DIGITAL FILTER
DSE: Group 04 - Lab 6

Andrei Gheorghiu, Tianqi Li, Yanxi Xie, Zhang Zherui

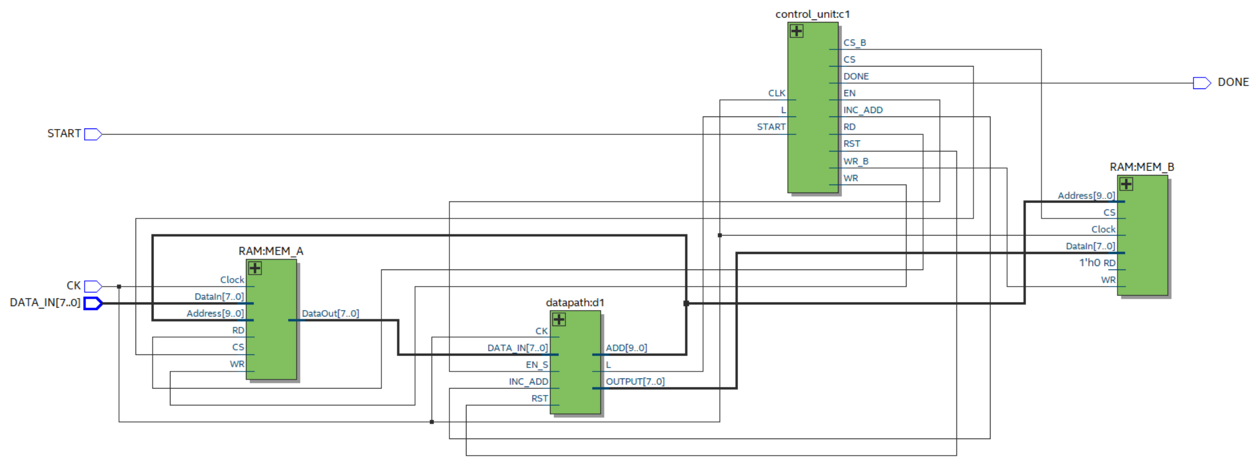
May 2021

1 Introduction

In this laboratory we design a digital processor working as a finite impulse response filter with characteristic equation

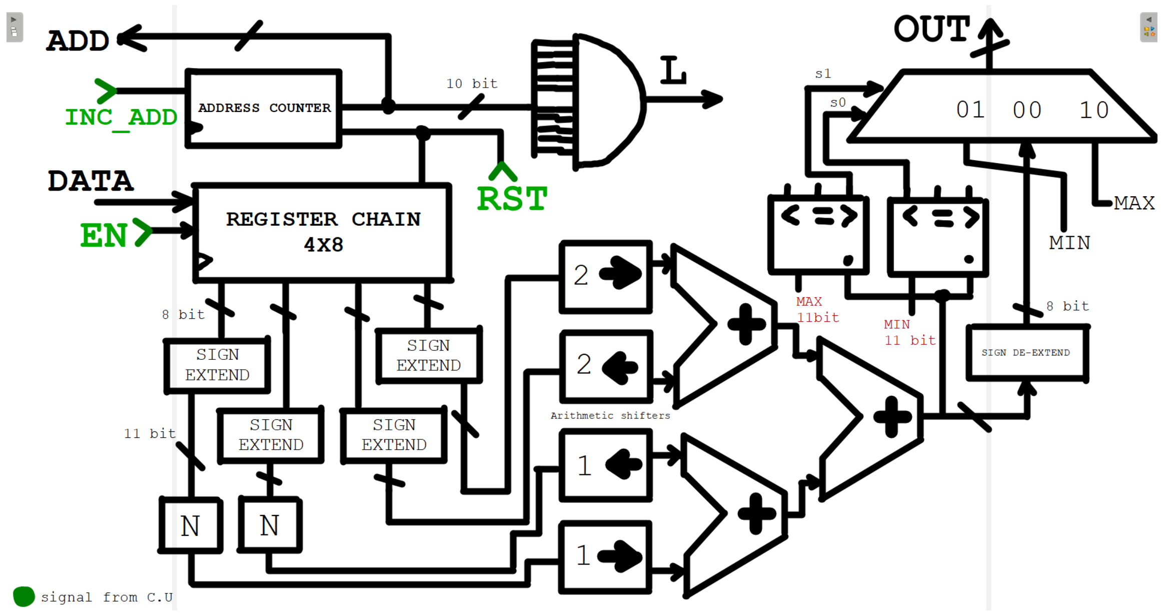
$$Y(n) = -0.5X(n) - 2X(n-1) + 4X(n-2) + 0.25X(n-3)$$

The filter first acquires sampled data from the DATA_IN input and loads it into MEM_A, once the memory is filled the processor starts to compute the $Y(n)$ values and load them into MEM_B until the end of the memory is reached again and an output pin signals the end of the algorithm.



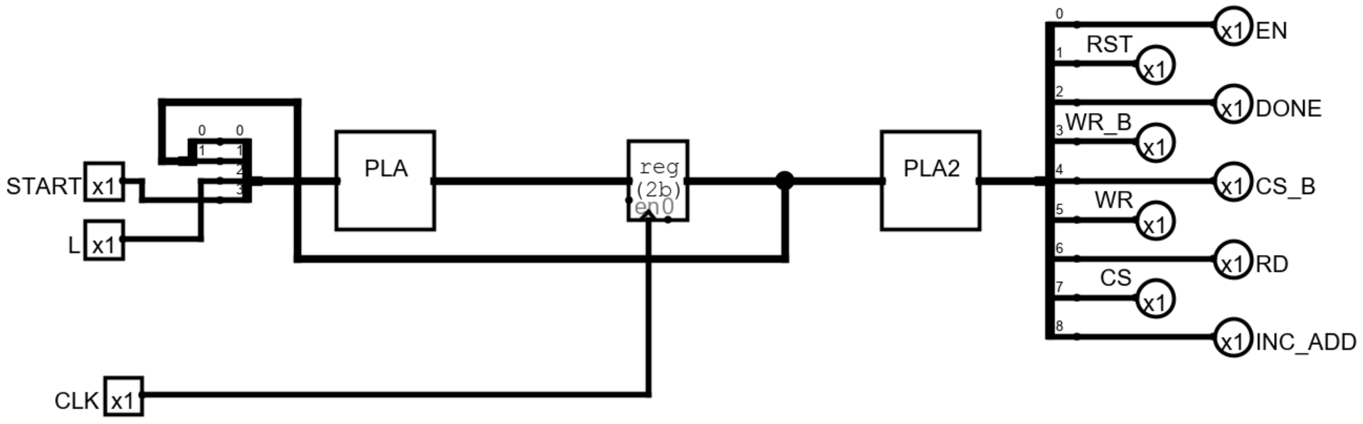
2 DATAPATH

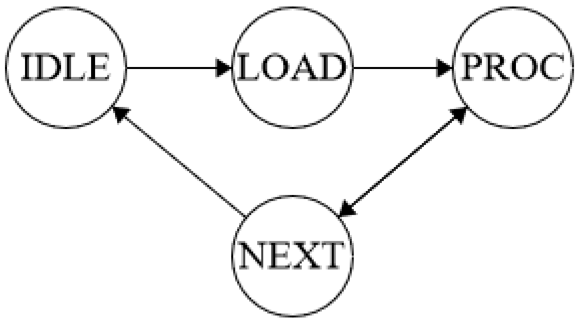
The architecture of the datapath starts with a 10 bit counter keeping track of the address selected on both memories, it gets an initial reset signal (RST) for initialization purposes. The 'L' flag activates once the address counter has reached a maximum meaning that we are at the end of the memories. The register chain is composed of 8 shift registers with length of 4 bits, the 8 bit sampled data is loaded into all the first bits of the registers at the rising edge of the clock if the EN signal is high, on the next rising edges the whole 8 bit word is shifted along the register chain like in a normal shift register, this allows us to work not only on the selected sample in the memory, but also on the preceding 3 numbers. The sign extension circuits perform sign extension from 8 bits to 11 bits, this avoids overflow and underflow problems which will be checked easily at the very end using a pair of 2's complement comparators. The shifting circuits implement arithmetic combinational shifting, they serve as the coefficient multipliers in this circuit. After the sum of all the 4 components, we cut the 11 bit result into it's 8 bit form, two comparators check for overflow or underflow and their outputs select the inputs of a multiplexer which will output the final result to be loaded into the memory.



3 CONTROL UNIT

We use a Moore type FSM to implement the Control Unit. There are 4 states, the first is called IDLE and it coincides with the starting and 'done' state of the system, the done output is high in this state; The LOAD state is where the samples are loaded into MEM_A, the PROC state is the processing of the final result, while the L flag is low, PROC goes to NEXT in the following rising edge and NEXT goes to PROC in the same way until L signals the end of the memory and we get back to IDLE and the 'done' output goes high. To avoid problems PROC does not allow the address counter to increment, only NEXT does. The truth tables for the transition & output logic are presented in the following page.





TRANSITION LOGIC (PLA)

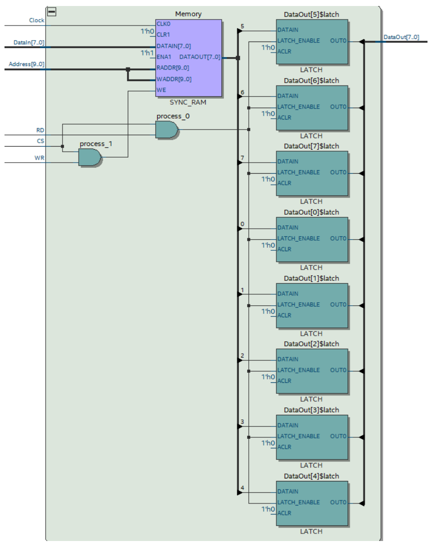
0000	00
0001	01
0010	11
0011	10
0100	00
0101	10
0110	11
0111	00
1000	01
1001	01
1010	11
1011	10
1100	01
1101	10
1110	11
1111	00

OUTPUT LOGIC (PLA2)

00	000000110
01	110100000
10	011000001
11	111011000

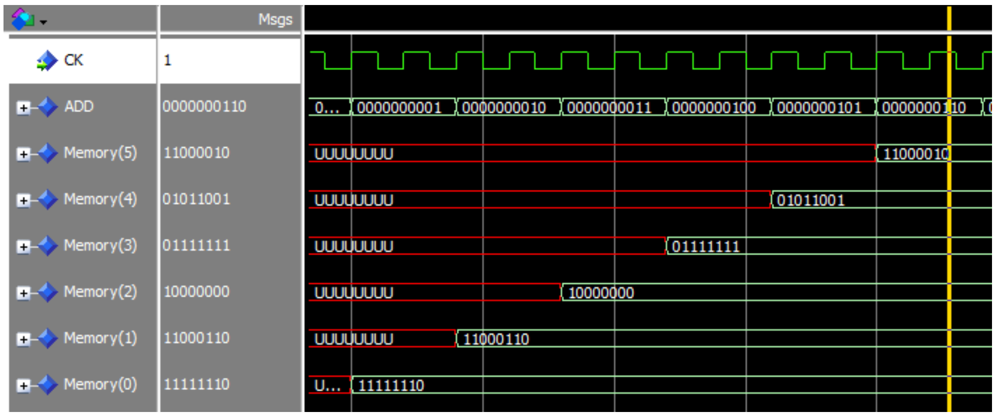
4 RAM

We use a behavioural VHDL description to make a 1kB register file identical in functionality to the one demanded by the laboratory assignment.



5 Testing

To test the functionality of the system we gave it the vector : $[4, 100, 8, -16, 0, 0, \ldots]$ which will test both positive,negative,overflowed and underflowed conditions (all the edge cases), the output should be : $[-2, -58, MIN : -128, MAX : 127, 89, -62, \ldots]$



	Msgs
CK	1
ADD	0000000110
Memory(5)	11000010
Memory(4)	01011001
Memory(3)	01111111
Memory(2)	10000000
Memory(1)	11000110
Memory(0)	11111110

As can be seen in the picture, the memory contains the correct values represented in 2's complement form.