LAB 3

Dor Livne 204528251

Amir Tsur 203795828

Full System:

The diagram illustrating the system is

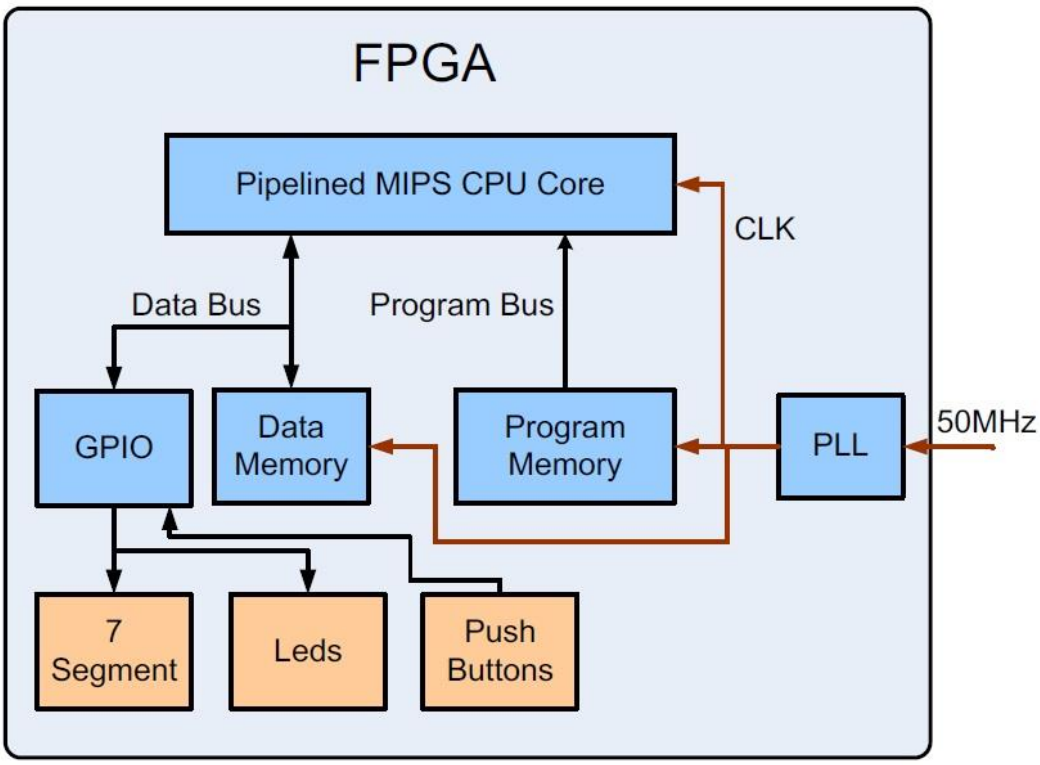


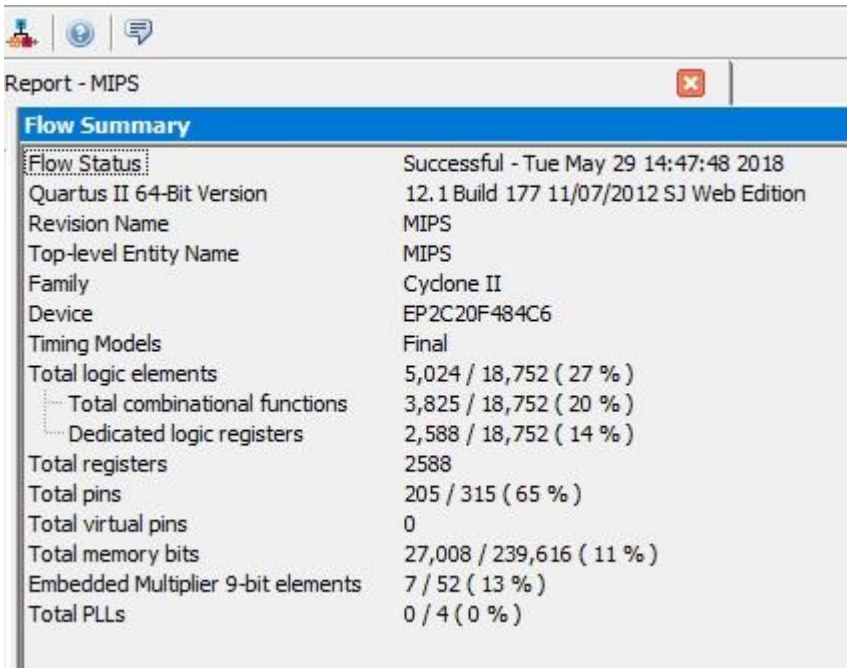
Figure 1 : System architecture

Port table:

Port	direction	size	functionality
Sw8	in	1 bit	Hex(4 downto 1)<-FPU_Hi if 1 else FPU_LO
clk	in	1 bit	Clk<-Clock_50
Status	Out	6 bit	LEDG<-Status
Hex1,Hex2,Hex3,Hex4	Out	4 bit each	7_seg<-Hex

The full system design is fed from predetermined program, written in assembly using MARS.

Logic elements usage:

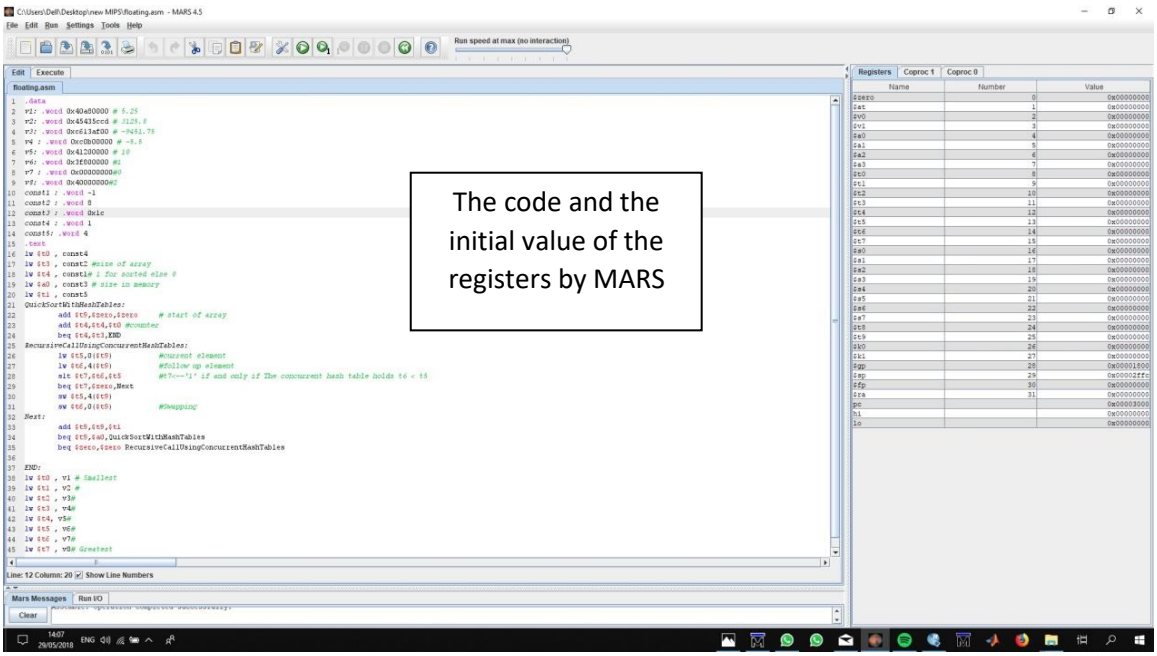


Report - MIPS

Flow Summary

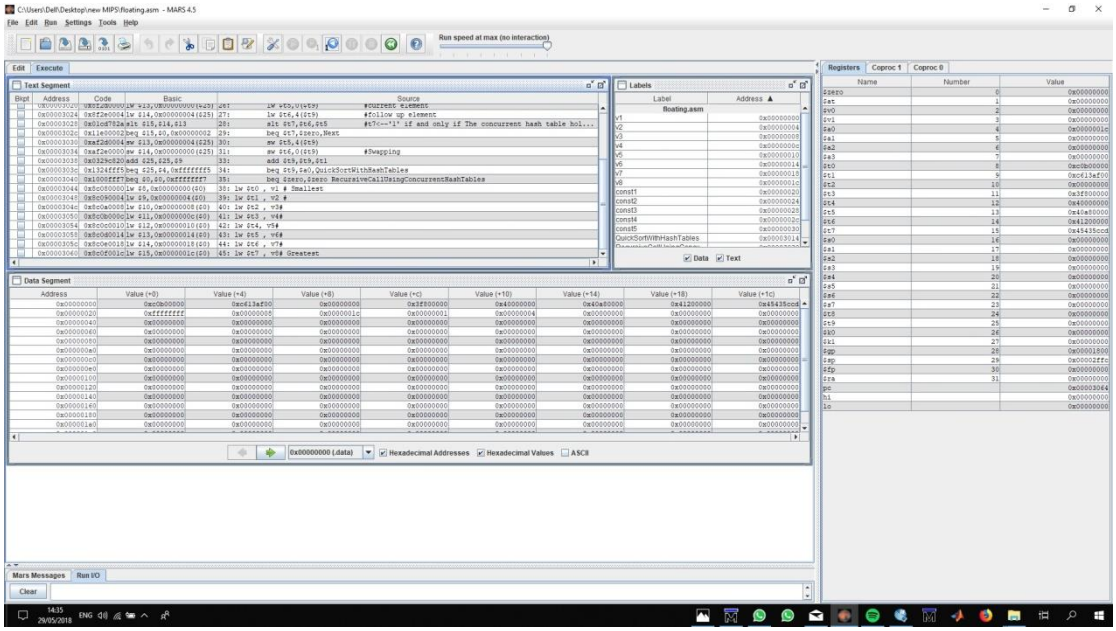
Flow Status	Successful - Tue May 29 14:47:48 2018
Quartus II 64-Bit Version	12.1 Build 177 11/07/2012 SJ Web Edition
Revision Name	MIPS
Top-level Entity Name	MIPS
Family	Cyclone II
Device	EP2C20F484C6
Timing Models	Final
Total logic elements	5,024 / 18,752 (27 %)
Total combinational functions	3,825 / 18,752 (20 %)
Dedicated logic registers	2,588 / 18,752 (14 %)
Total registers	2588
Total pins	205 / 315 (65 %)
Total virtual pins	0
Total memory bits	27,008 / 239,616 (11 %)
Embedded Multiplier 9-bit elements	7 / 52 (13 %)
Total PLLs	0 / 4 (0 %)

Mars simulation

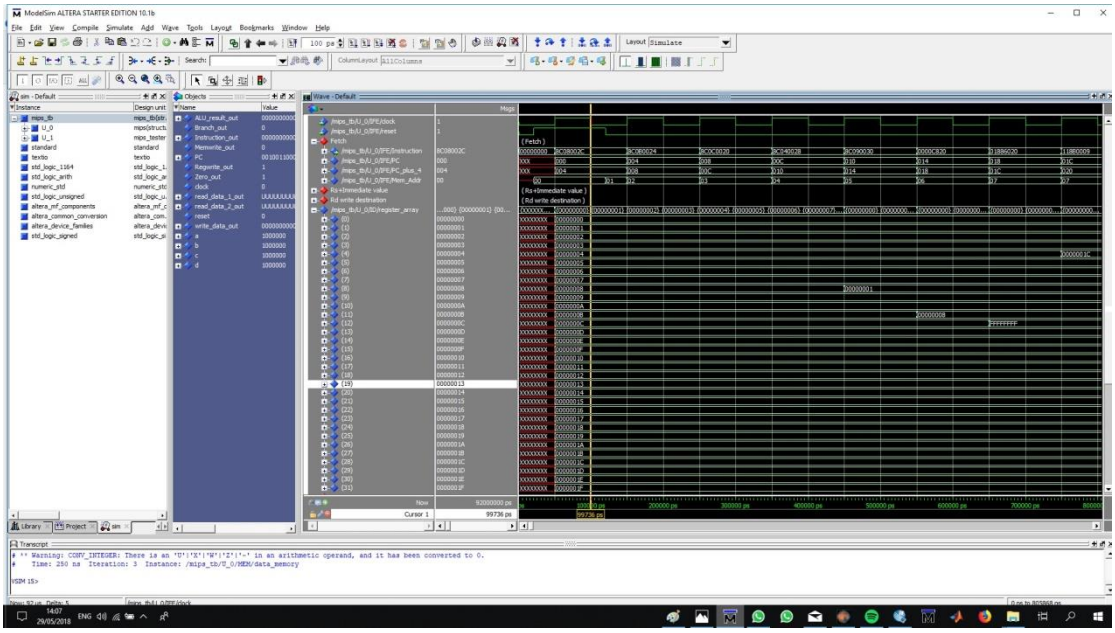


The code and the initial value of the registers by MARS

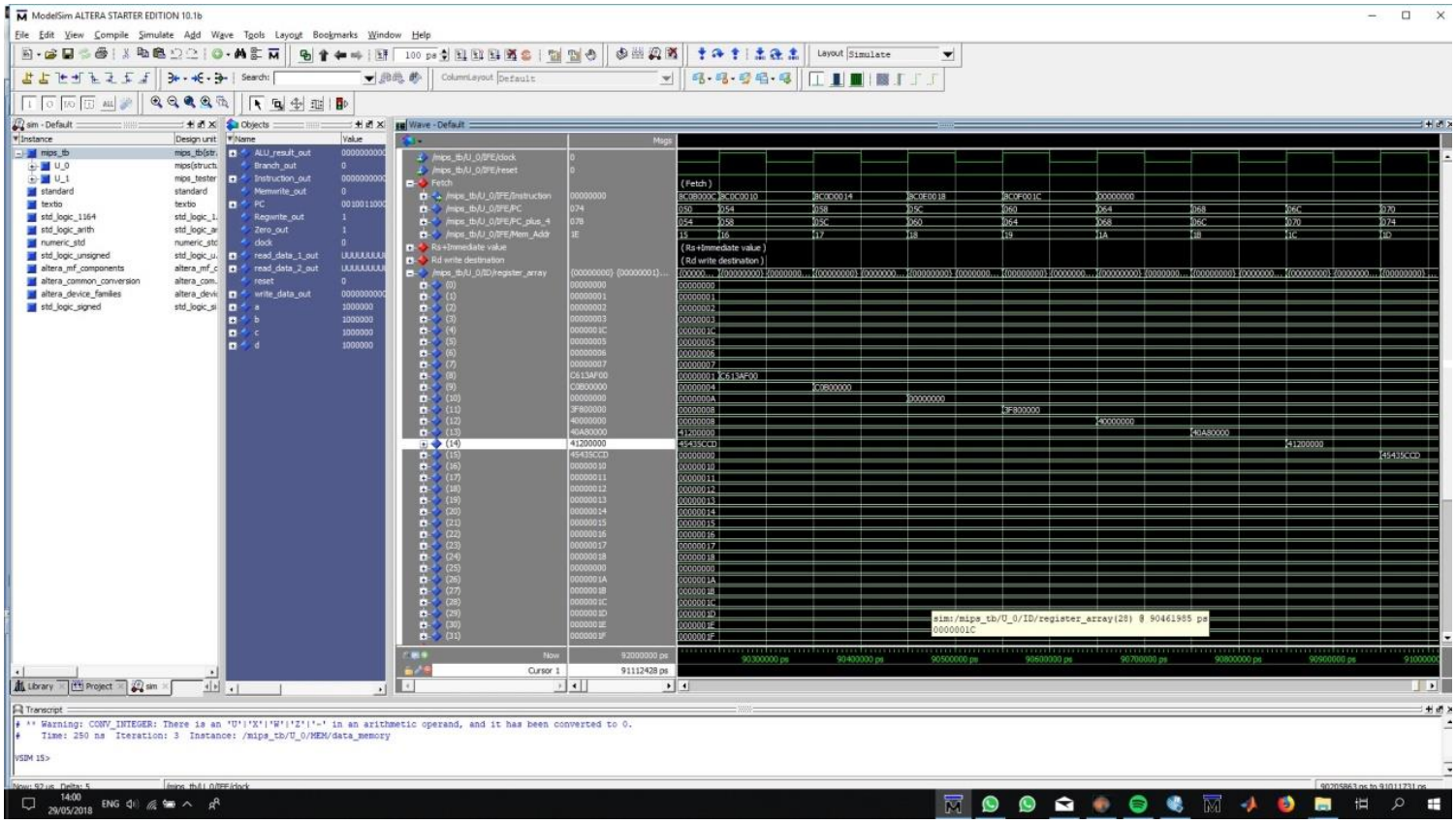
Simulation by mars:



Initial values by ModelSim

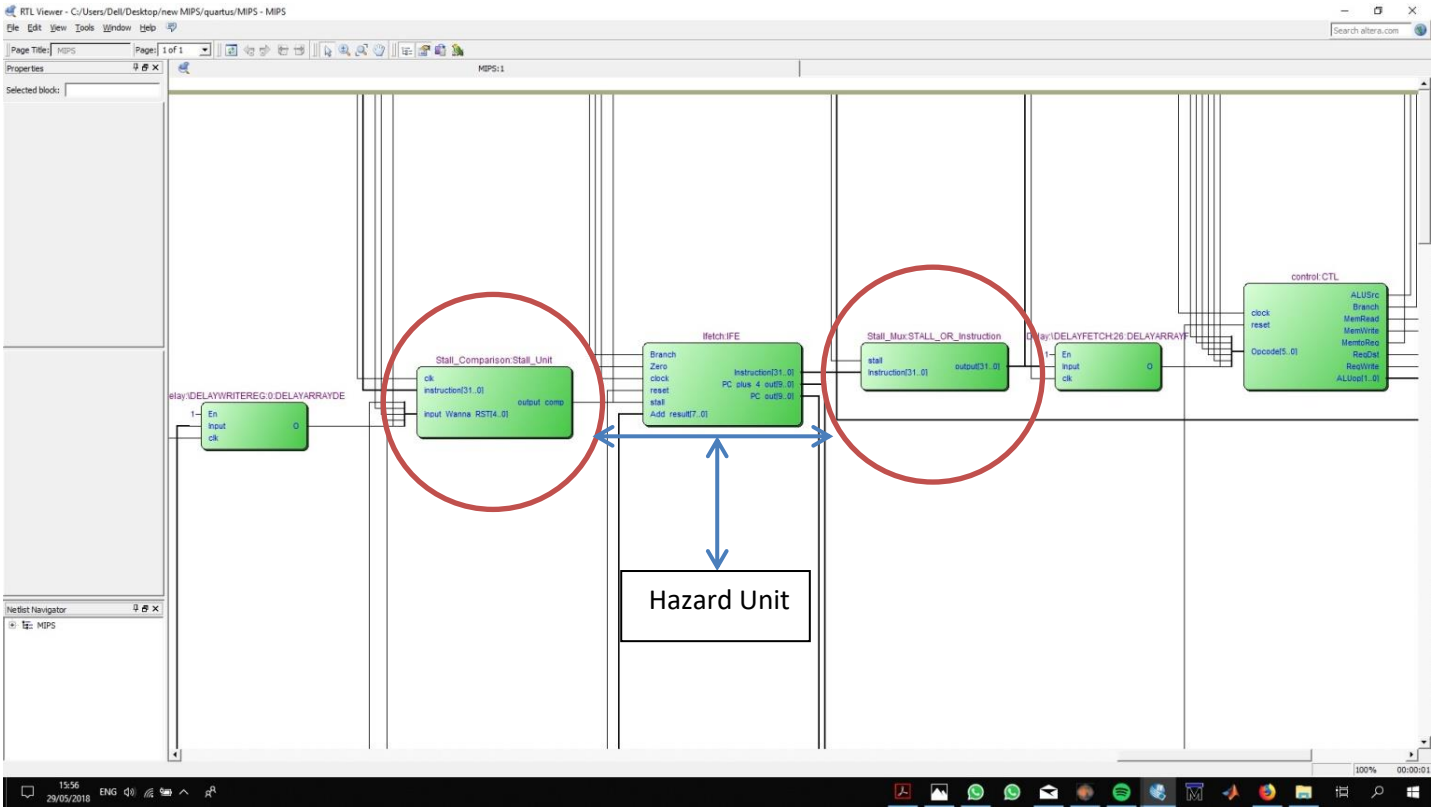


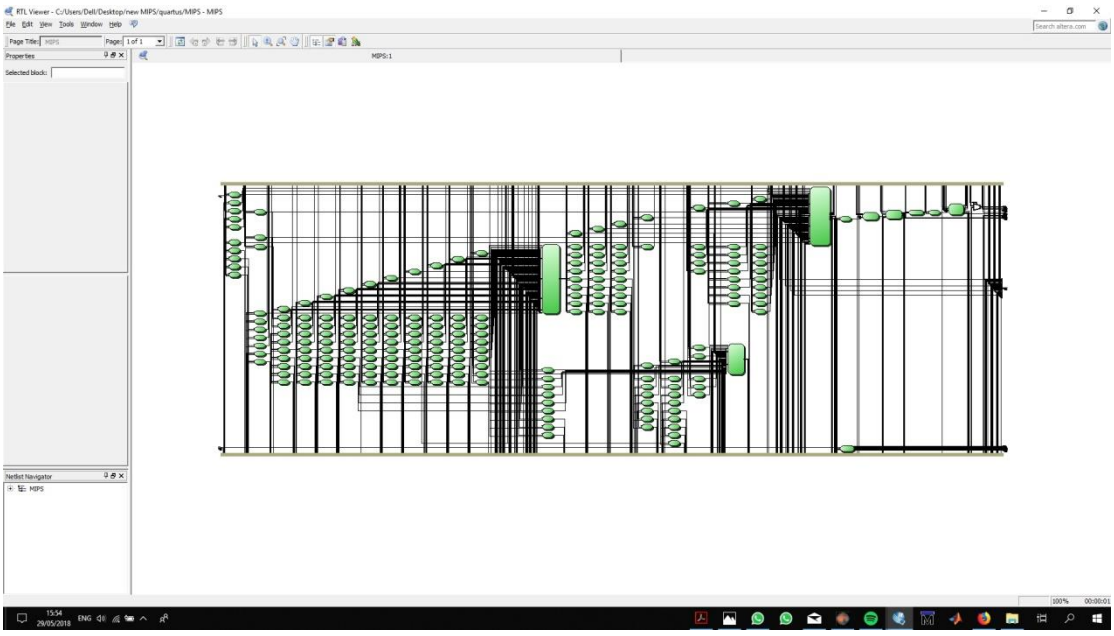
Output – sorted vector by ModelSim



The result is the same to mars simulation.

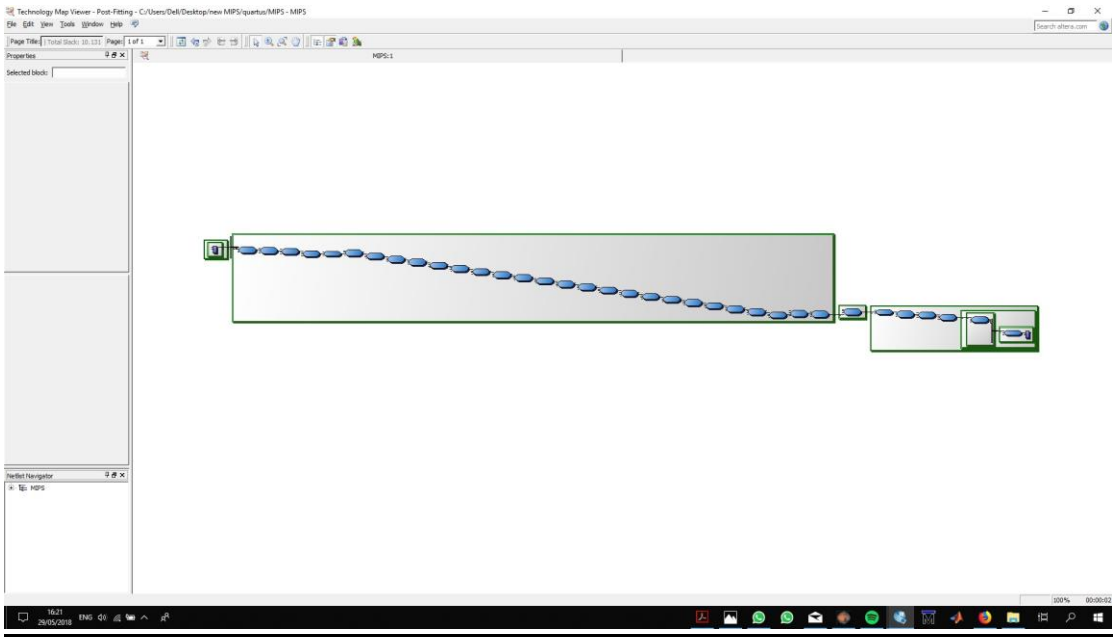
RTL view:


Stall_Comparison:Stall_Unit
lfetch:IFE
Stall_Mux:STALL_OR_Instruction
control:CTL
Hazard Unit



Critical Path:

Our Fmax is 15.1MHZ (slow model).



The critical path consists of the floating point system.

HAZRAD UNIT

This unit detects and deals with 3 types of hazards: (source – Pipeline lectures)

1. *Data Hazards –* An Instruction in flight wants to use a data value that's not "done" yet – Instruction depends on result of prior computation which is not ready yet.
2. *Structural Hazards-* Hardware can't support a combination of instructions.
3. *Control Hazards –* Pipeline of branches and other instructions which change the PC.

Measures taken to deal with each hazard:

A. Stall.
B. Double Pump.
C. Branch Detection and Computation at Fetch Phase.
D. Redesign of resources – alternation of several command to behave differently (XOR, NOR and SLT).

Regular cases that were checked:

A. Instructions without hazards
B. Dependency of branches (taken/not taken) and the following instructions
C. Usage of the redesigned of resources.

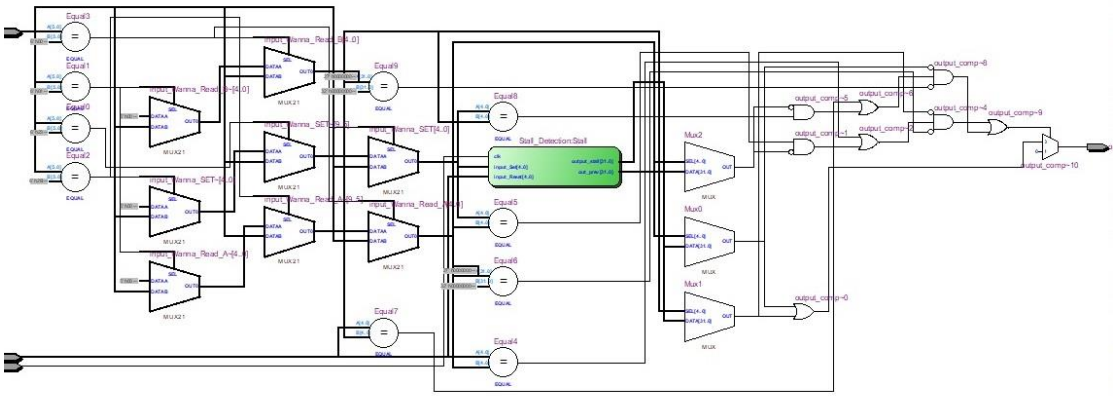
Extreme cases that were checked:

A. Reading from the same register we want to write to – no hazard, the system should continue in fetching (eg. Add $t1,$t1,$t1).
B. Resetting and Acquiring the same register in the same clock – (e.g add $t1(restes),$t1,$t1, clock, clock, add $t1(acquire),$t1,$t1)

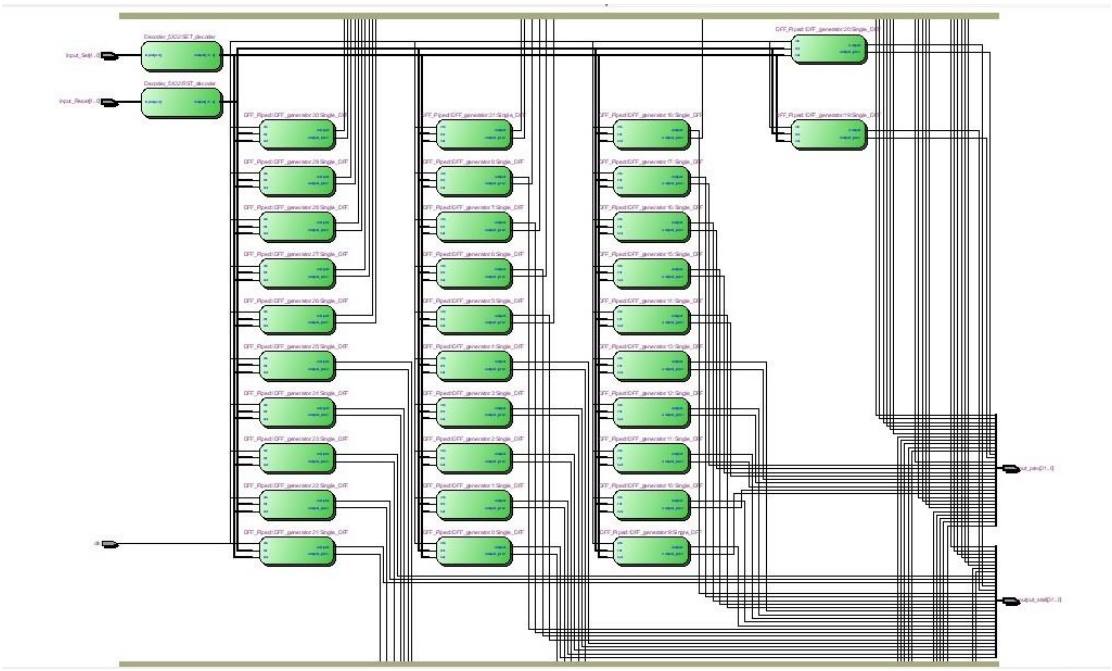
Forbidden cases that were not checked:

A. Support of new intsructions (addi,lui, etc).
B. Support of comparison and usage with scalars, instead if a premade register.

Hazard Unit



Stall-unit



Port	direction	size	functionality
clk	In	1	clock
Wanna_Reset	In	32	
Instruction	in	32	
Output_cpmp	out	1	Out←hazard/safe

This unit gets the current instruction and analyzes its components:

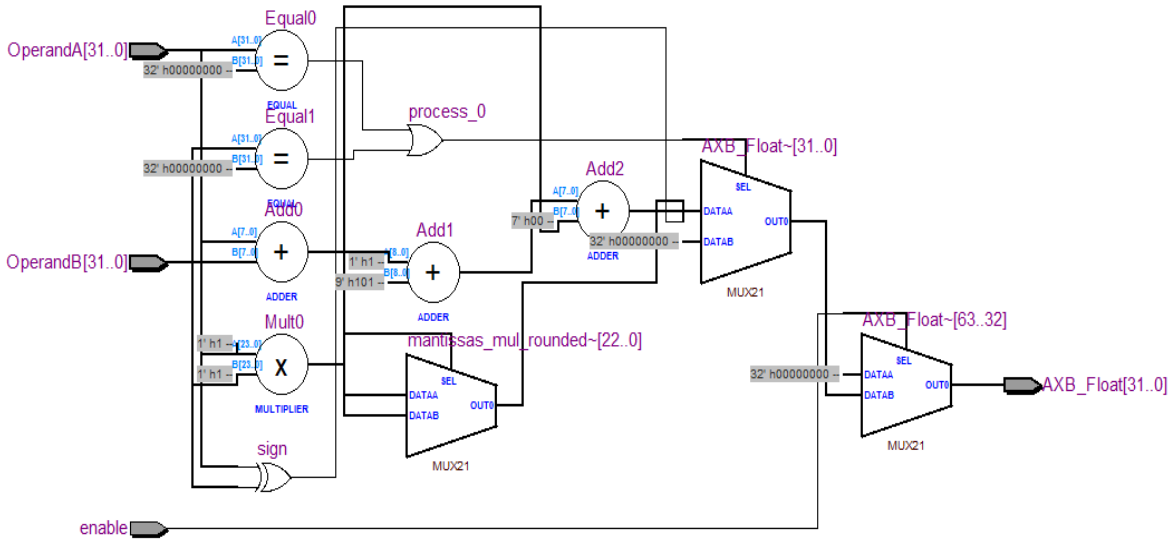
A. Opcode type – R-format,I-Type,J-Type
B. Registers needed to read from – depends on the opcode type – each type handled differently.

The unit also gets the register now cleared (Wanna_Reset) – when we start the WB phase.

The unit also stores list of occupied/available registers for current time and the clock beforehand (by the equation $Q(n) = D(n) * (\overline{Rst(n)} * Q(n-1))$

The unit then decides whether we are good to go or we detected a hazard.

FPM:



Port	direction	size	functionality
OperandA	In	32	Floating point IEEE
OperandB	In	32	Floating point IEEE
enable	In	1	
AxB_Float	out	32	Out←A*B

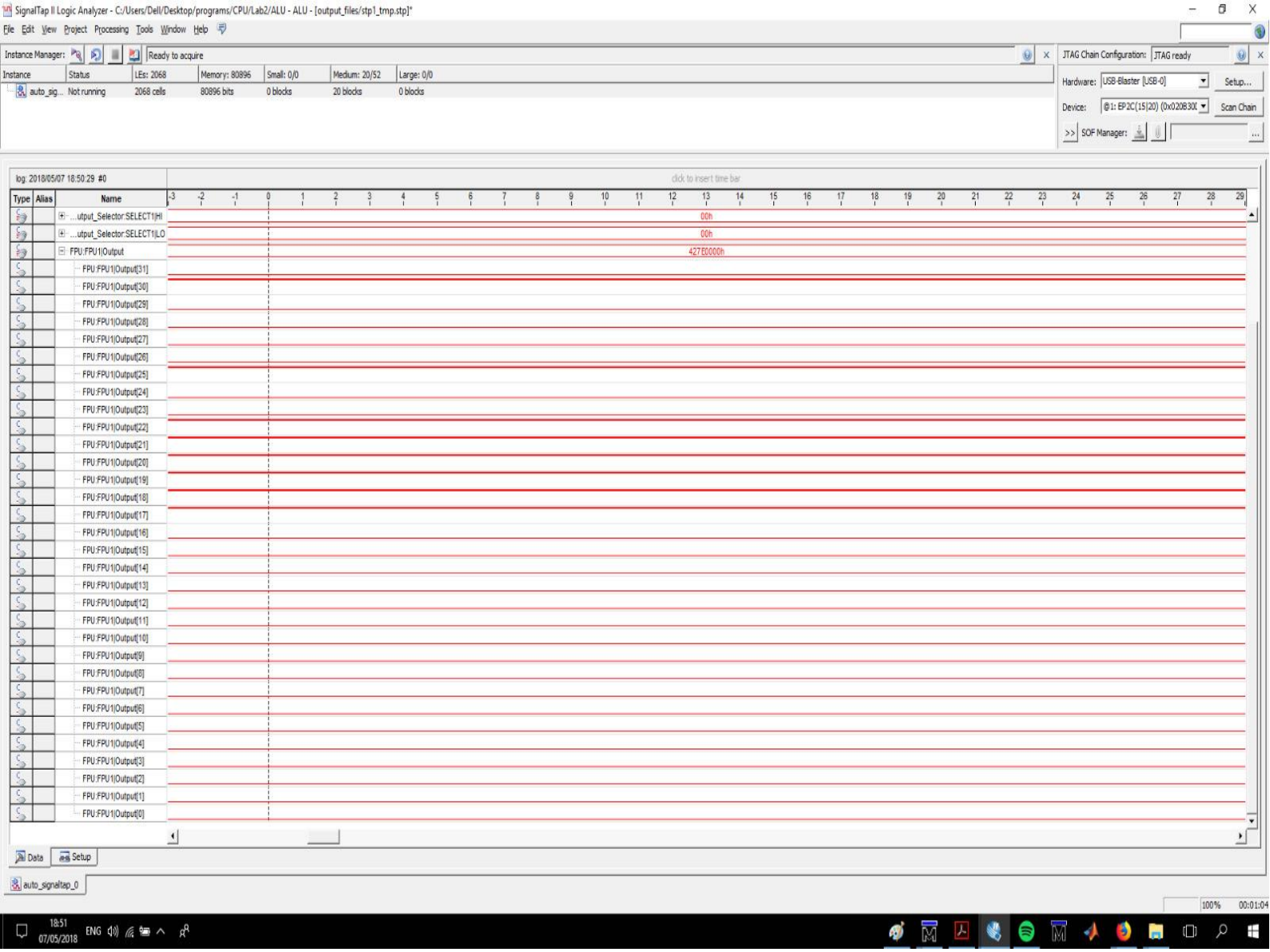
This Unit is responsible for multiplying two floats, it seperates the two floats to

Two mantissas and two exponents

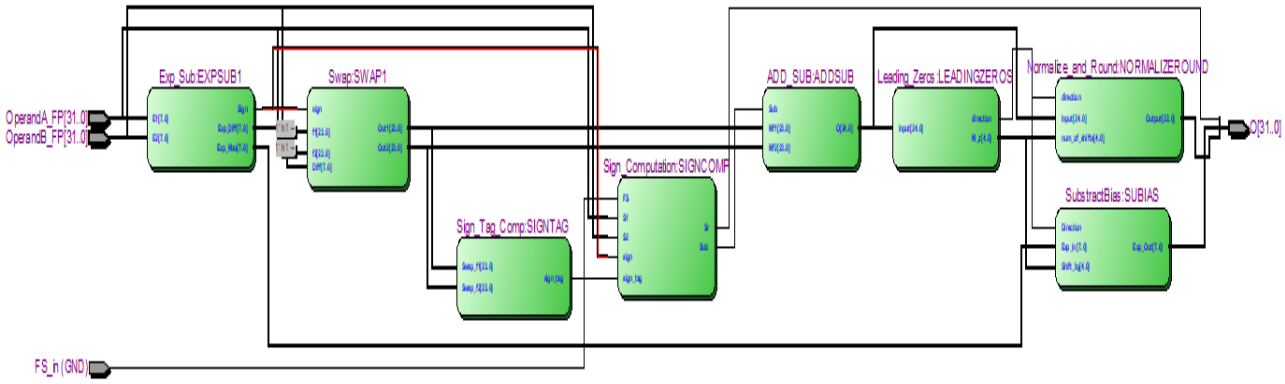
The two manitssas are multiplied while the two exponents are added together

Eventually we put the result in AXB_Float and transmit it out of this module.

FPM SIGNAL TAP

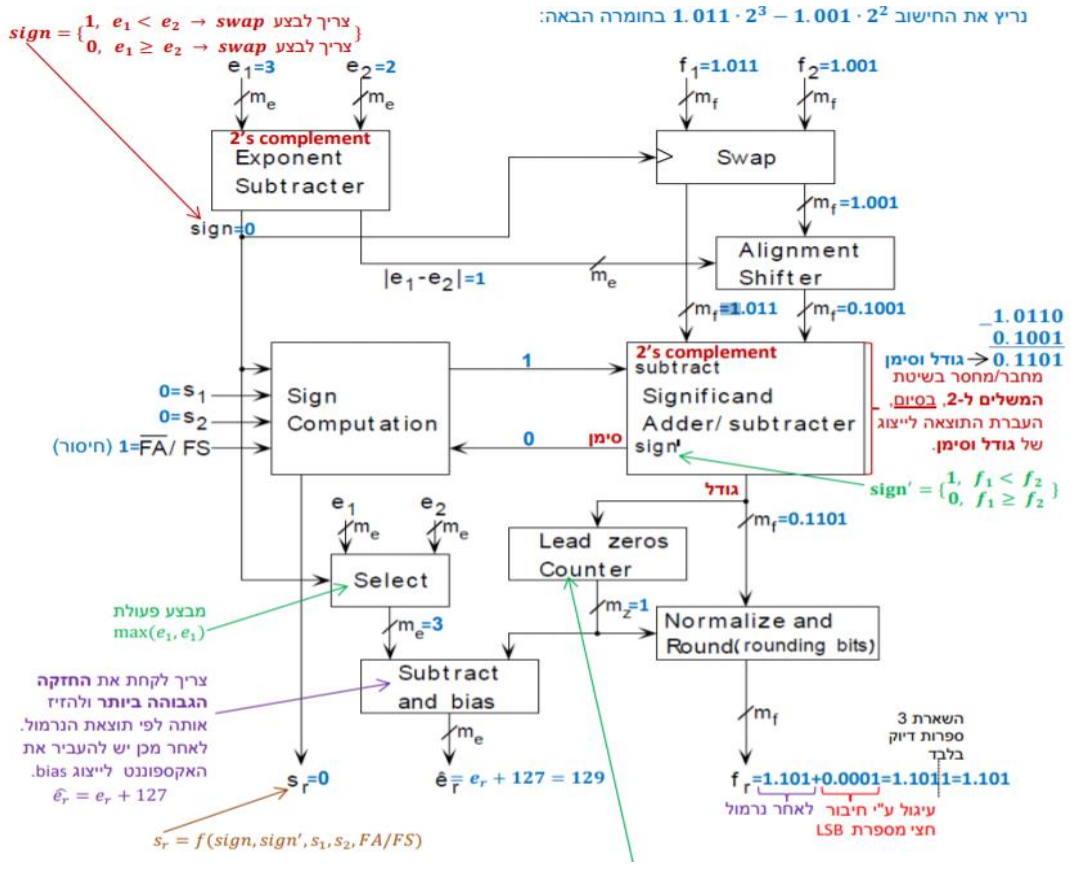


FPA



Port	direction	size	functionality
OperandA	In	32	Floating point IEEE
OperandB	In	32	Floating point IEEE
Fs_in	in	1	Fs='1'→A-B,Fs='0'→A+B
O	out	32	Out←A+/-B

OperandA and OperandB are two floats and O is A+B or A-B according to the value of Fs. This unit implements the next diagram :



Quick explanation of the units (same to previous assignment):

Exp_Sub:

the input to this unit is the E1 – Exponent of A, E2-Exponent of B
The output of this system is Diff ,sign and Exp_Max where :

$$\text{sign} = \begin{cases} 1 & e1 < e2 \\ 0 & else \end{cases}$$

$$\text{Diff} = |\text{E1} - \text{E2}|$$

$$\text{Exp\_Max} = \max\{\text{E1}, \text{E2}\}$$

Swap:

the input of this unit is F1- mantissa of A normalized which is 23 bit,
F2- mantissa of B normalized which is 23 bit,
sign-'1'-> switch,
Difff-> how much to shift left Swap_out2
The output of this unit is
Out1 and Out2 and they are the inputs to ADD_SUB

Sign_Tag_Computation:

the input of this unit is Out1,Out2
The output is sign' which is determine by:

$$\text{Sign}' = \begin{cases} 1 & F1 < F2 \\ 0 & else \end{cases}$$

ADD_SUB:

the inputs of this unit is Out1,Out2 and FS .
the output is O which is of size 24 bit

$$\text{O} = \begin{cases} Out1 + Out2 & FS = '0' \\ Out1 - Out2 & FS = '1' \end{cases}$$

Leading_Zeros:

Input is O which is 24 bits
Output is M_z and direction which M_z is the how much we need to shift the O inorder to make in normalized and direction is to which side to shift.

Normalize and Round:

Input is O ,direction,M_z
Output is *Output* which is O normalized and which is 22 bit(Rounded O)

Sign_Computation:

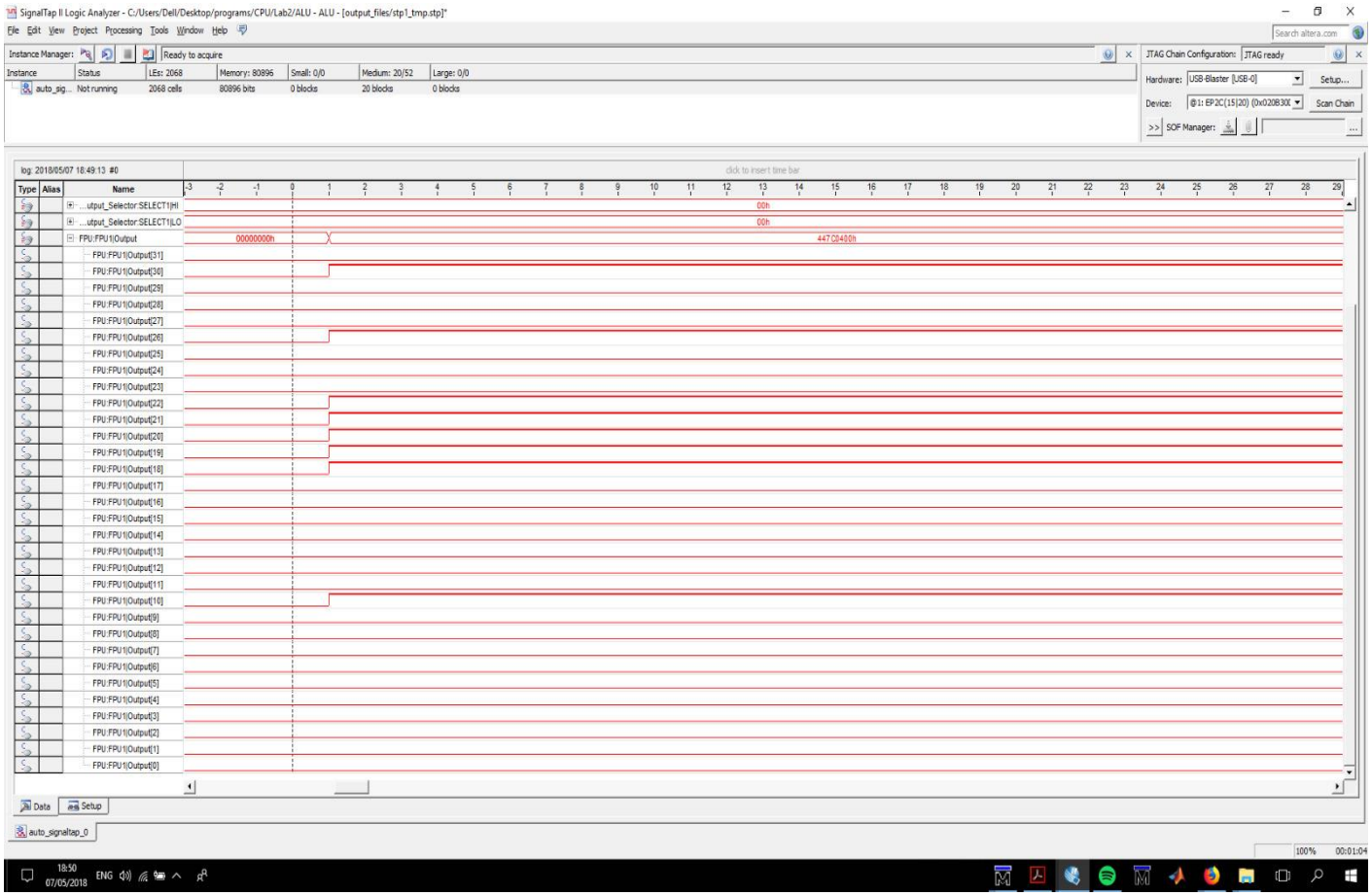
Input is sign,S1->the sign of A ,S2->the sign of B,Fs,Sign'
Output is Sr-> the sign of the output and Subtract-> this is the FS_in of ADD_SUB Unit.

SubtractBias:

Input is Exp_Max ,direction and M_z as described above
Output is Exp_Out which is 7 bit and is described as

$$\text{Exp\_Out} = \begin{cases} Exp\_Max - M\_z & direction = 1 \\ Exp\_Max + M\_z & direction = 0 \end{cases}$$

FPA SIGNAL TAP



Conclusions and further improvements

We believe that we managed to accomplish this lab's targets.

Always there is a room for improvement – given the time to do so.

We believe we could accelerate the system's CPI and it's Period time (Fmax) with several modifications:

A. Deepen the Pipeline's depth – break the Instruction computation to more phases – shorter operations→Fmax increases.
B. Support I-type instructions – less stages in application layer.
C. Make floating point computation compatible with the pipeline – this is the critical path and break it into stages will make a great performance improvement.