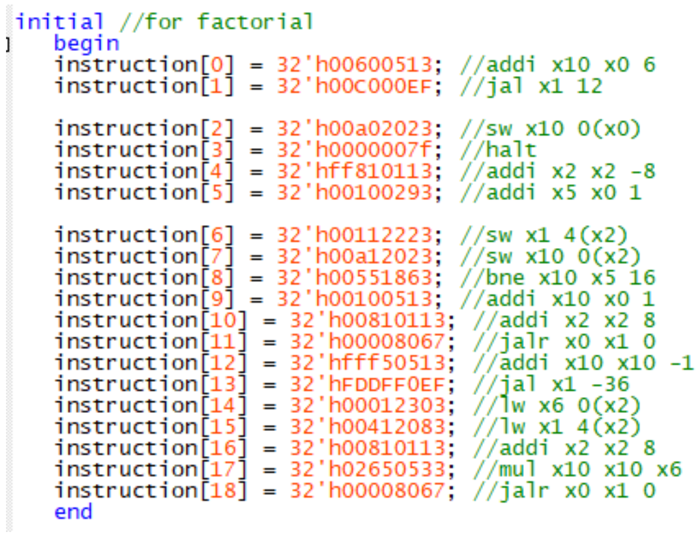
Homework 7 Report

Question 1

Figure 1 shows the code for the factorial program, the spaces are where the nops were originally placed and expected. Prior to submission the hazard detector exported to hex displays and a key was used to advance the clock to ensure the hazards were detected at the right place. Proper functioning is shown by the memory contents of figure 2 which matches that in assignment 6. To test the number of clock cycles used the LEDs were incremented at each clock edge so long as the instruction was not a halt. In part a the jalr instruction required a nop as the addition was dependent on the alu module, though in this part it was changed. For that reason the counts were a bit different with and without the hazard detector, being at 71 cycles (figure 3) now while in part a it was 83 cycles (figure 4).

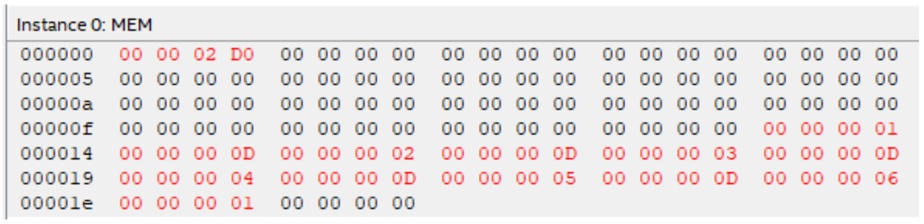


```
initial //for factorial
    begin
    instruction[0] = 32'h00600513; //addi x10 x0 6
    instruction[1] = 32'h00C000EF; //jal x1 12

    instruction[2] = 32'h00a02023; //sw x10 0(x0)
    instruction[3] = 32'h0000007f; //halt
    instruction[4] = 32'hff810113; //addi x2 x2 -8
    instruction[5] = 32'h00100293; //addi x5 x0 1

    instruction[6] = 32'h00112223; //sw x1 4(x2)
    instruction[7] = 32'h00a12023; //sw x10 0(x2)
    instruction[8] = 32'h00551863; //bne x10 x5 16
    instruction[9] = 32'h00100513; //addi x10 x0 1
    instruction[10] = 32'h00810113; //addi x2 x2 8
    instruction[11] = 32'h00008067; //jalr x0 x1 0
    instruction[12] = 32'hfff50513; //addi x10 x10 -1
    instruction[13] = 32'hFDDFF0EF; //jal x1 -36
    instruction[14] = 32'h00012303; //lw x6 0(x2)
    instruction[15] = 32'h00412083; //lw x1 4(x2)
    instruction[16] = 32'h00810113; //addi x2 x2 8
    instruction[17] = 32'h02650533; //mul x10 x10 x6
    instruction[18] = 32'h00008067; //jalr x0 x1 0
    end
```

Figure 1. Factorial Assembly and Machine Code



```
Instance 0: MEM
000000  00 00 02 D0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000005  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000a  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000f  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 01
000014  00 00 00 0D  00 00 00 02  00 00 00 0D  00 00 00 03  00 00 00 0D
000019  00 00 00 04  00 00 00 0D  00 00 00 05  00 00 00 0D  00 00 00 06
00001e  00 00 00 01  00 00 00 00
```

Figure 2. Factorial Program Memory Contents

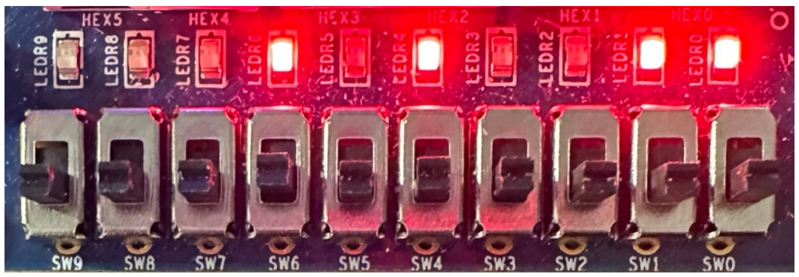


Figure 3. Clock Cycle Count for Factorial Program After Hazard Detection and JALR Change (71)

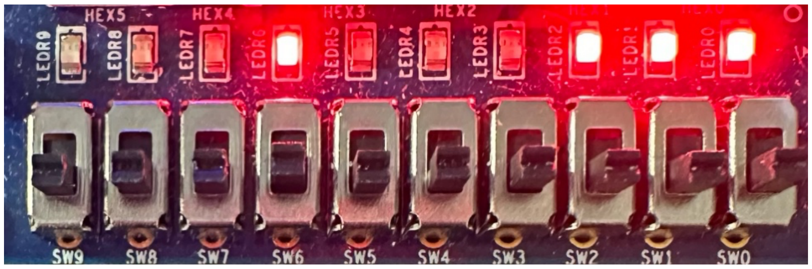
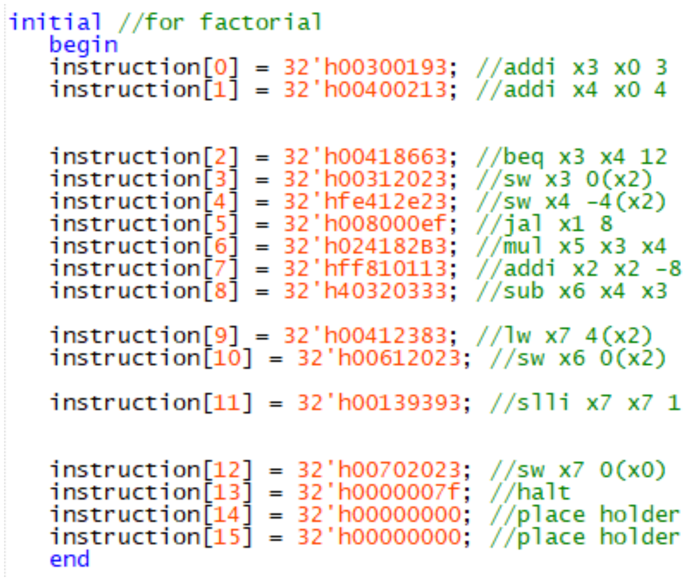


Figure 4. Clock Cycle Count for Factorial Program Before Hazard Detection and JALR Change (83)

Figure 5 shows the code for the verification program. Registers 3 and 4 are loaded and subsequently used with many of the supported functions. I made sure to use beq (since bne is used in the factorial program) and jal, as well as mul, sub (r-types) and addi, sllli (i-types). Compute and use hazards are detected at instruction the beq instructions with registers 3 and 4. Load and use hazards are detected at instruction 11 with register 7. Proper functioning is shown by the memory contents of figure 6, which matches that expected by Venus. Similar cycle count methods used for the factorial program were used here showing a cycle count of 14 with the hazard detector (figure 7) and 18 without (figure 8).

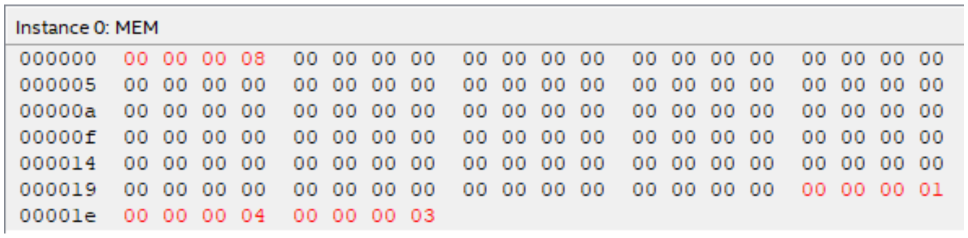

```
initial //for factorial
   begin
   instruction[0] = 32'h00300193; //addi x3 x0 3
   instruction[1] = 32'h00400213; //addi x4 x0 4


   instruction[2] = 32'h00418663; //beq x3 x4 12
   instruction[3] = 32'h00312023; //sw x3 0(x2)
   instruction[4] = 32'hfe412e23; //sw x4 -4(x2)
   instruction[5] = 32'h008000ef; //jal x1 8
   instruction[6] = 32'h024182B3; //mul x5 x3 x4
   instruction[7] = 32'hff810113; //addi x2 x2 -8
   instruction[8] = 32'h40320333; //sub x6 x4 x3

   instruction[9] = 32'h00412383; //lw x7 4(x2)
   instruction[10] = 32'h00612023; //sw x6 0(x2)

   instruction[11] = 32'h00139393; //slli x7 x7 1


   instruction[12] = 32'h00702023; //sw x7 0(x0)
   instruction[13] = 32'h0000007f; //halt
   instruction[14] = 32'h00000000; //place holder
   instruction[15] = 32'h00000000; //place holder
   end
```
Figure 5. Verification Assembly and Machine Code


```
Instance 0: MEM
000000  00 00 00 08  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000005  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000a  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00000f  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000014  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
000019  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 01
00001e  00 00 00 04  00 00 00 03
```
Figure 6. Verification Program Memory Contents

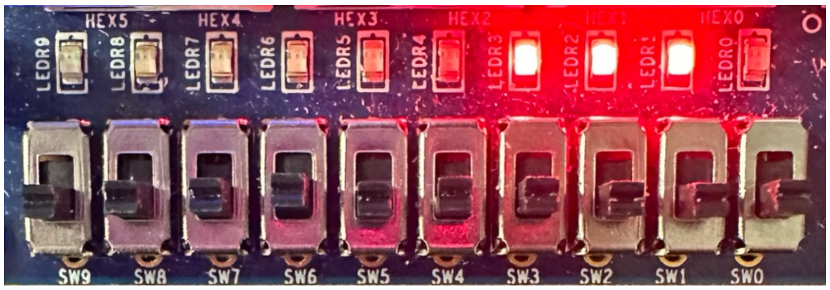


Figure 7. Clock Cycle Count for Verification Program After Hazard Detection and JALR Change (14)

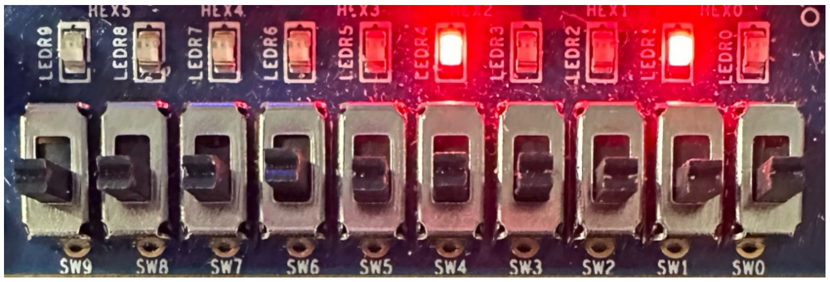
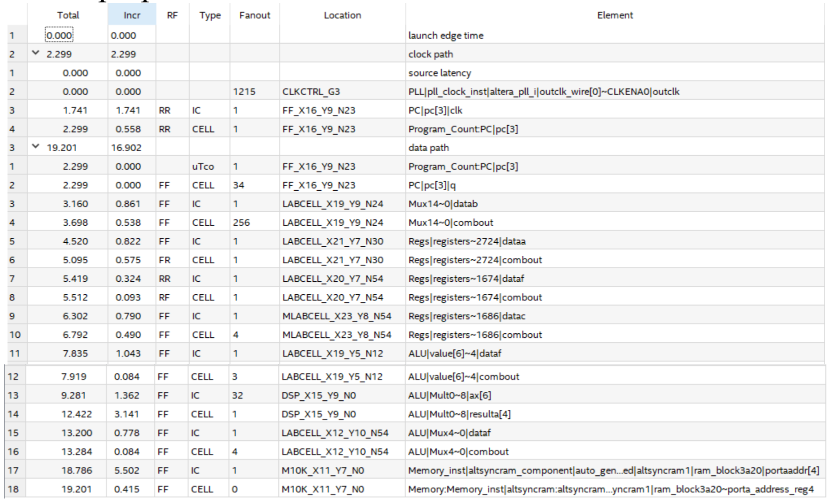


Figure 8. Clock Cycle Count for Verification Program Before Hazard Detection and JALR Change (18)

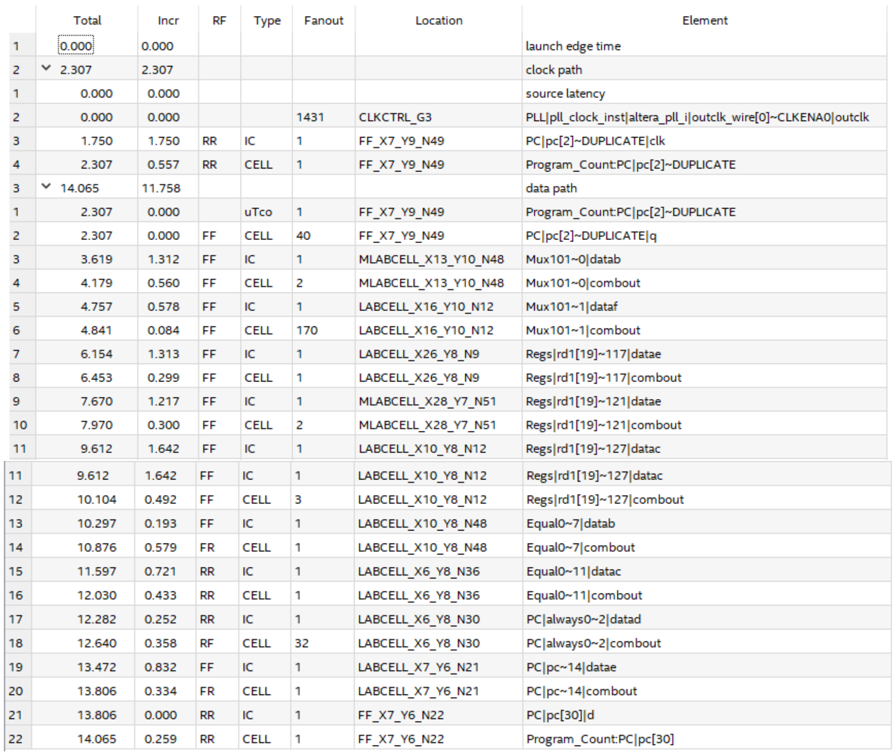
Question 2

Using the timing quest analyzer on the single cycle design to determine where to partition the processor, the critical path is shown in figure 9. Using this information I decided for a three stage pipeline which isolates the alu. The program count, registers, hazard, control and immediate generator are in the first stage, the alu in the second, and the memory in the third. Using the timing quest analyzer on the pipeline processor (figure 10), the critical path is now shown to be in incrementing the pc and comparing the register data for the branch instructions, making the new max frequency of about 71 MHz. This path can be seen in Question 5 via the PC and Register partitions. We will find that this is not accurate, seeing as I used optimized fitter and synthesis settings for speed. This may have come at the expense of area or other factors, but works very well for the purposes of this lab.



	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	2.299	2.299					clock path
1	0.000	0.000					source latency
2	0.000	0.000			1215	CLKCTRL_G3	PLL|pll_clock_inst|altera_pll_i|outclk_wire[0]~CLKENA0|outclk
3	1.741	1.741	RR	IC	1	FF_X16_Y9_N23	PC|pc[3]|clk
4	2.299	0.558	RR	CELL	1	FF_X16_Y9_N23	Program_Count:PC|pc[3]
3	19.201	16.902					data path
1	2.299	0.000		uTco	1	FF_X16_Y9_N23	Program_Count:PC|pc[3]
2	2.299	0.000	FF	CELL	34	FF_X16_Y9_N23	PC|pc[3]|q
3	3.160	0.861	FF	IC	1	LABCELL_X19_Y9_N24	Mux14~0|datab
4	3.698	0.538	FF	CELL	256	LABCELL_X19_Y9_N24	Mux14~0|combout
5	4.520	0.822	FF	IC	1	LABCELL_X21_Y7_N30	Regs|registers~2724|dataa
6	5.095	0.575	FR	CELL	1	LABCELL_X21_Y7_N30	Regs|registers~2724|combout
7	5.419	0.324	RR	IC	1	LABCELL_X20_Y7_N54	Regs|registers~1674|dataf
8	5.512	0.093	RF	CELL	1	LABCELL_X20_Y7_N54	Regs|registers~1674|combout
9	6.302	0.790	FF	IC	1	MLABCELL_X23_Y8_N54	Regs|registers~1686|datac
10	6.792	0.490	FF	CELL	4	MLABCELL_X23_Y8_N54	Regs|registers~1686|combout
11	7.835	1.043	FF	IC	1	LABCELL_X19_Y5_N12	ALU|value[6]~4|dataf
12	7.919	0.084	FF	CELL	3	LABCELL_X19_Y5_N12	ALU|value[6]~4|combout
13	9.281	1.362	FF	IC	32	DSP_X15_Y9_N0	ALU|Mult0~8|ax[6]
14	12.422	3.141	FF	CELL	1	DSP_X15_Y9_N0	ALU|Mult0~8|resulta[4]
15	13.200	0.778	FF	IC	1	LABCELL_X12_Y10_N54	ALU|Mux4~0|dataf
16	13.284	0.084	FF	CELL	4	LABCELL_X12_Y10_N54	ALU|Mux4~0|combout
17	18.786	5.502	FF	IC	1	M10K_X11_Y7_N0	Memory_inst|altsyncram_component|auto_gen...ed|altsyncram1|ram_block3a20|portaaddr[4]
18	19.201	0.415	FF	CELL	0	M10K_X11_Y7_N0	Memory:Memory_inst|altsyncram:altsyncram...yncram1|ram_block3a20~porta_address_reg4

Figure 9. Timing Quest Analyzer for the Single Cycle Processor, Showing Critical Path Through ALU Multiplier.

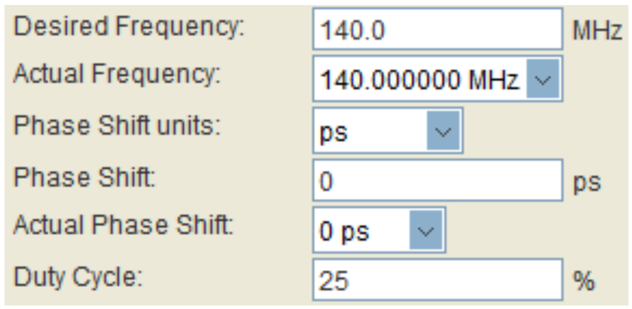


	Total	Incr	RF	Type	Fanout	Location	Element
1	0.000	0.000					launch edge time
2	2.307	2.307					clock path
1	0.000	0.000					source latency
2	0.000	0.000			1431	CLKCTRL_G3	PLL|pll_clock_inst|altera_pll_i|outclk_wire[0]~CLKENA0|outclk
3	1.750	1.750	RR	IC	1	FF_X7_Y9_N49	PC|pc[2]~DUPLICATE|clk
4	2.307	0.557	RR	CELL	1	FF_X7_Y9_N49	Program_Count:PC|pc[2]~DUPLICATE
3	14.065	11.758					data path
1	2.307	0.000		uTco	1	FF_X7_Y9_N49	Program_Count:PC|pc[2]~DUPLICATE
2	2.307	0.000	FF	CELL	40	FF_X7_Y9_N49	PC|pc[2]~DUPLICATE|q
3	3.619	1.312	FF	IC	1	MLABCELL_X13_Y10_N48	Mux101~0|datab
4	4.179	0.560	FF	CELL	2	MLABCELL_X13_Y10_N48	Mux101~0|combout
5	4.757	0.578	FF	IC	1	LABCELL_X16_Y10_N12	Mux101~1|dataf
6	4.841	0.084	FF	CELL	170	LABCELL_X16_Y10_N12	Mux101~1|combout
7	6.154	1.313	FF	IC	1	LABCELL_X26_Y8_N9	Regs|rd1[19]~117|datae
8	6.453	0.299	FF	CELL	1	LABCELL_X26_Y8_N9	Regs|rd1[19]~117|combout
9	7.670	1.217	FF	IC	1	MLABCELL_X28_Y7_N51	Regs|rd1[19]~121|datae
10	7.970	0.300	FF	CELL	2	MLABCELL_X28_Y7_N51	Regs|rd1[19]~121|combout
11	9.612	1.642	FF	IC	1	LABCELL_X10_Y8_N12	Regs|rd1[19]~127|datac
11	9.612	1.642	FF	IC	1	LABCELL_X10_Y8_N12	Regs|rd1[19]~127|datac
12	10.104	0.492	FF	CELL	3	LABCELL_X10_Y8_N12	Regs|rd1[19]~127|combout
13	10.297	0.193	FF	IC	1	LABCELL_X10_Y8_N48	Equal0~7|datab
14	10.876	0.579	FR	CELL	1	LABCELL_X10_Y8_N48	Equal0~7|combout
15	11.597	0.721	RR	IC	1	LABCELL_X6_Y8_N36	Equal0~11|datac
16	12.030	0.433	RR	CELL	1	LABCELL_X6_Y8_N36	Equal0~11|combout
17	12.282	0.252	RR	IC	1	LABCELL_X6_Y8_N30	PC|always0~2|datad
18	12.640	0.358	RF	CELL	32	LABCELL_X6_Y8_N30	PC|always0~2|combout
19	13.472	0.832	FF	IC	1	LABCELL_X7_Y6_N21	PC|pc~14|datae
20	13.806	0.334	FR	CELL	1	LABCELL_X7_Y6_N21	PC|pc~14|combout
21	13.806	0.000	RR	IC	1	FF_X7_Y6_N22	PC|pc[30]|d
22	14.065	0.259	RR	CELL	1	FF_X7_Y6_N22	Program_Count:PC|pc[30]

Figure 10. Timing Quest Analyzer for the Pipeline Processor, Showing Critical Path Through the PC for Branches.

Question 3

Incrementing the PLL on the actual board, the maximum clock setting for the pipeline processor are shown in Figure 11. The maximum was 140MHz (7.1428 with a duty cycle of 25% as the memory is read on the posedge and written on the negedge) making the total run time for 71 cycles about 507 nanoseconds, a major improvement from the 71 cycles at 70 Mhz for a run time of about 1014 nanoseconds. As discussed before this may be because of the fact that a lot of the settings were chosen to optimize for speed. It may also be with limitations in the modeling of the timing quest analyzer, the PLL strategy (having a 25% duty cycle) may interact differently with the board than what is expected in the modeling (skews, noise, etc.). I also used what I think to be a fast board, so it may be due in part to manufacturing variability in the boards. Additionally, like with the multiplier this likely assumed that the hardware is used to its fullest extend when in fact the read data and comparing of the registers uses small numbers instead of the whole range of the register, making it much faster than what is expected. This is seen in the critical path being through the 22nd bit of the register, when in fact we only use the first few bits.



Desired Frequency:	140.0	MHz
Actual Frequency:	140.000000 MHz	
Phase Shift units:	ps	
Phase Shift:	0	ps
Actual Phase Shift:	0 ps	
Duty Cycle:	25	%

Figure 11. Maximum Clock Frequency for the FPGA Board.

Question 4

For branches and jumps I decided to resolve them in the first partition in my pipeline to avoid extra clock cycles. For branches the read data from registers 1 and 2 are compared immediate to generate the zero register. Taking this along with the branch output from the control unit the pc can immediately update without having to move on to the next stage of the pipeline. For jump and link the register file is hard coded to store the pc into register one when a jal instruction and link is encountered, at which point the pc intercepts the immediate value from the immediate generator and is added to update the pc. Similarly when a jalr instruction is encountered the read data from register 1 is intercepted and added to the pc and immediate to update the pc. The control unit effectively makes these instructions nops so they can run through the processor but the pc updates immediate, avoiding many of the nops that would have to occur otherwise.

Question 5

The floorplan partitions for the hazard detector, immediate generator, alu, memory, pipeline, program count, and registers is shown in figure 12. The resource usage summary post-fitting is showing in figure 13 showing a usage of 2012 ALMs, 1580 dedicated logic registers, 2 DSPs, and 1 PLL.

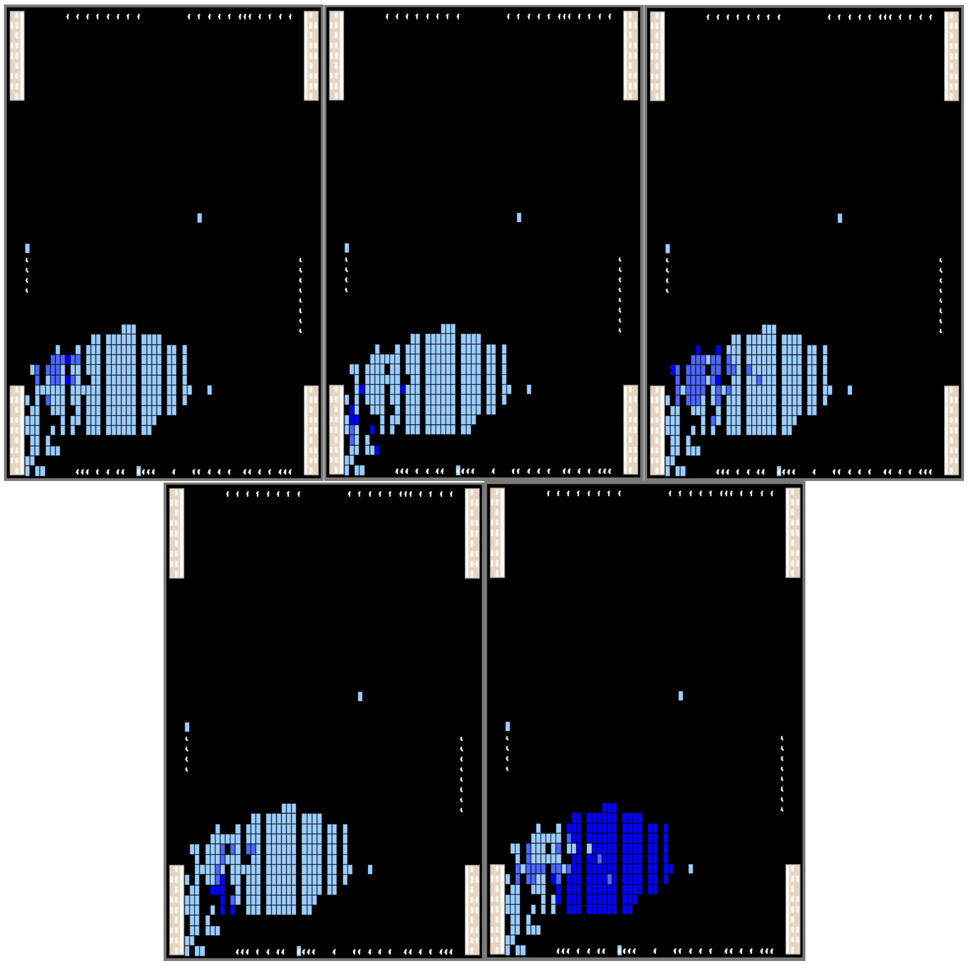
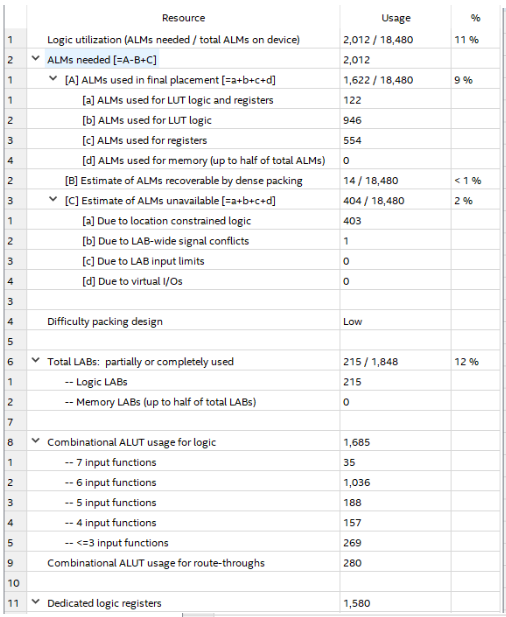
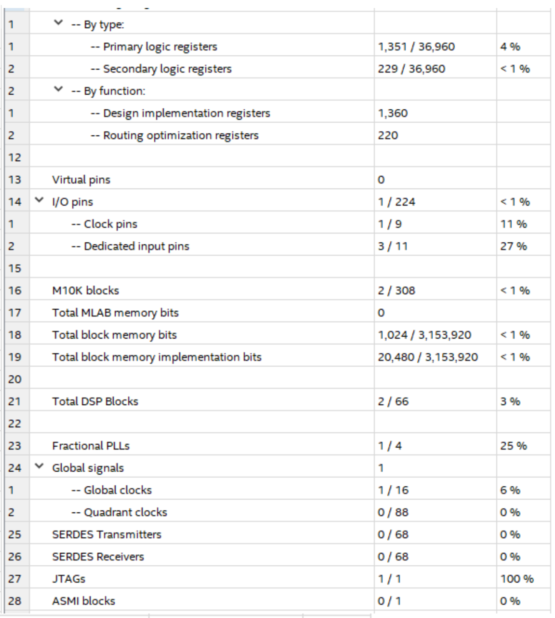


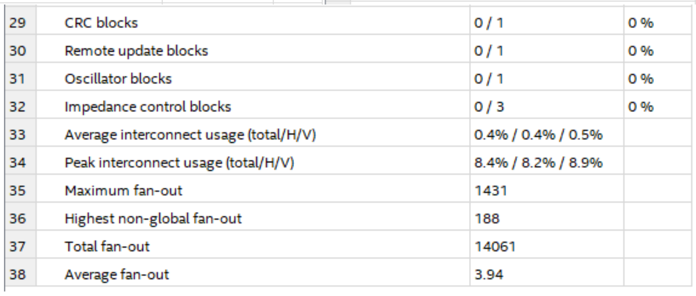
Figure 12. Partition Floor Plan of the ALU, Memory, and Pipeline (Top Row, Respectively) and Program Count, Registers (Bottom Row, Respectively)



	Resource	Usage	%
1	Logic utilization (ALMs needed / total ALMs on device)	2,012 / 18,480	11 %
2	ALMs needed [=A-B+C]	2,012	
1	[A] ALMs used in final placement [=a+b+c+d]	1,622 / 18,480	9 %
1	[a] ALMs used for LUT logic and registers	122	
2	[b] ALMs used for LUT logic	946	
3	[c] ALMs used for registers	554	
4	[d] ALMs used for memory (up to half of total ALMs)	0	
2	[B] Estimate of ALMs recoverable by dense packing	14 / 18,480	< 1 %
3	[C] Estimate of ALMs unavailable [=a+b+c+d]	404 / 18,480	2 %
1	[a] Due to location constrained logic	403	
2	[b] Due to LAB-wide signal conflicts	1	
3	[c] Due to LAB input limits	0	
4	[d] Due to virtual I/Os	0	
3			
4	Difficulty packing design	Low	
5			
6	Total LABs: partially or completely used	215 / 1,848	12 %
1	-- Logic LABs	215	
2	-- Memory LABs (up to half of total LABs)	0	
7			
8	Combinational ALUT usage for logic	1,685	
1	-- 7 input functions	35	
2	-- 6 input functions	1,036	
3	-- 5 input functions	188	
4	-- 4 input functions	157	
5	-- <=3 input functions	269	
9	Combinational ALUT usage for route-throughs	280	
10			
11	Dedicated logic registers	1,580	



1	-- By type:		
1	-- Primary logic registers	1,351 / 36,960	4 %
2	-- Secondary logic registers	229 / 36,960	< 1 %
2	-- By function:		
1	-- Design implementation registers	1,360	
2	-- Routing optimization registers	220	
12			
13	Virtual pins	0	
14	I/O pins	1 / 224	< 1 %
1	-- Clock pins	1 / 9	11 %
2	-- Dedicated input pins	3 / 11	27 %
15			
16	M10K blocks	2 / 308	< 1 %
17	Total MLAB memory bits	0	
18	Total block memory bits	1,024 / 3,153,920	< 1 %
19	Total block memory implementation bits	20,480 / 3,153,920	< 1 %
20			
21	Total DSP Blocks	2 / 66	3 %
22			
23	Fractional PLLs	1 / 4	25 %
24	Global signals	1	
1	-- Global clocks	1 / 16	6 %
2	-- Quadrant clocks	0 / 88	0 %
25	SERDES Transmitters	0 / 68	0 %
26	SERDES Receivers	0 / 68	0 %
27	JTAGs	1 / 1	100 %
28	ASMI blocks	0 / 1	0 %



29	CRC blocks	0 / 1	0 %
30	Remote update blocks	0 / 1	0 %
31	Oscillator blocks	0 / 1	0 %
32	Impedance control blocks	0 / 3	0 %
33	Average interconnect usage (total/H/V)	0.4% / 0.4% / 0.5%	
34	Peak interconnect usage (total/H/V)	8.4% / 8.2% / 8.9%	
35	Maximum fan-out	1431	
36	Highest non-global fan-out	188	
37	Total fan-out	14061	
38	Average fan-out	3.94	

Figure 13. Resource Usage (Post-Fitting) Showing Resources Used in the Final Design of the Pipeline Processor.