Elettronica dei Sistemi Digitali
Digital Systems Electronics
Lab#6

A simple digital filter

This lab aims at developing your digital design skills. You will deal with a realistic problem starting from the functional specification and you will develop a circuit made of several blocks including memories, control unit and data path. You will write and simulate the VHDL of the whole design. Differently from previous labs, **You are NOT requested to download the code onto the FPGA circuit to test it** , but you must design the circuit, document it, simulate it in order to validate your work, and then write a final report with the following items:

- Description of the Design
- The complete scheme of the Datapath and the Control Unit so that whoever can implement a PCB (Printed Circuit Board) for your design starting uniquely form your schemes!
- The logic procedure you used to validate your design by means of the TestBench
- Some Modelsim Simulation fragments to demonstrate your Work.

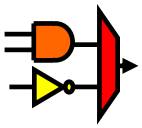
Abbreviations and acronyms:
ASM – Algorithmic State Machine
VHDL – Very high speed integrated circuits Hardware Description Language
CU – Control Unit

1 – A simple digital filter

A circuit that implements a simple digital filter, shown in figure 1, can be described as follows.

A memory (MEM_A), whose size is 1 kByte (see figure 2), is used to store 1024 samples represented as 8 bit wide 2's complement values. The circuit has an input signal, referred to as START. When the circuit samples START at '1', the processing starts, namely from the next clock cycle the circuit starts to store the data coming from input DATA_IN into MEM_A. The writing operation is synchronous with the positive edge of the clock signal CLK (one datum per clock cycle is written to the memory). The samples are stored in order, namely the first sample is stored at the location 0 and the last at the location 1023.

The memory is implemented as a *register file* so the writing operation is synchronous, whereas the reading is asynchronous:



- if WR='1' on the rising edge of CLK, and the memory is selected, i.e. CS='1', then the datum available at DATA_IN is written at the location pointed by ADDRESS
- if CS='1' and RD='1' the DATA_OUT output is combinational and it is the value stored at the location pointer by ADDRESS.

Once all the data are stored in MEM_A, the circuit must fill a second memory MEM_B (equal to MEM_A) with

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

where each $X(i)$ value represents one datum in MEM_A and the four data in eq. (1) are consecutive. Thus, to compute the first 3 values of Y, we assume that the values not available in the memory (the ones whose index is negative) are zero:

1. $Y(0) = -0.5X(0) - 2X(-1) + 4X(-2) + 0.25X(-3) = -0.5X(0)$
2. $Y(1) = -0.5X(1) - 2X(0) + 4X(-1) + 0.25X(-2) = -0.5X(0)\ ) - 2X(0)$
3. $Y(2) = -0.5X(2) - 2X(1) + 4X(0) + 0.25X(-1) = -0.5X(2) - 2X(1) + 4X(0)$
4. $Y(3) = -0.5X(3) - 2X(2) + 4X(1) + 0.25X(0)$
5. …
6. ...

   ...

1024. $Y(1023) = -0.5X(1023) - 2X(1022) + 4X(1021) + 0.25X(1020)$

The algorithm must rely on a multiplierless datapath made of only one adder. It means that the multiplications must be performed only using add and shift operations (not too complex, since all the constants are powers of 2). The parallelism of the datapath has to be sized to compute the values of Y as in eq. (1). However, since MEM_B is equal to MEM_A if $Y(n)$ cannot be represented on 8 bits it must be saturated:

- If $Y(n)$ is greater than the maximum positive value represented on 8 bits, then it is saturated to the maximum positive value ($Y(n)$=01111111)
- If $Y(n)$ is lower than the minimum negative value represented on 8 bits, then it is saturated to the minimum negative value ($Y(n)$=10000000)

Once the algorithm is completed the circuit drives to '1' the output DONE.
Finally, the circuit waits for a new START (0 → 1) before starting again the algorithm.

For the circuit described above, you have to perform the following steps:

1. **Write** the pseudocode of the algorithm.
2. **Draw** the datapath of the algorithm with all necessary control signals, status signals, connections, parallelism of the busses, and all relevant details.
3. **Prepare** the ASM chart of the algorithm.
4. **Detail** the ASM chart of the control unit.
5. **Write** the timing of the circuit both during the load of the samples and the computation of $Y(n)$. **Note** it is important to show only a meaningful fragment of the timing showing the evolution of the status register, the data and the control signals. This written by-hand timing will be compared with the simulated circuit timing to be sure that it works according to your requirements.
6. **Write the VHDL** of the whole system including the memories.
7. **Prepare a testbench and simulate** the behavior of your circuit.
8. **Write a final report** including all the materials requested in the previous items, plus your comments about the design choices you made, the testing algorithm you used, the by-hand timing you prepared during the design, some Modelsim simulations fragments to demonstrate that it works.

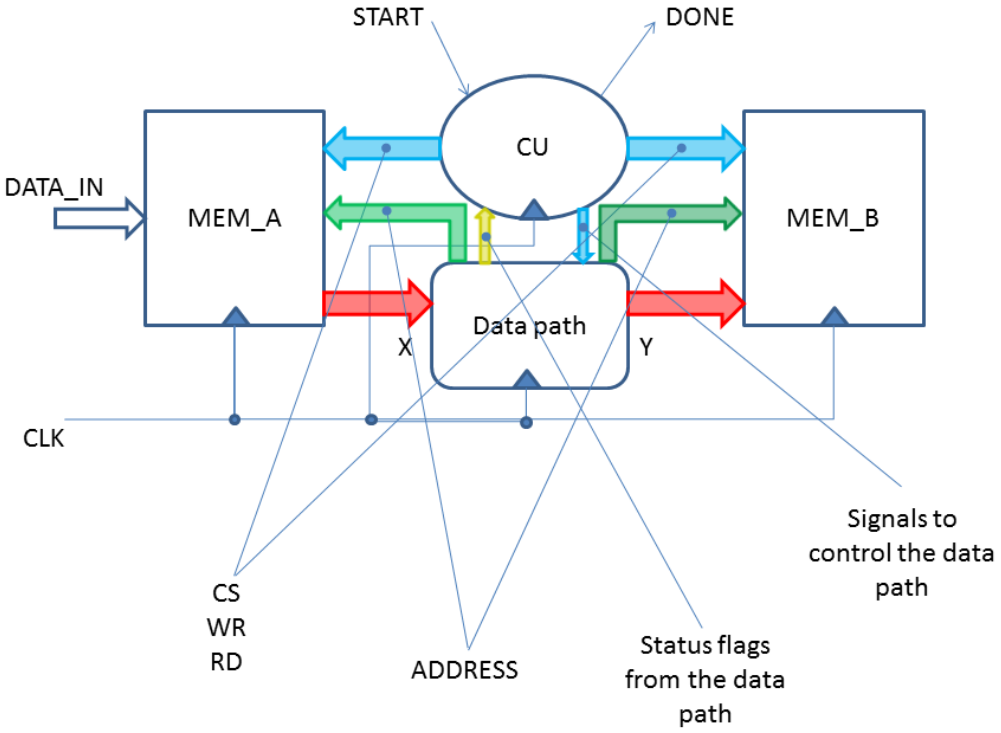


Figure 1 simple filter architecture

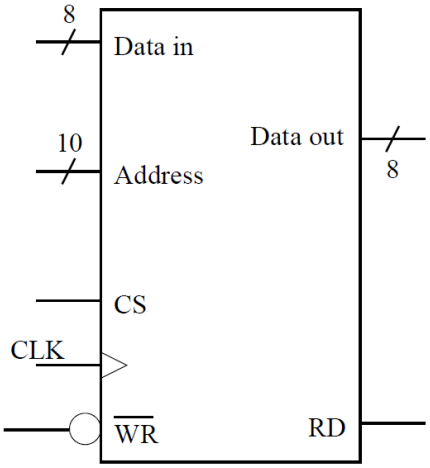


Figure 2 - Memory pinout