

valid <=0;

empty <=0;

data <= 63'b0;

newcrc <=32'hFFFFFFFF;

newcrc8 <= 32'hFFFFFFFF;

end

if(eop_flag)begin

eop<=1;

eop_flag<=0;

eop_flag_buff<=0;

valid_flag<=0;

valid_flag_buff<=0;

end

///

if(checksum_flag==1)begin

checksum_flag<=0;

checksum_err <= !(newcrc8==32'hC704DD7B);

end

else begin

checksum_err <=0;

```
                                        end
                                        if(checksum_buff != 0)begin
                                                checksum_flag <= 1;
                                                checksum_buff<=0;
                                        end
                                end
                        end
endmodule
```

MAC TX:

```
module mac_tx(

                                input logic clk,
                                input logic [63:0] stdata,// input data
                                input logic reset,
                                input logic [31:0]c,
                                input logic sopin,
                                input logic eopin,
                                input logic valid,
                                input logic [2:0] empty,

                                output logic ready,
                                output logic [63:0]xgmiirev,
                                output logic [71:0] xgmii,
                                output logic [31:0] newcrc,
                                output logic [31:0] cc);


        logic [63:0]swapxgmiidata;
        logic eop1;
        logic eop2;
```

```
                logic eop3;
                logic valid1;
                logic [63:0] stdata1;
                logic sop;
                logic eop;


                assign sop = sopin && valid;
                assign eop = eopin && valid;
                assign ready = 1;

                always @(posedge clk)
                        begin
                                eop1 <=eop;
                                eop2 <=eop1;
                                eop3 <=eop2;
                                valid1 <= valid;
                                stdata1 <= stdata;
                        end

  integer i;
  integer g;
  integer k;

          initial begin
                newcrc<=32'hFFFFFFFF;
                xgmii <= {1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07, 1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07,1'b1,
8'h07};
          end


  always@* begin
```

```
    for (i=0;i<8;i++)begin
     for(g=0;g<8;g++)begin
       xgmiirev[g+i*8]=stdata[(7-g)+i*8];
     end
    end


    for (k=0;k<32;k++)begin
      cc[k] = ~newcrc[31-k];
    end
  end


                always @(posedge clk) begin

                                        if (reset == 1)begin
                                                newcrc<=32'hFFFFFFFF;
                                        end else if (sop)
                                                begin
                                                        xgmii <= {1'b0, 8'h55,1'b0,
8'h55,1'b0, 8'h55,1'b0, 8'hD5,1'b1, 8'hFB,1'b0, 8'h55,1'b0, 8'h55, 1'b0, 8'h55};
                                                end else if (valid1) //middle cases
                                                        begin
                                                        xgmii <= {1'b0,  stdata1[31:24],
1'b0, stdata1[23:16], 1'b0, stdata1[15:8], 1'b0, stdata1[7:0], 1'b0, stdata1[63:56], 1'b0, stdata1[55:48], 1'b0,
stdata1[47:40], 1'b0, stdata1[39:32]};
                                                        end else if (eop2)
                                                        begin
                                                                xgmii <= { 1'b1, 8'hFD,1'b1,
8'hBC,1'b1, 8'hBC, 1'b1, 8'hBC,1'b0, cc[7:0], 1'b0, cc[15:8], 1'b0, cc[23:16], 1'b0, cc[31:24]};
                                                        end else if ( eop3)
                                                                begin
                                                                        xgmii <= {1'b1,
8'h07,1'b1, 8'h07,1'b1, 8'h07, 1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07};
```

```
                                                                    newcrc
<=32'hFFFFFFFF;
                                                                    end else
                                                                        begin

        xgmii <= {1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07, 1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07,1'b1, 8'h07};
                                                                    end




                                                        if(valid)
                                                            begin
                                                                    newcrc[0]   =
xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^
xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[37] ^ xgmiirev[34] ^
xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[25] ^
xgmiirev[24] ^ xgmiirev[16] ^ xgmiirev[12] ^ xgmiirev[10] ^ xgmiirev[9] ^ xgmiirev[6] ^ xgmiirev[0] ^ c[0] ^ c[2] ^
c[5] ^ c[12] ^ c[13] ^ c[15] ^ c[16] ^ c[18] ^ c[21] ^ c[22] ^ c[23] ^ c[26] ^ c[28] ^ c[29] ^ c[31];

                                                                    newcrc[1]   =
xgmiirev[63] ^ xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[53] ^
xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[38] ^
xgmiirev[37] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[24] ^
xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[13] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[9] ^ xgmiirev[7] ^ xgmiirev[6] ^
xgmiirev[1] ^ xgmiirev[0] ^ c[1] ^ c[2] ^ c[3] ^ c[5] ^ c[6] ^ c[12] ^ c[14] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[24]
^ c[26] ^ c[27] ^ c[28] ^ c[30] ^ c[31];

                                                                    newcrc[2]   =
xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^
xgmiirev[44] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[32] ^
xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[16] ^
xgmiirev[14] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[2] ^ xgmiirev[1] ^
xgmiirev[0] ^ c[0] ^ c[3] ^ c[4] ^ c[5] ^ c[6] ^ c[7] ^ c[12] ^ c[19] ^ c[20] ^ c[21] ^ c[23] ^ c[25] ^ c[26] ^ c[27];

                                                                    newcrc[3]   =
xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^
xgmiirev[45] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[33] ^
xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[27] ^ xgmiirev[25] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[17] ^
xgmiirev[15] ^ xgmiirev[14] ^ xgmiirev[10] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[3] ^ xgmiirev[2] ^
xgmiirev[1] ^ c[0] ^ c[1] ^ c[4] ^ c[5] ^ c[6] ^ c[7] ^ c[8] ^ c[13] ^ c[20] ^ c[21] ^ c[22] ^ c[24] ^ c[26] ^ c[27] ^ c[28];

                                                                    newcrc[4]   =
xgmiirev[63] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[47] ^
xgmiirev[46] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[38] ^
xgmiirev[33] ^ xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[20] ^
xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[15] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[6] ^ xgmiirev[4] ^
```

xgmiirev[3] ^ xgmiirev[2] ^ xgmiirev[0] ^ c[1] ^ c[6] ^ c[7] ^ c[8] ^ c[9] ^ c[12] ^ c[13] ^ c[14] ^ c[15] ^ c[16] ^ c[18] ^ c[25] ^ c[26] ^ c[27] ^ c[31];

newcrc[5] = xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[37] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[24] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[13] ^ xgmiirev[10] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[5] ^ c[7] ^ c[8] ^ c[9] ^ c[10] ^ c[12] ^ c[14] ^ c[17] ^ c[18] ^ c[19] ^ c[21] ^ c[22] ^ c[23] ^ c[27] ^ c[29] ^ c[31];

newcrc[6] = xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[38] ^ xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[25] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[14] ^ xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[2] ^ xgmiirev[1] ^ c[6] ^ c[8] ^ c[9] ^ c[10] ^ c[11] ^ c[13] ^ c[15] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[23] ^ c[24] ^ c[28] ^ c[30];

newcrc[7] = xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[54] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[39] ^ xgmiirev[37] ^ xgmiirev[34] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[10] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[5] ^ xgmiirev[3] ^ xgmiirev[2] ^ xgmiirev[0] ^ c[0] ^ c[2] ^ c[5] ^ c[7] ^ c[9] ^ c[10] ^ c[11] ^ c[13] ^ c[14] ^ c[15] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[24] ^ c[25] ^ c[26] ^ c[28];

newcrc[8] = xgmiirev[63] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[54] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[46] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[40] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[23] ^ xgmiirev[22] ^ xgmiirev[17] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[10] ^ xgmiirev[8] ^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[0] ^ c[1] ^ c[2] ^ c[3] ^ c[5] ^ c[6] ^ c[8] ^ c[10] ^ c[11] ^ c[13] ^ c[14] ^ c[18] ^ c[19] ^ c[20] ^ c[22] ^ c[25] ^ c[27] ^ c[28] ^ c[31];

newcrc[9] = xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[55] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[41] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[18] ^ xgmiirev[13] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[9] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[2] ^ xgmiirev[1] ^ c[0] ^ c[1] ^ c[2] ^ c[3] ^ c[4] ^ c[6] ^ c[7] ^ c[9] ^ c[11] ^ c[12] ^ c[14] ^ c[15] ^ c[19] ^ c[20] ^ c[21] ^ c[23] ^ c[26] ^ c[28] ^ c[29];

newcrc[10] = xgmiirev[63] ^ xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[52] ^ xgmiirev[50] ^ xgmiirev[42] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[19] ^ xgmiirev[16] ^ xgmiirev[14] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[5] ^ xgmiirev[3] ^ xgmiirev[2] ^ xgmiirev[0] ^ c[0] ^ c[1] ^ c[3] ^ c[4] ^ c[7] ^ c[8] ^ c[10] ^ c[18] ^ c[20] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[28] ^ c[30] ^ c[31];

newcrc[11] = xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[36] ^ xgmiirev[33] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[20] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[14] ^

xgmiirev[12] ^ xgmiirev[9] ^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[1] ^ c[4] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[13] ^ c[15] ^ c[16] ^ c[18] ^ c[19] ^ c[22] ^ c[23] ^ c[24] ^ c[25] ^ c[26] ^ c[27];

newcrc[12] = xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[27] ^ xgmiirev[24] ^ xgmiirev[21] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[15] ^ xgmiirev[13] ^ xgmiirev[12] ^ xgmiirev[9] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[2] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[9] ^ c[10] ^ c[14] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^ c[20] ^ c[21] ^ c[22] ^ c[24] ^ c[25] ^ c[27] ^ c[29] ^ c[31];

newcrc[13] = xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[25] ^ xgmiirev[22] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[16] ^ xgmiirev[14] ^ xgmiirev[13] ^ xgmiirev[10] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[3] ^ xgmiirev[2] ^ xgmiirev[1] ^ c[0] ^ c[10] ^ c[11] ^ c[15] ^ c[16] ^ c[18] ^ c[19] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[25] ^ c[26] ^ c[28] ^ c[30];

newcrc[14] = xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[26] ^ xgmiirev[23] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[17] ^ xgmiirev[15] ^ xgmiirev[14] ^ xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[2] ^ c[0] ^ c[1] ^ c[11] ^ c[12] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[24] ^ c[26] ^ c[27] ^ c[29] ^ c[31];

newcrc[15] = xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[30] ^ xgmiirev[27] ^ xgmiirev[24] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[18] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[12] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[3] ^ c[1] ^ c[2] ^ c[12] ^ c[13] ^ c[17] ^ c[18] ^ c[20] ^ c[21] ^ c[22] ^ c[23] ^ c[24] ^ c[25] ^ c[27] ^ c[28] ^ c[30];

newcrc[16] = xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[51] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[37] ^ xgmiirev[35] ^ xgmiirev[32] ^ xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[19] ^ xgmiirev[17] ^ xgmiirev[13] ^ xgmiirev[12] ^ xgmiirev[8] ^ xgmiirev[5] ^ xgmiirev[4] ^ xgmiirev[0] ^ c[0] ^ c[3] ^ c[5] ^ c[12] ^ c[14] ^ c[15] ^ c[16] ^ c[19] ^ c[24] ^ c[25];

newcrc[17] = xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[52] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[38] ^ xgmiirev[36] ^ xgmiirev[33] ^ xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[27] ^ xgmiirev[25] ^ xgmiirev[23] ^ xgmiirev[22] ^ xgmiirev[20] ^ xgmiirev[18] ^ xgmiirev[14] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[6] ^ xgmiirev[5] ^ xgmiirev[1] ^ c[1] ^ c[4] ^ c[6] ^ c[13] ^ c[15] ^ c[16] ^ c[17] ^ c[20] ^ c[25] ^ c[26];

newcrc[18] = xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[53] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[46] ^ xgmiirev[39] ^ xgmiirev[37] ^ xgmiirev[34] ^ xgmiirev[32] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[21] ^ xgmiirev[19] ^ xgmiirev[15] ^ xgmiirev[14] ^ xgmiirev[10] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[2] ^ c[0] ^ c[2] ^ c[5] ^ c[7] ^ c[14] ^ c[16] ^ c[17] ^ c[18] ^ c[21] ^ c[26] ^ c[27];

newcrc[19] = xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[54] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[47] ^

xgmiirev[40] ^ xgmiirev[38] ^ xgmiirev[35] ^ xgmiirev[33] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[22] ^ xgmiirev[20] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[3] ^ c[0] ^ c[1] ^ c[3] ^ c[6] ^ c[8] ^ c[15] ^ c[17] ^ c[18] ^ c[19] ^ c[22] ^ c[27] ^ c[28];

newcrc[20] = xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[55] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[41] ^ xgmiirev[39] ^ xgmiirev[36] ^ xgmiirev[34] ^ xgmiirev[33] ^ xgmiirev[30] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[25] ^ xgmiirev[23] ^ xgmiirev[21] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[12] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[4] ^ c[1] ^ c[2] ^ c[4] ^ c[7] ^ c[9] ^ c[16] ^ c[18] ^ c[19] ^ c[20] ^ c[23] ^ c[28] ^ c[29];

newcrc[21] = xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[56] ^ xgmiirev[53] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[49] ^ xgmiirev[42] ^ xgmiirev[40] ^ xgmiirev[37] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[22] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[13] ^ xgmiirev[10] ^ xgmiirev[9] ^ xgmiirev[5] ^ c[2] ^ c[3] ^ c[5] ^ c[8] ^ c[10] ^ c[17] ^ c[19] ^ c[20] ^ c[21] ^ c[24] ^ c[29] ^ c[30];

newcrc[22] = xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[52] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[41] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[16] ^ xgmiirev[14] ^ xgmiirev[12] ^ xgmiirev[11] ^ xgmiirev[9] ^ xgmiirev[0] ^ c[2] ^ c[3] ^ c[4] ^ c[5] ^ c[6] ^ c[9] ^ c[11] ^ c[12] ^ c[13] ^ c[15] ^ c[16] ^ c[20] ^ c[23] ^ c[25] ^ c[26] ^ c[28] ^ c[29] ^ c[30];

newcrc[23] = xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[42] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[34] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[15] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[6] ^ xgmiirev[1] ^ xgmiirev[0] ^ c[2] ^ c[3] ^ c[4] ^ c[6] ^ c[7] ^ c[10] ^ c[14] ^ c[15] ^ c[17] ^ c[18] ^ c[22] ^ c[23] ^ c[24] ^ c[27] ^ c[28] ^ c[30];

newcrc[24] = xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[55] ^ xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[43] ^ xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[35] ^ xgmiirev[32] ^ xgmiirev[30] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[16] ^ xgmiirev[14] ^ xgmiirev[10] ^ xgmiirev[7] ^ xgmiirev[2] ^ xgmiirev[1] ^ c[0] ^ c[3] ^ c[4] ^ c[5] ^ c[7] ^ c[8] ^ c[11] ^ c[15] ^ c[16] ^ c[18] ^ c[19] ^ c[23] ^ c[24] ^ c[25] ^ c[28] ^ c[29] ^ c[31];

newcrc[25] = xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[52] ^ xgmiirev[51] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[44] ^ xgmiirev[41] ^ xgmiirev[40] ^ xgmiirev[38] ^ xgmiirev[37] ^ xgmiirev[36] ^ xgmiirev[33] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[17] ^ xgmiirev[15] ^ xgmiirev[11] ^ xgmiirev[8] ^ xgmiirev[3] ^ xgmiirev[2] ^ c[1] ^ c[4] ^ c[5] ^ c[6] ^ c[8] ^ c[9] ^ c[12] ^ c[16] ^ c[17] ^ c[19] ^ c[20] ^ c[24] ^ c[25] ^ c[26] ^ c[29] ^ c[30];

newcrc[26] = xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[54] ^ xgmiirev[52] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[47] ^ xgmiirev[44] ^ xgmiirev[42] ^ xgmiirev[41] ^ xgmiirev[39] ^ xgmiirev[38] ^ xgmiirev[31] ^ xgmiirev[28] ^ xgmiirev[26] ^ xgmiirev[25] ^ xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[22] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[18] ^ xgmiirev[10] ^ xgmiirev[6] ^ xgmiirev[4] ^ xgmiirev[3] ^ xgmiirev[0] ^ c[6] ^ c[7] ^ c[9] ^ c[10] ^ c[12] ^ c[15] ^ c[16] ^ c[17] ^ c[20] ^ c[22] ^ c[23] ^ c[25] ^ c[27] ^ c[28] ^ c[29] ^ c[30];

```
                                                                    newcrc[27] =
xgmiirev[63] ^ xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[55] ^
xgmiirev[53] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[48] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^
xgmiirev[40] ^ xgmiirev[39] ^ xgmiirev[32] ^ xgmiirev[29] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[25] ^
xgmiirev[24] ^ xgmiirev[23] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[19] ^ xgmiirev[11] ^ xgmiirev[7] ^ xgmiirev[5]
^ xgmiirev[4] ^ xgmiirev[1] ^ c[0] ^ c[7] ^ c[8] ^ c[10] ^ c[11] ^ c[13] ^ c[16] ^ c[17] ^ c[18] ^ c[21] ^ c[23] ^ c[24] ^
c[26] ^ c[28] ^ c[29] ^ c[30] ^ c[31];

                                                                    newcrc[28] =
xgmiirev[63] ^ xgmiirev[62] ^ xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[56] ^ xgmiirev[54] ^
xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[49] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[41] ^
xgmiirev[40] ^ xgmiirev[33] ^ xgmiirev[30] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[25] ^
xgmiirev[24] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[20] ^ xgmiirev[12] ^ xgmiirev[8] ^ xgmiirev[6] ^ xgmiirev[5] ^
xgmiirev[2] ^ c[1] ^ c[8] ^ c[9] ^ c[11] ^ c[12] ^ c[14] ^ c[17] ^ c[18] ^ c[19] ^ c[22] ^ c[24] ^ c[25] ^ c[27] ^ c[29] ^
c[30] ^ c[31];

                                                                    newcrc[29] =
xgmiirev[63] ^ xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[58] ^ xgmiirev[57] ^ xgmiirev[55] ^ xgmiirev[52] ^
xgmiirev[51] ^ xgmiirev[50] ^ xgmiirev[47] ^ xgmiirev[45] ^ xgmiirev[44] ^ xgmiirev[42] ^ xgmiirev[41] ^
xgmiirev[34] ^ xgmiirev[31] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[25] ^
xgmiirev[23] ^ xgmiirev[22] ^ xgmiirev[21] ^ xgmiirev[13] ^ xgmiirev[9] ^ xgmiirev[7] ^ xgmiirev[6] ^ xgmiirev[3] ^
c[2] ^ c[9] ^ c[10] ^ c[12] ^ c[13] ^ c[15] ^ c[18] ^ c[19] ^ c[20] ^ c[23] ^ c[25] ^ c[26] ^ c[28] ^ c[30] ^ c[31];

                                                                    newcrc[30] =
xgmiirev[63] ^ xgmiirev[61] ^ xgmiirev[59] ^ xgmiirev[58] ^ xgmiirev[56] ^ xgmiirev[53] ^ xgmiirev[52] ^
xgmiirev[51] ^ xgmiirev[48] ^ xgmiirev[46] ^ xgmiirev[45] ^ xgmiirev[43] ^ xgmiirev[42] ^ xgmiirev[35] ^
xgmiirev[32] ^ xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[26] ^ xgmiirev[24] ^
xgmiirev[23] ^ xgmiirev[22] ^ xgmiirev[14] ^ xgmiirev[10] ^ xgmiirev[8] ^ xgmiirev[7] ^ xgmiirev[4] ^ c[0] ^ c[3] ^
c[10] ^ c[11] ^ c[13] ^ c[14] ^ c[16] ^ c[19] ^ c[20] ^ c[21] ^ c[24] ^ c[26] ^ c[27] ^ c[29] ^ c[31];

                                                                    newcrc[31] =
xgmiirev[62] ^ xgmiirev[60] ^ xgmiirev[59] ^ xgmiirev[57] ^ xgmiirev[54] ^ xgmiirev[53] ^ xgmiirev[52] ^
xgmiirev[49] ^ xgmiirev[47] ^ xgmiirev[46] ^ xgmiirev[44] ^ xgmiirev[43] ^ xgmiirev[36] ^ xgmiirev[33] ^
xgmiirev[31] ^ xgmiirev[30] ^ xgmiirev[29] ^ xgmiirev[28] ^ xgmiirev[27] ^ xgmiirev[25] ^ xgmiirev[24] ^
xgmiirev[23] ^ xgmiirev[15] ^ xgmiirev[11] ^ xgmiirev[9] ^ xgmiirev[8] ^ xgmiirev[5] ^ c[1] ^ c[4] ^ c[11] ^ c[12] ^
c[14] ^ c[15] ^ c[17] ^ c[20] ^ c[21] ^ c[22] ^ c[25] ^ c[27] ^ c[28] ^ c[30];

                                                                    end

                                        end


endmodule
```

TX FIFO:

```
module txbuffer(

                                        input logic clk,

                                        input logic reset,

                                        input  logic  [63:0] stdata,            //  stream_src.data
```

```
                                    input  logic valid,              //          .valid
                                    input  logic sopin,       //          .startofpacket
                                    input  logic eopin,       //          .endofpacket
                                    input  logic [2:0] emptyin,             //          .empty

                                    output  logic  [63:0] databuff,
        output  logic validout,
        output  logic readybuff,
                                    output  logic sopout,
        output  logic eopout,
        output  logic [2:0]emptybuff,
                                    output  logic readytx             //          .ready
);

                                    logic [7:0] cnt;
                                    logic [7:0] cnt2;
                                    logic [0:190][63:0] fifo;
                                    logic flag;
                                    logic sendflag;
                                    logic validbuff;
                                    logic sopbuff;
                                    logic eopbuff;

                                    assign readytx  = ! sendflag;




                always @(posedge clk)
                                    begin
                                                        validout  <=validbuff;
            sopout    <=sopbuff;
```

```
        eopout    <=eopbuff;
                                end


                        initial
                                begin
                                                databuff <= 64'b0;
                                                cnt <= 8'b0;
                                                cnt2<=8'b0;
                                                sendflag <= 0;
                                end

                always @(posedge clk)
                                begin
                                        if (reset == 1)
                                                begin
                                                        databuff <= 64'b0;
                                                        cnt <= 8'b0;
                                                        cnt2<=8'b0;
                                                        sendflag <= 0;
                                           end
                                        else
                                                begin
                                                                ////////////read from dc
fifo//////////////
                                                                if (valid && readytx)
                                                                                begin

        fifo[cnt][63:0] <= stdata;
                                                                                                cnt <=
cnt +1;
                                                                                                flag <=1;
                                                                                end
```

```
                                                    if (eopin && valid)
                                                            begin
                                                                    sendflag
<= 1;
                                                    end
                                                    /////////////////write to
mac///////////////
                                                    else if (sendflag)
                                                            begin
                                                                    if (flag)

        begin

        sopbuff <=1;

        flag <=0;

        databuff <= fifo[cnt2][63:0];

        cnt2 <= cnt2+1;

        cnt <= cnt -1;

        validbuff <=1;
                                                                    end


                                                                    if (cnt ==0)

        begin

                eopbuff  <=0;

                sendflag <=0;

                validbuff <=0;

                databuff <= fifo[cnt2][63:0];
```

```
                                                                    end
                                                                    else if (cnt ==8'b1)
        begin
        eopbuff<=1;
        databuff <= fifo[cnt2][63:0];
        cnt2 <= cnt2+1;
        cnt <= cnt -1;
                                                                    end
                                                                    else if (flag ==0)
        begin
        databuff <= fifo[cnt2][63:0];
        cnt2 <= cnt2+1;
        cnt <= cnt -1;
        sopbuff <=0;
                                                                        end
                                                            else
                                                                begin
                                                                    databuff <= 64'b0;
                                                                    cnt2 <= 8'b0;
                                                                end
                                                        end
                                    end
                            end
endmodule
```

AVALON MM to ST:

```
// avl_st_bytes_packet.v

// This file was auto-generated as a prototype implementation of a module
// created in component editor.  It ties off all outputs to ground and
// ignores all inputs.  It needs to be edited to make it do something
// useful.
//
// This file will not be automatically regenerated.  You should check it in
// to your version control system if you want to keep it.

module avl_st_bytes_packet (
                input  logic       clk,            //       clk.clk
                input  logic       reset,          //     reset.reset


                //Avalon ST interface (sink) --------- bytes in
                output logic       in_ready,
                input  logic       in_valid,       //          .valid
                input  logic [31:0] in_data,       //          .data
                /*
                //Avalon ST interface (src) --------- bytes out

                input   logic      out_ready,      //  bytes_out.ready
                output  logic      out_valid,      //          .valid
                output  logic [31:0] out_data,     //          .data

                /Avalon ST interface (sink) --------- packet in
                output logic       sink_ready,     // packets_in.ready
                input  logic       sink_valid,     //          .valid
                input  logic [63:0] sink_data,     //          .data
```

```
        input  logic       sink_sop,         //         .startofpacket
        input  logic       sink_eop,         //         .endofpacket
        input  logic  [2:0] sink_empty,       //         .empty
        */
        //Avalon ST interface (src) --------- packet out

        input   logic       src_ready,       // packets_out.ready
        output  logic       src_valid,       //         .valid
        output  logic [63:0] src_data,       //         .data
        output  logic       src_sop,         //         .startofpacket
        output  logic       src_eop,         //         .endofpacket
        output  logic  [2:0] src_empty,      //         .empty

        //Avalon MM interface (slave) --------- used for module controlling

        input  logic       write,          // control_mm.write
        input  logic [7:0]  writedata,      //         .writedata
        input  logic       read,           //         .read
        output  logic [7:0]  readdata,      //         .readdata
        input  logic [3:0]  address,        //         .address
        input  logic       cs              //         .chipselect
);

        //=======================================================
        //  REG/logic declarations
        //=======================================================
        logic[31:0] in_data_reverse;
        logic                       in_frame;
        logic[31:0]                 data_reg;
        logic                       sop_reg;
        logic[1:0]                  eop_reg;
        logic[2:0]                  empty_reg;
```

```
                logic                   esc_reg;
                logic                   isFull;
                logic   ready_reg;
                logic[7:0]
control_reg1,control_reg2,control_reg3,control_reg4,control_reg5,control_reg6,control_reg7;
                logic[31:0] pktincnt,pktoutcnt;



  /*logic                 in_frame_out;
                logic[2:0]                      isFull_out;
                logic[0:1][31:0]    data_reg_out;
                logic                   esc_reg_out;*/
                //=========================================================
                //  Structural coding
                //=========================================================
  assign in_ready = ready_reg;
        assign in_data_reverse= {in_data[7-:8],in_data[15-:8],in_data[23-:8],in_data[31-:8]};


                        always_ff@(posedge clk) begin
                                if (reset) begin
                                        ready_reg <= 0;
                                        control_reg1 <= 0;
                                        control_reg2 <= 0;
                                        control_reg3 <= 0;
                                        control_reg4 <= 0;
                                        control_reg5 <= 0;
                                        control_reg6 <= 0;
                                        control_reg7 <= 0;
                                end else begin
                                  if (cs) begin
                                        if (write) begin
                                                case (address)
```

```
                                        3'h0 : ready_reg <= writedata[0];
                                        3'h1 : control_reg1 <= writedata;
                                        3'h2 : control_reg2 <= writedata;
                                        3'h3 : control_reg3 <= writedata;
                                        3'h4 : control_reg4 <= writedata;
                                        3'h5 : control_reg5 <= writedata;
                                        3'h6 : control_reg6 <= writedata;
                                        3'h7 : control_reg7 <= writedata;
                                        default;//do nothing;
                                endcase
                        end else if (read) begin
                                case (address)
                                        3'h0 : readdata <= pktincnt[7:0];
                                        3'h1 : readdata <= pktincnt[15:8];
                                        3'h2 : readdata <= pktincnt[23:16];
                                        3'h3 : readdata <= pktincnt[31:24];
                                        3'h4 : readdata <= pktoutcnt[7:0];
                                        3'h5 : readdata <= pktoutcnt[15:8];
                                        3'h6 : readdata <= pktoutcnt[23:16];
                                        3'h7 : readdata <= pktoutcnt[31:24];
                                        default;//do nothing;
                                endcase
                        end
                end
////////////////////////////////////From Bytes to
Packets/////////////////////////////////////////////////////////////////
                        if (in_valid) begin
                                case (in_data_reverse)
                                  32'h7a7a7a7a : begin       //0x7a7a7a7a Indicates Start Of
Packet;
                                        src_valid <= 0;
                                                src_eop <= 0;
                                        sop_reg <= 1;
```

```
                eop_reg <= 0;
                in_frame <= 1;
                                                        end
                                                    32'h7b7b7b7b : begin       //0x7b7b7b7b Indicates End Of
Packet;
                                                      src_valid <= 0;
                                                      sop_reg <= 0;
                in_frame <= 0;
                eop_reg <= 2'b01;
                                                        end
                                                    32'h7d7d7d7d : begin       //0x7d7d7d7d Indicates Escape,
Data after it should XOR with 0x20202020;
                                                      src_valid <= 0;
                                                      if (in_frame) esc_reg <= 1;
                                                        end
                                                    default: begin
                                                       if (isFull) begin
                                                            if (in_frame) begin        //In Data Frame:
Data Out
                                                                src_valid <= 1;
                                                                isFull <= 0; //isFull cleared
                                                                if (esc_reg) begin     //A Escape Flag
Was Triggered
                                                                  src_data <=
{data_reg,in_data_reverse ^ 32'h20202020};
                                                                  esc_reg <= 0;
                                                                end else src_data <=
{data_reg,in_data_reverse};
                                                                src_sop <= sop_reg;
                                                                sop_reg <= 0;
                                                            end else if (eop_reg[0]) begin  //In End Of
Packet State: Data Mask
                                                                case (in_data_reverse)
                                                                  32'hff000000: empty_reg <= 3'h3;
                                                                  32'hffff0000: empty_reg <= 3'h2;
```

```
                                        32'hffffff00: empty_reg <= 3'h1;
                                        32'hffffffff: empty_reg <= 3'h0;
                                        default: empty_reg <= 0;
                                    endcase
                                    src_eop <= 1;
                                    eop_reg <= 2'b10;
                                src_valid <= 0;
                            end else if (eop_reg[1]) begin //In End Of
Packet State: Final Data Out With Empty Signal
                                    src_valid <= 1;
                                    isFull <= 0;  //isFull cleared
                                    src_sop <= 0;
                                    src_eop <= 1;
                                    if (esc_reg) begin     //A Escape Flag
Was Triggered
                                      src_data <= {in_data_reverse ^
32'h20202020,data_reg};
                                      esc_reg <= 0;
                                    end else src_data <=
{data_reg,in_data_reverse};
                                    src_empty <= empty_reg;
                                    empty_reg <= 0;
                                    eop_reg <= 2'b00;
                                end else begin
                                    src_eop <= 0;
                                    src_sop <= 0;
                    src_valid <= 0;
                                end
                            end else begin
                                if (in_frame) begin        //In Data Frame:
No Data Out
                                    src_valid <= 0;
                                    isFull <= 1; //isFull set
```

```
                                        if (esc_reg) begin     //A Escape Flag
Was Triggered
                                          data_reg <= in_data_reverse ^
32'h20202020;
                                          esc_reg <= 0;
                                        end else data_reg <=
in_data_reverse;
                                        src_sop <= sop_reg;
                                      end else if (eop_reg[0]) begin  //In End Of
Packet State: Data Mask
                                        src_valid <= 0;
                                        case (in_data_reverse)
                                          32'hff000000: empty_reg <= 3'h7;
                                          32'hffff0000: empty_reg <= 3'h6;
                                          32'hffffff00: empty_reg <= 3'h5;
                                          32'hffffffff: empty_reg <= 3'h4;
                                          default: empty_reg <= 3'h4;
                                        endcase
                                        src_eop <= 1;
                                        eop_reg <= 2'b10;
                                      end else if (eop_reg[1]) begin //In End Of
Packet State: Final Data Out With Empty Signal
                                        src_valid <= 1;
                                        isFull <= 0;  //isFull cleared
                                        src_sop <= 0;
                                        src_eop <= 1;
                                        if (esc_reg) begin     //A Escape Flag
Was Triggered
                                          src_data <=
{32'h0,in_data_reverse ^ 32'h20202020};
                                          esc_reg <= 0;
                                        end else src_data <=
{in_data_reverse,32'h0};
                                        src_empty <= empty_reg;
                                        empty_reg <= 0;
```

```
                                                    eop_reg <= 2'b00;
                                                end else begin
                                                    src_eop <= 0;
                                                    src_sop <= 0;
                                    src_valid <= 0;
                                                end
                                            end
                                        end

                                            endcase
                                    end else begin
                                            src_eop <= 0;
                                            src_sop <= 0;
                                      src_valid <= 0;
                                            src_empty <= 0;
                                            src_data <= 0;
                                    end
////////////////////////////////////From Packets to
Bytes/////////////////////////////////////////////////////////////////
                                /*if (out_ready) begin
                                  out_valid <= 1;
                                  if (sink_valid) begin
                                        if (sink_sop) begin
                                          in_frame_out <= 1;
                                          isFull_out <= 0;
                                        end
                                        if (isFull_out==) begin
                                          out_data <= data_reg_out;
                                          isFull_out <= 0;
                                          sink_ready <= 1;
                                        end else begin
                                          sink_ready <= 0;
```

```
                                        data_reg_out <= sink_data;
                                        if
(sink_data[63:32]==32'h00000000||sink_data[63:32]==32'h7a7a7a7a||sink_data[63:32]==32'h7b7b7b7b||sink_d
ata[63:32]==32'h7d7d7d7d) begin
                                            out_data <= 32'7d7d7d7d;
                                            data_reg_out[0] <= sink_data[63:32] ^ 32'h20202020;
                                            data_reg_out[1] <= sink_data[31:0];
                                        else begin


                                    end
                                end else begin


                                end
                            end else begin
                                out_valid <= 0;
                                out_data <= 0;
                            end*/
                        end
                end


endmodule
```

AVALON ST to MM:

```
// avl_st_bytes_packet.v


// This file was auto-generated as a prototype implementation of a module
// created in component editor.  It ties off all outputs to ground and
// ignores all inputs.  It needs to be edited to make it do something
// useful.
//
// This file will not be automatically regenerated.  You should check it in
```

```
// to your version control system if you want to keep it.

module avl_st_packet_bytes (
                input  logic       clk,            //       clk.clk
                input  logic       reset,          //     reset.reset



                //Avalon ST interface (src) --------- bytes out

                input   logic       out_ready,       //  bytes_out.ready
                output  logic       out_valid,       //          .valid
                output  logic [31:0] out_data,       //          .data

                //Avalon ST interface (sink) --------- packet in
                output logic       sink_ready,       // packets_in.ready
                input  logic       sink_valid,       //          .valid
                input  logic [63:0] sink_data,       //          .data
                input  logic       sink_sop,         //          .startofpacket
                input  logic       sink_eop,         //          .endofpacket
                input  logic  [2:0] sink_empty,      //          .empty



                //Avalon MM interface (slave) --------- used for module controlling

                input  logic       write,          // control_mm.write
                input  logic [7:0]  writedata,      //          .writedata
                input  logic       read,           //          .read
                output  logic [7:0]  readdata,      //          .readdata
                input  logic [3:0]  address,        //          .address
                input  logic       cs              //          .chipselect
        );
```

```
                //=======================================================
                //  REG/logic declarations
                //=======================================================
                logic[31:0] sink_data_rh;
                logic[31:0] sink_data_rl;
                logic[0:4][31:0] data_reg;
                logic                           sop_reg;
                logic[1:0]                  esc_need;
                logic[2:0]                  esc_proc;
                logic                                       proc_ready;



                //=======================================================
                //  Structural coding
                //=======================================================
    assign sink_data_rl= {sink_data[7-:8],sink_data[15-:8],sink_data[23-:8],sink_data[31-:8]};
    assign sink_data_rh= {sink_data[39-:8],sink_data[47-:8],sink_data[55-:8],sink_data[63-:8]};
    assign esc_need[1] =
sink_data_rh==32'h7a7a7a7a||sink_data_rh==32'h7b7b7b7b||sink_data_rh==32'h7d7d7d7d||sink_data_rh==32'
h7e7e7e7e;
    assign esc_need[0] =
sink_data_rl==32'h7a7a7a7a||sink_data_rl==32'h7b7b7b7b||sink_data_rl==32'h7d7d7d7d||sink_data_rl==32'h7
e7e7e7e;
    //assign sink_ready = proc_ready & out_ready;
    assign sink_ready = proc_ready;


  always_ff@(posedge clk) begin
    if (reset) begin
      data_reg[0] <= 32'h7e7e7e7e;
                            data_reg[1] <= 32'h7e7e7e7e;
                            data_reg[2] <= 32'h7e7e7e7e;
                            data_reg[3] <= 32'h7e7e7e7e;
      out_data <= 32'h7e7e7e7e;  // The default output when no income data is 0x7e7e7e7e
```

```
        proc_ready <= 1;
        esc_proc <= 0;
    end
                else
                begin
                        if (cs)
                        begin
                                if (write)
                                begin
                                        case (address)
                                                default;//do nothing;
                                        endcase
                                end
                                else if (read)
                                begin
                                        case (address)
                                                default;//do nothing;
                                        endcase
                                end
                         end
////////////////////////////////////From Packets to
Bytes//////////////////////////////////////////////////////////////////
                        if (out_ready) begin
                                out_valid <= 1;
                                        if (esc_proc==0)
                                        begin
                                                if (sink_valid)
                                                        begin
                                                                ///////////////// recieve new data
(64bits), generate code, buffer ///////////////////////
                                                                        proc_ready <= 0;
                                                                ///////////////// if
startofpacket///////////////
```

```
                                                                    if (sink_sop)
                                                                    begin
                                                                   out_data <= 32'h7a7a7a7a;
                                                                        case (esc_need)
                                                                            2'b00:
begin
        data_reg[0]<= sink_data_rl;
        data_reg[1]<= sink_data_rh;
        esc_proc <= 3'h2;
                                                                            end
        2'b10:begin
        data_reg[0]<= sink_data_rl;
        data_reg[1]<= sink_data_rh^ 32'h20202020;
        data_reg[2]<= 32'h7d7d7d7d;
        esc_proc <= 3'h3;
                                                                            end
        2'b01:begin
        data_reg[0]<= sink_data_rl^ 32'h20202020;
        data_reg[1]<= 32'h7d7d7d7d;
        data_reg[2]<= sink_data_rh;
        esc_proc <= 3'h3;
                                                                            end
        2'b11:begin
```

```
            data_reg[0]<= sink_data_rl^ 32'h20202020;
            data_reg[1]<= 32'h7d7d7d7d;
            data_reg[2]<= sink_data_rh^ 32'h20202020;
            data_reg[3]<= 32'h7d7d7d7d;
            esc_proc <= 3'h4;
                                                                    end
                                                                    endcase
                                                            end
                                                        /////////////// if
endofpacket/////////////////
                                                            else if (sink_eop)
                                                            begin
                                                            case (sink_empty[2])
                                                            1'b0: begin
                                                                    case (esc_need)
            2'b00:begin
            data_reg[0]<= sink_data_rl;
            data_reg[1]<= {30'h0,sink_empty[1:0]};
            data_reg[2]<= 32'h7b7b7b7b;
            out_data <= sink_data_rh;
            esc_proc <= 3'h3;
                                                                    end
            2'b10:begin
            data_reg[0]<= sink_data_rl;
```

```
data_reg[1]<= {30'h0,sink_empty[1:0]};
data_reg[2]<= 32'h7b7b7b7b;
data_reg[3]<= sink_data_rh^ 32'h20202020;
out_data<= 32'h7d7d7d7d;
esc_proc <= 3'h4;
                                                                    end
2'b01:begin
data_reg[0]<= sink_data_rl^ 32'h20202020;
data_reg[1]<= 32'h7d7d7d7d;
data_reg[2]<= {30'h0,sink_empty[1:0]};
data_reg[3]<= 32'h7b7b7b7b;
out_data<= sink_data_rh;
esc_proc <= 3'h4;
                                                                    end
2'b11:begin
data_reg[0]<= sink_data_rl^ 32'h20202020;
data_reg[1]<= 32'h7d7d7d7d;
data_reg[2]<= {30'h0,sink_empty[1:0]};
data_reg[3]<= 32'h7b7b7b7b;
data_reg[4]<= sink_data_rh^ 32'h20202020;
```

```
            out_data<= 32'h7d7d7d7d;
            esc_proc <= 3'h5;
                                                                    end
                                                                endcase
                                                            end
                                                            1'b1: begin
                                                                case (esc_need)
                                                                    2'b00:
begin
            data_reg[0]<= sink_data_rh;
            data_reg[1]<= {30'h0,sink_empty[1:0]};
            out_data<= 32'h7b7b7b7b;
            esc_proc <= 3'h2;
                                                                    end
            2'b10:begin
            data_reg[0]<= sink_data_rh^ 32'h20202020;
            data_reg[1]<= 32'h7d7d7d7d;
            data_reg[2]<= {30'h0,sink_empty[1:0]};
            out_data<= 32'h7b7b7b7b;
            esc_proc <= 3'h3;
                                                                    end
                                                                endcase
                                                            end
                                                            endcase
                                                            end
```

```
                                                    /////////////// if in frame
//////////////////
                                                        else
                                                        begin
                                                        case (esc_need)
                                                            2'b00: begin
            data_reg[0]<= sink_data_rl;
                                                                out_data
<= sink_data_rh;
                                                                esc_proc
<= 3'h1;
                                                            end
                                                            2'b10:begin
            data_reg[0]<= sink_data_rl;
            data_reg[1]<= sink_data_rh;
                                                                out_data
<= 32'h7d7d7d7d;
                                                                esc_proc
<= 3'h2;
                                                            end
                                                            2'b01:begin
            data_reg[0]<= sink_data_rl;
            data_reg[1]<= 32'h7d7d7d7d;
                                                                out_data
<= sink_data_rh;
                                                                esc_proc
<= 3'h2;
                                                            end
                                                            2'b11:begin
            data_reg[0]<= sink_data_rl;
            data_reg[1]<= 32'h7d7d7d7d;
```

```
            data_reg[2]<= sink_data_rh;
                                                                                    out_data
<= 32'h7d7d7d7d;
                                                                                    esc_proc
<= 3'h3;
                                                                                        end
                                                                                    endcase
                                                                                    end
                                                                            end
                                                                    else
                                                                            begin
                                                                                    out_data <=
32'h7e7e7e7e;
                                                                                    data_reg[0] <=
32'h7e7e7e7e;
                                                                                    data_reg[1] <=
32'h7e7e7e7e;
                                                                                    data_reg[2] <=
32'h7e7e7e7e;
                                                                                    data_reg[3] <=
32'h7e7e7e7e;
                                                                                    out_valid <= 1;
                                                                                    proc_ready <= 1;
                                                                            end
                                                            end
                                                                    ///////// stream out data in buffer ////////
                                                            else if (esc_proc > 0)
                                                                    begin
                                                                            esc_proc <= esc_proc - 1;
                                                                            out_data <= data_reg [esc_proc - 1];
                                                                            if (esc_proc == 1) proc_ready <= 1;
                                                                            else  proc_ready <= 0;
                                                                    end
                                                    else
```

```
                                begin
                                        out_data <= 32'h7e7e7e7e;
                                        out_valid <= 0;
                                end
    end
        end
end
endmodule
```