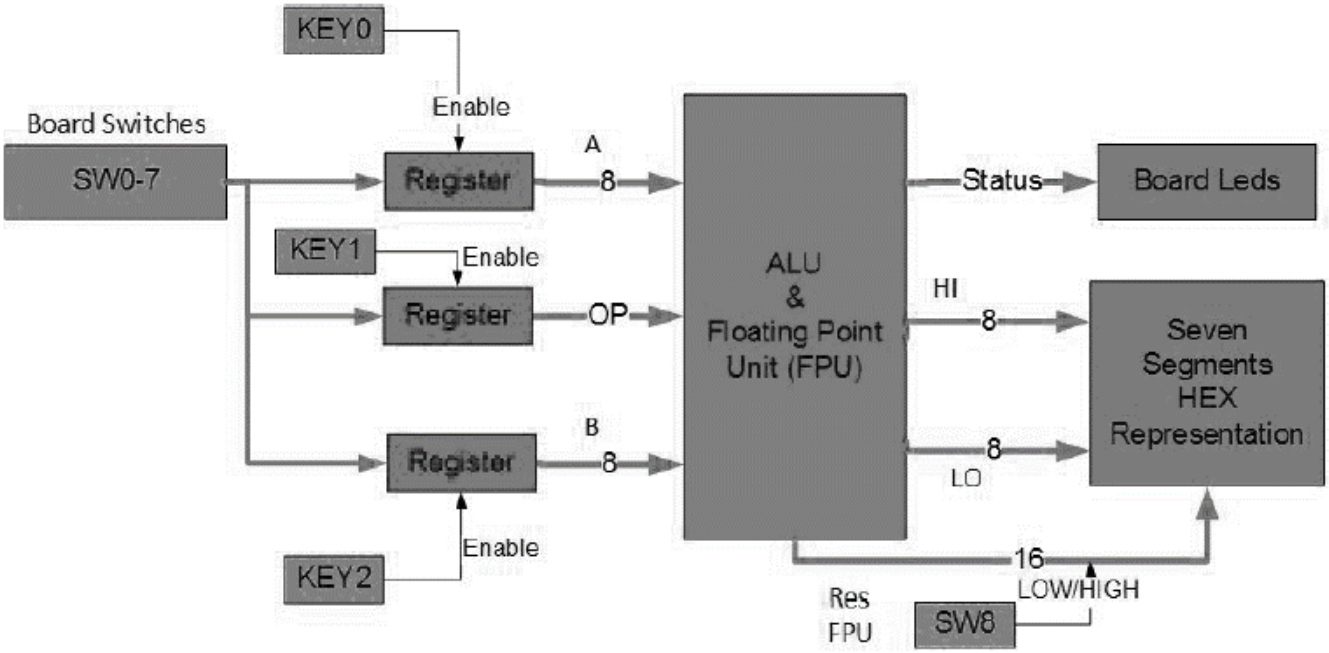
Lab 2

Architecture of CPU

Designing a basic ALU with FPU

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, proper analysis and understanding of architecture design, Understanding arithmetic synthesis and FPGA arithmetic limitations and Floating-point design.

1.2 Assignment definition

In this laboratory you will have to synthesize an ALU from the first assignment for the Cyclone II FPGA with impact on performance and logic usage. You will have to do the two following test cases for the 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)
- Floating-point IEEE standard commands

1.3 Workspace & language

- ModelSim ALTERA STARTER EDITION 10.1b
- VHDL (2008's syntax)
- ATOM editor version 1.25.1
- Quartus II 12.1 Web Edition (32-Bit) & Altera DE1 FPGA

System Design

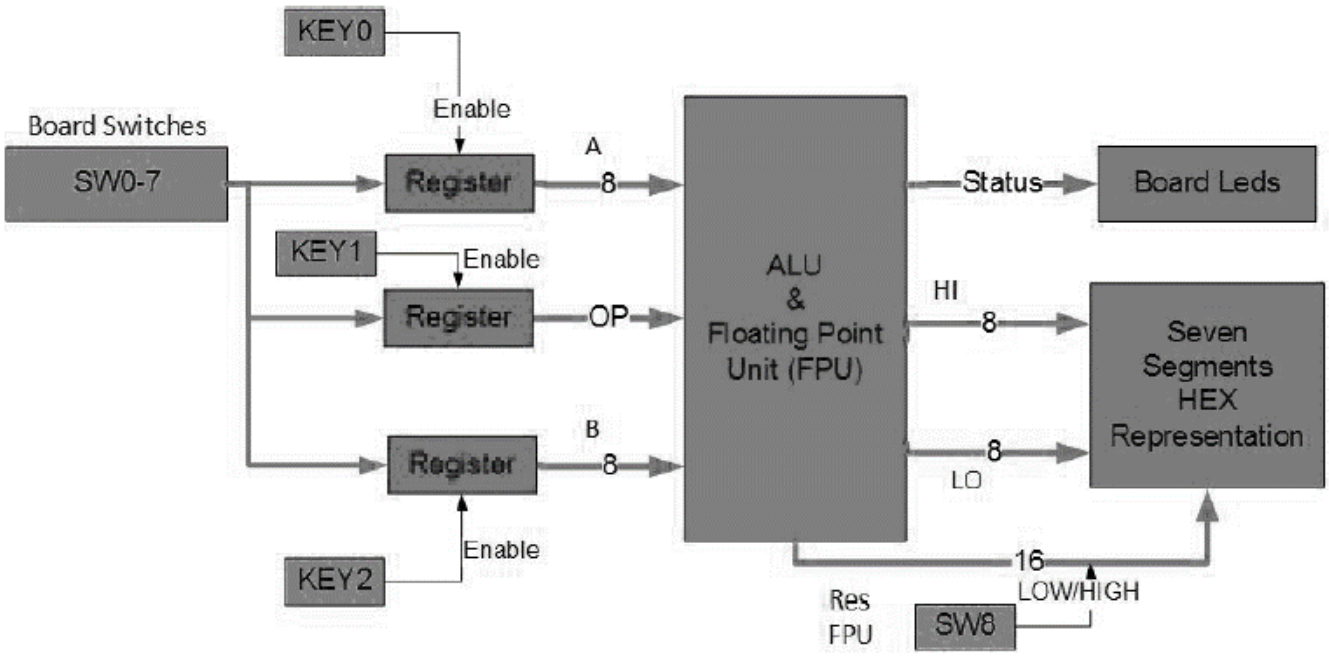


Figure 1: Overall system operation

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
ADDF	1010	$(HI, LO) = A + B$	ieee 754 2-numbers addition
MULLF	1000	$(HI, LO) = A * B$	ieee 754 2 numbers multiplication

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

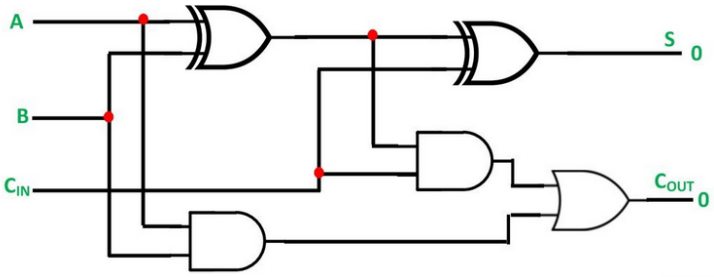


Figure 3.1 : Graphical description for : Full Adder

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

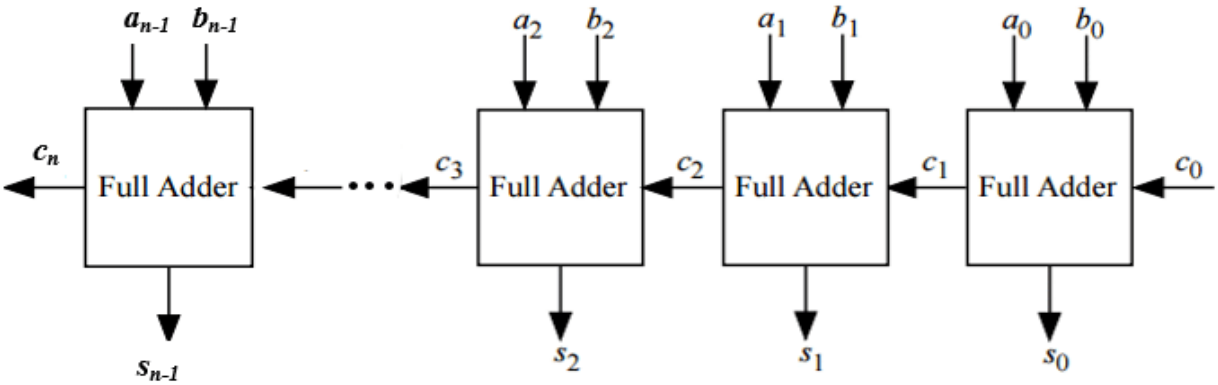


Figure 3.2 : Graphical description for : Adder n-bit

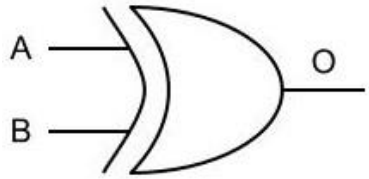
Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
Cin	*in*	*std_logic (1 bit)*	*Carry bit in*
A	*in*	*signed (N bits)*	*Number A*
B	*in*	*signed (N bits)*	*Number B*
Sum	*out*	*signed (N bits)*	*Sum = A+B*
Cout	*out*	*std_logic (1 bit)*	*Carry bit out*

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



A	B	O
0	0	0
0	1	1
1	0	1
1	1	0

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

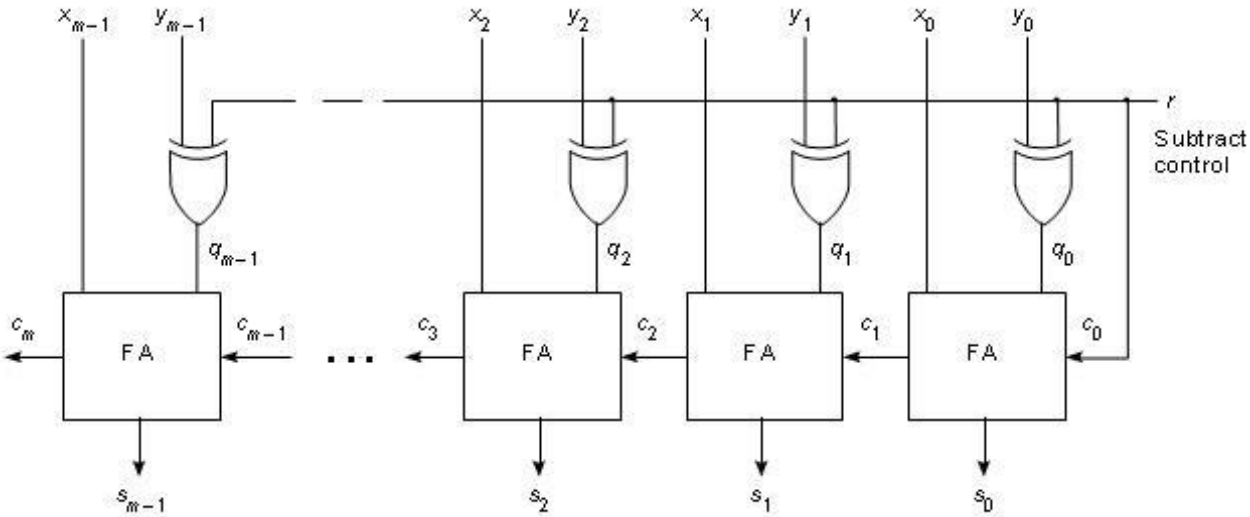


Figure 3.4: Graphical description for: Adder/Subtractor n-bit

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

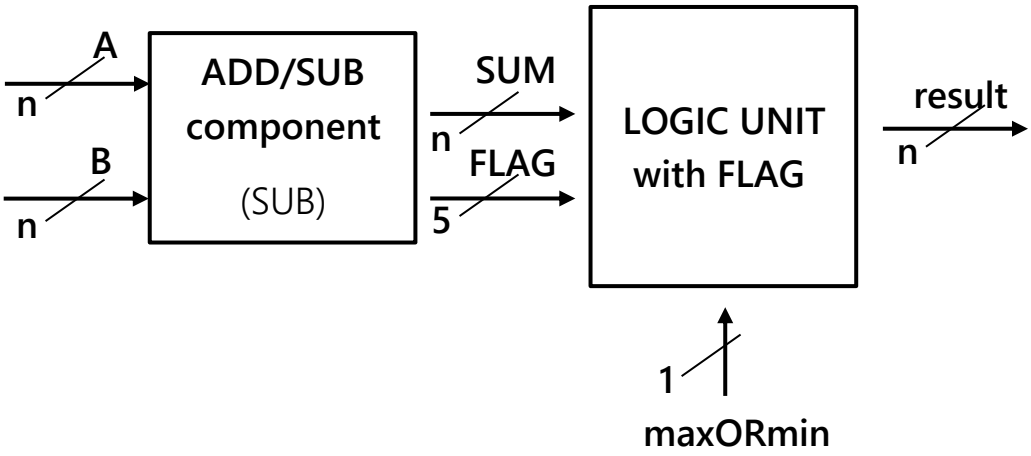


Figure 3.5: Graphical description for: MAX/MIN operation

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

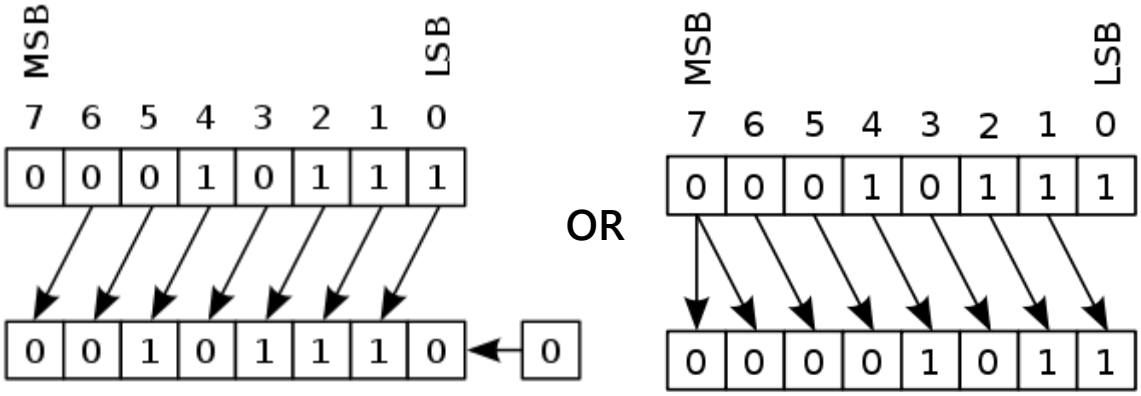


Figure 3.6: Graphical description for: Shift Left/Right (1-bit shifter)

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

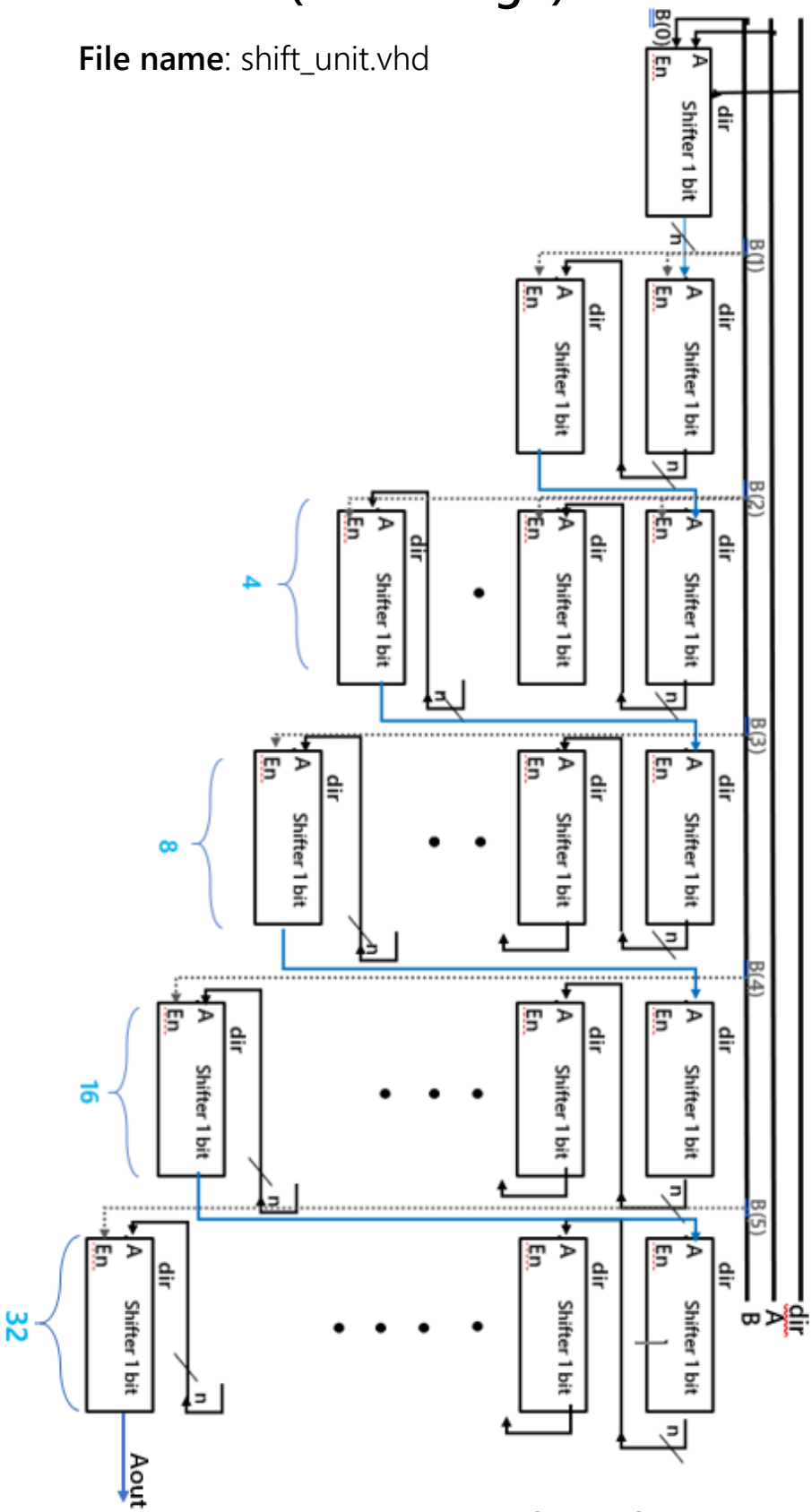


Figure 3.8: Graphical description for: Shift Unit (B-bits shifter)

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

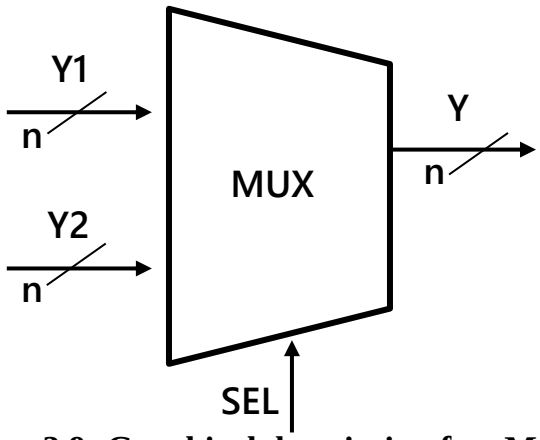


Figure 3.9: Graphical description for: Mux 2N-N bit

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

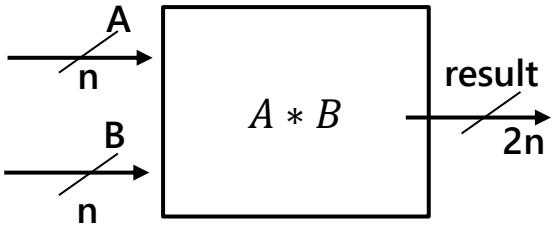


Figure 3.10: Graphical description for: MUL operation

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

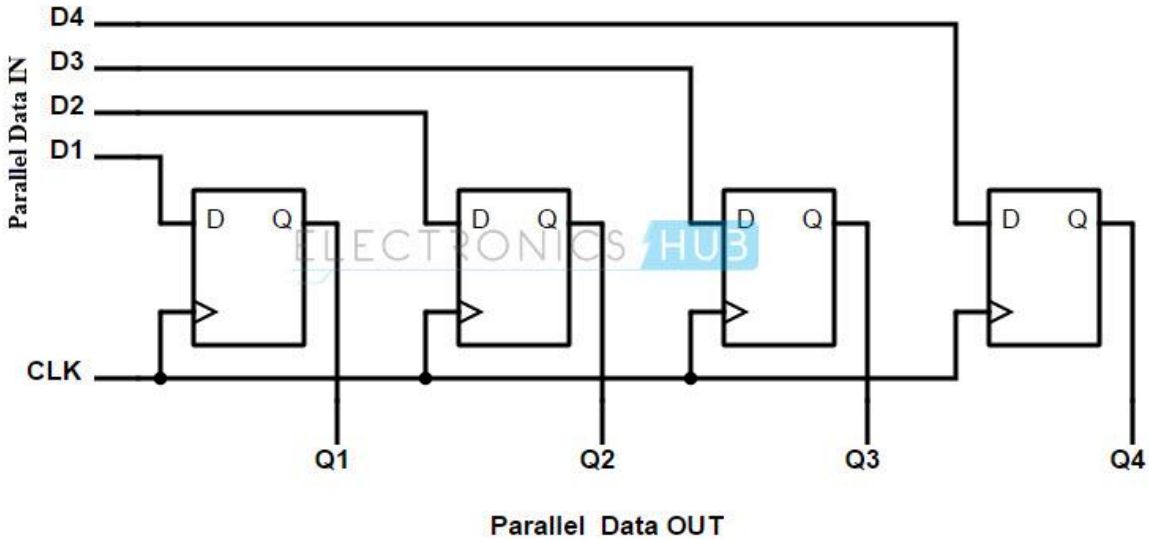


Figure 3.11: Graphical description for: MAC operation, example for 4-bits.

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

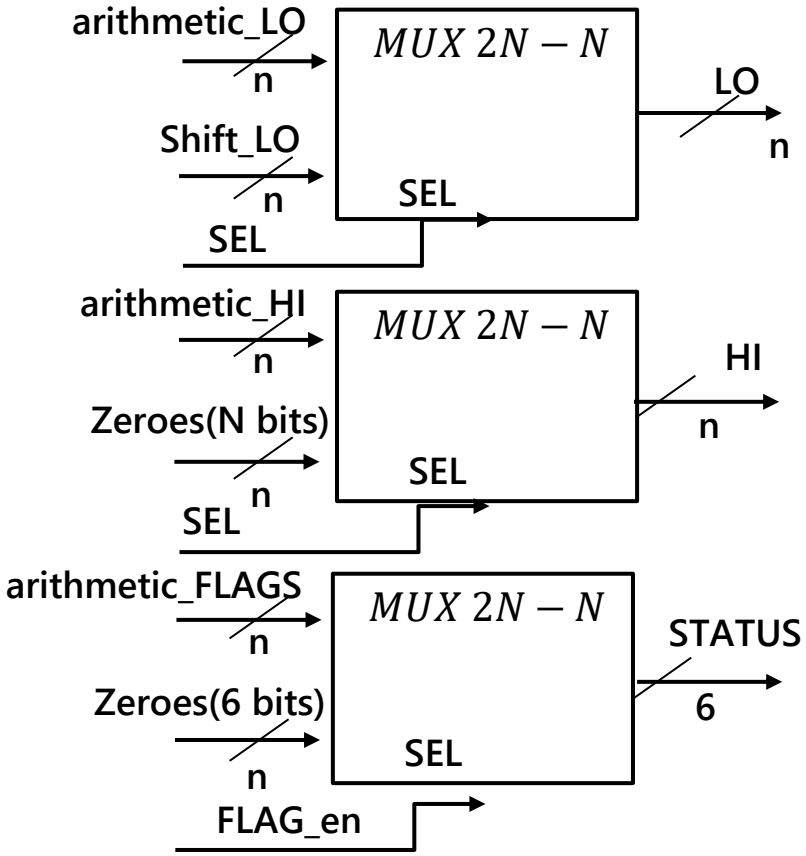


Figure 3.12: Graphical description for: Output Selector UNIT

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
SEL	*In*	*std_logic (1 bit)*	*'0' : arithmetic, '1' : shift*
FPU_SW	*in*	*std_logic (1 bit)*	*FPU MSB bits or LSB bits*
FPU_SEL	*In*	*std_logic (1 bit)*	*SHIFT UNIT or FPU UNIT*
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

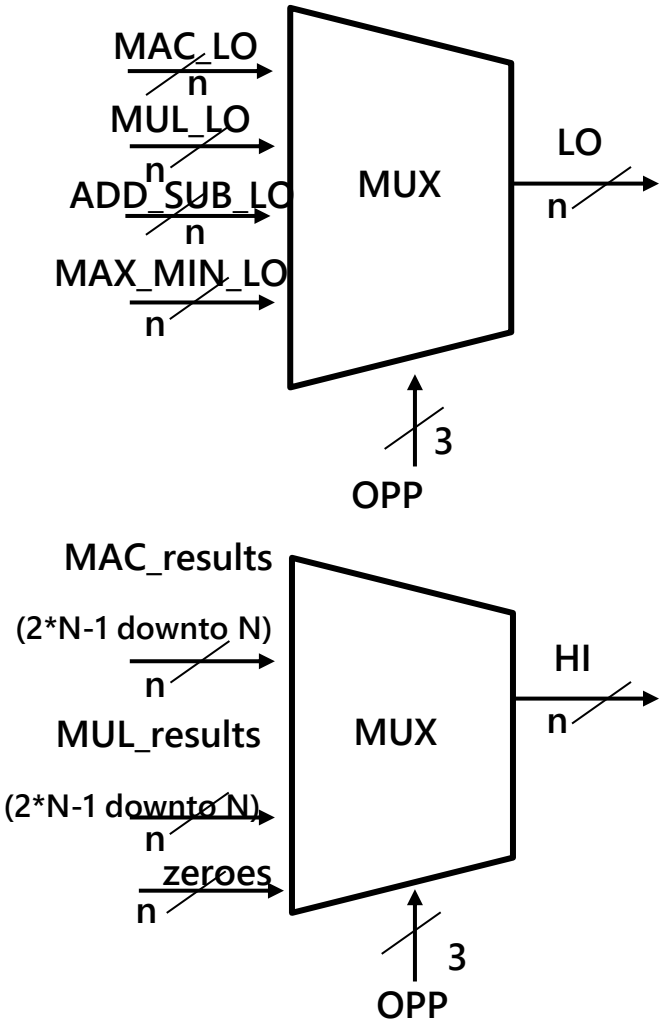


Figure 3.13: Graphical description for: Arithmetic Selector entity

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)

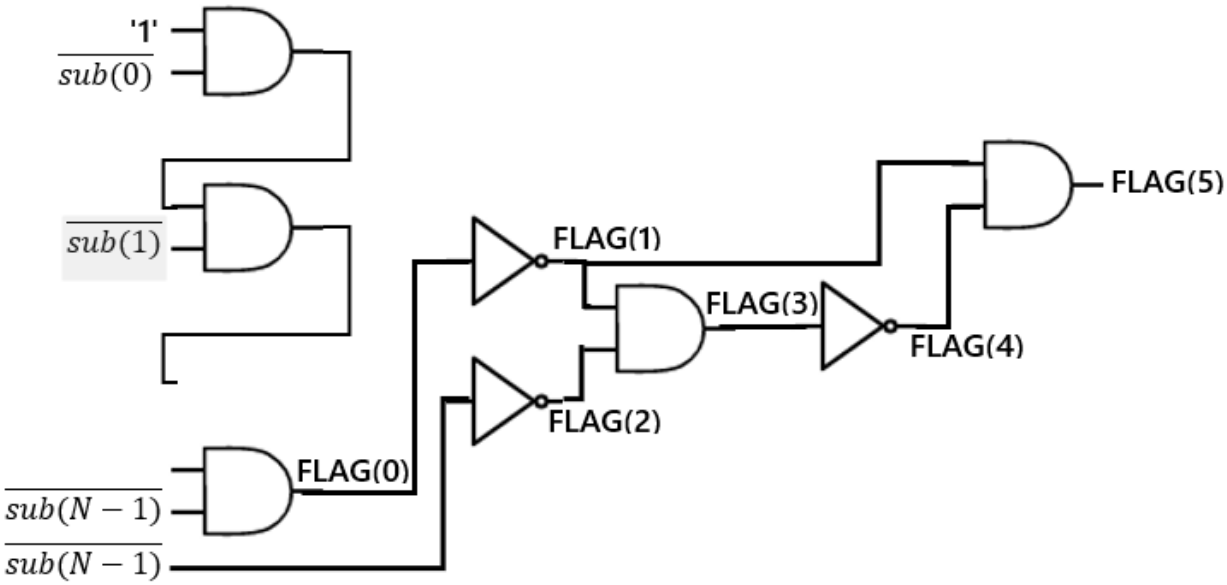


Figure 3.14: Graphical description for: FLAGS handling.

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*
FPU_SW	*in*	*std_logic (1 bit)*	*FPU MSB bits or LSB bits*
OPP	*In*	*std_logic (3 bits)*	*OPP CODE*
A	*In*	*signed (N bits)*	*Num A*
B	*in*	*signed (N bits)*	*Num B*
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 **structural architecture**. The main entity of the ALU and the only entity that the user communicates with. **Components**: Arithemethic_unit, Shift_unit, Output_Selector, FPU_Unit, MUX_Nbits, FloatingPointConvertor. If we dealing with floating point commands then the inputs are 2-8bits number, we need to convert them to 32bit ieee-754 standard -> inputs to the FPU.

3.16 basic d-flip-flop (dff)

File name: dff_1bit.vhd

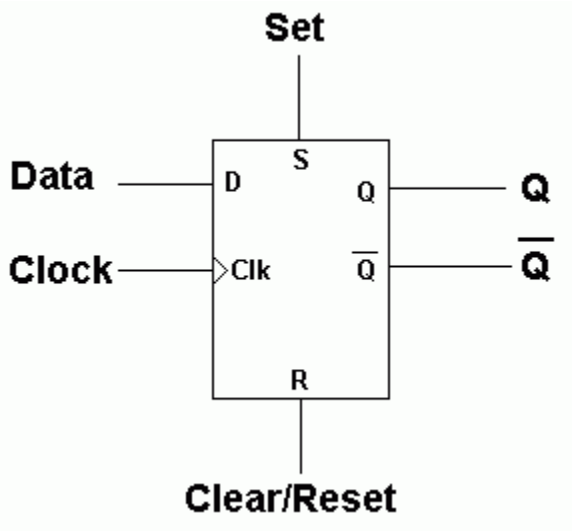


Figure 3.16: Graphical description for: 1-bit dff entity.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
en	*In*	*std_logic (1 bit)*	*Enable bit*
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

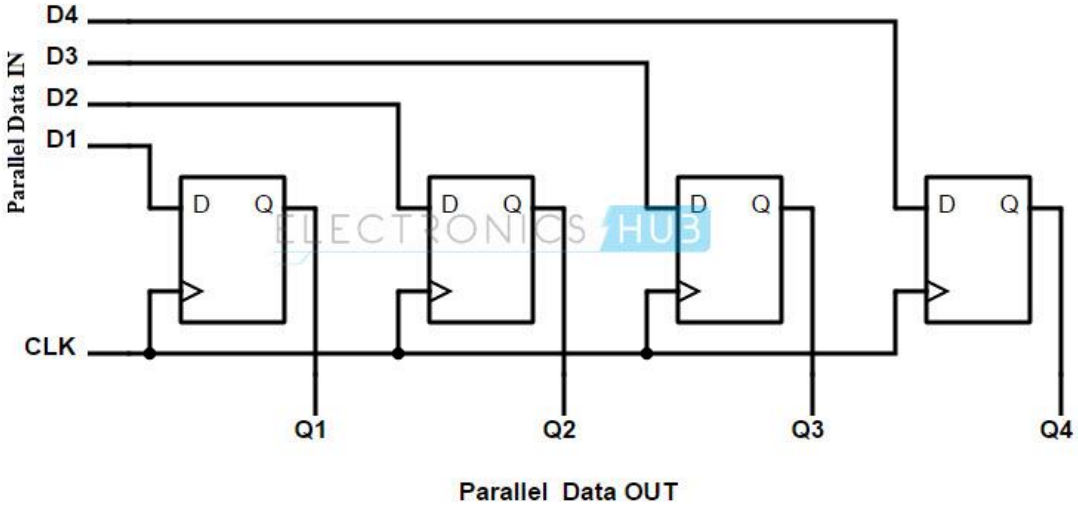


Figure 3.17: Graphical description for:(example N=4) N-bit dff entity.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
clk	*In*	*std_logic (1 bit)*	*clock bit*
enable	*In*	*std_logic (1 bit)*	*Enable 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.

3.18 Floating Point 8to32 bits Convertor

File name: FloatinPointConvertor.vhd

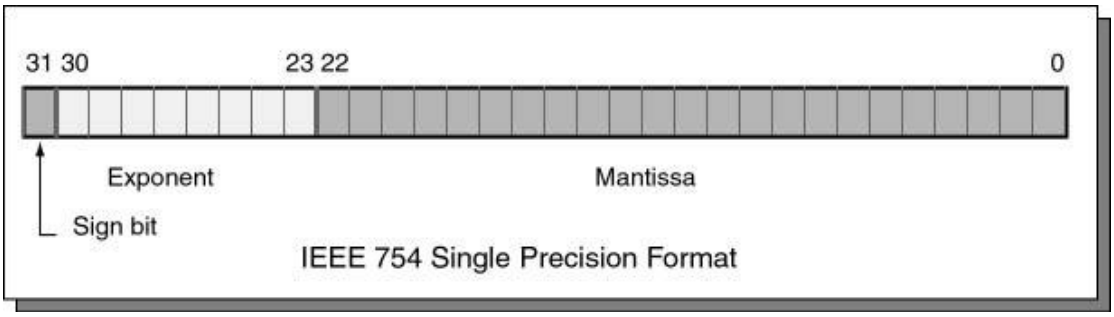


Figure 3.18: Graphical description for: 32 bits ieee 754 standard.

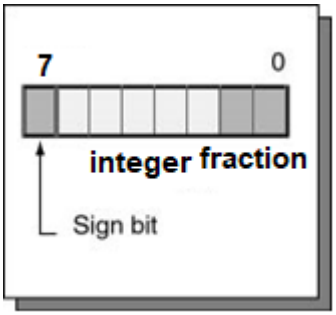


Figure 3.18.2: Graphical description for: 8 bit integer+fraction input.

Port name	direction	type & size	functionality
A	*in*	*Signed (8 bit)*	*Number A*
B	*In*	*Signed (8 bit)*	*Number B*
OUT1	*Out*	*Signed (32 bit))*	*Ieee 32bit A*
OUT2	*iut*	*Signed (32 bit)*	*Ieee 32bit B*

Table 3.18: Port Table for: Floating Point 8to32 bits Convertor entity.

Description: 8 bit integer+fraction to 32bit ieee-754 convertor . Designed with **structural architecture** as an aid component for ALU top designe. **Components:** LeadingZeroes_counter, Shift_unit, ADD_SUB.

3.19 Swap Nbits numbers

File name: Swap.vhd

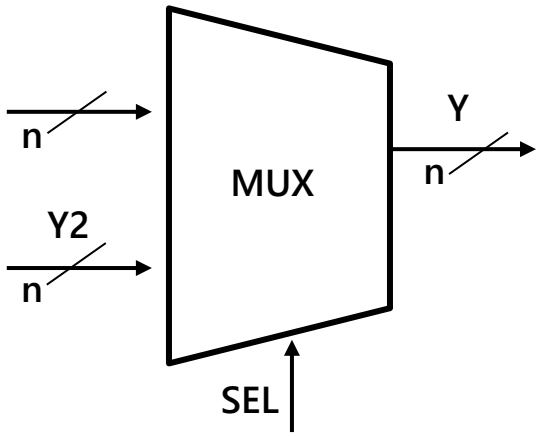


Figure 3.19: Graphical description for Swap Nbits numbers entity.

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 3.19: Port Table for: Swap Nbits numbers entity.

Description: Swap between 2 Nbits numbers. Designed with **structural architecture** as an aid component for FPU unit.

3.20 ADD\SUB Floating Point numbers

File name: ADD_SUB_FPU.vhd

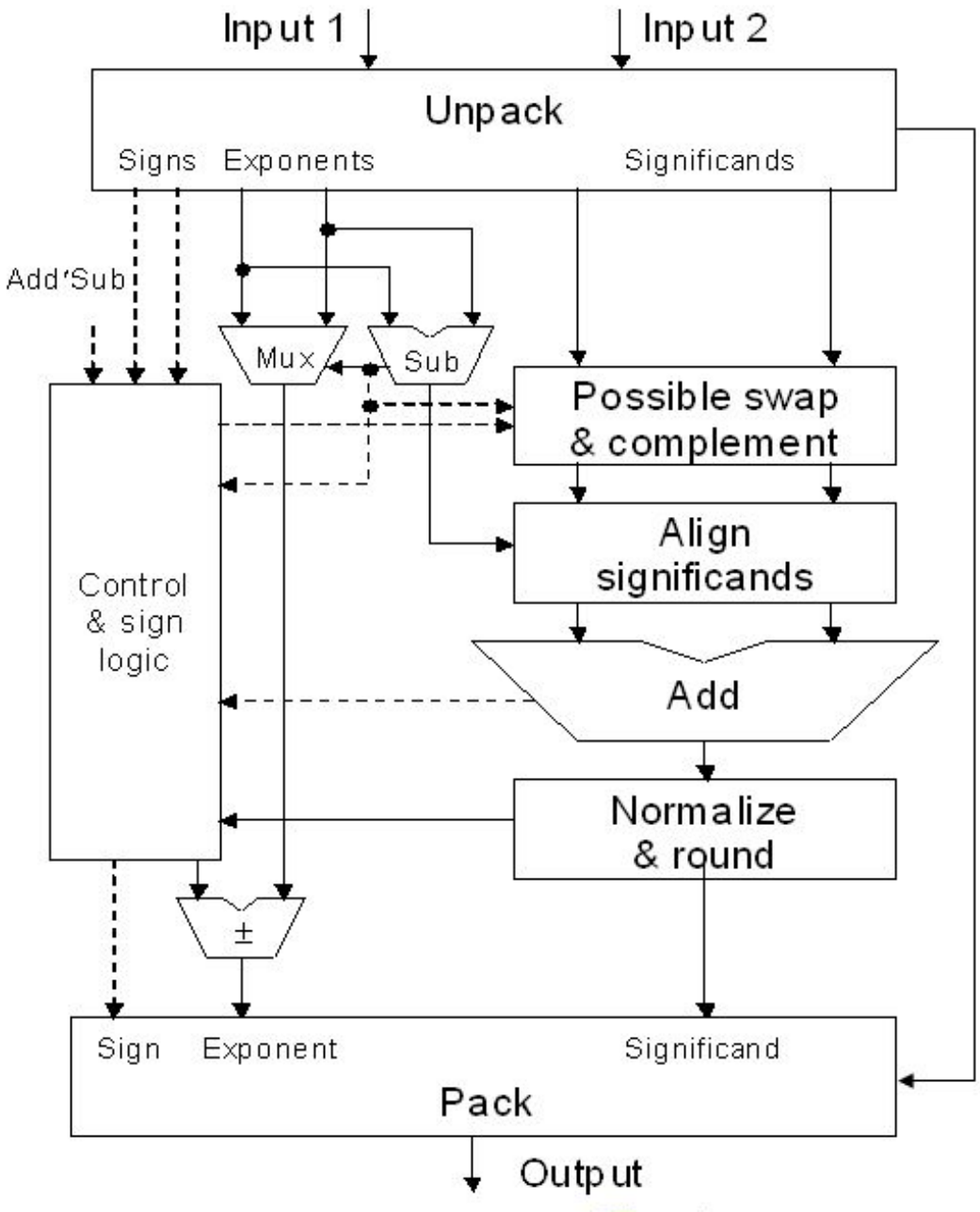


Figure 3.20: Graphical description for ADD\SUB Floating Point numbers entity.

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

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

Description: Add\Sub floating point numbers. Designed with **structural architecture** as an aid component for FPU unit.
Components: ADD_SUB, Swap, shift_unit, MUX_Nbits, MAX_MIN.

3.21 MUL Floating Point numbers

File name: MUL_FPUvhd

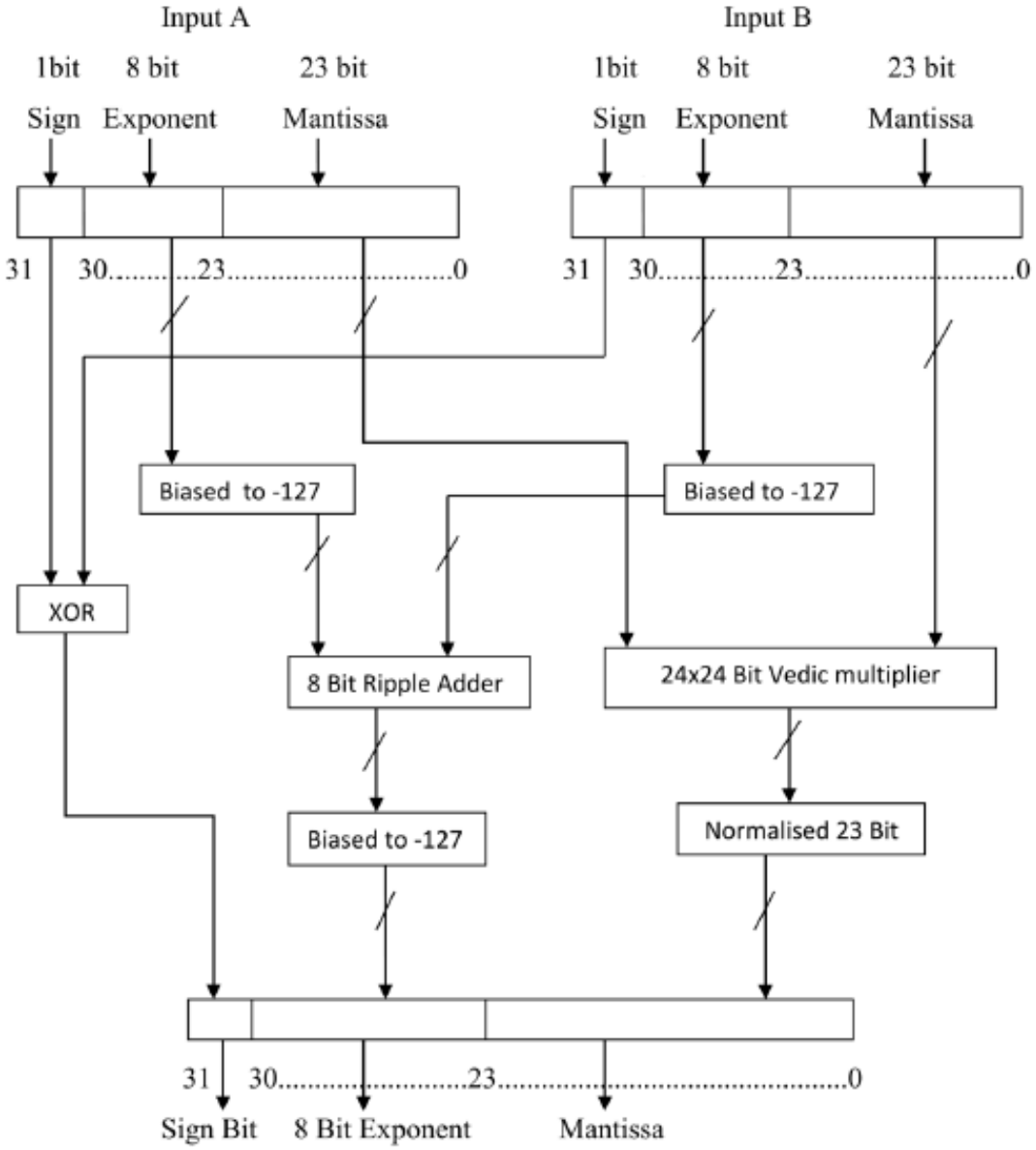


Figure 3.21: Graphical description for: MUL Floating Point numbers entity.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
A	*in*	*signed (N bits)*	*ieee A*
B	*in*	*signed (N bits)*	*ieee B*
Sum	*out*	*signed (N bits)*	*Ieee C = A*B*

Table 3.21: Port Table for: MUL Floating Point numbers entity.

Description: Multiply floating-point numbers. Designed with **structural architecture** as an aid component for FPU unit.
Components: ADD_SUB, MUL, Swap, LeadingZeroes_counter.

3.22 FPU output selector

File name: FPU_selector.vhd

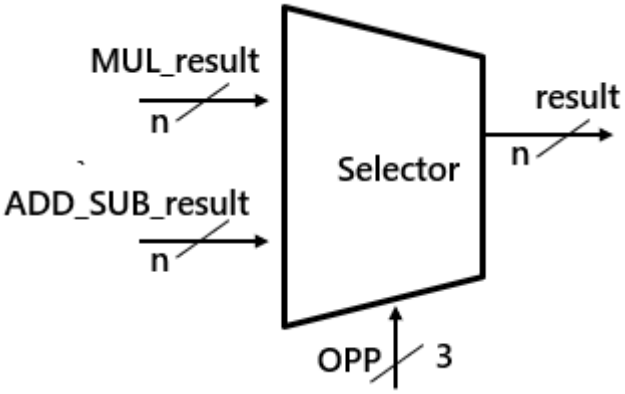


Figure 3.22: Graphical description for: FPU output selector entity.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
OPP	*in*	*Std_logic_vector (3 bits)*	*OPP code LSBS*
MUL_result	*in*	*signed (N bits)*	*ieee MUL*
ADD_SUB_result	*in*	*signed (N bits)*	*Ieee ADD\SUB*
result	*in*	*signed (N bits)*	*ieee select*

Table 3.22: Port Table for: N-bit dff entity

Description: FPU output selector. Designed with **structural architecture** as an aid component for FPU top design.

3.23 FPU top design unit

File name: FPU_Unit.vhd

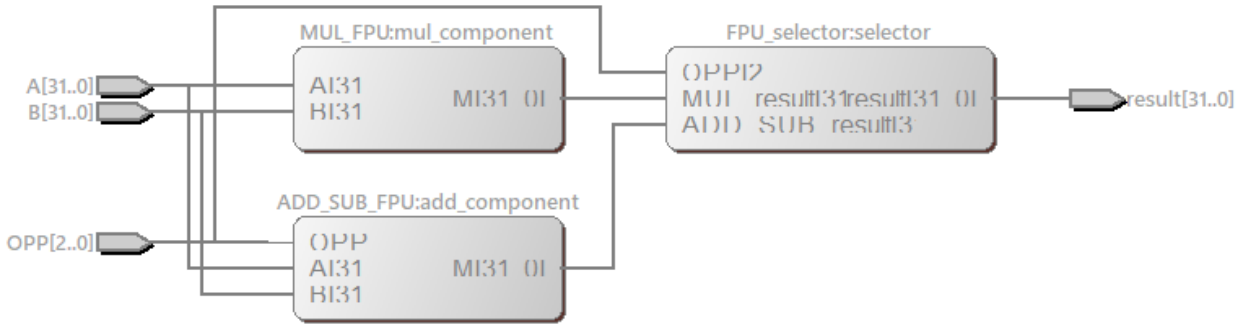


Figure 3.22: Graphical description for FPU top design unit entity.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
OPP	*in*	*Std_logic_vector (3 bits)*	*OPP code LSBS*
A	*in*	*signed (N bits)*	*ieee A*
B	*in*	*signed (N bits)*	*Ieee B*
result	*in*	*signed (N bits)*	*ieee result*

Table 3.22: Port Table for: FPU top design unit entity.

Description: FPU top design unit Designed with **structural architecture**. **Components**: ADD_SUB_FPU, MUL_FPU, FPU_Selector.

3.24 LeadingZeros counter

File name: LeadingZeros_counter.vhd

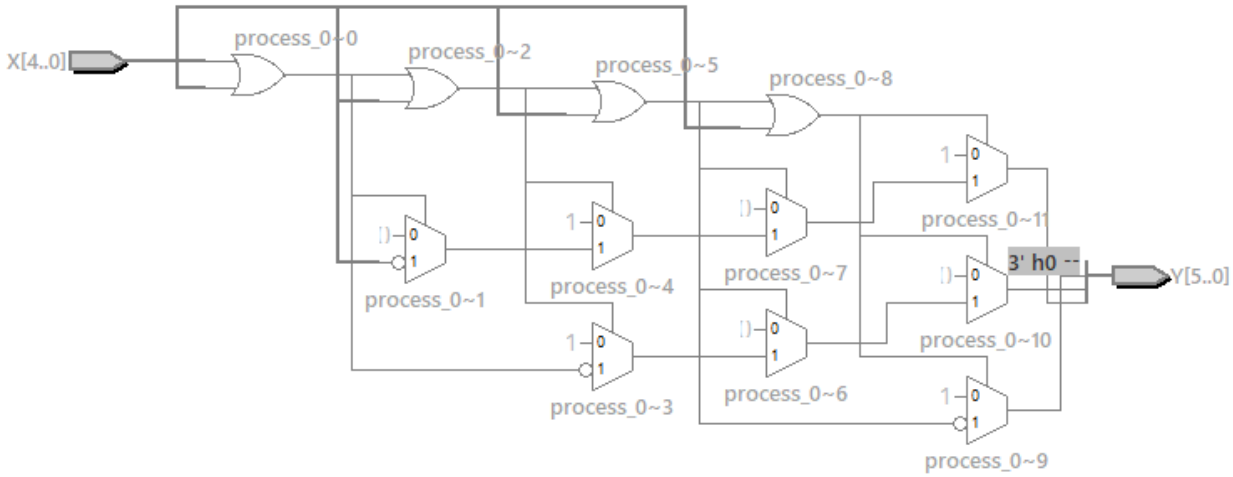


Figure 3.24: Graphical description for LeadingZeros counter entity.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
X	*in*	*signed (N bits)*	*Number X*
Y	*out*	*std_logic (6 bits)*	*Y'leading zeros*

Table 3.24: Port Table for: LeadingZeros counter entity.

Description: Leading Zeros counter. Designed with **structural architecture** as an aid component for FPU commands.

3.25 FPGA TOP design

File name: FPGA_design.vhd

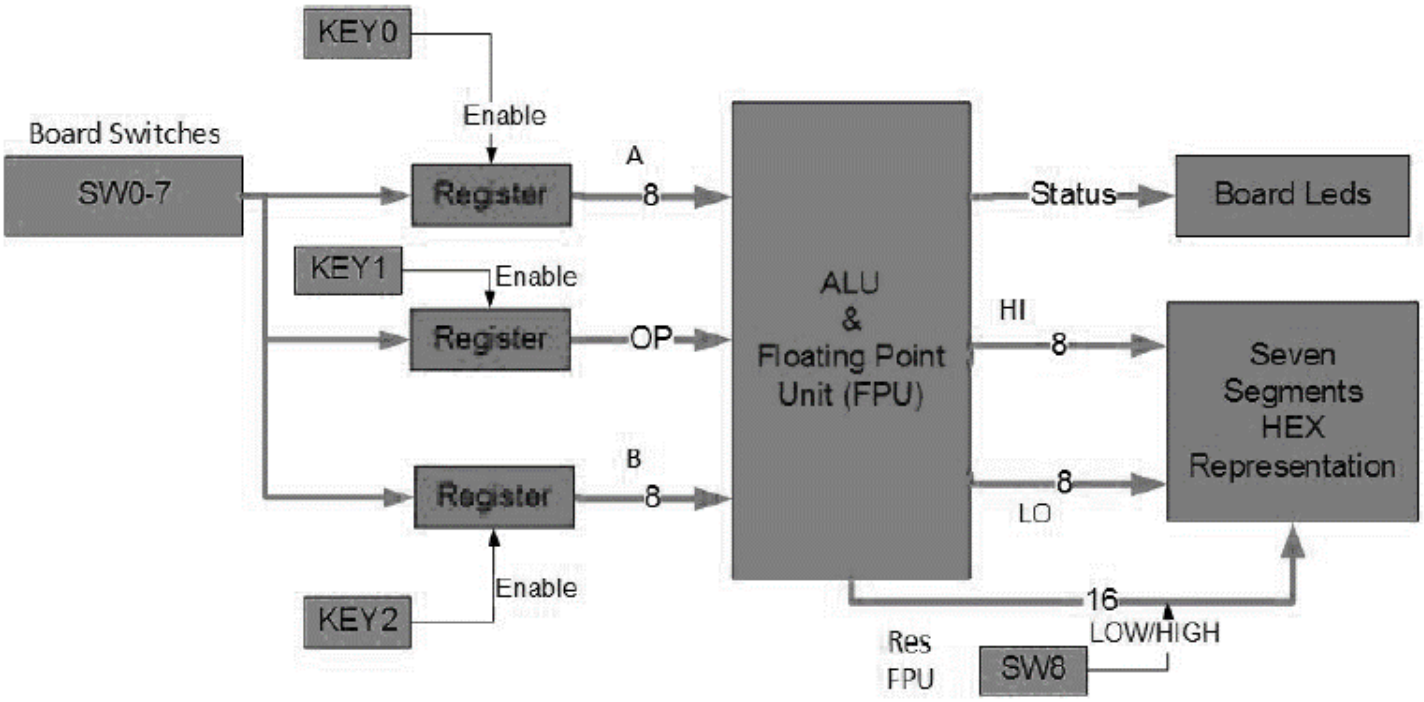


Figure 3.25: Graphical description for: FPGA TOP design entity.

Port name	direction	type & size	functionality
N	*in*	*generic integer*	*How many bits*
clk	*In*	*std_logic (1 bit)*	*clock bit*
numin	*In*	*std_logic (N bits)*	*Number*
FPU_SW_8	*In*	*std_logic (1 bit)*	*FPU MSB\LSB select bit*
KEY0	*in*	*std_logic (1 bit)*	*KEY0 enable number A*
KEY1	*in*	*std_logic (1 bit)*	*KEY1 enable OPP*
KEY2	*in*	*std_logic (1 bit)*	*KEY2 enable number B*
KEY3	*in*	*std_logic (1 bit)*	*RST registers*
LO_1	*out*	*std_logic (7 bits)*	*LOW LSBs*
LO_2	*out*	*std_logic (7 bits)*	*HIGH LSBs*
HI_1	*out*	*std_logic (7 bits)*	*LOW MSBs*

HI_2	*out*	*std_logic (7 bits)*	*HIGH MSBs*
STATUS	*out*	*std_logic (6 bits)*	*STATUS bits*

Table 3.25: Port Table for: FPGA TOP design entity.

Description: FPGA TOP design. Wrap the ALU, registers & 7-segment components. Designed with **structural architecture**.

3.26 Hex to 7-Segment

File name: 7-Segment_8_bit.vhd

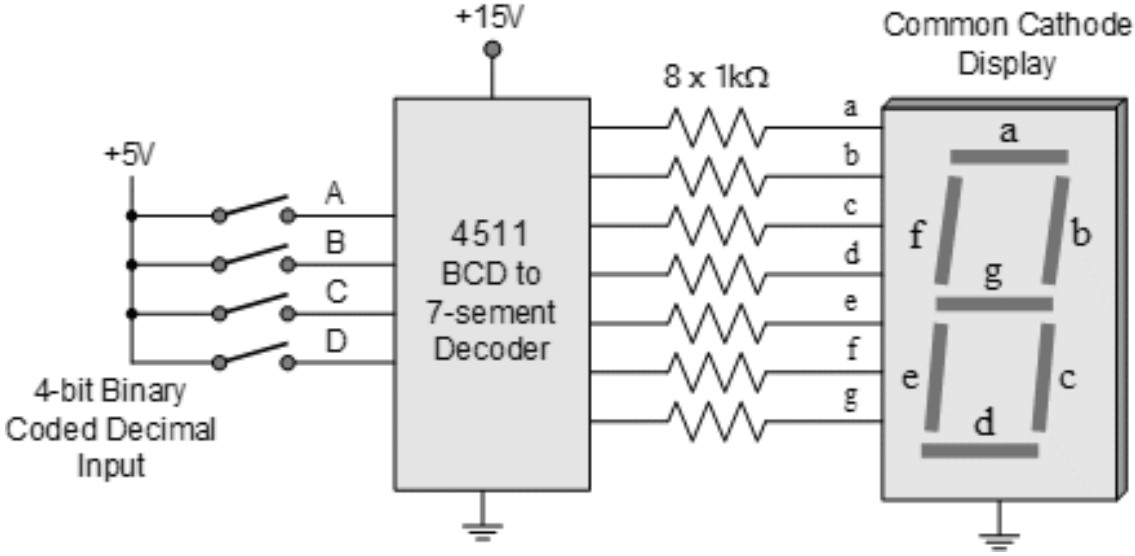


Figure 3.26: Graphical description for: Hex to 7-Segment entity.

Port name	direction	type & size	functionality
q	*in*	*std_logic (8 bits)*	*Number Q*
Segment1	*Out*	*std_logic (7 bits)*	*Segment1*
Segment2	*out*	*std_logic (7 bits)*	*Segment2*

Table 3.26: Port Table for: Hex to 7-Segment entity.

Description: 8 bit (2 hex number) to 7-segment display on the FPGA. Designed with **behavioral architecture** as an aid component for FPGA top design entity register.

Performance Test Case

4.1 ALU

File name: ALU.vhd

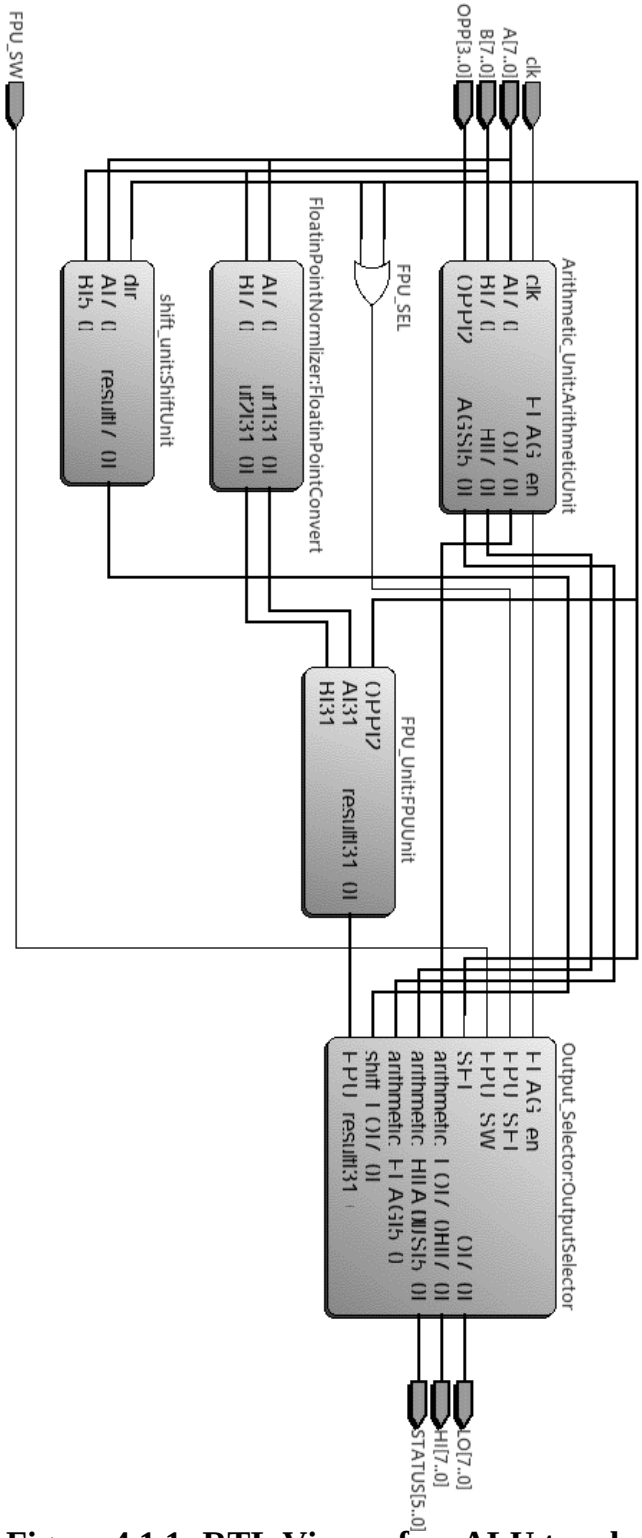


Figure 4.1.1: RTL Viewer for: ALU top design entity.

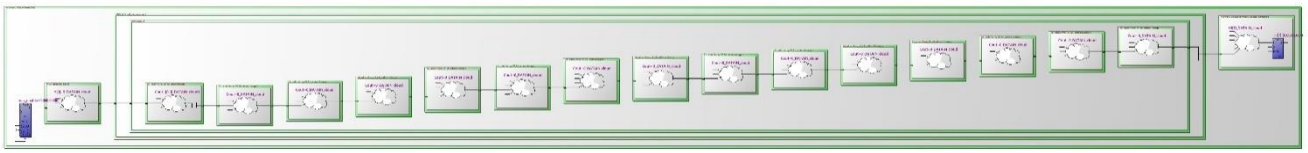


Figure 4.1.2: Critical path for: ALU top design entity – full png in DOC

Analyze: $Arithmetic\_unit -> \ ADD\_SUB -> ADD(stage1)$

$\rightarrow FullAdders \rightarrow Arithmetic\ selector$

If we check the MAX frequency for the ALU TOP design only, we get an increase from the overall system (without the inputs\output registers & keys) :

Slow Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	114.61 MHz	114.61 MHz	clk	

4.2 FPU unit

File name: FPU_unit.vhd

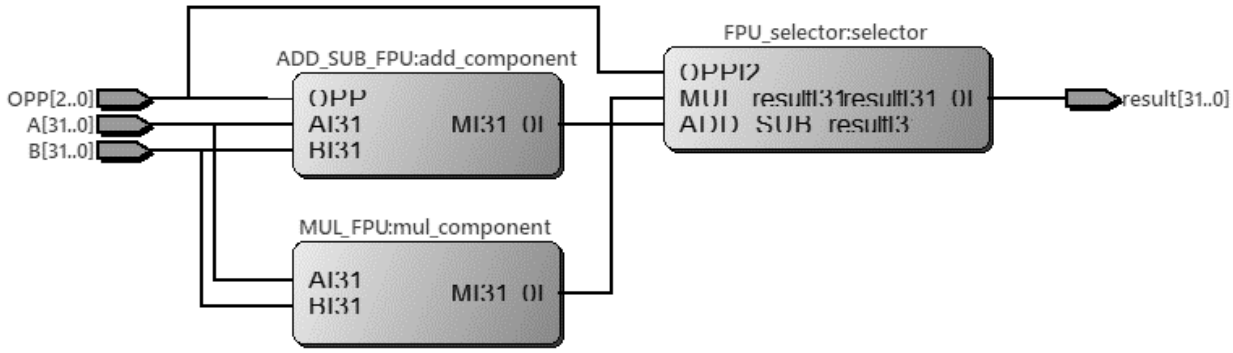


Figure 4.2.1: RTL Viewer for: FPU top design entity.

4.3 Floating Point Convertor

File name: FloatingPointConvertor.vhd

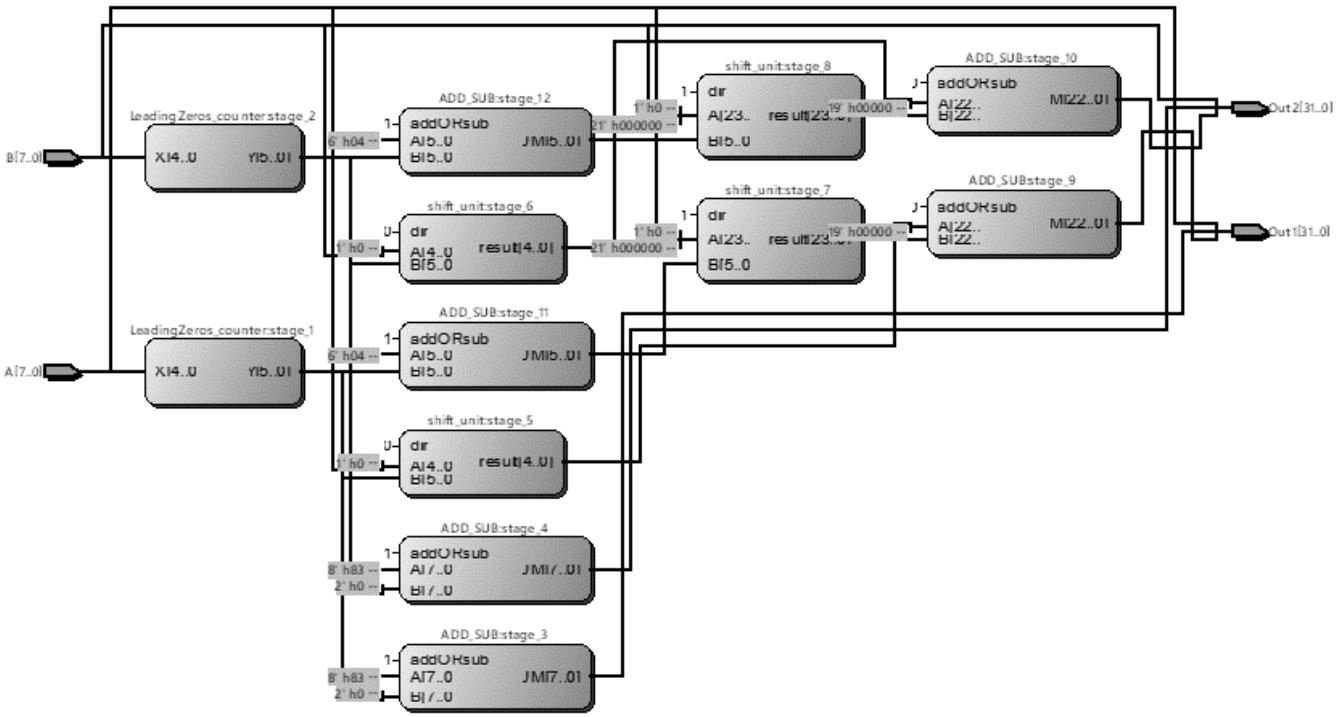


Figure 4.3.1: RTL Viewer for: Floating Point Convertor entity.

4.4 Floating Point Adder

File name: ADD_SUB_FPU.vhd

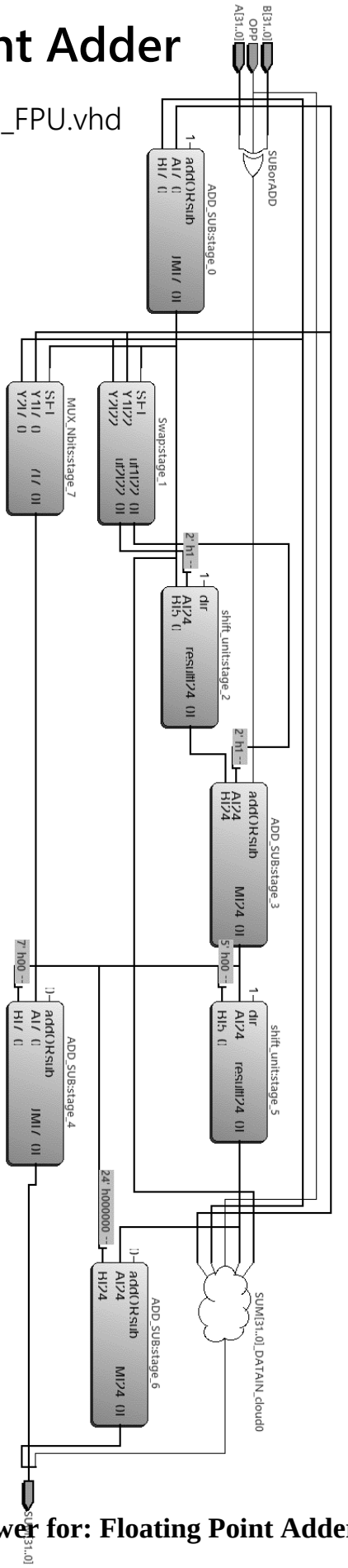


Figure 4.4.1: RTL Viewer for: Floating Point Adder entity.

4.5 Floating Point Multiplier

File name: MUL_FPU.vhd

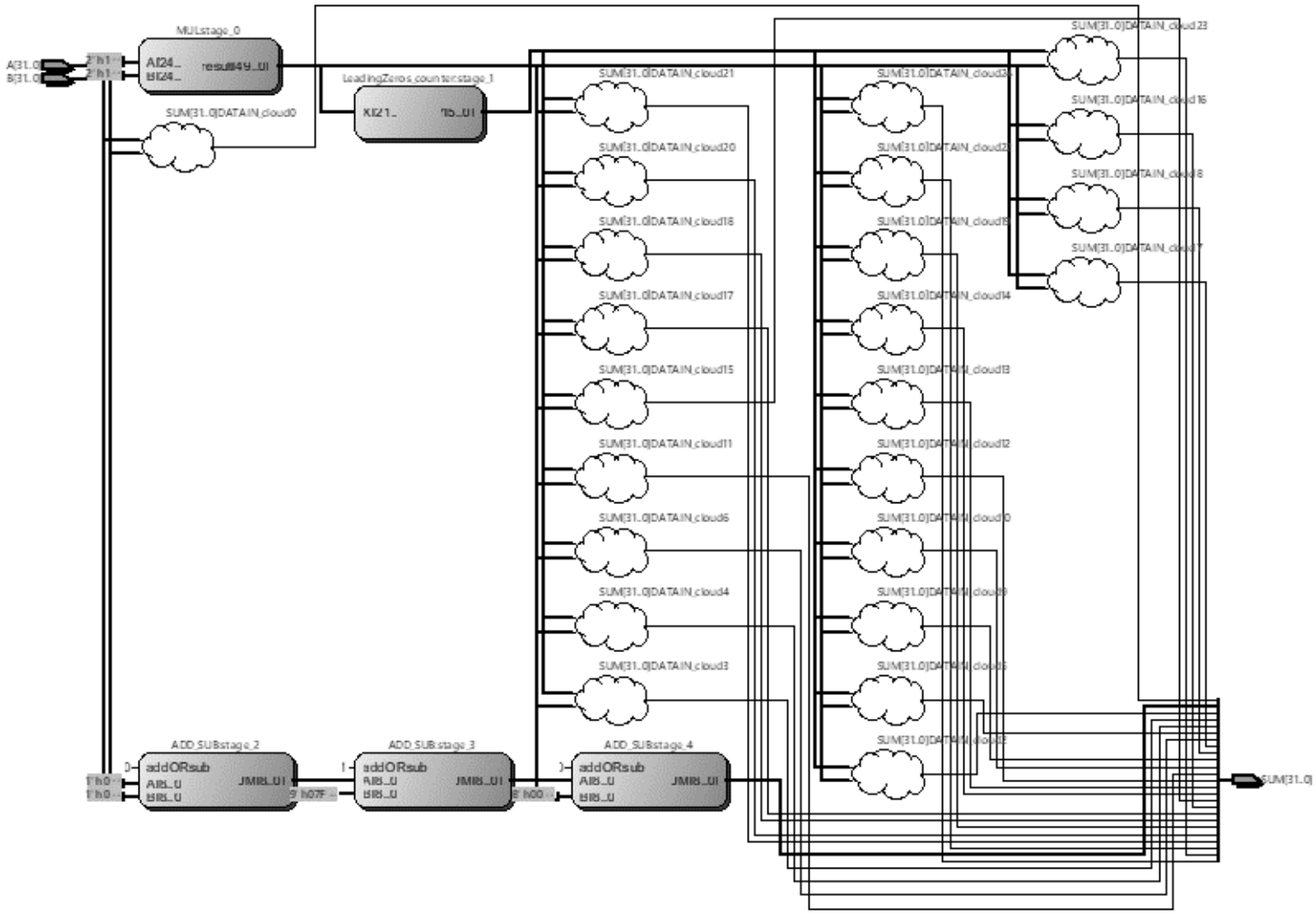


Figure 4.5.1: RTL Viewer for: Floating Point Multiplier entity.

4.6 FPGA top design – overall system

File name: FPGA_design.vhd

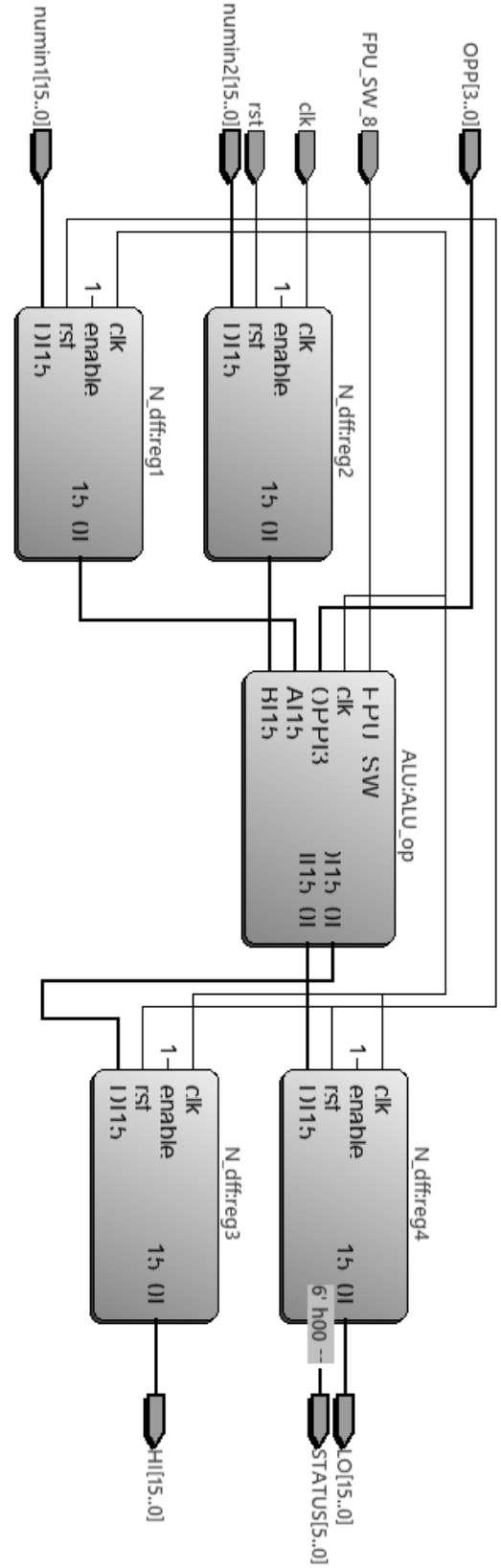


Figure 4.6.1: RTL Viewer for: overall system entity.

Flow Summary	
Flow Status	Successful - Mon May 07 16:31:30 2018
Quartus II 32-bit Version	12.1 Build 177 11/07/2012 SJ Web Edition
Revision Name	test
Top-level Entity Name	FPGA_test
Family	Cyclone II
Device	EP2C20F484C6
Timing Models	Final
Total logic elements	1,348 / 18,752 (7 %)
Total combinational functions	1,344 / 18,752 (7 %)
Dedicated logic registers	132 / 18,752 (< 1 %)
Total registers	132
Total pins	77 / 315 (24 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	9 / 52 (17 %)
Total PLLs	0 / 4 (0 %)

1	Estimated Total logic elements	1,344
2		
3	Total combinational functions	1344
4	Logic element usage by number of LUT inputs	
1	-- 4 input functions	848
2	-- 3 input functions	399
3	-- <=2 input functions	97
5		
6	Logic elements by mode	
1	-- normal mode	1282
2	-- arithmetic mode	62
7		
8	Total registers	132
1	-- Dedicated logic registers	132
2	-- I/O registers	0
9		
10	I/O pins	77
11	Embedded Multiplier 9-bit elements	9
12	Maximum fan-out	375
13	Total fan-out	5203
14	Average fan-out	3.33

Figures 4.6.2: Logic usage for: overall system entity.

tity	Logic Cells	Dedicated Logic Registers	DSP Elements	DSP 9x9	DSP 18x18	Pins	LUT-Only LCs	Register-Only LCs	LUT/Registe
Cyclone II: EP2C20F484C6									
FPGA_test	1348 (1)	132 (0)	9	1	4	77	1216 (1)	4 (0)	128 (0)
ALU:ALU_op	1138 (0)	68 (0)	9	1	4	0	1040 (0)	0 (0)	98 (0)
Arithmetic_Unit:...	264 (5)	68 (3)	2	0	1	0	191 (2)	0 (0)	73 (3)
FloatingPointCon...	51 (0)	0 (0)	0	0	0	0	46 (0)	0 (0)	5 (0)
FPU_Unit:FPUUnit	341 (0)	0 (0)	7	1	3	0	341 (0)	0 (0)	0 (0)
ADD_SUB_FP...	98 (3)	0 (0)	0	0	0	0	98 (3)	0 (0)	0 (0)
MUL_FPU:mul...	230 (121)	0 (0)	7	1	3	0	230 (121)	0 (0)	0 (0)
FPU_selector:s...	13 (13)	0 (0)	0	0	0	0	13 (13)	0 (0)	0 (0)
Output_Selector:...	18 (0)	0 (0)	0	0	0	0	12 (0)	0 (0)	6 (0)
shift_unit:ShiftUnit	464 (0)	0 (0)	0	0	0	0	450 (0)	0 (0)	14 (0)

Figures 4.6.2: Logic usage for: each main entity.

Analyze: We zoom in on the main components. We can see that the MUL_FPU required more logic units than the ADD_SUB_FPU, because the MUL_FPU required the MUL hardware. The FPU_UNIT using most of the components of the ALU, such as shift_unit, MUL, ADD\SUB – therefor it required more logic units than the Arithmetic_Unit.

Figure 4.6.3: Critical path for: overall system entity, full-png in DOC.

Analyze: $Registers \rightarrow FPGA\ TOP -> ALU\ TOP \rightarrow SHIFT\ UNIT \rightarrow OutputSelector \rightarrow Registers$

Frequency limiting operation: shift operations - using the **Shift Unit.** The shift unit using Ndff to shift the number, which require 1 clk per shifter (1 register).

Propose solution for CPU frequency improvements in two cases (current ALU):

1. The problematic operation is commonly used in software :
   ⇨ Design shift unit with dedicated hardware (with separate clock), and refactor the output selector not to wait for the shift result.

2. The problematic operation is almost unused:
   ⇨ Then we can refactor the system with enable to this specific hardware. The hardware will not be in use until the uncommon operation will be required.

Maximal operating clock:

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	40.8 MHz	40.8 MHz	clk	

Hardware Test Case

In the hardware test case you will have to test an 8bit ALU and floating point at the FPGA board. A board switches (SW) and push buttons (KEY) will be used to select the input data and output data.

The ALU input data is generated using 3 registers connected in parallel to the board switched with 3 push buttons serving as register enables.

All the system must work from a single 50MHz (non-gated clock). All the system must be connected to the Altera board interfaces according to the following diagram. For floating point commands (described above) the eight-bit number will be transfer to floating point format while first two bits will be describing the fraction (LSB) and entire six bits the integer part, MSB describe sign of the number.

The result of floating point commands will be present on seven segment displays in floating point format. While each segment (total 4) can be used for present 4 bit in HEX format. Using SW8 ones will be present 16 LSB bits and ones 16 MSB bits, for example:
SW8 = 1 => 16 MSB bits presented.
SW8 = 0 => 16 LSB bits presented.

5.1 FPGA top design – overall system

File name: FPGA_design.vhd

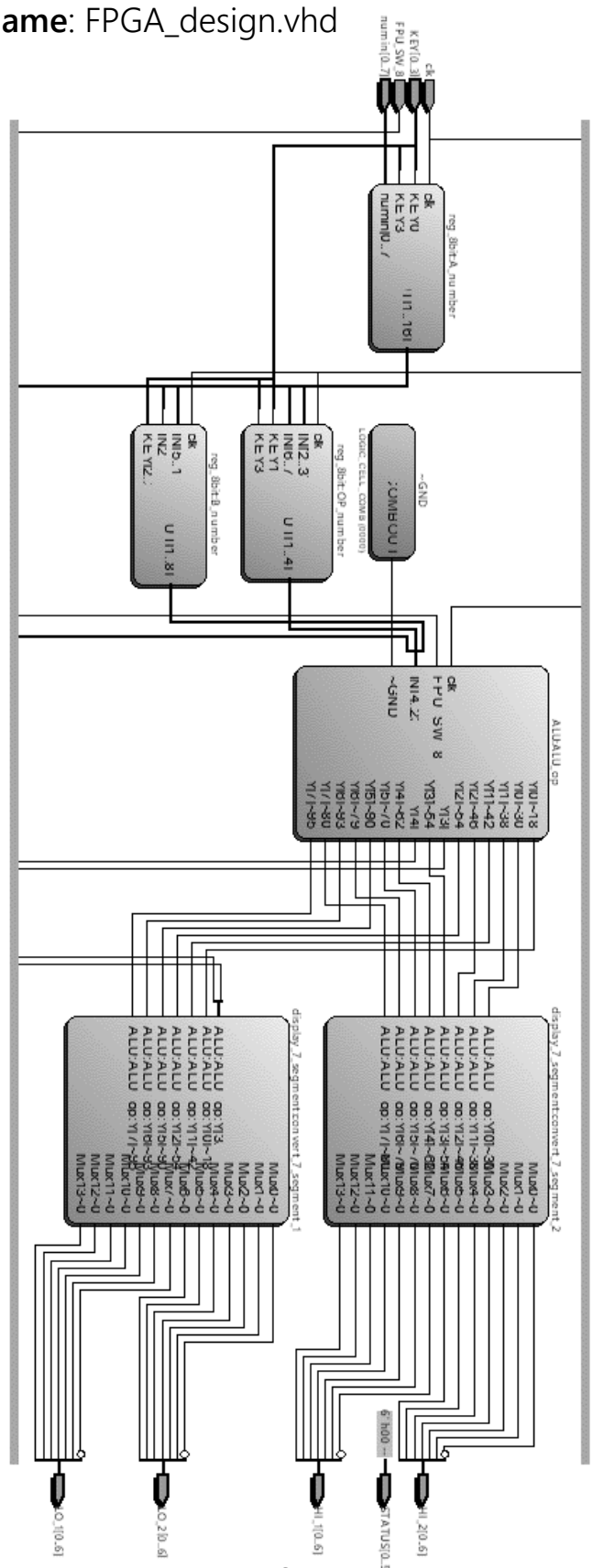


Figure 4.6.1: RTL Viewer for: overall system entity.

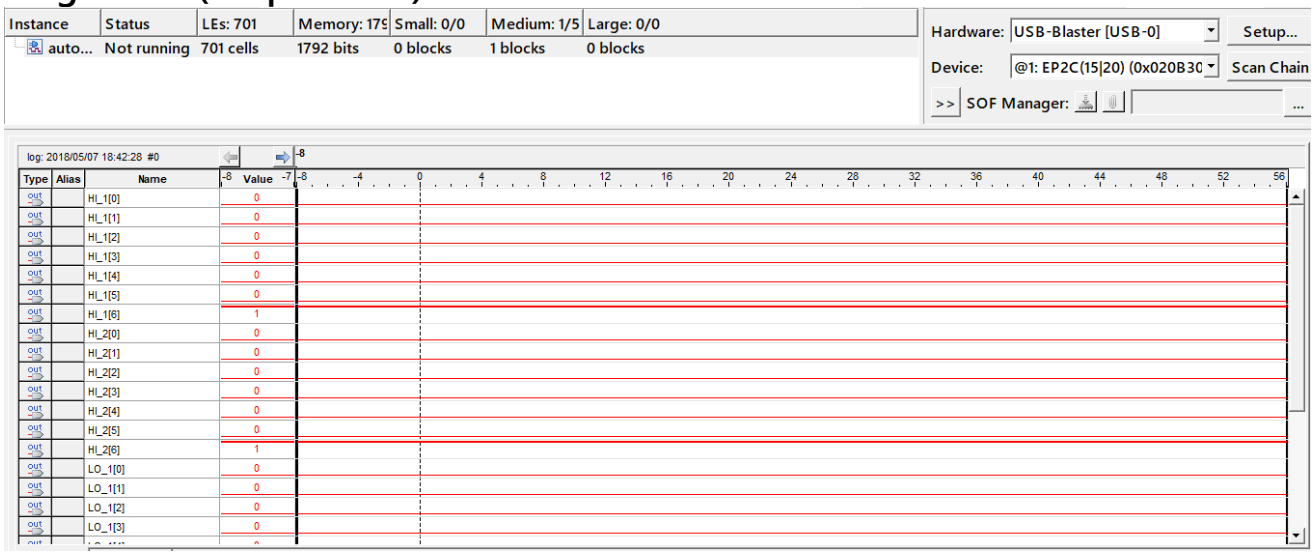
Flow Summary	
Flow Status	Successful - Mon May 07
Quartus II 32-bit Version	12.1 Build 177 11/07/2012 S
Revision Name	ALU
Top-level Entity Name	FPGA_design
Family	Cyclone II
Device	EP2C20F484C6
Timing Models	Final
Total logic elements	795 / 18,752 (4 %)
Total combinational functions	790 / 18,752 (4 %)
Dedicated logic registers	56 / 18,752 (< 1 %)
Total registers	56
Total pins	48 / 315 (15 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	8 / 52 (15 %)
Total PLLs	0 / 4 (0 %)

MAX Frequency:

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	80.21 MHz	80.21 MHz	clk	

Single TAP (outputs bits):



Also, this design has been physically tested successfully.

Bench Tests

6.1 full_adder.VHD

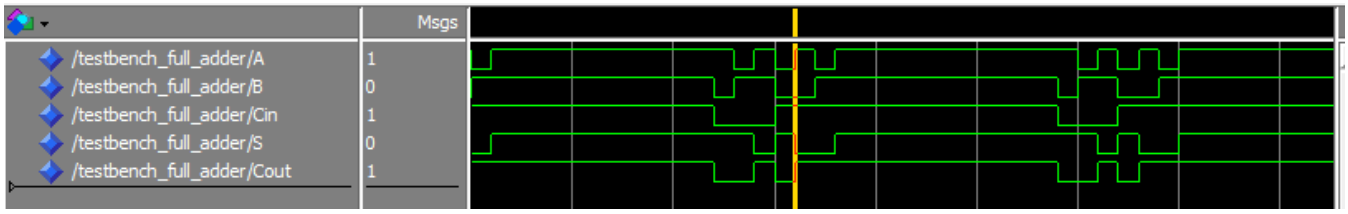


Figure 6.1: Test Bench for: full_adder entity

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

6.2 ADD_SUB.VHD

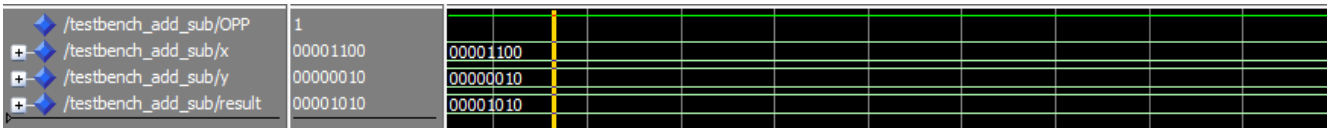


Figure 6.2: Test Bench for: ADD_SUB operation

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

6.3 ADD.VHD

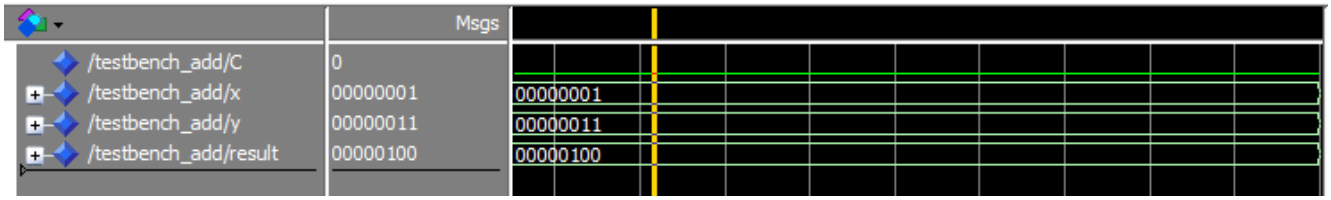


Figure 6.3: Test Bench for: ADD entity

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

6.4 MAX_MIN.VHD

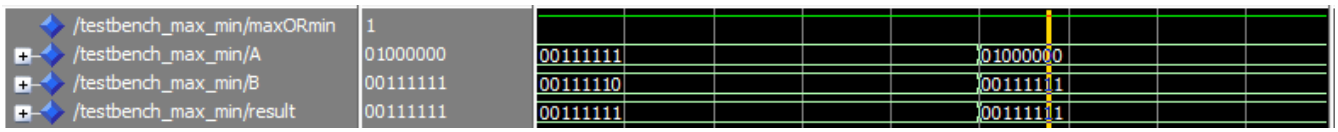


Figure 6.4: Test Bench for: MAX_MIN operation

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

6.5 Output_Selector.VHD

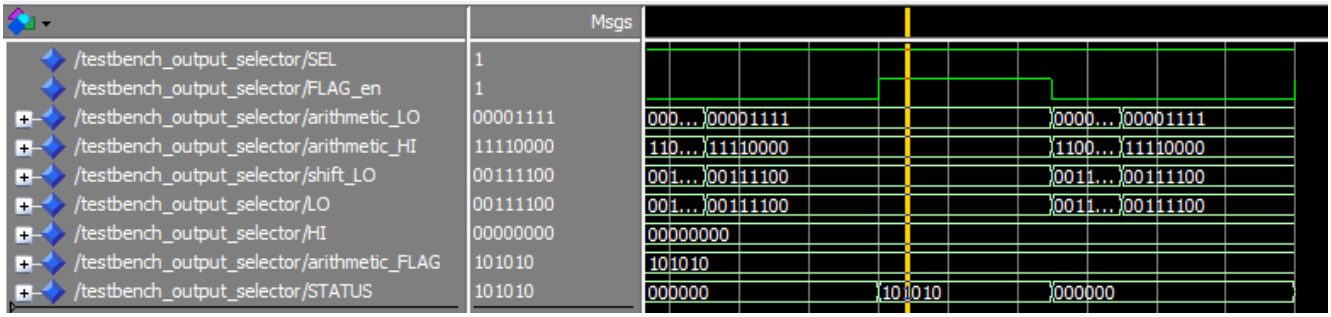


Figure 6.5: Test Bench for: Output_Selector unit

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).

6.6 MAC.VHD

Figure 6.6: Test Bench for: MAC entity

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

6.7 MUL.VHD

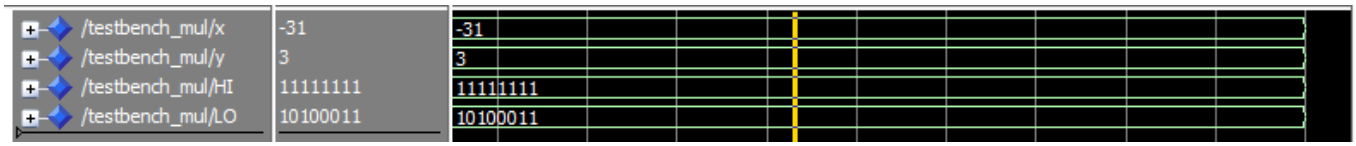


Figure 6.7: Test Bench for: MUL entity

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

6.8 diff_1bit.VHD

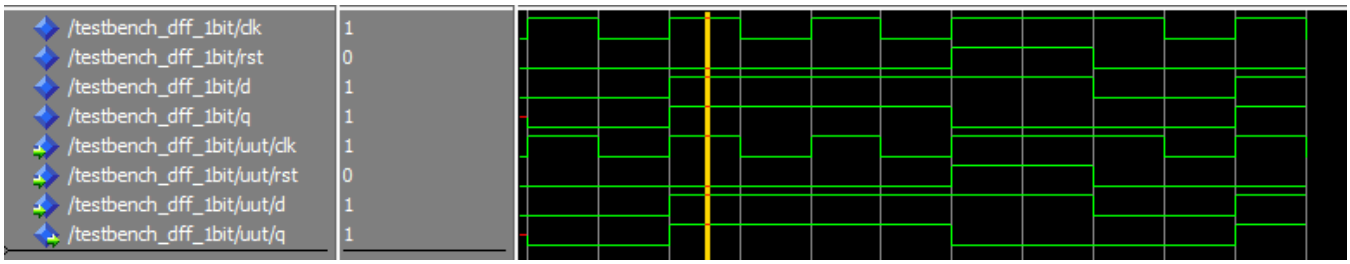


Figure 6.8: Test Bench for: diff_1bit entity

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

6.9 arithmetic_selector.VHD

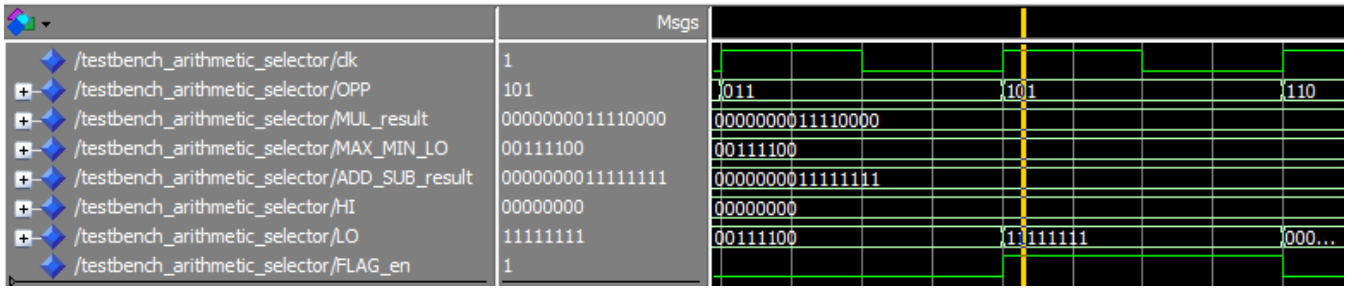


Figure 6.9: Test Bench for: arithmetic_selector entity

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

6.10 arithmetic_unit.VHD

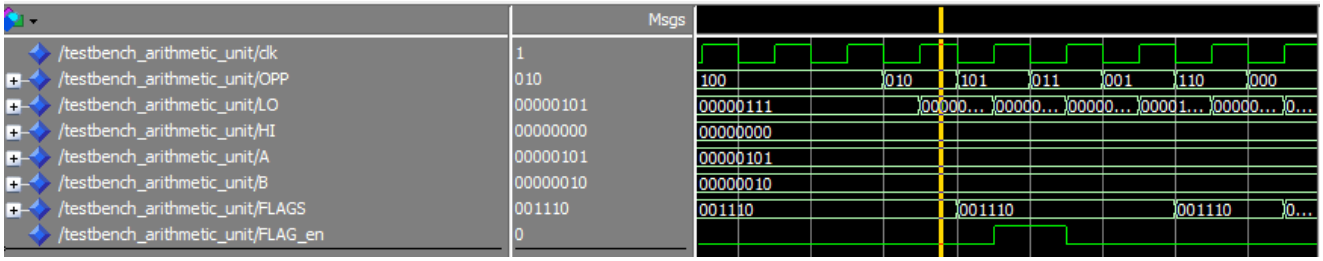


Figure 6.10: Test Bench for: arithmetic unit

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

6.11 N_dff.VHD

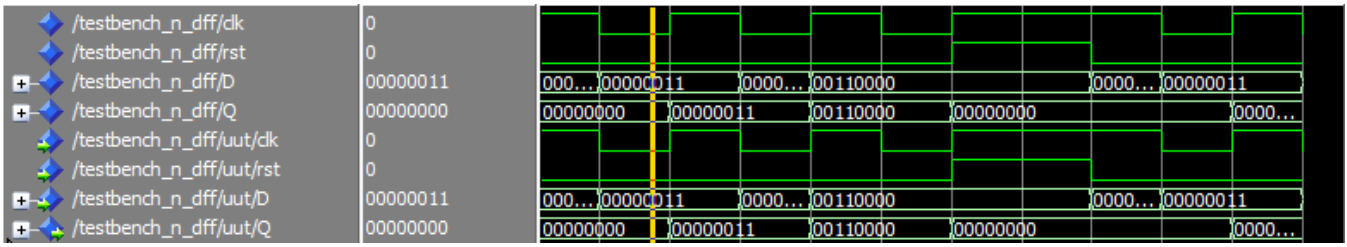


Figure 6.11: Test Bench for: N_dff entity

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

6.12 mux_Nbits.VHD

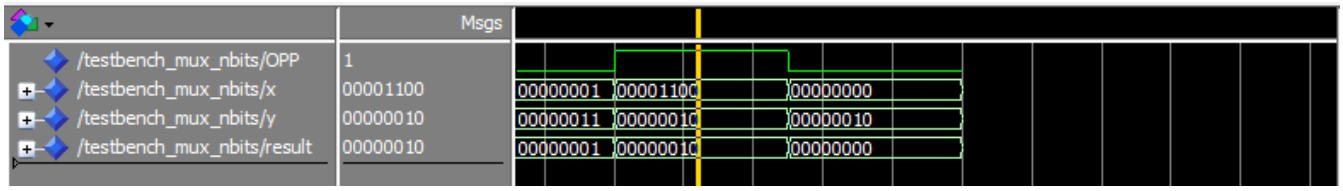


Figure 6.12: Test Bench for: mux_Nbits entity

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

6.13 shift_Nbits.VHD

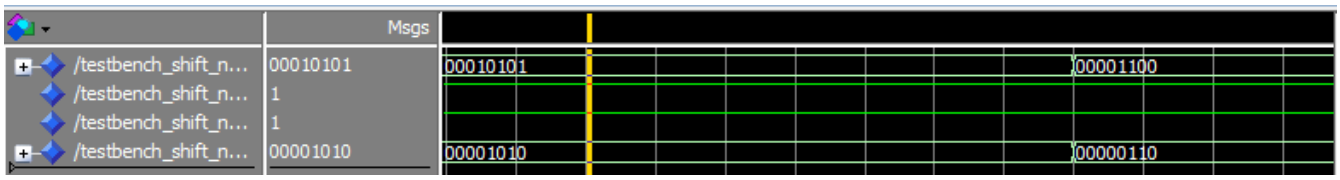


Figure 5.13: Test Bench for: shift_Nbits entity

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

6.14 ALU.VHD

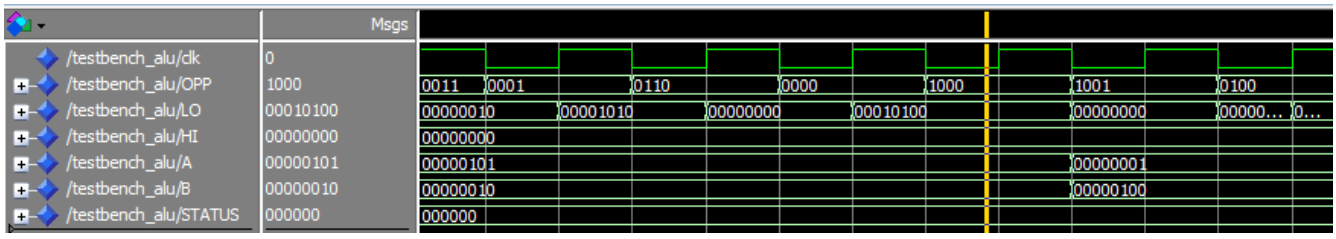


Figure 6.14: Test Bench for: ALU unit

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) .

6.15 ADD_SUB_FPU.VHD

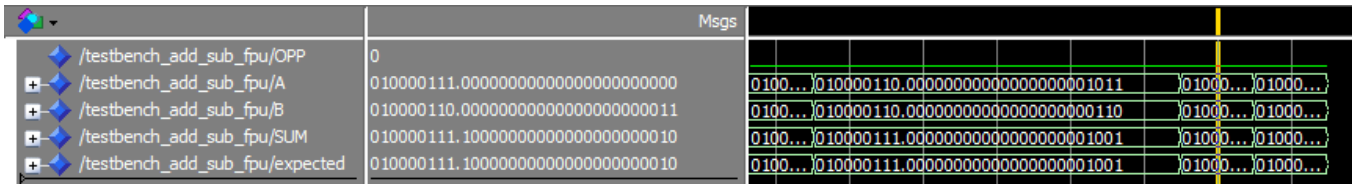


Figure 5.15: Test Bench for: ADD\SUB FPU entity

Description: As shown the SUM is as expected (expected result calculated using online calculators).

6.16 FloatingPointConvertor.VHD

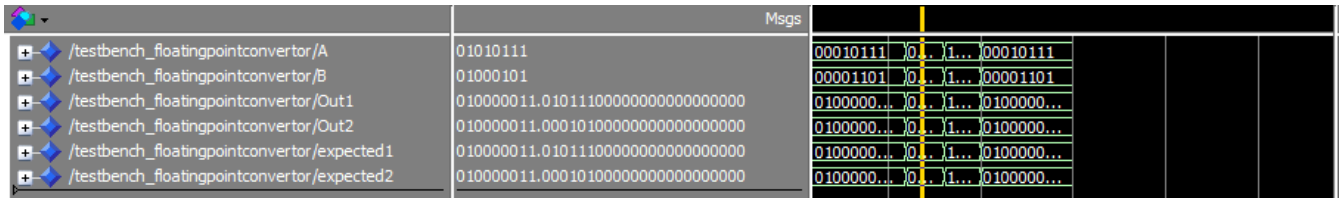


Figure 5.16: Test Bench for: FloatingPointConvertor 8bitTO32bit entity

Description: As shown the outputs is as expected (expected result calculated using online convertors).

6.17 MUL_FPU.VHD

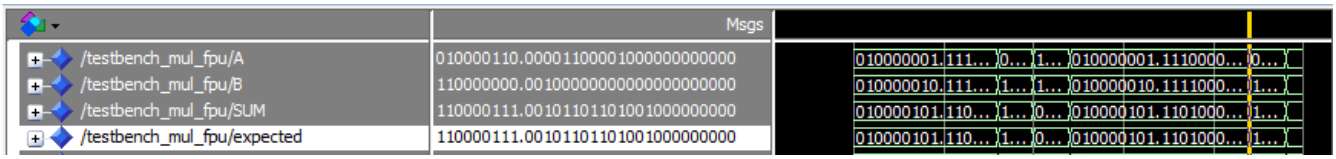


Figure 5.17: Test Bench for: MUL FPU entity

Description: As shown the result is as expected (expected result calculated using online calculators).

6.18 LeadingZeros_Counter.VHD

Figure 6.18: Test Bench for: Leading Zeros Counter entity

Description: The input number are with 3 zeros, and the result is 3 as expected.

6.19 FPU_Unit.VHD

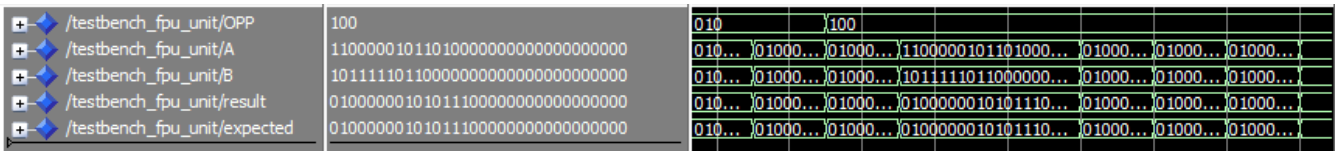


Figure 6.19: Test Bench for: FPU_Unit entity

Description: OPP = "1100" -> MUL F. As shown the result is as expected (expected result calculated using online calculators).

6.20 shift_Unit.VHD

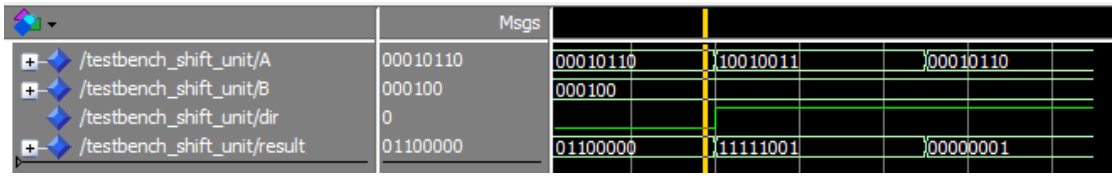
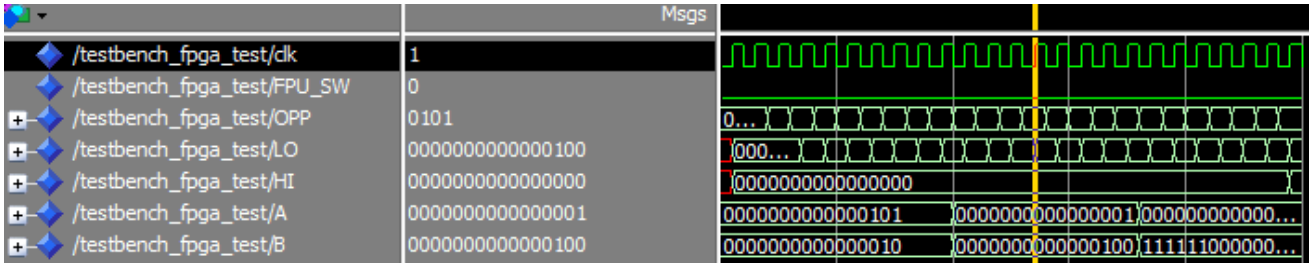


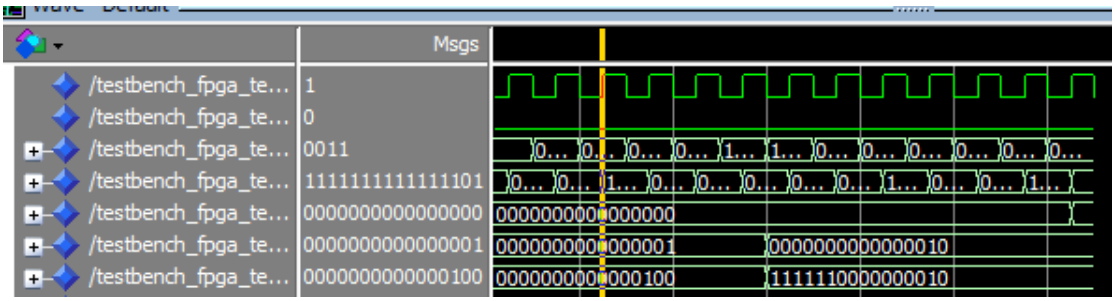
Figure 6.20: Test Bench for: shift_Unit entity

Description: dir = '0' (shift to the left), B = 4, the output is the number A shifted B times to the left as expected.

6.21 FPGA_design.VHD – test performence



Description : OPP="0101" -> SUB 1-4, we can see in the next cycle that the result is updating in the output register.



Attached files

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