CPU Architecture

LAB1 preparation report

VHDL part1 – Concurrent code

Hanan Ribo

16/03/2025

Table of contents

1. Aim of the Laboratory

- Obtaining skills in VHDL part1 code, which contains Code Structure, Data Types, Operators and Attributes, Concurrent Code, Design Hierarchy, Packages and Components.
- Obtaining basic skills in ModelSim (multi-language HDL simulation environment).
- General knowledge rehearsal in digital systems.
- Proper analysis and understanding of architecture design.

2. System Design ISA

Function Kind	**Decimal value**	**ALUFN**	**Operation**	**Note**
Arithmetic	8	**01**000	Res=Y+X	
	9	**01**001	Res=Y-X	Used also for comparison operation
	10	**01**010	Res=neg(X)	
	11	**01**011	Res=Y+1	Increment of Y in one
	12	**01**100	Res=Y-1	Decrement of Y in one
Shift	16	**10**000	Res=SHL Y,X(k-1 to 0)	Shift Left Y of q≜X(k-1 ..0) times Res=Y(n-1-q…0)#(q@0) $\boldsymbol{When\ k = log_2 n}$
	17	**10**001	Res=SHR Y,X(k-1 to 0)	Shift Right Y of q≜X(k-1 ..0) times Res=(q@0)#Y(n-1...q) $\boldsymbol{When\ k = log_2 n}$
Boolean	24	**11**000	Res=not(Y)	
	25	**11**001	Res=Y or X	
	26	**11**010	Res=Y and X	
	27	**11**011	Res=Y xor X	
	28	**11**100	Res=Y nor X	
	29	**11**101	Res=Y nand X	
	30	**11**111	Res=Y xnor X	

Table 1: Selected operations

3. Status Bits

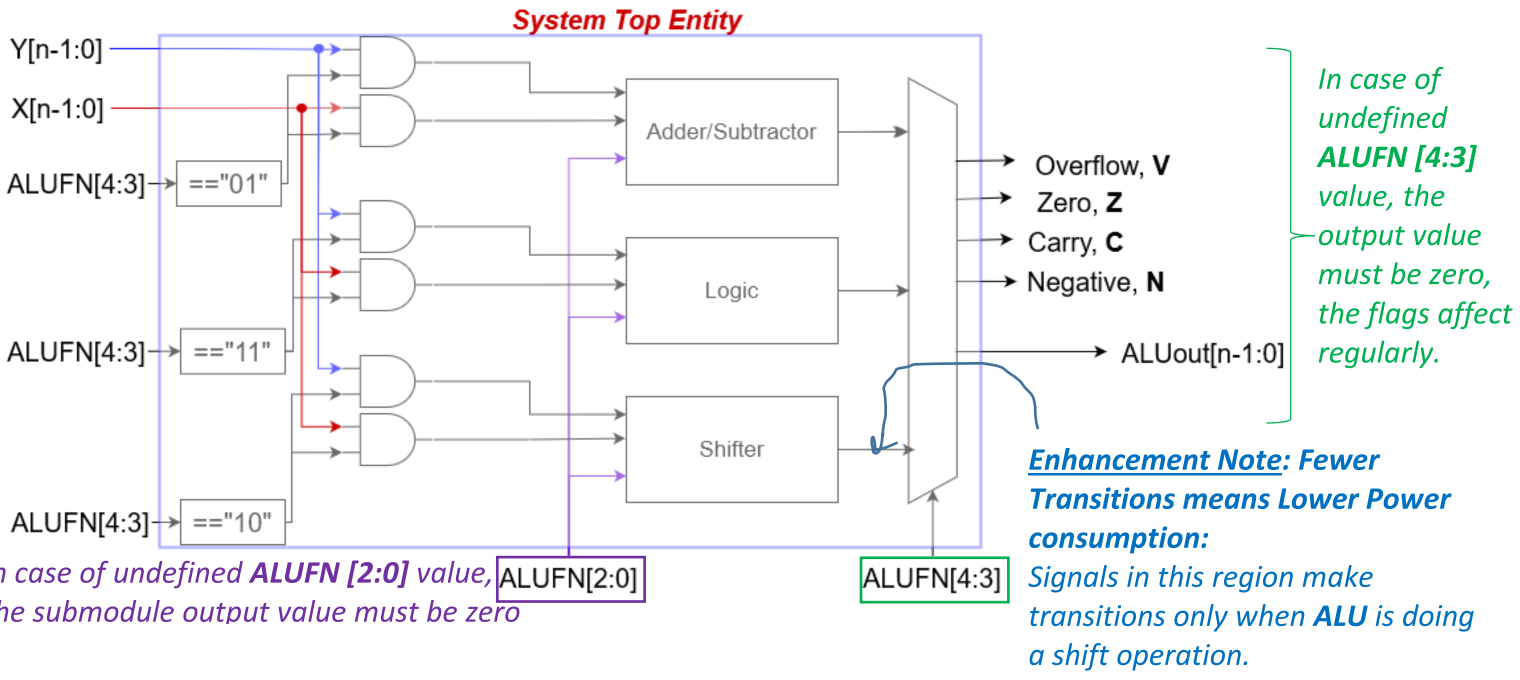
Status Bit	Description
V	Overflow bit. This bit is set when the result of an arithmetic operation overflows the signed-variable range. **1.** In case of ADD operation: V=1: • Positive + Positive = Negative • Negative + Negative = Positive otherwise, V=0. **2.** In case of SUB operation: V=1: • Positive - Negative = Negative • Negative - Positive = Positive otherwise, V=0. **<u>Note</u>:** It is mandatory to first manually write an optimized Boolean equation for V flag using above terms before its implementation.
Z	Zero bit. This bit is set when the result is 0 and cleared when the result is not 0.
C	Carry bit. This bit is set when the result produced a carry and cleared when no carry occurred.
N	Negative bit. This bit is set when the result is negative and cleared when the result is not negative.

Table 2: Status Bits details

4. System Design Micro-Architecture

In this laboratory you will design a module which contains the next three sub-modules:

- Generic Adder/Subtractor module between two vectors Y, X of size n-bit (the default is n=8).
- Generic Shifter module based on Barrel-Shifter n-bit size (the default is n=8).
- Boolean Logic operates bitwise.
- The generic n value must be verified for 4,8,16,32 (set from ***tb.vhd*** file).
- You are required to design the whole system and make a test bench for testing.


System Top Entity
Y[n-1:0]
X[n-1:0]
ALUFN[4:3]
=="01"
=="11"
=="10"
Adder/Subtractor
Logic
Shifter
Overflow, V
Zero, Z
Carry, C
Negative, N
ALUout[n-1:0]
ALUFN[2:0]
ALUFN[4:3]
In case of undefined ALUFN [2:0] value, the submodule output value must be zero
In case of undefined ALUFN [4:3] value, the output value must be zero, the flags affect regularly.
Enhancement Note: Fewer Transitions means Lower Power consumption:
Signals in this region make transitions only when ALU is doing a shift operation.

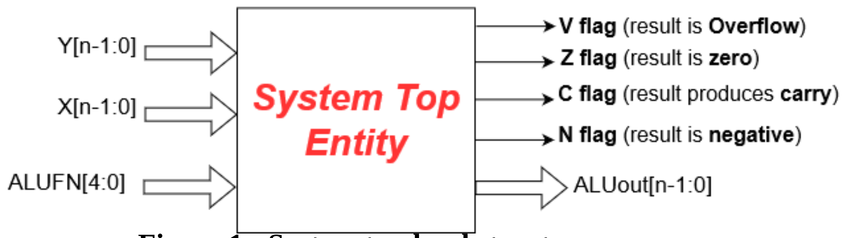
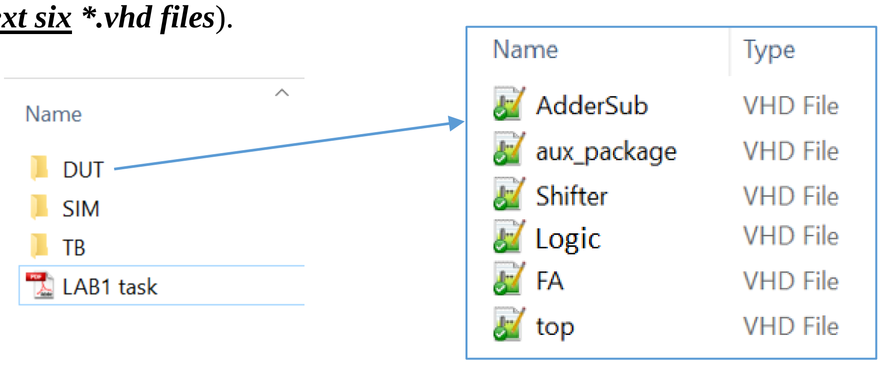

Y[n-1:0]
X[n-1:0]
ALUFN[4:0]
System Top Entity
V flag (result is Overflow)
Z flag (result is zero)
C flag (result produces carry)
N flag (result is negative)
ALUout[n-1:0]

Figure 1 : System top level structure

- The Top Level design must be Structural (***your DUT must contain the <u>exact</u> <u>next six</u> *.vhd files***).



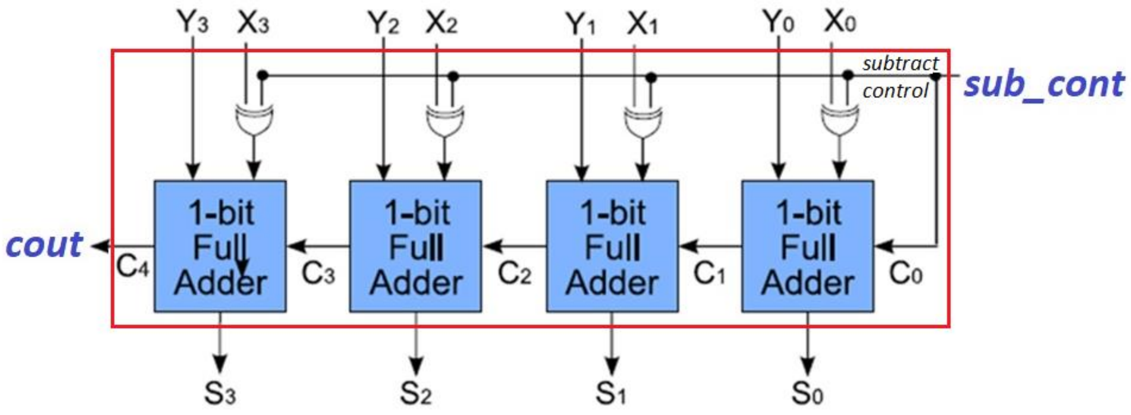
Name
DUT
SIM
TB
LAB1 task

Name	Type
AdderSub	VHD File
aux_package	VHD File
Shifter	VHD File
Logic	VHD File
FA	VHD File
top	VHD File

- **Note:** in the given two files **top.vhd**, **aux_package.vhd (**you can only add code to these files, ***you are not allowed to erase anything*****).**
- The submitted project must be compiled using the given **tb_ref1.vhd** file (otherwise the submitted project will be considered as a failure).
- The submitted assignments get through copy checking machine, in this case, both sides' assignments will be disqualified.

5. Generic Adder/Subtractor module based on a single ripple carry adder:

- You are required to design a Generic Adder/Subtractor between two vectors Y, X of size n-bit (the default is n=8), using the next diagram. The design must be Structural of a least two levels.
- In order to do so, you are asked to use the Generic Adder code that was given in Moodle.



Note: this figure illustrates $Y \pm X$ operation

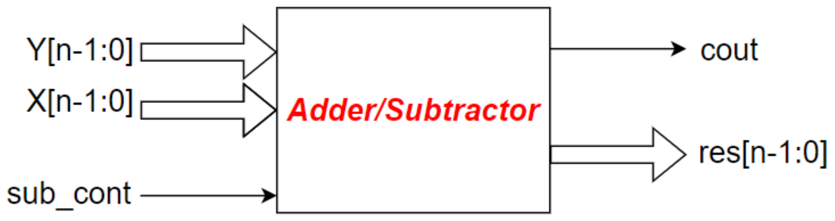
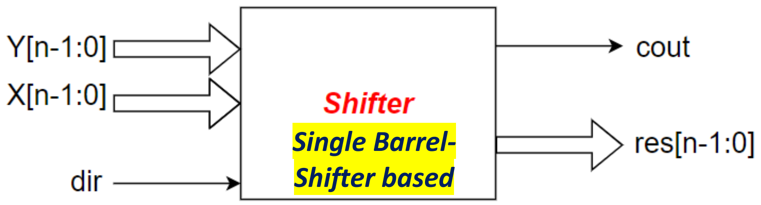


Figure 2: Generic Adder/Subtractor (based on a single ripple carry adder)

6. Shifter Module based on a single Barrel-Shifter n-bit:

Note: using **sll, srl** operators are forbidden and causes to disqualification of this clause.
Hint: In order to meet the requirements you must use ***generate*** concurrent statement



Note: this figure illustrates the operation ***Shift** X, Y (2 **downto** 0)*

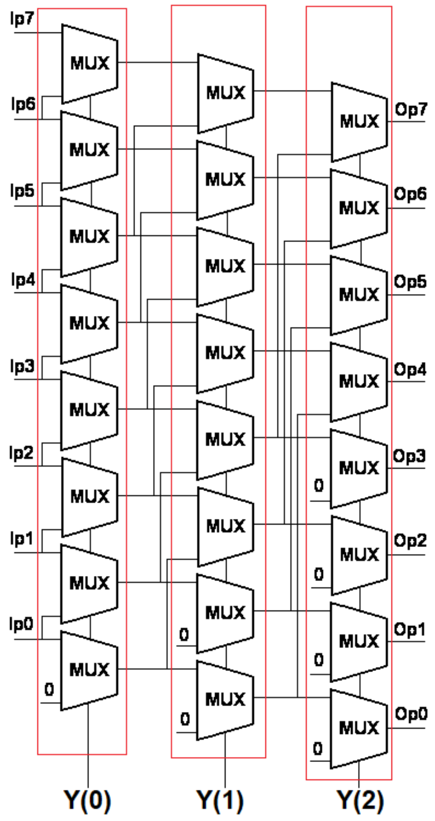


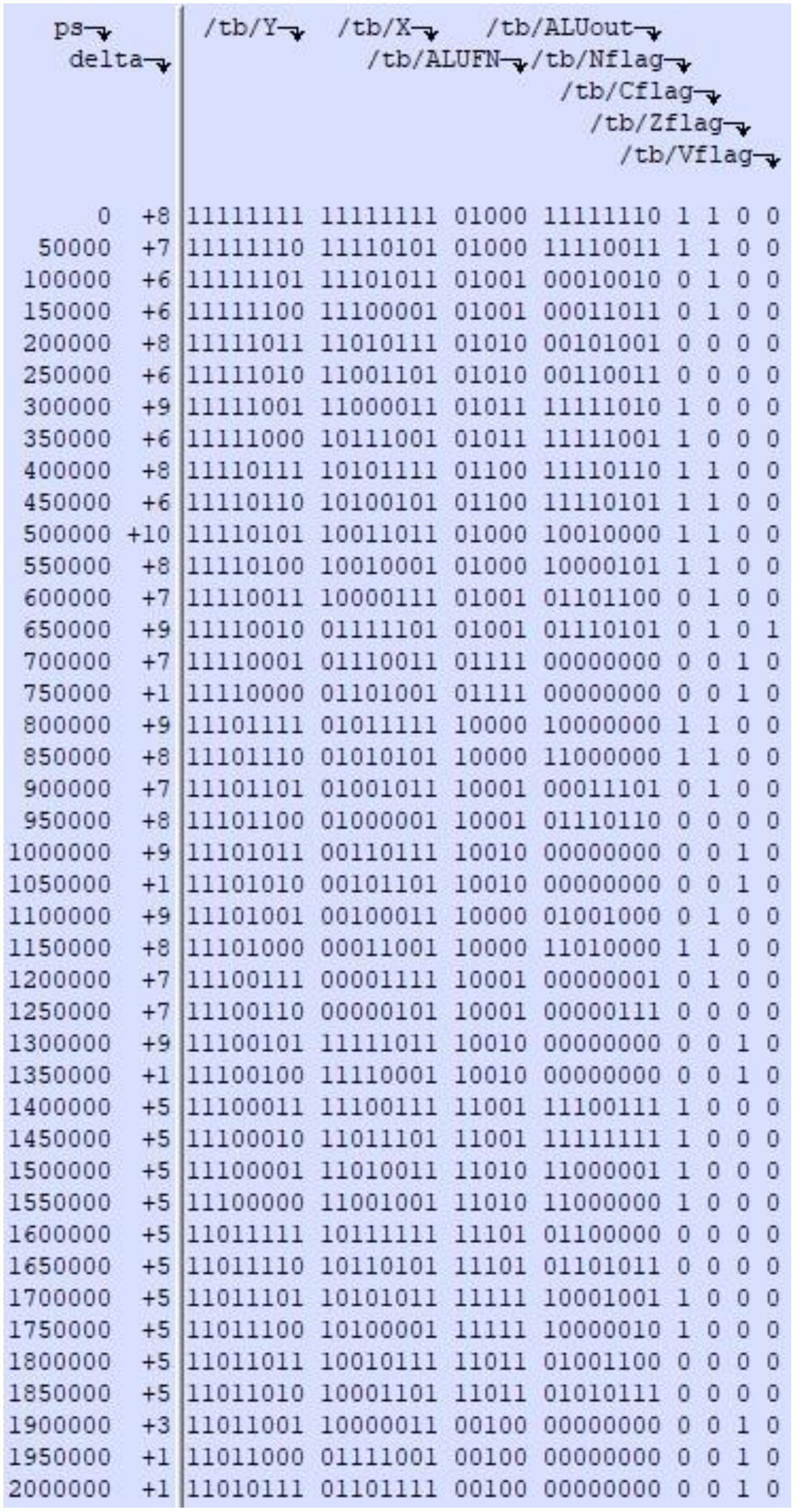
Figure 3: Example of 8-bit Barrel Shifter

7. Test and Timing:

- Design a test bench which tests all the system.
- Analyze the results by zooming on the important transactions in the waveforms. explain these (input/output/internal signals of the system).
- You are welcome to use the Internet also as reference.
- **Good tip for beginners**: Build a test bench for each module you are designing for easy debugging, otherwise you will waste a lot of time for whole system debug.
- The timing of the system will be ideal (means a functional simulation).

8. System Output Example (for n=8)

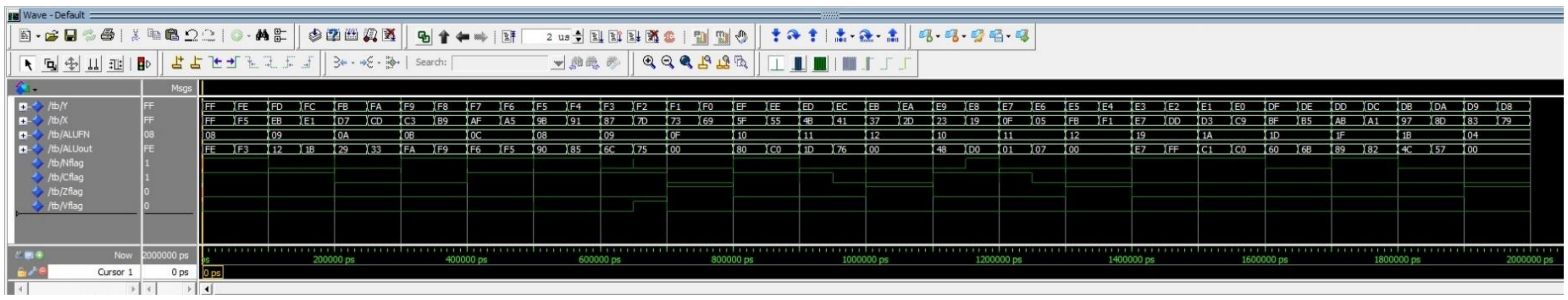
Signal vectors are shown in binary format



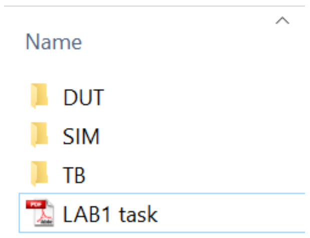
```
      ps-     /tb/Y-   /tb/X-    /tb/ALUout-
     delta-              /tb/ALUFN-/tb/Nflag-
                                     /tb/Cflag-
                                       /tb/Zflag-
                                         /tb/Vflag-

      0   +8 |11111111 11111111 01000 11111110 1 1 0 0
  50000   +7 |11111110 11110101 01000 11110011 1 1 0 0
 100000   +6 |11111101 11101011 01001 00010010 0 1 0 0
 150000   +6 |11111100 11100001 01001 00011011 0 1 0 0
 200000   +8 |11111011 11010111 01010 00101001 0 0 0 0
 250000   +6 |11111010 11001101 01010 00110011 0 0 0 0
 300000   +9 |11111001 11000011 01011 11111010 1 0 0 0
 350000   +6 |11111000 10111001 01011 11111001 1 0 0 0
 400000   +8 |11110111 10101111 01100 11110110 1 1 0 0
 450000   +6 |11110110 10100101 01100 11110101 1 1 0 0
 500000  +10 |11110101 10011011 01000 10010000 1 1 0 0
 550000   +8 |11110100 10010001 01000 10000101 1 1 0 0
 600000   +7 |11110011 10000111 01001 01101100 0 1 0 0
 650000   +9 |11110010 01111101 01001 01110101 0 1 0 1
 700000   +7 |11110001 01110011 01111 00000000 0 0 1 0
 750000   +1 |11110000 01101001 01111 00000000 0 0 1 0
 800000   +9 |11101111 01011111 10000 10000000 1 1 0 0
 850000   +8 |11101110 01010101 10000 11000000 1 1 0 0
 900000   +7 |11101101 01001011 10001 00011101 0 1 0 0
 950000   +8 |11101100 01000001 10001 01110110 0 0 0 0
1000000   +9 |11101011 00110111 10010 00000000 0 0 1 0
1050000   +1 |11101010 00101101 10010 00000000 0 0 1 0
1100000   +9 |11101001 00100011 10000 01001000 0 1 0 0
1150000   +8 |11101000 00011001 10000 11010000 1 1 0 0
1200000   +7 |11100111 00001111 10001 00000001 0 1 0 0
1250000   +7 |11100110 00000101 10001 00000111 0 0 0 0
1300000   +9 |11100101 11111011 10010 00000000 0 0 1 0
1350000   +1 |11100100 11110001 10010 00000000 0 0 1 0
1400000   +5 |11100011 11100111 11001 11100111 1 0 0 0
1450000   +5 |11100010 11011101 11001 11111111 1 0 0 0
1500000   +5 |11100001 11010011 11010 11000001 1 0 0 0
1550000   +5 |11100000 11001001 11010 11000000 1 0 0 0
1600000   +5 |11011111 10111111 11101 01100000 0 0 0 0
1650000   +5 |11011110 10110101 11101 01101011 0 0 0 0
1700000   +5 |11011101 10101011 11111 10001001 1 0 0 0
1750000   +5 |11011100 10100001 11111 10000010 1 0 0 0
1800000   +5 |11011011 10010111 11011 01001100 0 0 0 0
1850000   +5 |11011010 10001101 11011 01010111 0 0 0 0
1900000   +3 |11011001 10000011 00100 00000000 0 0 1 0
1950000   +1 |11011000 01111001 00100 00000000 0 0 1 0
2000000   +1 |11010111 01101111 00100 00000000 0 0 1 0
```

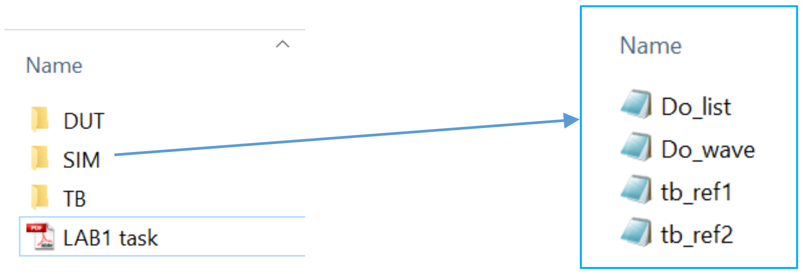
Signals ***Y, X,ALUFN, ALUout*** are shown in HEX format (the same exact example as above).



In addition, you are given two test bench files ***tb_ref1.vhd, tb_ref2.vhd*** (in TB folder) and their associate do and list files (in SIM folder).

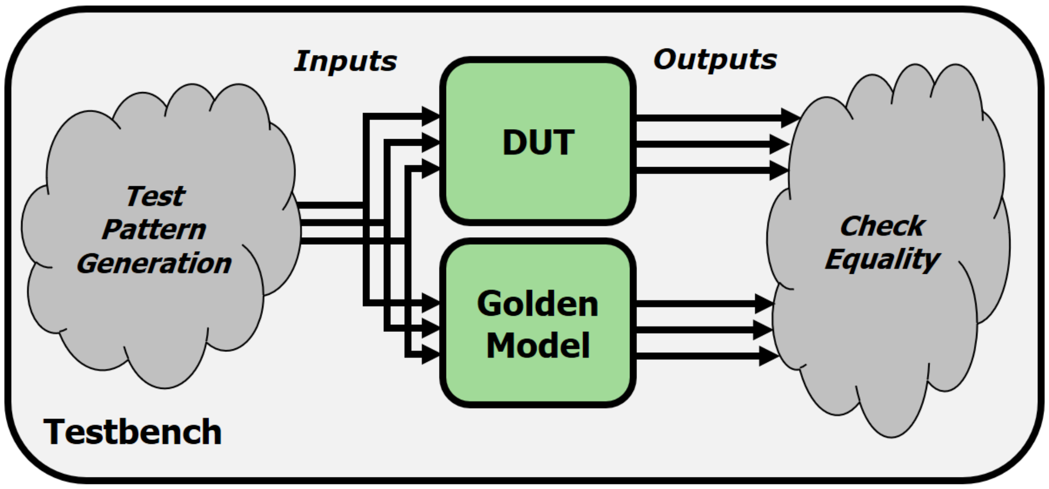


In order to use golden model based functional verification, you are given TextDiff application (download and double click the ***TextDiff.exe*** file) in order to compare your developing design results to the golden model results as part of design developing chain.



Automatic Testbench

The DUT **output** is compared against the **golden model**



Note: in comparison between the given ***tb_ref1.lst, tb_ref2.lst*** files and yours, you should ignore the ***delta cycle*** column

9. Requirements

1. The lab assignment is in pairs (as shown in the inlay file).
2. The design must be well commented.
3. **Important:** For each of two submodules:
   - Graphical description (a square with ports going in and out) and short descriptions.
4. Elaborated analysis and wave forms:
   - Remove irrelevant signals.
   - Zoom on regions of interest.
   - Draw clouds on the waveform with explanations of what is happening (Figure 4).
   - Change the waveform colors in ModelSim for clear documentation **(Tools->Edit Preferences->Wave Windows).**
5. A ZIP file in the form of **id1_id2.zip** (where id1 and id2 are the identification number of the submitters, and id1 < id2) *must be upload to Moodle only by student with id1* (any of these rules violation disqualify the task submission).

6. The **ZIP** file will contain (***only the exact next sub folders***):

Directory	**Contains**	**Comments**
DUT	Project VHDL files	**Only VHDL files of DUT**, excluding test bench **Note: your project files must be well compiled without errors as a basic condition before submission**
TB	***Four*** VHDL files that are used for test bench	AdderSub, Logic, shifter, System (top)
SIM	***Four*** ModelSim DO files (wave, list)	AdderSub, Logic, shifter, System (top)
DOC	Project documentation	• ***readme.txt*** (list of the DUT **.vhd* files with their brief functional description) • ***pre1.pdf*** (report file that includes brief explanation of the four modules with their wave diagrams as shown in figure 4)

Table 2: Directory Structure

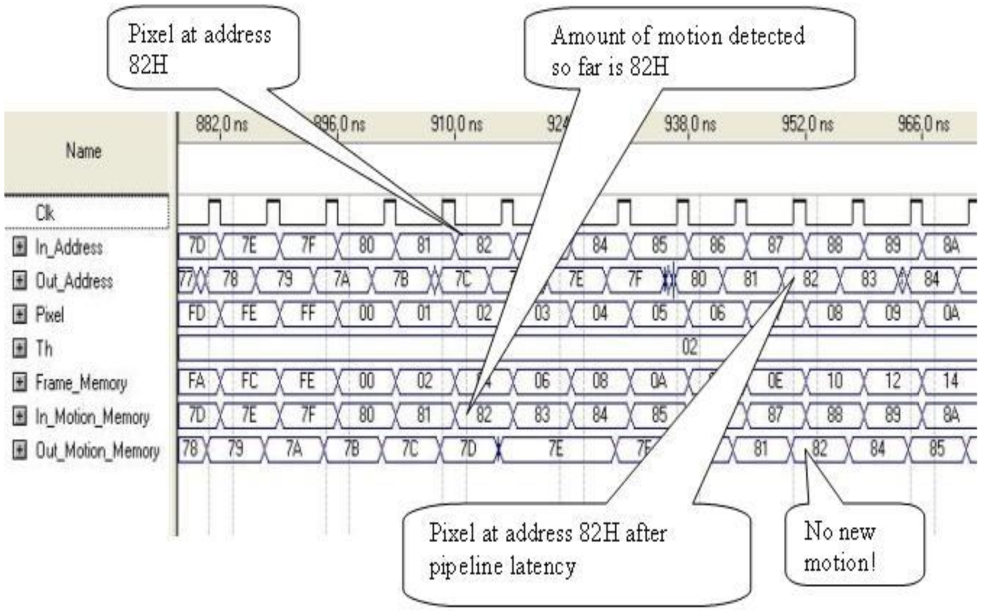


Figure 4: Clouds over the waveform

10. Grading Policy

Weight	Task	Description
3%	Correct submission	The required four folders are the immediate down folders under the submission id1_id2.zip folder
7%	Documentation	The "clear" way in which you presented the requirements and the analysis and conclusions on the work you've done
90%	Analysis and Test	The correct analysis of the system (under the requirements)

Table 1 : Grading

Under the above policies you'll be also evaluated using common sense:

- Your files will be compiled and checked; the system must work.
- Your design and architecture must be intelligent, minimal, effective, and well organized.