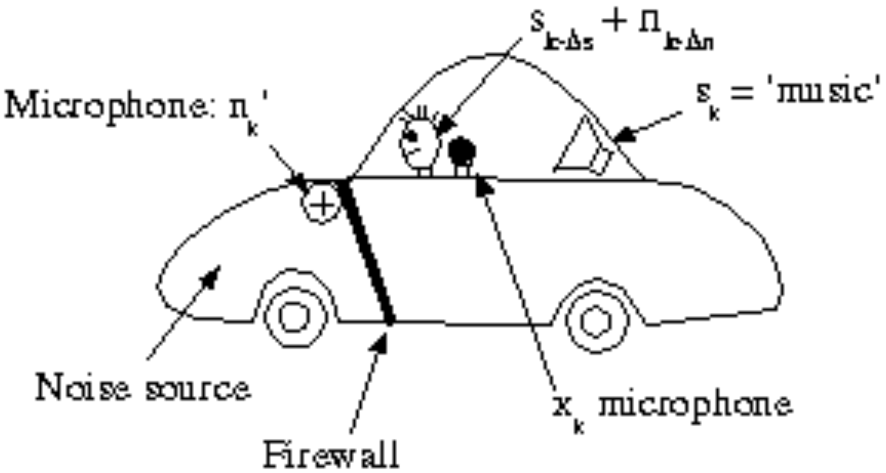
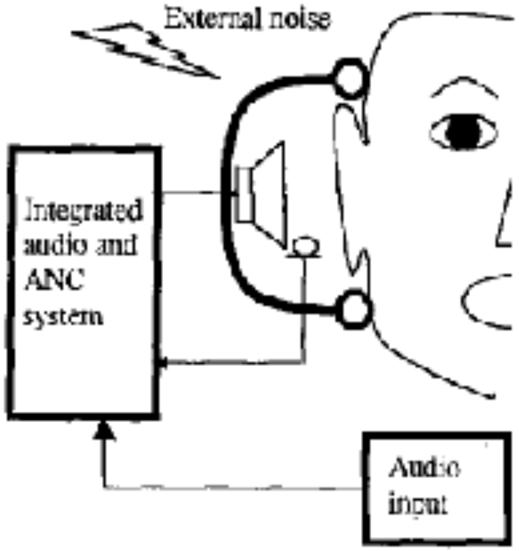
Adaptive Noise Cancellation

Ashwin Karthik Tamilselvan : at3103
Gikku Stephen Geephilip: gg2624
Richa Glenn Netto: rn 2324
Rishikanth : rc3022

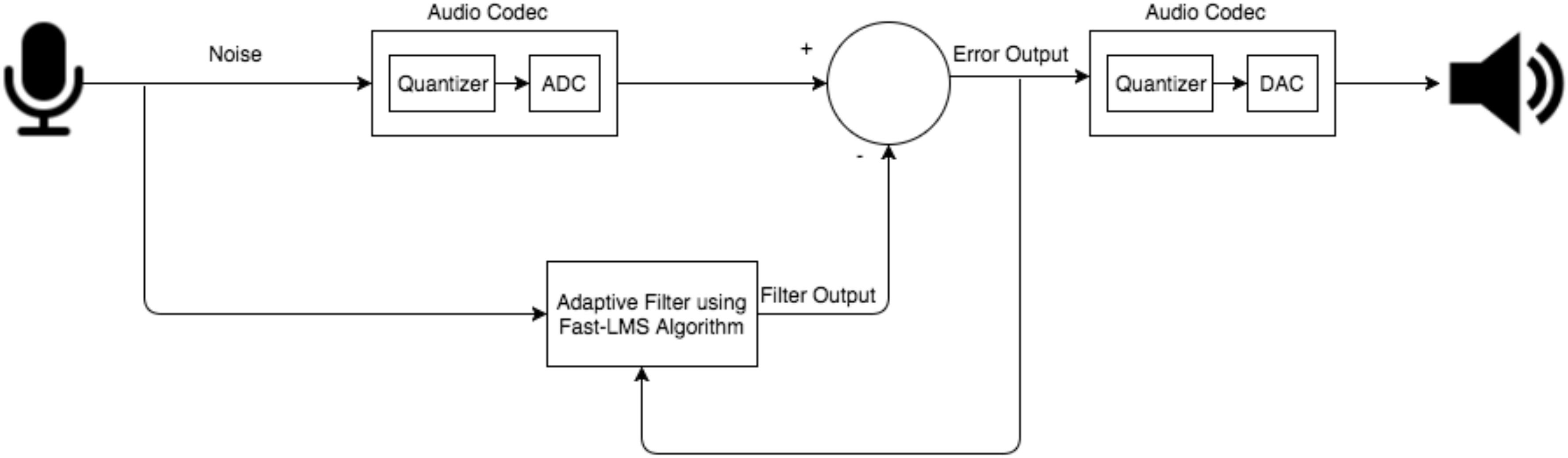
OVERVIEW

- Implement Adapative Noise Cancellation on FPGA



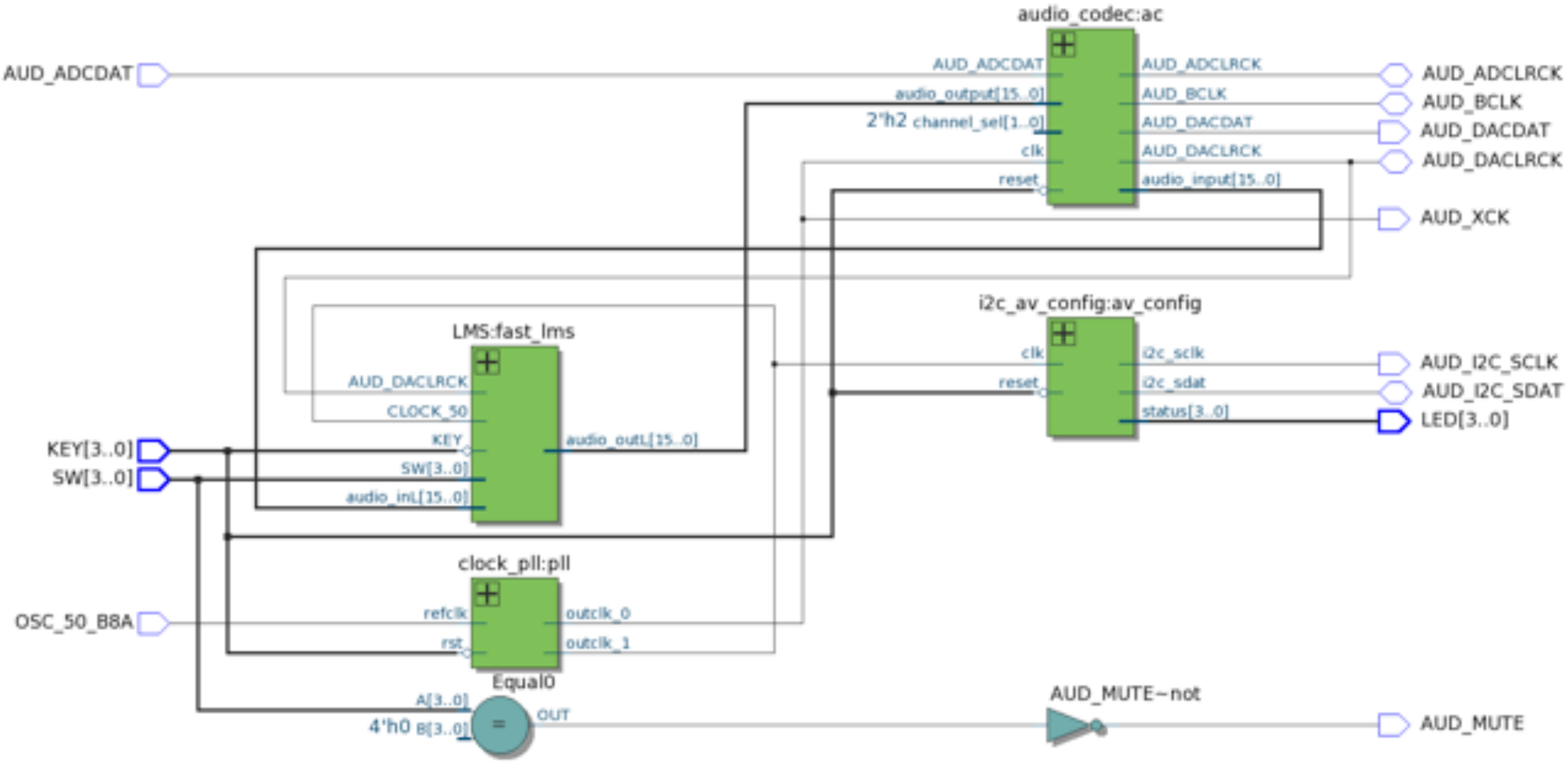


PRINCIPLE IDEA



$$W_{k+1}=W_k+e(k)*sign(u_k)>>>n$$

SYSTEM DESIGN



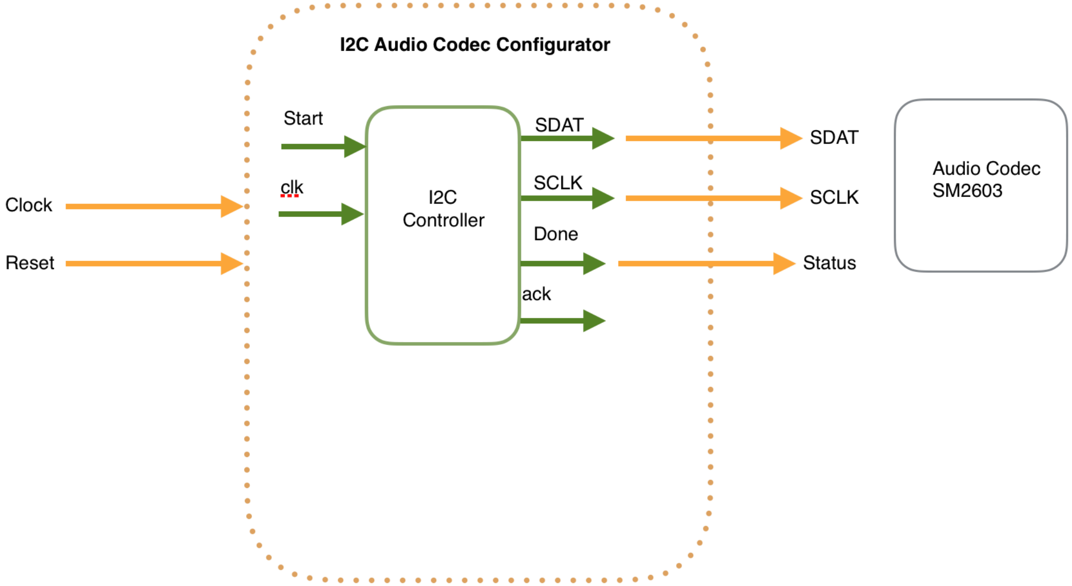
SYSTEM DESIGN

I2C Audio Codec Configurator

- The audio codec peripheral(SM2603) present on SoCKit is controlled using I2C communication.
- The Audio Codec has a set of configuration registers with each bit position controlling a specific aspect of the functionality.
  - These registers are configured via I2C using 16 bit data words as specified by the datasheet of SM2603

- We designed a generic I2C controller module, that takes care of the I2C protocol specifications and managing address and data words.

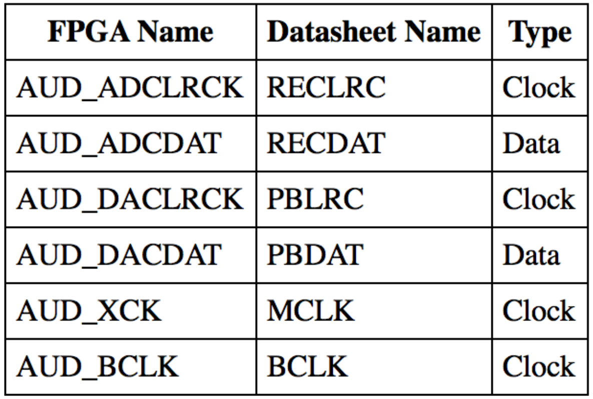
- The I2C module generates the I2CSCLK clock for the I2C bus using dividers from the main clock. It runs at the speed of 526KHz which is the maximum speed that the Audio Codec can run at



SYSTEM DESIGN

Audio Codec

- The Audio codec has different modes of operations and configurations.
- For our setup we are recording the mic input and sampling it via the ADC and the sampled values are passed to the LMS module
- The output of the LMS value is then fed to the DAC on the audio codec to play on the output port.

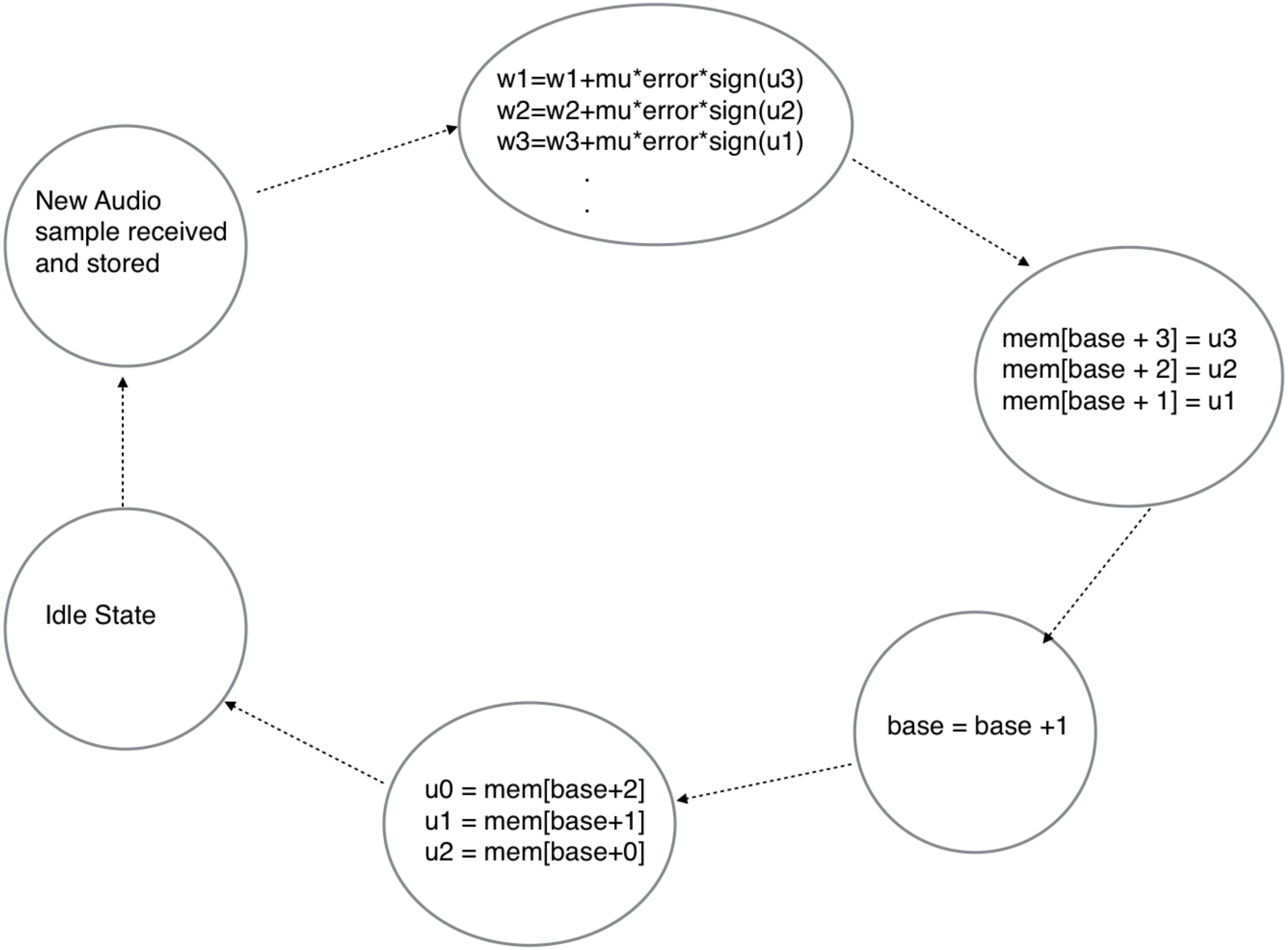


FPGA Name	Datasheet Name	Type
AUD_ADCLRCK	RECLRC	Clock
AUD_ADCDAT	RECDAT	Data
AUD_DACLRCK	PBLRC	Clock
AUD_DACDAT	PBDAT	Data
AUD_XCK	MCLK	Clock
AUD_BCLK	BCLK	Clock

- The 3 clocks specified in the datasheet needs to be generated
- The MCLK and BCLK are generated using the onboard PLL
- The LRC clock is then generated by dividing the BCLK
- The ADC is sampling at the rate of 44.1 KHz

LMS – Fast LMS

- We implement the Fast LMS using a state machine that handles the various events and sequential processes involved


New Audio sample received and stored

w1=w1+mu*error*sign(u3)
w2=w2+mu*error*sign(u2)
w3=w3+mu*error*sign(u1)
.
.

mem[base + 3] = u3
mem[base + 2] = u2
mem[base + 1] = u1

base = base +1

u0 = mem[base+2]
u1 = mem[base+1]
u2 = mem[base+0]

Idle State

CHALLENGES

- Figuring out the Audio Codec
  - The various clocks and timing cycles for sending and receiving data
- Understanding LMS and optimizing it for fast implementation on FPGA for faster convergence
- For our setup we are recording the mic input and sampling it via the ADC and the sampled values are passed to the LMS module
- The output of the LMS value is then fed to the DAC on the audio codec to play on the output port.

Thank You