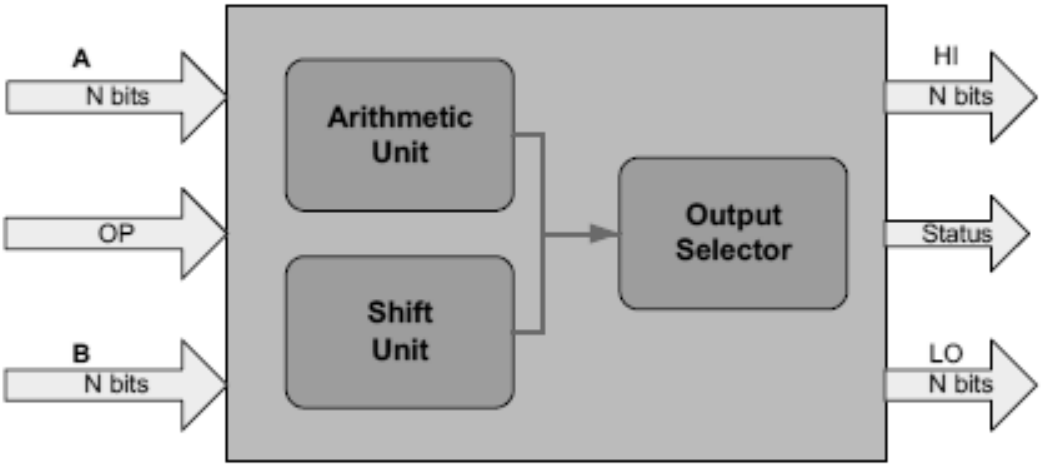
Lab 1

Architecture of CPU

Designing a basic ALU

VHDL & Modelsim



Maor Assayag 318550746
Refael Shetrit 204654891
Facilitators: Boris Braginsky & Prof. Hugo Guterman

Table of contents

General Description

1.1 Aims of the Laboratory

The aims of this laboratory are to Obtaining basic skills in VHDL and ModelSim, General knowledge rehearsal in digital systems and proper analysis and understanding of architecture design.

1.2 Assignment definition

In this laboratory you will design a basic ALU. The ALU will have following features:

- Configurable input bus width between 8 and 32bit (using generic variable N)
- Two input Busses
- Two Output buses
- One status bus output and an op-code input
- **Design choice**: the numbers on the bus will be **represent as signed** number (MSB is '0' for positive, '1' for negative)

1.3 Workspace & language

- ModelSim ALTERA STARTER EDITION 10.1b
- VHDL (2008's syntax)
- ATOM editor version 1.25.1

System Design

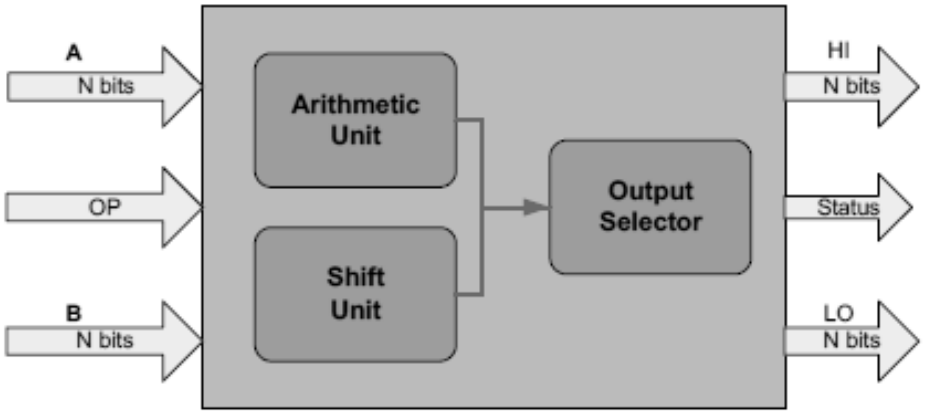


Figure 1: Overall system operation

The following table describes all the available ALU Op codes and operations:

OP	OP code	Operation	Description
MUL	0001	$(HI, LO) = A * B$	Multiply two signed numbers (Result is N*2 bits)
MAC	0000	$MAC = MAC + A * B$ $(HI, LO) = MAC$	Multiply Accumulate (MAC is internal N*2 bits register) signed numbers
MAX	0010	$LO = Max\,(A, B)$	Return maximum between A and B
MIN	0011	$LO = Min\,(A, B)$	Return minimum between A and B
ADD	0100	$LO = A + B$	Arithmetic Add
SUB	0101	$LO = A - B$	Arithmetic Sub
RST	0110	$MAC = 0$	Reset MAC
SHR	1001	$LO = A >> B$	SHR LO = A >> B Shift right
SHL	1000	$LO = A << B$	Shift left

Table 1: ALU Op Codes

Do not use additional arithmetic hardware for MAC operation; utilize the ADD/SUB and MULT hardware instead.
The Status bus of the ALU outputs the comparison status of the ALU in case of SUB operation, do not use comparators for each condition, but utilize the ADD/SUB hardware instead (use carry output of the adder and comparison of output to zero).

The following table describes the status bus signals:

Status Flag Name	Condition
$\boldsymbol{eq}$	$A = B$
$\boldsymbol{ne}$	$A != B$
$\boldsymbol{ge}$	$A >= B$
$\boldsymbol{gt}$	$A > B$
$\boldsymbol{le}$	$A <= B$
$\boldsymbol{lt}$	$A < B$

Table 2: Status Bus

The Top Level ALU design must be structural and contain the following entities:

- Arithmetic Entity (For MUL, MAC, ADD and SUB operations)
- Shift Entity (For SHL and SHR operations)
- Output selector that formulates the output busses and status

The adder for Add/Sub function must be designed both behaviorally and **structurally** (using basic gates AND, OR, NOT, XOR, NAND, NOR). Other entities can be designed behaviorally, structurally or mixed.

The synchronous parts of the system will be constructed using Flip-Flops (DFF). Other entities can be designed behaviorally, structurally or mixed.

Modules Description

3.1 Full Adder

File name: full_adder.vhd

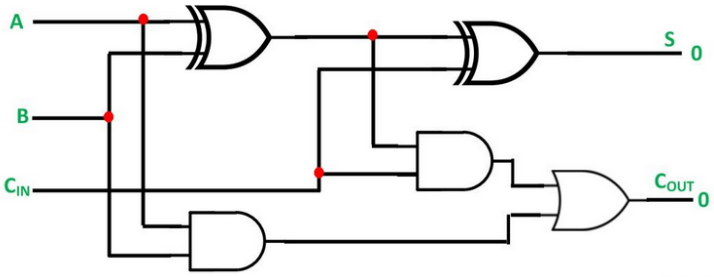


Figure 3.1 : Graphical description for : Full Adder

Port name	direction	type & size	functionality
a	*in*	*std_logic*	*bit A*
b	*in*	*std_logic*	*bit B*
Cin	*in*	*std_logic*	*bit Cin*
s	*out*	*std_logic*	*bit S*
Cout	*out*	*std_logic*	*bit Cout*

Table 3.1.1: Port Table for : Full Adder

Description: 2-bit full adder with carry in\out. Designed as a component for the Adder **structural** architecture entity..

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 3.1.2 : Logic Table for : Full Adder

3.2 Adder

File name: add.vhd

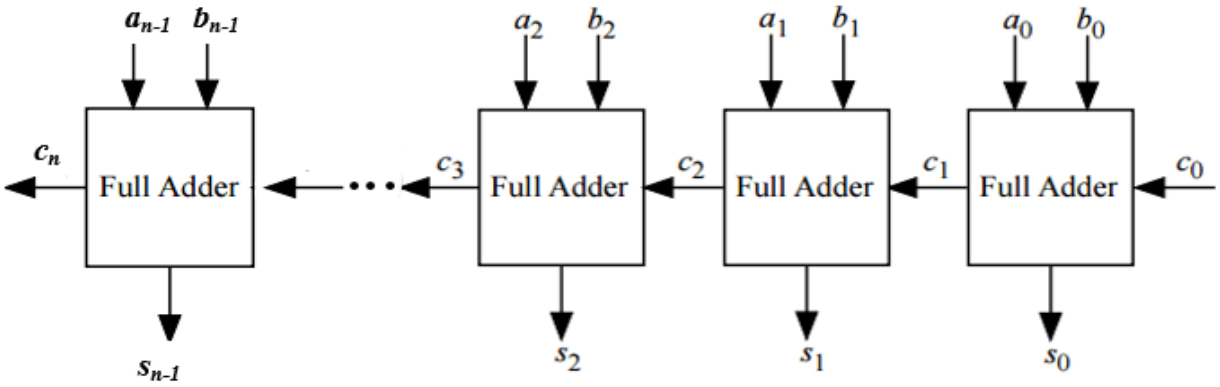


Figure 3.2 : Graphical description for : Adder n-bit

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
Cin	*in*	*std_logic (1 bit)*	*Carry bit in*
A	*in*	*signed (N bits)*	*Number A*
B	*in*	*signed (N bits)*	*Number B*
Sum	*out*	*signed (N bits)*	*Sum = A+B*
Cout	*out*	*std_logic (1 bit)*	*Carry bit out*

Table 3.2: Port Table for: Adder

Description: n-bit Adder designed using 2-bit full-adders with carry in\out (for-generate). The design is with structural architecture as an aid component for ADD/SUB entities.

3.3 XOR GATE

File name: xor.vhd

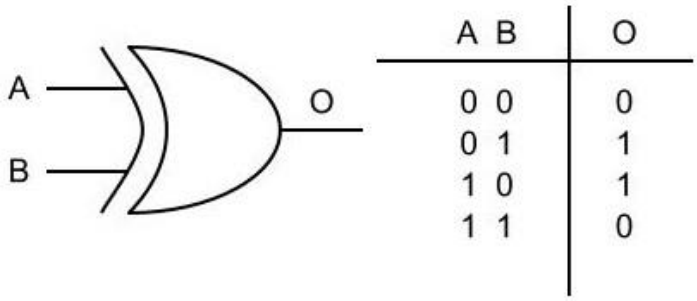


Figure 3.3: Graphical description for: XOR gate

Port name	direction	type & size	functionality
A	*in*	*Std_logic, 1 bit*	*Bit A*
B	*in*	*Std_logic 1 bit*	*Bit B*
C	*out*	*Std_logic 1 bit*	*C = A XOR B*

Table 3.3: Port Table for: XOR gate

Description: xor gate of 2 inputs (1 bit each) that generate 1-bit output. Design with behavioral architecture as an aid component for ADD/SUB entities.

3.4 ADD/SUB operations

File name: ADD_SUB.vhd

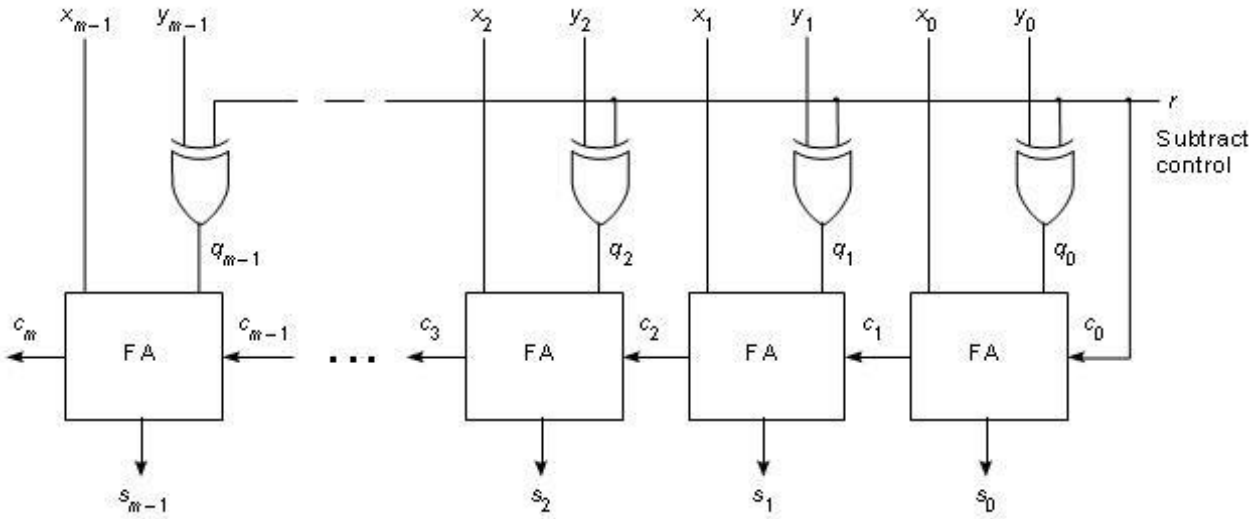


Figure 3.4: Graphical description for: Adder/Subtractor n-bit

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
addORsub	*In*	*std_logic (1 bit)*	*'1' for sub, '0' for add*
A	*in*	*signed (N bits)*	*Number A*
B	*in*	*signed (N bits)*	*Number B*
Sum	*out*	*signed (N bits)*	*Sum = A+B*

Table 3.4: Port Table for: Adder/Subtractor n-bit

Description: n-bit Adder/Subtractor designed using 2-bit full-adders with carry in\out (ADD component). The design is with structural architecture.

3.5 MAX/MIN operation

File name: MAX_MIN.vhd

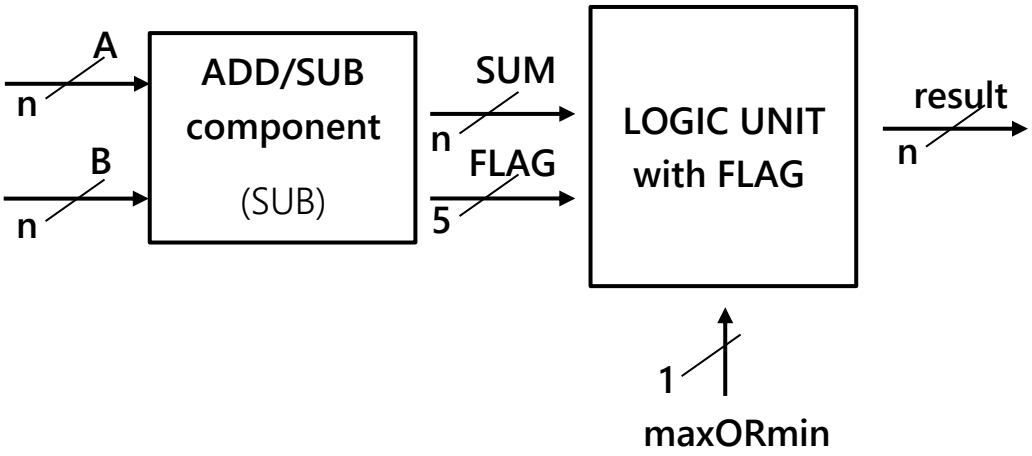


Figure 3.5: Graphical description for: MAX/MIN operation

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
maxORmin	*in*	*std_logic (1 bit)*	*Operation selector bit*
A	*in*	*signed (N bits)*	*Number A*
B	*in*	*signed (N bits)*	*Number B*
result	*out*	*signed (N bits)*	*C = MAX/MIN(A,B)*

Table 3.5: Port Table for: MAX/MIN operation

Description: max/min operation. Input 2 Numbers (N bits each) and 1-bit operation selector for max or min operation. The design is with behavioral architecture with the aid component ADD_SUB. After using the SUB operation, we can know the order between A and B from calculate the FLAGS.

3.6 Shift Left/Right (1-bit shifter)

File name: shift_Nbits.vhd

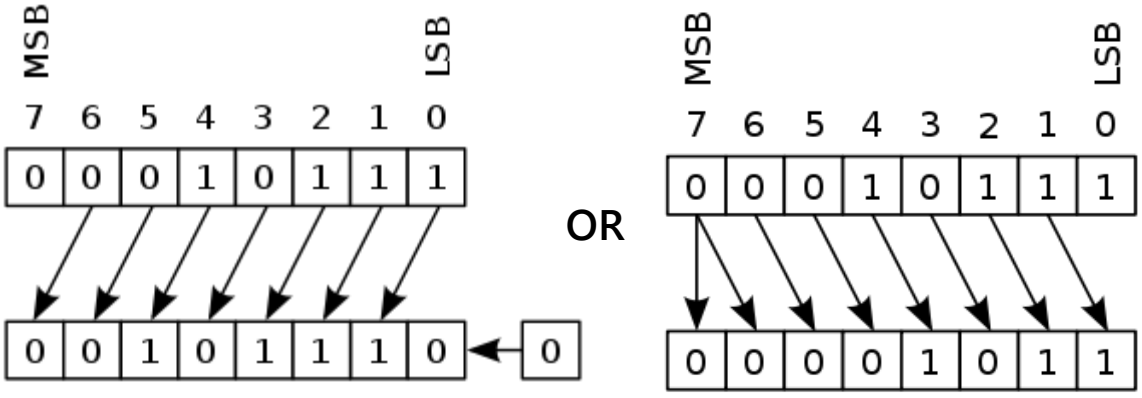


Figure 3.6: Graphical description for: Shift Left/Right (1-bit shifter)

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
dir	*In*	*std_logic (1 bit)*	*'0' left '1' right*
enable	*In*	*std_logic (1 bit)*	*'0' – Aout=A, '1' – Aout is the shifted number*
A	*in*	*signed (N bits)*	*Number A*
Aout	*out*	*signed (N bits)*	*Shifted Number A*

Table 3.6: Port Table for: Shift Left/Right (1-bit shifter)

Description: 1-bit shifter (1 bit to the left/right) that generate N bit output. The design is with **structural architecture** as an aid component for the TOP design shift unit. If enable = '0' then the output will be the input A. The shift unit will generate 64 shifters and will passing enables according to the required number B.

3.8 Shift Unit (TOP design)

File name: shift_unit.vhd

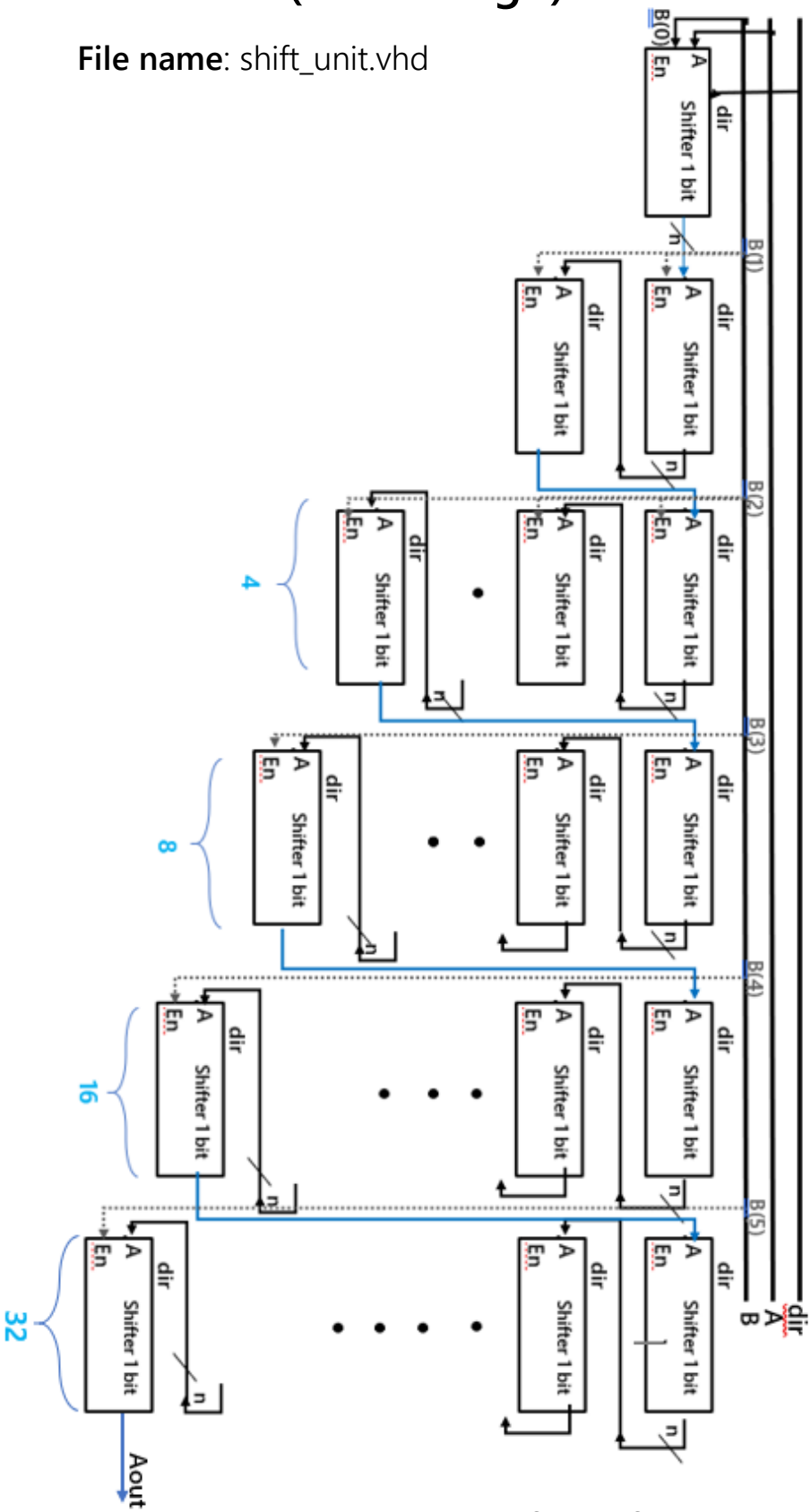


Figure 3.8: Graphical description for: Shift Unit (B-bits shifter)

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
dir	*In*	*std_logic (1 bit)*	*'0' left '1' right*
A	*in*	*signed (N bits)*	*Number A*
B	*in*	*signed (N bits)*	*Number B*
result	*out*	*signed (N bits)*	*Result = A >> B*

Table 3.8: Port Table for: Shift Unit (B-bits shifter)

Description: |B|-bits shifter (to the right/left) that generate N bit output with **Barrel** logic. The design is with **structural architecture** with the aid of the structural component shift_Nbits as required.

3.9 Mux 2N-N bit

File name: MUX_Nbits.vhd

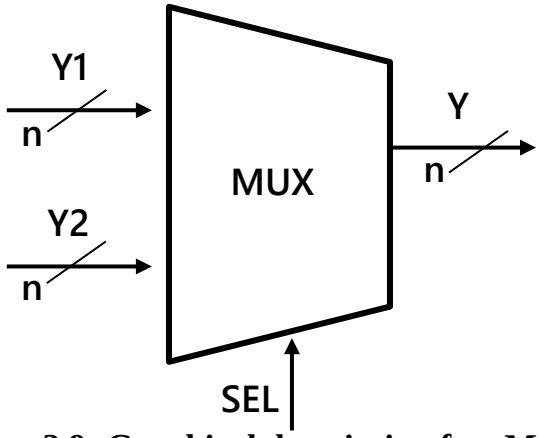


Figure 3.9: Graphical description for: Mux 2N-N bit

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
SEL	*In*	*std_logic(1 bit)*	*Selection bit*
Y1	*in*	*signed (N bit)*	
Y2	*In*	*signed (N bit)*	
Y	*out*	*signed (N bit)*	*Y1 \ Y2, according to SEL*

Table 3.9: Port Table for: Mux 2N-N bit

Description: 2N-N mux with behavioral architecture.
Designed as an aid component for general use.

3.10 MUL operation

File name: MUL.vhd

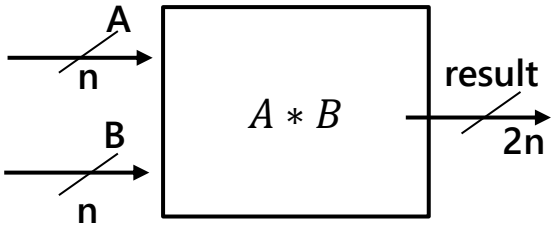


Figure 3.10: Graphical description for: MUL operation

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
A	*in*	*signed (N bits)*	*Number A*
B	*in*	*signed (N bits)*	*Number B*
result	*out*	*signed (2*N bits)*	*A*B*

Table 3.10: Port Table for: MUL operation

Description: MUL operation. Input 2 Numbers (N bits each). The design is with **behavioral architecture**. Support Signed numbers.

3.11 MAC operation

File name: MAC.vhd

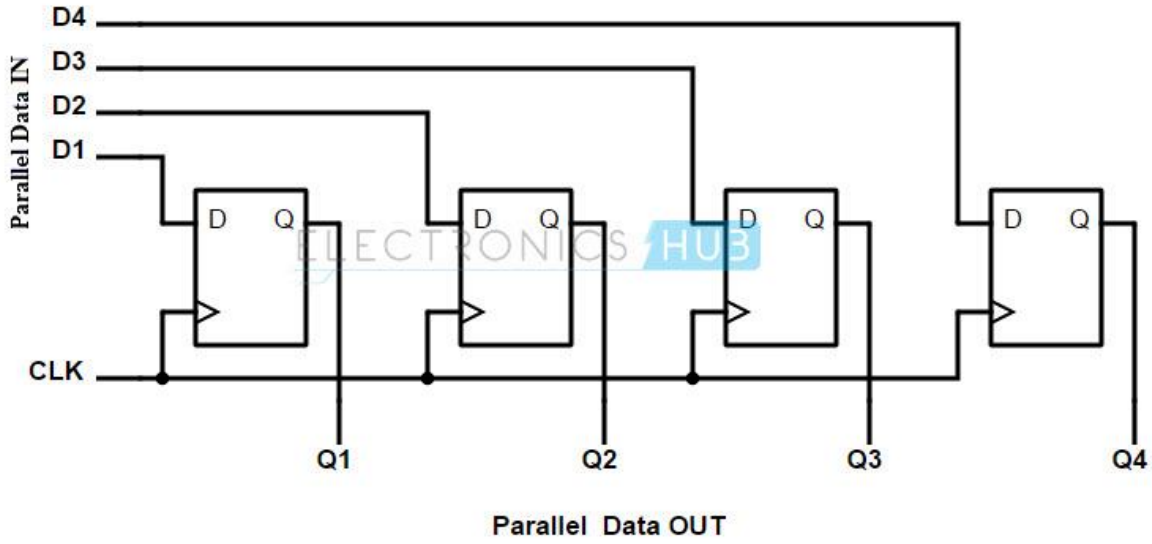


Figure 3.11: Graphical description for: MAC operation, example for 4-bits.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
mac_rst	*In*	*std_logic (1 bit)*	*Reset bit*
clk	*In*	*std_logic (1 bit)*	*clock bit*
enable	*In*	*std_logic (1 bit)*	*enable bit*
LO_BITS	*in*	*signed (N bits)*	*Number A*
HI_BITS	*in*	*signed (N bits)*	*Number B*
MAC_result	*out*	*signed (2*N bits)*	*= MAC+ A*B*

Table 3.11: Port Table for: MAC operation

Description: MAC operation. Designed as N dff with the inputs clk, enable & mac_rst (reset the register), Designed with **behavioral architecture** with the aid components dff_1bit.

3.12 Output Selector UNIT

File name: Output_Selector.vhd

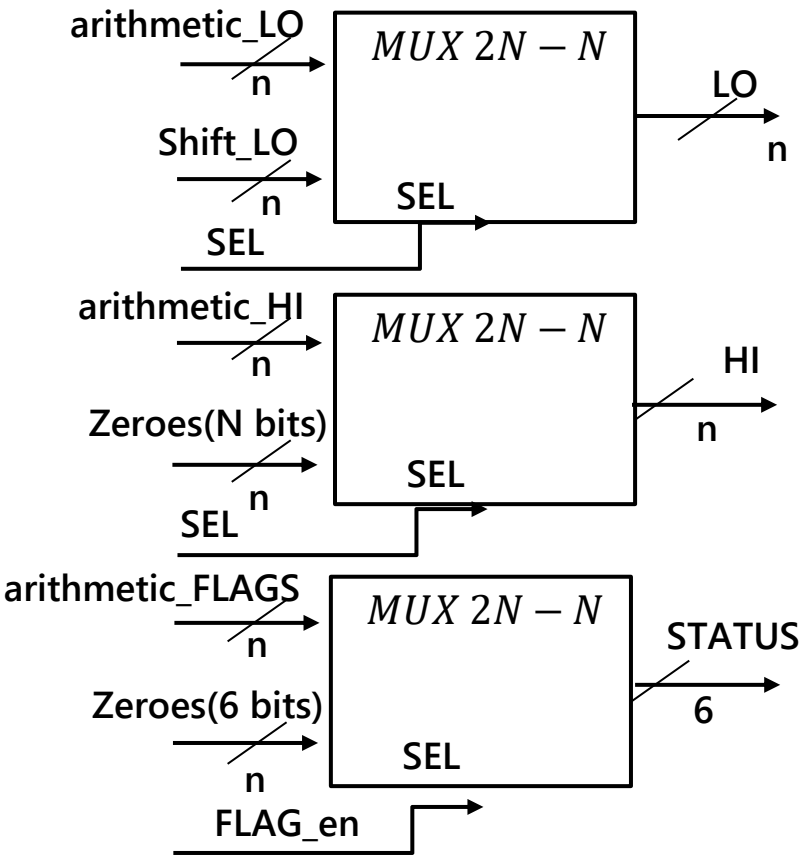


Figure 3.12: Graphical description for: Output Selector UNIT

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
SEL	*In*	*std_logic (1 bit)*	*'0' : arithmetic, '1' : shift*
FLAG_en	*In*	*std_logic (1 bit)*	*'1' : SUB OPP*
arithmetic_LO	*in*	*signed (N bits)*	
arithmetic_HI	*in*	*signed (N bits)*	
arithmetic_FLAG	*in*	*signed (N bits)*	*SYSTEM FLAGS*
Shift_LO	*in*	*signed (N bits)*	
LO	*out*	*signed (N bits)*	
HI	*out*	*signed (N bits)*	
STATUS	*out*	*signed (6 bits)*	*SYSTEM FLAGS*

Table 3.12: Port Table for: Output Selector UNIT

Description: Output Selector UNIT – TOP design. Input 2 numbers that represent the result from the arithmetic unit(N bits each), the result from the shift UNIT(N bit), SEL which is the selector bit from the OPP(3) and FLAG_en which indicate that the OPP was SUB (then we need to update the STATUS bus). The design is with **structural architecture**.

3.13 Arithmetic Selector

File name: Arithmetic_selector.vhd

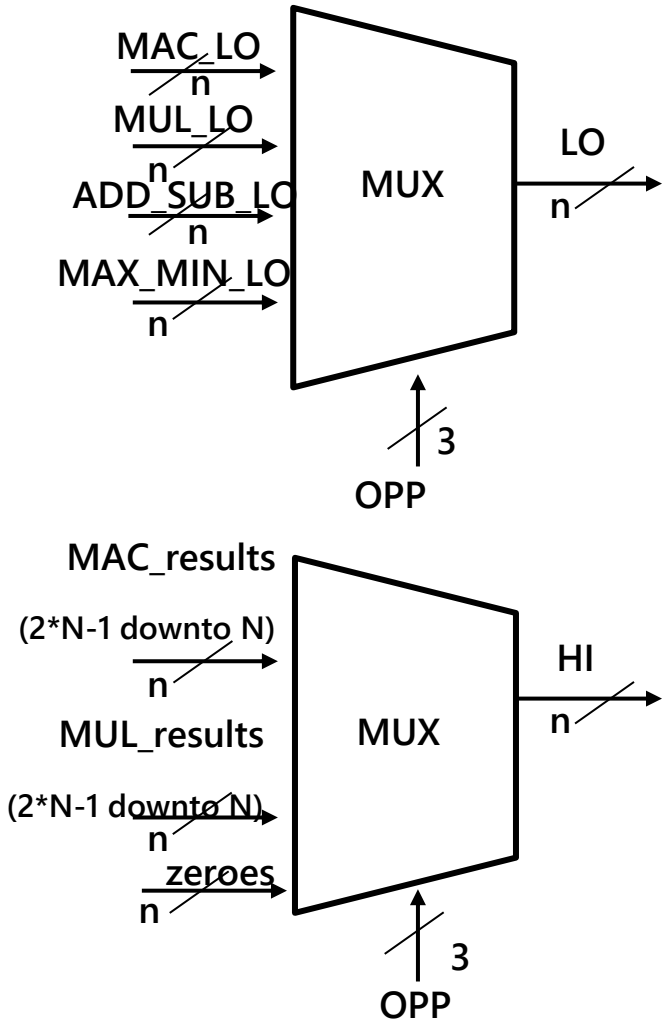


Figure 3.13: Graphical description for: Arithmetic Selector entity

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
OPP	*In*	*std_logic (3 bit)*	*OPP CODE*
MUL_results	*in*	*signed (2*N bits)*	
MAC_results	*in*	*signed (2*N bits)*	
MAX_MIN_LO	*in*	*signed (N bits)*	
ADD_SUB_LO	*in*	*signed (N bits)*	
LO	*out*	*signed (N bits)*	*Result*
HI	*out*	*signed (N bits)*	*result*
FLAG_en	*out*	*std_logic (1 bit)*	*'1' : SUB OPP*

Table 3.13: Port Table for: Arithmetic Selector entity

Description: Arithmetic selector entitiy with **behavioral architecture**. Designed as an aid component for the Arithmetic TOP design UNIT.

3.14 Arithmetic UNIT (Top Design)

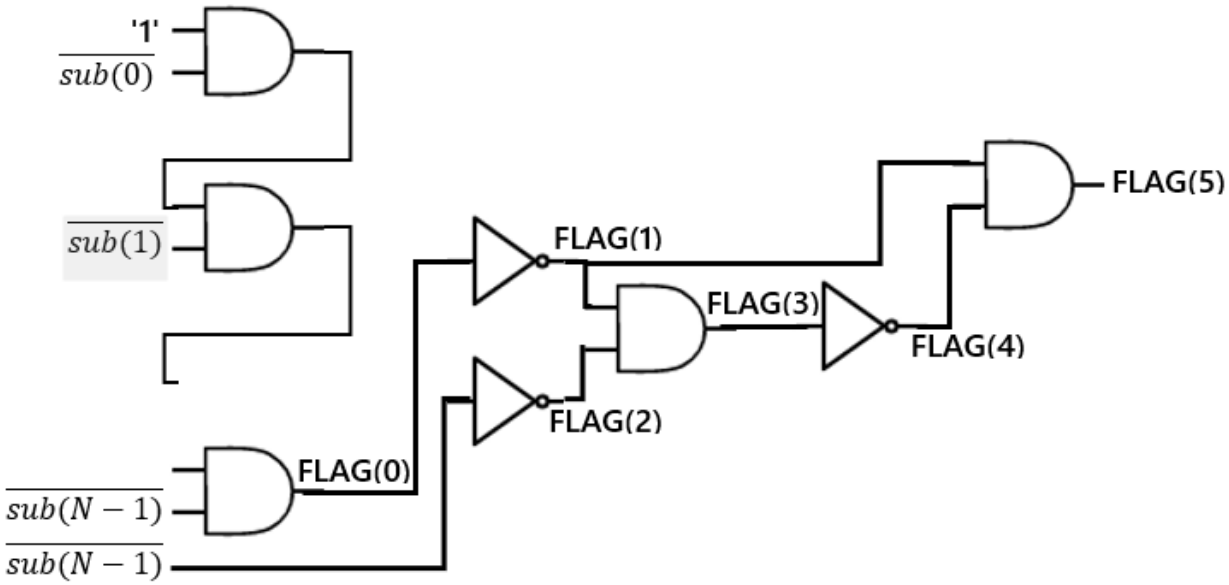


Figure 3.14: Graphical description for: FLAGS handling.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
clk	*In*	*std_logic (1 bit)*	*clock bit*
OPP	*In*	*std_logic (3 bits)*	*OPP CODE*
A	*in*	*signed (N bits)*	
B	*in*	*signed (N bits)*	
LO	*out*	*signed (N bits)*	*Result*
HI	*out*	*signed (N bits)*	*result*
FLAGS	*out*	*signed (6 bits)*	*FLAGS*
FLAG_en	*out*	*std_logic (1 bit)*	*'1' : SUB OPP*

Table 3.14: Port Table for: Arithmetic UNIT

Description: Arithmetic UNIT with **behavioral architecture**. The operation will be compute according to the OPP signal. Components: ADD_SUB, MUL, MAC, MAX_MIN, 2 MUX_2N & Arithmetic_selector.

2 MUX (N bits each) will **select** (internal signal's : selectedA, selectedB) the inputs of ADD_SUB hardware (MAC inputs OR

A,B), this way we can utilize the same hardware for both operations.

Last, we compute the SYSTEM FLAGS, which will update the STATUS BUS if the operation is SUB (later in the output selector unit).

3.15 ALU (Top Design)

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
Clk	*In*	*std_logic (1 bit)*	*System clock*
OPP	*In*	*std_logic (3 bits)*	*OPP CODE*
LO	*out*	*signed (N bits)*	*Result LOW bits*
HI	*out*	*signed (N bits)*	*Result HIGH bits*
STATUS	*out*	*signed (6 bit)*	*System STATUS*

Table 3.15: Port Table for: ALU (Top Design)

Description: ALU unit (Top Design) with **behavioral architecture**. The main entity of the ALU and the only entity that the user communicates with. **Components**: Arithemethic_unit, Shift_unit, Output_Selector.

3.16 basic d-flip-flop (dff)

File name: dff_1bit.vhd

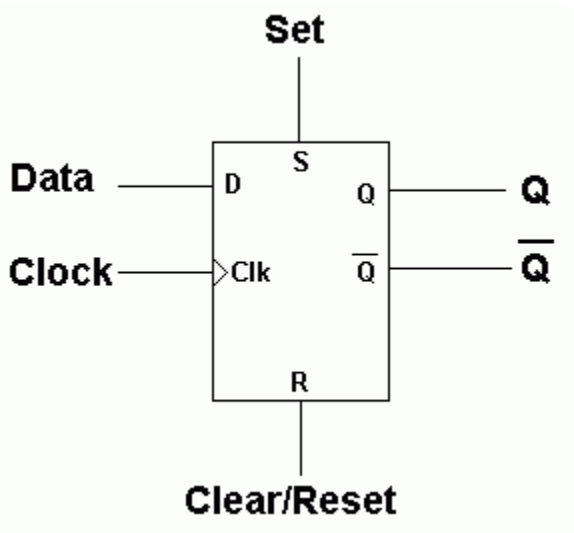


Figure 3.16: Graphical description for: 1-bit dff entity.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
clk	*In*	*std_logic (1 bit)*	*clock bit*
rst	*In*	*std_logic (1 bit)*	*reset bit*
d	*in*	*std_logic (1 bit)*	*bit d*
q	*in*	*std_logic (1 bit)*	*bit q*

Table 3.16: Port Table for: 1-bit dff entity

Description: 1-bit dff. Designed with **structural architecture** as an aid component for N-bits dff.

3.17 N dff's

File name: N_dff.vhd

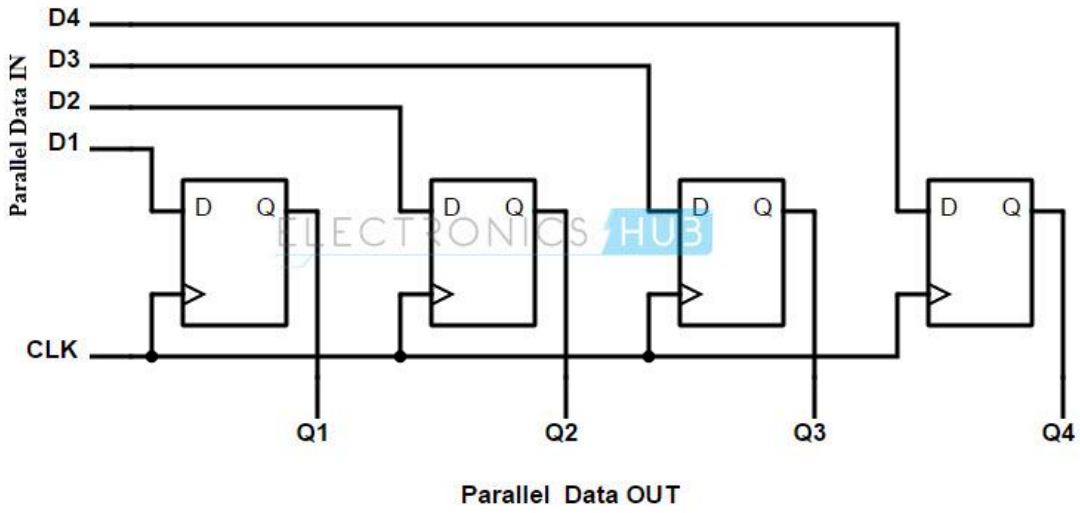


Figure 3.17: Graphical description for:(example N=4) N-bit dff entity.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
clk	*In*	*std_logic (1 bit)*	*clock bit*
rst	*In*	*std_logic (1 bit)*	*reset bit*
d	*in*	*std_logic (N bits)*	*Number D*
q	*in*	*std_logic (N bits)*	*Number Q*

Table 3.17: Port Table for: N-bit dff entity

Description: N-bit dff (which is really N 1-bit dffs). Designed with **structural architecture** as an aid component for MAC register. Component: 1bit_dff.

Bench Tests

4.1 full_adder.VHD

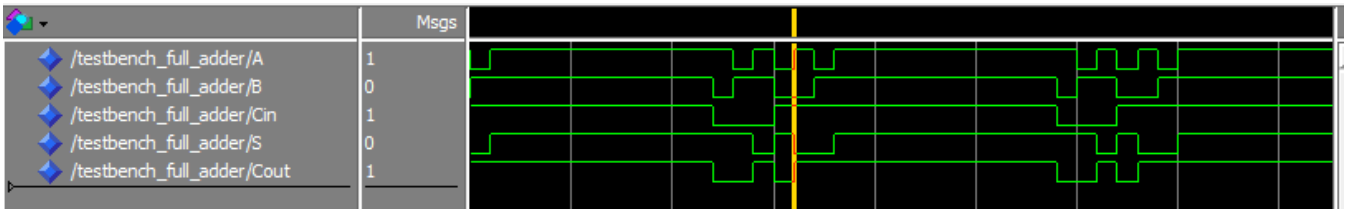


Figure 4.1: Test Bench for: full_adder entity

Description: '1'(A) + '0'(B) + '1'(Cin) = '0' ('1'' Cout)

4.2 ADD_SUB.VHD

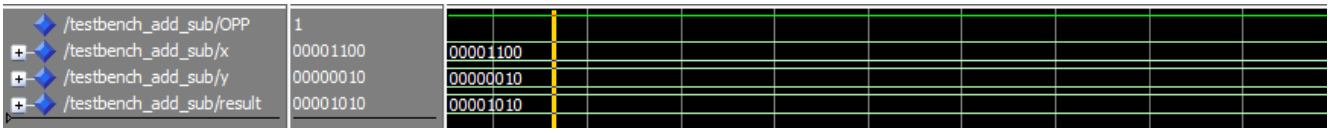


Figure 4.2: Test Bench for: ADD_SUB operation

Description: OPP: SUB ('1'): "1100"(12, x) – "0010"(2, y) = "1010"(10, result)

4.3 ADD.VHD

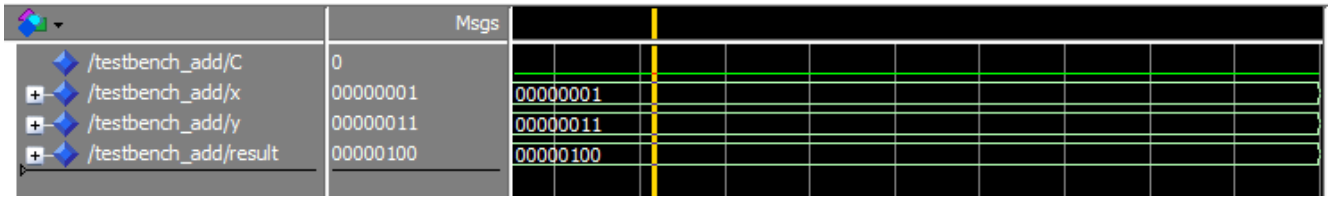


Figure 4.3: Test Bench for: ADD entity

Description: '0' (C) + "001"(1, x) + "011"(3, y) = "100"(4, result)

4.4 MAX_MIN.VHD

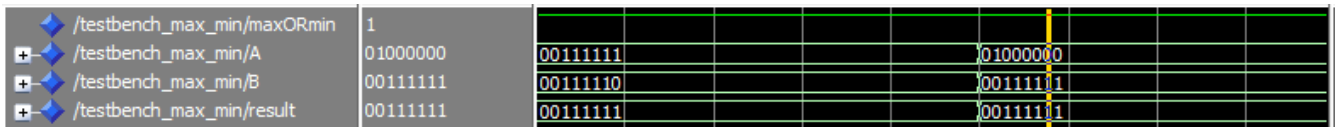


Figure 4.4: Test Bench for: MAX_MIN operation

Description: OPP : MIN (maxORmin = '1') : result = min(A,B) = B.

4.5 Output_Selector.VHD

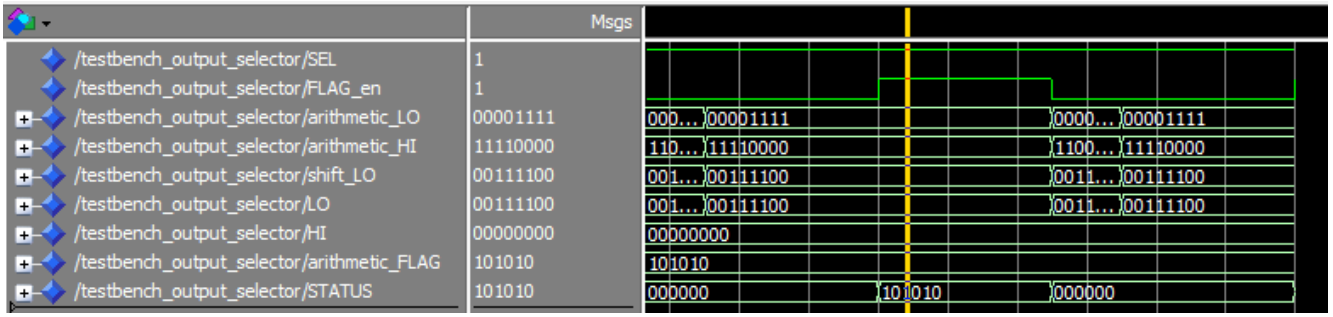


Figure 4.5: Test Bench for: Output_Selector unit

Description: shift_LO selected (SEL = '1'), FLAG_en update the STATUS with FLAG (this test isn't represent a real scenario, FLAG_en='1' indicate arithmetic opp, and SEL = '1' indicate shift opp).

4.6 MAC.VHD

Figure 4.6: Test Bench for: MAC entity

Description: MAC_result output only on rising edge, we can see the behavior of N dffs (N bit register).

4.7 MUL.VHD

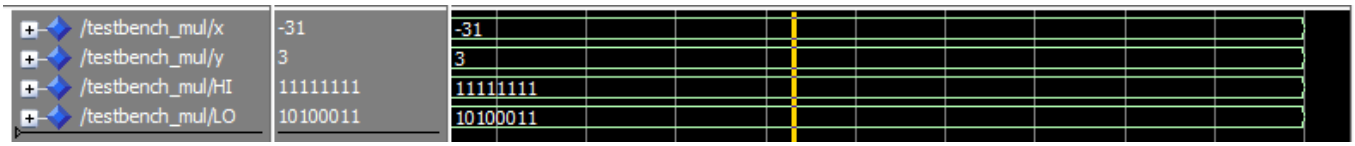


Figure 4.7: Test Bench for: MUL entity

Description: result = (HI,LO) = "1111111110100011" (-93) = -31(x) * 3(y).

4.8 diff_1bit.VHD

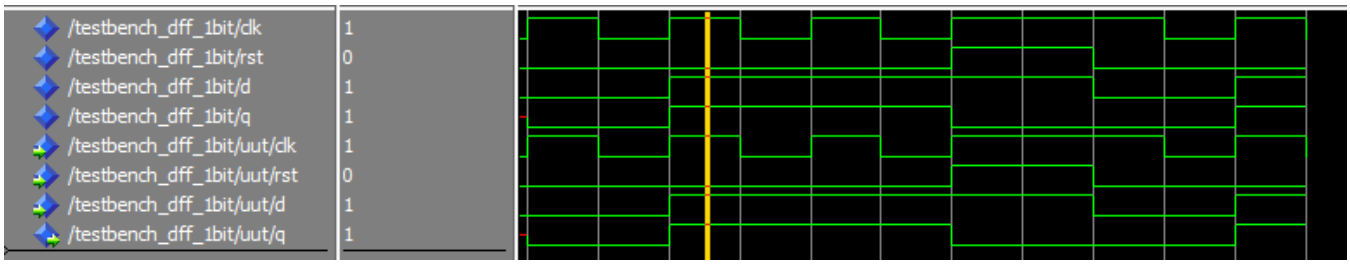


Figure 4.8: Test Bench for: diff_1bit entity

Description: if clk is rising edge then $d \rightarrow q$.

4.9 arithmetic_selector.VHD

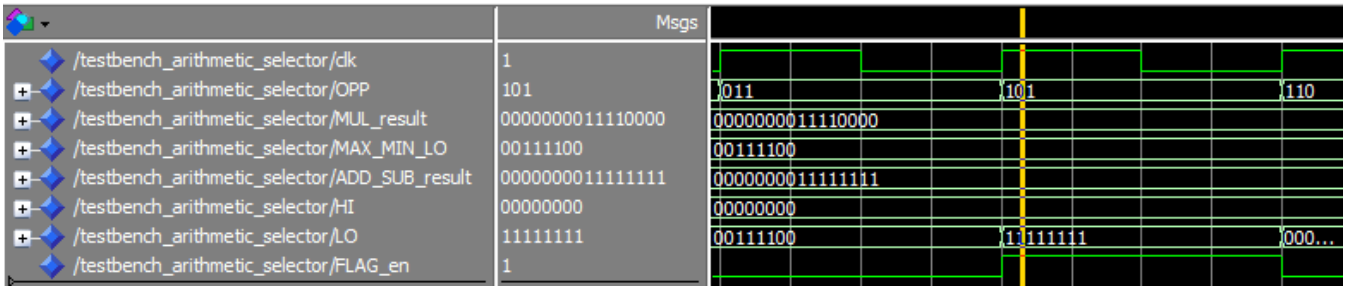


Figure 4.9: Test Bench for: arithmetic_selector entity

Description: OPP : SUB("0101"), then (HI,LO) = (zeroes, ADD_SUB_result(N-1 downto 0)).

4.10 arithmetic_unit.VHD

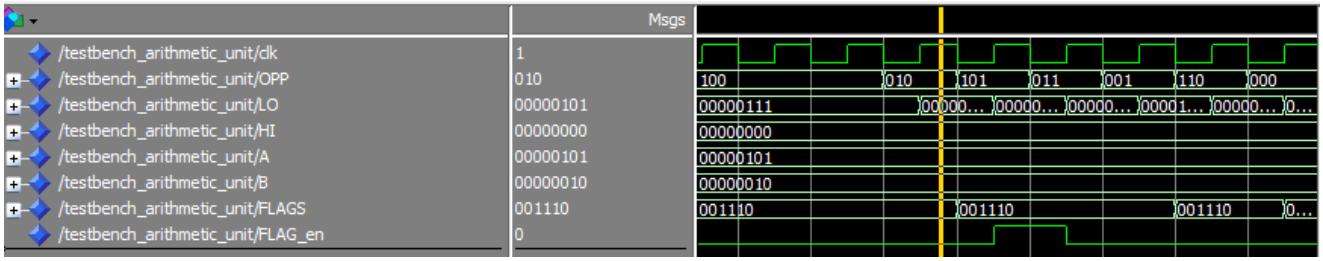


Figure 4.10: Test Bench for: arithmetic unit

Description: OPP : MAX ("0010"), HI=zeroes, LO = max(A,B)=A, FLAG_en = '0' (not a sub operation).

4.11 N_dff.VHD

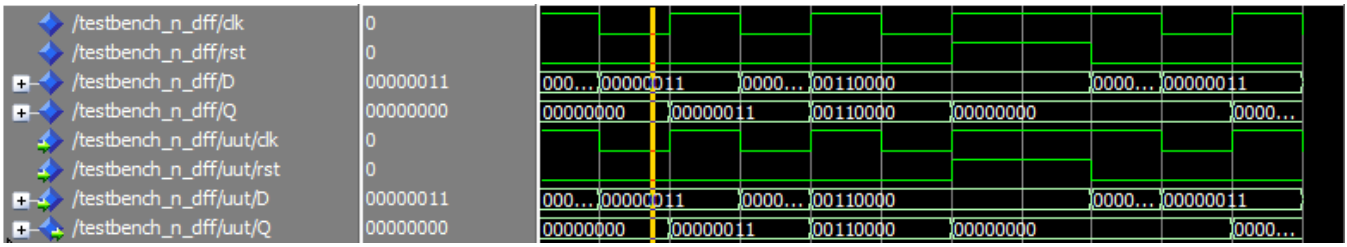


Figure 4.11: Test Bench for: N_dff entity

Description: if clk is rising edge then $D \rightarrow Q$.

4.12 mux_Nbits.VHD

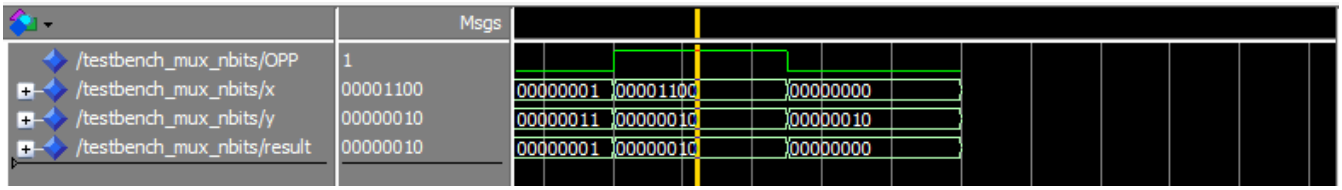


Figure 4.12: Test Bench for: mux_Nbits entity

Description: OPP(local signal in the test bench, it is SEL) : ='1', then result = y ('0' for result = x).

4.13 shift_Nbits.VHD

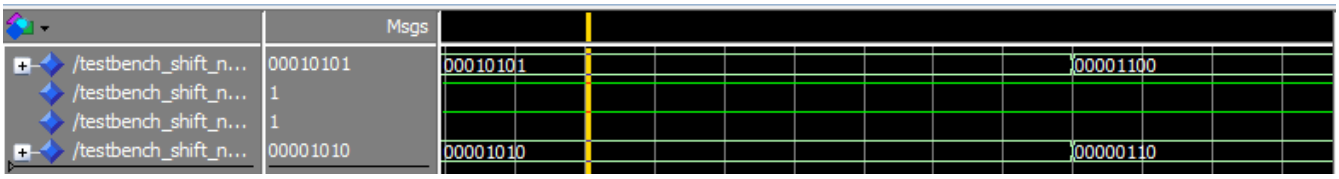


Figure 4.13: Test Bench for: shift_Nbits entity

Description: dir = '1' (shift to the right), enable = '1' (then shift the number instead of result=input).

4.14 ALU.VHD

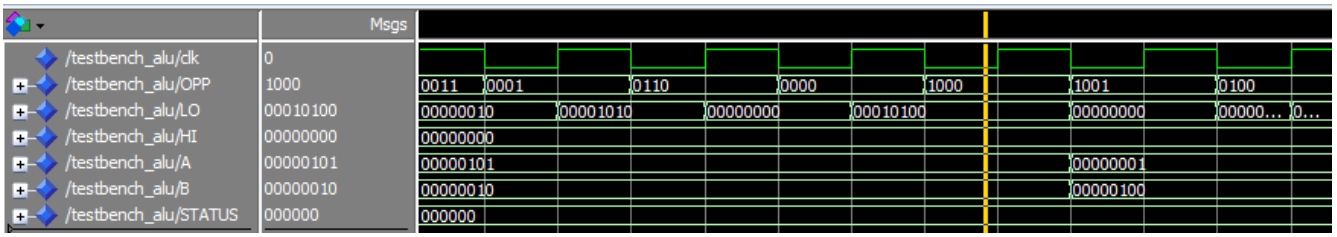


Figure 4.14: Test Bench for: ALU unit

Description: OPP : mac ("0000"), HI=zeroes, LO = "10100"(20), status = zeroes (not a sub operation), this is the second mac from the last RST, then the MAC=2 * A * B(=20) .

Attached files

- VHDL/ VHDL files
- TB/ Test Bench files
- DOC/ readme.txt – compilation order