Ananya Mukherjee
Scotty Kenney

Goal: Our goal was to design a Finite Impulse Response (FIR) filter, where, N represents window length (5 in our test benches), x is the vector of input data, y is the vector of output data, and h is the vector of coefficients.

$$y[n] = \sum_{k=0}^{N-1} h[k].x[n-k]$$

Procedure:

Test	Test Scenario
FUNCTIONALITY TESTS	
1	1) Write 3 coefficients into the coefficient buffer. 2) Go to reset state.
2	1) Write data_in in internal coefficients buffer until it is full (5 clock cycles). 2) Read from data_in when sample_enable is high as samples and calculate the dot product.
3	1) Go to reset state from s_coeff state . 2) Go to reset state from s_sample state.
4	1) coeff_enable high for 10 clock cycles.
5	1) Update coefficients after beginning sampling. 2) Sample again after new set of coefficients are read.
ERROR TESTS	
1	Coef_enable becomes ‘zero’ before 5 clock cycles
2	No coefficients are read, sample goes high.
3	Sample_enable and coef_enable both high in the same clock
4	System in error, reset occurs, but sample and coef are still high. Stay in error

FSM design: The specifications of a simplified Mealy machine are shown below in Figure 1. The system can be simplified into 4 distinct states, s_reset, s_coeff, s_sample, and s_error. The reset state is the return point for a system failure. Here everything is returned to 0 and the system is refreshed when reset goes low. When reset goes high, and coef_enable is high, the coeff buffer stores data from data in, and transitions to s_coef state. Here, as long as coef_enable is high, reset is low, and the coeff register has not filled, it will continue to fill. If coef_en does not stay high long enough (multiples of 5 clock cycles) to fill the buffer, the state will transition to s_error and wait for a reset, otherwise it transistions to s_sample for sampling. In s_sample, data_out is computed with the dot product combinatorial block, and the sample count increases until sample_en goes low, and coef_enable goes high. If both are high at the same time, the state will transition to s_error. Out_enable will go low in either case when the s_sample state is left.

Synthesis: In our model, the data input and FIR coefficients would enter on 8-bit buses leading flip-flops to store the data in the sample and coef registers. As the system is clocked, the inputs are sampled, through a shifter into a buffer (coef registers of size 5) for multiple of 5 clock cycle. Then it moves to filter state for performing the dot product between the data_in and the coefficients in the coef register. The ‘current state’ which controls this counter would be synthesized as a decoder with inputs based on the combinational logic of the reset, coef_en, sample_en, and count. When the combination of those signals is determined, it would be stored in a buffer ‘next state’ to be loaded into ‘current state’. The ‘current state’ register’s output would be the input to a selector to decide which combinational logic circuits to follow, as described by our if statements. The dot product of the coefficients with the data_input is calculated within more combinational logic including the use of 4 of our 16-bit CLA-Adders. The output would leave the system on a 16-bit bus. This synthesis can be viewed more closely in Figure 2.

Figure 1: Extended FSM of mealy machine of FIFO

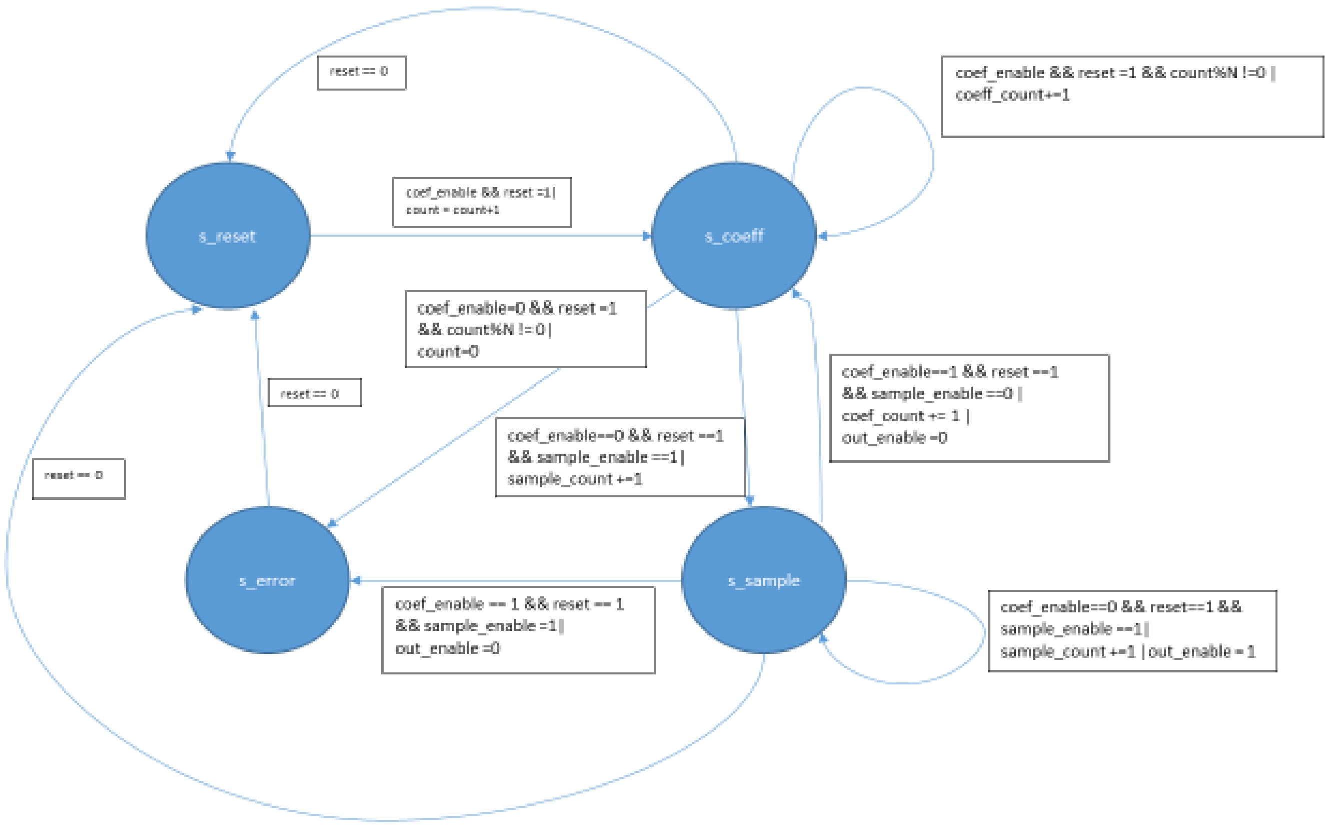
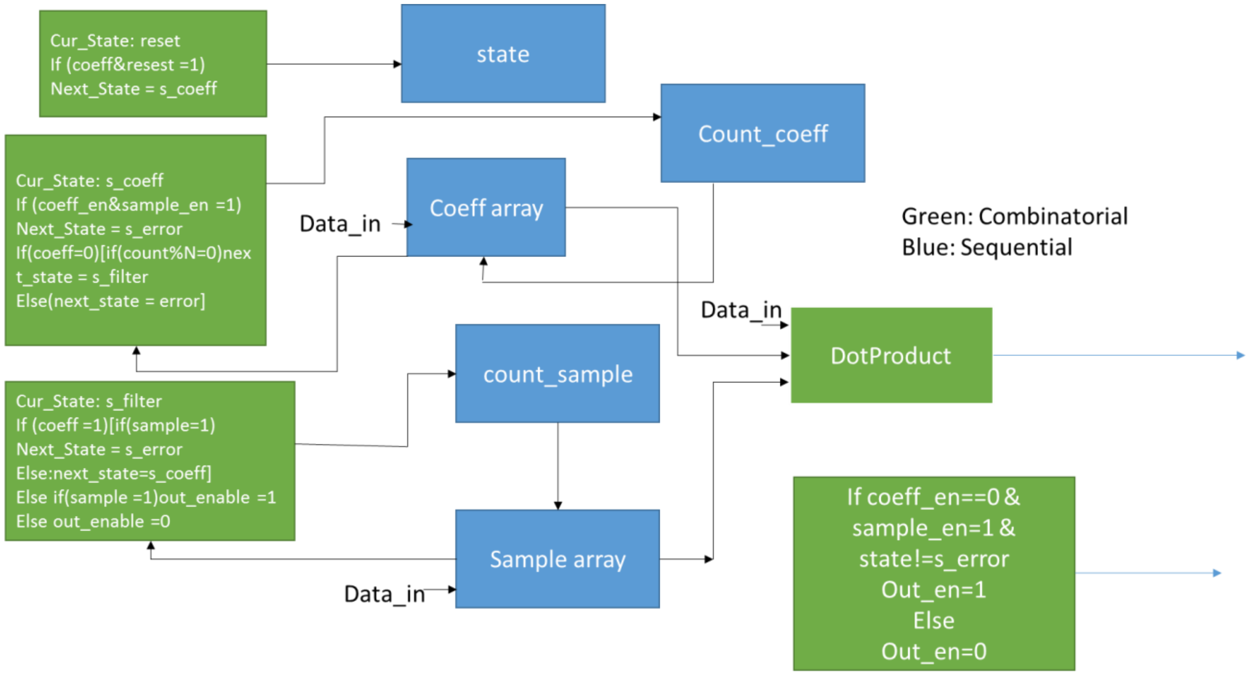


Figure 2: Diagram showing synthesized elements of FIFO


Cur_State: reset
If (coeff&resest =1)
Next_State = s_coeff
state
Count_coeff
Cur_State: s_coeff
If (coeff_en&sample_en =1)
Next_State = s_error
If(coeff=0)[if(count%N=0)next_state = s_filter
Else(next_state = error]
Data_in
Coeff array
Green: Combinatorial
Blue: Sequential
Data_in
DotProduct
count_sample
Cur_State: s_filter
If (coeff =1)[if(sample=1)
Next_State = s_error
Else:next_state=s_coeff]
Else if(sample =1)out_enable =1
Else out_enable =0
Sample array
Data_in
If coeff_en==0 &
sample_en=1 &
state!=s_error
Out_en=1
Else
Out_en=0