
POLITECNICO
DI TORINO

# – Elettronica dei Sistemi Digitali –
## Lab#6

# ASM chart

This lab aims at developing Your digital design skills. You will deal with a real 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.

In any case at the end of these pages You will be explained all the materials You have to give the teachers for the evaluation of the lab.

The evaluation of Your work will be based on Your report. Submit your report **before** the deadline published on "Portale della didattica" according with the following procedure:

1. Produce a pdf file of your report. **Note** pdf is the **ONLY** accepted format.
2. The name of the pdf **MUST** be obtained by concatenating in alphabetical order the surnames of the people who worked on the preparation of the report. Special characters including spaces and apostrophe **MUST** be avoided. Surnames are separated by the underscore '_' character.
   **Example:** the group XYY (*e.g. C14*) with *Lennon*, *McCartney*, *Harrison* and *Starr* will produce a file named *sqC_tav14_lab06_harrison_lennon_mccartney_starr.pdf.*
3. Upload the pdf file on "Portale della didattica" in the section named "elaborati".

**Note: the report can be part of the discussion during the exam.**

**Contents:**

1. A possible example of Proportional-Integral-Derivative (PID) controller

**Abbreviations and acronyms:**

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

# 1 – PID controller

In the following, You will find a simplified version of a Proportional-Integral-Derivative (PID) controller, which is employed in many control systems such as those for monitoring the speed of DC motors or for implementing a Phase-Locked Loop (PLL). A schematic view of a PID controller is available in Figure 1: the setpoint *r(t)* - *i.e.* the reference value to be reached - is compared with the process variable *y(t)* provided by the plant and their difference is equal to the error

$$e(t) = r(t) - y(t)$$

The error is then provided to the PID controller, which is made of three parts, each one multiplied by a coefficient $K_x$:

- Proportional (**P**), which provides a contribution related to the current error.
- Integral (**I**), which provides a contribution not only depending on the current error, but also on the previous ones.
- Derivative (**D**), which provides an estimate of the error's future trend based on its current state of change.

The overall control function is the continuous time domain is

$$u(t) = K_p e(t) + K_i \int_0^t e(t')\mathrm{d}t' + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}$$

In many controllers, a system for saturating the dynamics of the control function is available, whose output is provided to the plant in order to change its process variable.
In this project a discrete-time PID controller is going to be implemented, whose control function for the $k^{\text{th}}$ error (with $k \geq 0$) is

$$u[k] = K_p e[k] + K_i \sum_{l=0}^{k} e[l] + K_d\big(e[k] - e[k-1]\big)$$

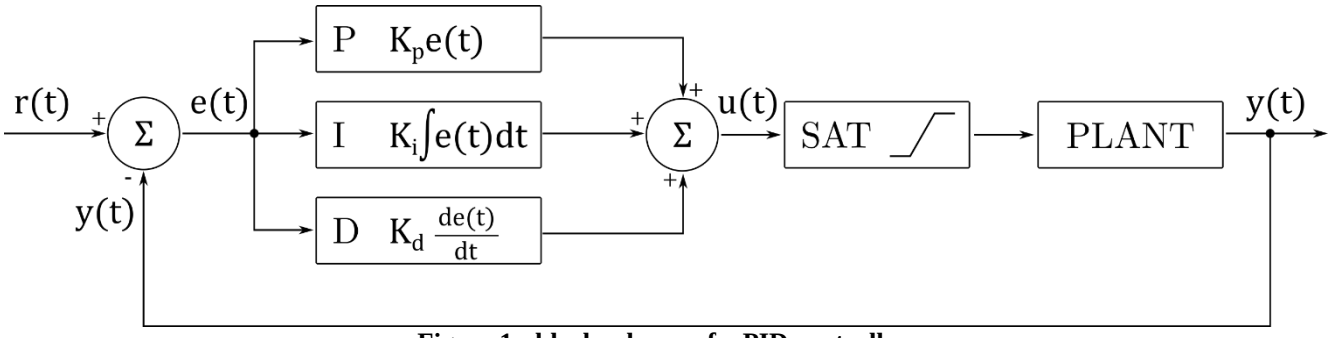
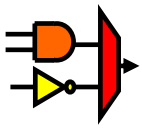



**Figure 1 - block scheme of a PID controller.**

The architecture to be designed describes the **circuit implementing the discrete-time control function**. We will assume that the clock period of the circuit to be designed is sufficiently low with the respect to the minimum time constant of the plant dynamical response ($T_{clk} << \min(\tau_{plant})$) so that the effect of a delay of some clock periods between the arrival of $e[k]$ and the provision of $u[k]$ is negligible. This assumption will let us reduce the number of processing elements in the controller to be designed.



In order to simplify the functional verification of Your implementation, it can be assumed that a memory (MEM_A), whose size is 1 kByte (see figure 2), stores 1024 errors represented with 8-bit wide 2's complement values. This implies that You do not have to implement the whole feedback system and the errors are assumed to be available *a priori*.
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 stores the errors $e[k]$ 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 both writing and reading operation are synchronous:

- 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' on the rising edge of CLK, the DATA_OUT value is equal to the content of the location pointed 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 the results provided by the control function with

$$K_p = 3.75$$
$$K_i = 2$$
$$K_d = 0.5$$

We assume that the values $e[k]$ for $k < 0$ are not available in the memory and they are treated as zero, so the control function outputs will be:

1. $u[0] = 3.75\ e[0] + 2\ e[0] + 0.5\ e[0]$
2. $u[1] = 3.75\ e[1] + 2\ (e[0] + e[1]) + 0.5\ (e[1] - e[0])$
3. $u[2] = 3.75\ e[2] + 2\ (e[0] + e[1] + e[2]) + 0.5\ (e[2] - e[1])$
4. $u[3] = 3.75\ e[3] + 2\ (e[0] + e[1] + e[2] + e[3]) + 0.5\ (e[3] - e[2])$
5. …
6. ...

...

1024. $u[1023] = 3.75\ e[1023] + 2\ (e[0] + e[1] + \ldots + e[1023]) + 0.5\ (e[1023] - e[1022])$

The algorithm must rely on a **multiplierless datapath made of one adder** for the computation of $u[k]$. It means that the multiplications must be performed only using add/sub and shift operations (not too complex, since all the constants are combinations of powers of 2) and the same adder must be employed for computing not only the sum of the three contributions, but also the derivative and integral ones on their own. The parallelism of the datapath has to be sized to compute the values of $u[k]$ **without overflow**. However, since MEM_B is equal to MEM_A, if $u[k]$ cannot be represented on 8 bits it must be saturated:

- If $u[k]$ is greater than the maximum positive value represented on 8 bits, then it is saturated to the maximum positive value ($u[k]$ = 01111111).
- If $u[k]$ is lower than the minimum negative value represented on 8 bits, then it is saturated to the minimum negative value ($u[k]$ = 10000000).

Once the control function has been evaluated for each error $e[k]$ 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 (that must be inserted in the final report You have to give the teachers!):

1. **Write** the pseudocode of the algorithm.
2. **Draw** the datapath of the algorithm with all the controls, status, connections, parallelism of the busses, etc...
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 considerations 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.

**Remember** that the evaluation of this lab will be made uniquely on the report You write, **so write a good report!!!!!**

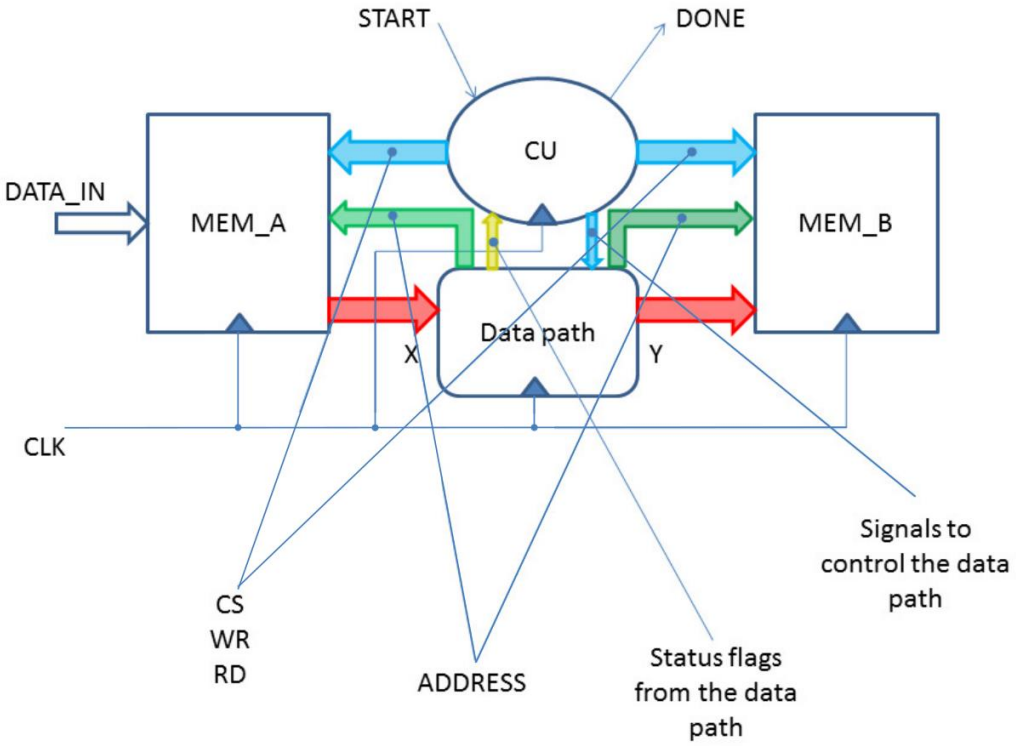



**Figure 2 - simple filter architecture**

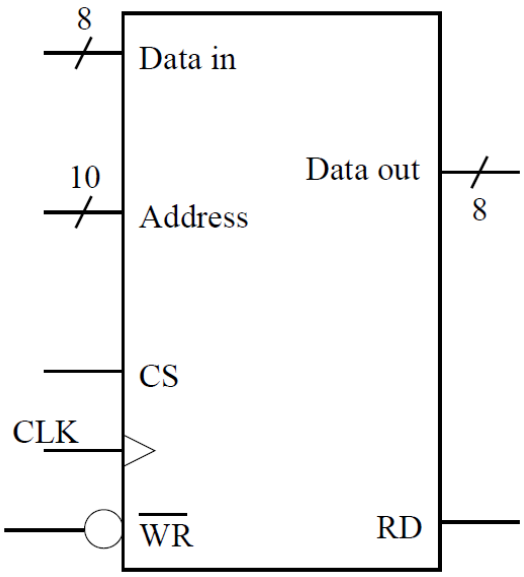



**Figure 3 - Memory pinout**