Tetris Video Game

Embedded Systems Design

Summer, 2005

David Sun, davidsun168@hotmail.com

Conrad Lin, cklin.iic93g@nctu.edu.tw

National Chiao Tung University, Taiwan, R.O.C

Table of Content

Abstract 3
I. Introduction 4
II. Game Design 5
III. Implementation Result 6
IV. Conclusion 8

Abstract - This document provides a detail of the "tetris video game". We are a team of two men, who complete this project in the last half of class during two weeks. The target platform is a development board called *Spartan-3 Board*, which has a *Spartan-3 FPGA* chip on it. Our contribution is modifying the OPB video controller code for MicroBlaze, which is a software processor core developed by Xilinx, and writing a C program with size lessen then eight kilo bytes, as the body of logic of the " tetris video game".

Index Terms - Tetris game, fpga, embedded system, two old men.

I. Introduction

The whole system roughly consists of two parts - software and hardware. The most important item, which executes the machine instructions, is *processor*. How can we obtain such a item, since all we have is a board with one FPGA chip? Two solutions are given by the vendor Xilinx - one is software processor simulator *MicroBlaze*, that could be downloaded into the FPGA chip as the *bitstream* format; the other solution is giving the board a "real" physical processor - *IBM PowerPC Processor*. The campus laboratory provides the previous one.

The other part of the whole system is software. With the development board, we can write software in two programming language, VHDL and C. We can write VHDL code for critical functions, OPB Video Controller and RS-232 Serial Communication, for example. We write C code for high level logic, in this project, is tetris game.

The developer can use a technology *OPB* by IBM, to let the VHDL code and C code communicate. For the C program, the VHDL could be accessed by *memory ports* which are some special memory addresses. For the VHDL code, the computed result could be put at some special memory addresses, which will be read by C program later.

In our design, the C program is tetris game body, and the screen data is sent by the sequential memory addresses, and no returned data needed. That does not mean the C program needs not incoming data. The C program needs player/user keyboard pressing data to determine what to do with the tetris block, like rotation, movement, dropping.

We describe how tetris game works. Bricks composed of four-squares drop from the top of screen sequentially. The player rotates the bricks and moves them to left or to right. A brick stops when touches the bottom of the screen or any brick piled up from the bottom of the screen. The game ends when the bricks pile touch the top of the screen. A row disappears when bricks fill a horizontal line, and the pile drops down, and more room is emptied out.

II. Game Design

There are 7 types of tetris. We assign 7 colors to all of them. Encodinig 7 colors needs 3 bits. The C program define each square as one char, which is a 8 bits wide data. We use the MSB half part of the char (4 bits/1 nibble) to store the color information(3 bits). And the LSB half part of the char is for *square status*, like moving, dropping, and stopped.

The board is defined as

```
unsigned char grid[MAX_ROW_BLOCK_NUM]
              [MAX_ROW_BLOCK_NUM];
```

where MAX_ROW_BLOCK_NUM is 40 and MAX_ROW_BLOCK_NUM is 18. Each char is split into two parts too. LSB 4 bits could be 0, 1 and 2, which means empty/no block, moving and inside, respectively.

The incoming key press code is captured in ISR within C program. The desired keys are A, S, D and W which means moving to left, dropping, moving to right and rotation. Once the desired key codes induce an interrupt, the ISR store the keys in a 16 elements queue. The seems-too-big queue is for buffering the quick player key presses. After each drop delaying for one unit time, the C program check the queue for doing correct responding.

The game ends when the scanning on grid[] shows that even the most top row is full and no more tetris block insertion is possible.

III. Implementation Result

The result is a playable tetris game. The first part of implementation is the modified *OPB Video Controller* VHDL code, which can identify our 7 colors/3 bits data and mix correct RGB values to raster.

The second part of implementation is our C program. The following list shows the main data structure, functions, and pseudo code:

Plane for store tetris blocks:

```
unsigned char grid[40][18];
```

7 types of tetris blocks:

```
short int blocksharp[7]={
      0x4444, 0x0660, 0xE440, 0x4460, 0x2260, 0x0C60,
0X0360
};
```

Data structure for an active block:

```
struct block
{
   unsigned char type;/*:I,O,T,L,J,Z,N*/
   int  x;
   int y;
   unsigned char color;
};
```

Important functions:

```
int    main();
int    check_bottom();
int    check_top();
int    check_lborder();
int    check_rborder();
int    del_line();
void   initial();
void   drawblock();
void   clear_block();
```

```
void   todelay(void);
void   buildblock();
struct  block create();
void   change(struct block cblock);
void   updated_memory(struct block cblock);
void   show_next(struct block drblock);
```

The pseudo code of main():

```
int main ()
   Enable CPU interrupts
   Initialize ISR handlers
   WHILE(TRUE)
      switch(key)
         case DOWN()
         case UP()
         case LEFT()
         case RIGHT()
      delay for a while
```

IV. Conclusion

During this class we learn little by little. From the theory to the practical. We study the tools including VHDL language, C language, Spartan-3 Board, EDK software, VGA control and finally the complete system - tetris game. All of the above items(except C language) are difficult for us. One of us(Sun) majored in EE and the other(Lin) majored in CS in college. So many times we can not communicate or have a consistent idea for one same problem or solution.

However we have an answer for the situation - the older man(Sun) knows more about the system. So Sun did the most jobs, from the labs to the project. Lin is the assistant. Our conclusion is that, with the trend of chip size shrinking and the more transistors available, it's always hard to conduct all layers together to do one task like one single system.