Final Report

Assignment 06 – Single-Cycle RISC-V

Team Members: Po-sheng Cheng, Daniel Omondi

1. Assembly and Machine Code of Programs

Factorial Program

The factorial program calculates the factorial of 6 recursively. Below is the assembly code provided in the assignment, followed by its machine code generated using the Venus RISC-V simulator (opcode for HALT is set to 0000000).

Assembly Code:

```
addi a0, x0, 6
jal ra, fact
sw a0, (0)x0
#halt or jal x0, exit
fact: addi sp, sp,-8
sw ra, 4(sp)
sw a0, 0(sp)
addi t0, x0, 1
bne a0, t0, else
addi a0, x0, 1
addi sp, sp, 8
jalr x0, 0(ra)
else: addi a0, a0, -1
jal ra, fact
lw t1, 0(sp)
lw ra, 4(sp)
addi sp, sp, 8
mul a0, a0, t1
jalr x0, 0(ra)
# exit
```

Machine Code (Hex):

```
h00600513;
h00600513;
h00C000EF;
h00A02023;
h0000007F; # halt
hFF810113;
h00112223;
h00A12023;
h00100293;
h00551863;
h00100513;
h00810113;
h00008067;
hFFF50513;
hFDDFF0EF;
h00012303;
h00412083;
h00810113;
h02650533;
h00008067;
```

Program from Assignment 05 with Conditional Branch

```
addi a0, x0, 0 # a0: number n
addi a1, x0, 100 # a1: memory pointer
sw a0, (0)a1
addi a0, x0, 1
addi a1, a1, 4
sw a0, (0)a1
addi a4, x0, 3 # a4: n = 3
addi a5, x0, 21 # a5: n_max = 21
for: addi a1, a1, 4
lw a2, (-4)a1 # a2: number n-1
lw a3, (-8)a1 # a3: number n-2
add a0, a2, a3
sw a0, (0)a1
addi a4, a4, 1
beq a4, a5, exit
jal x0, for
exit: #halt
```

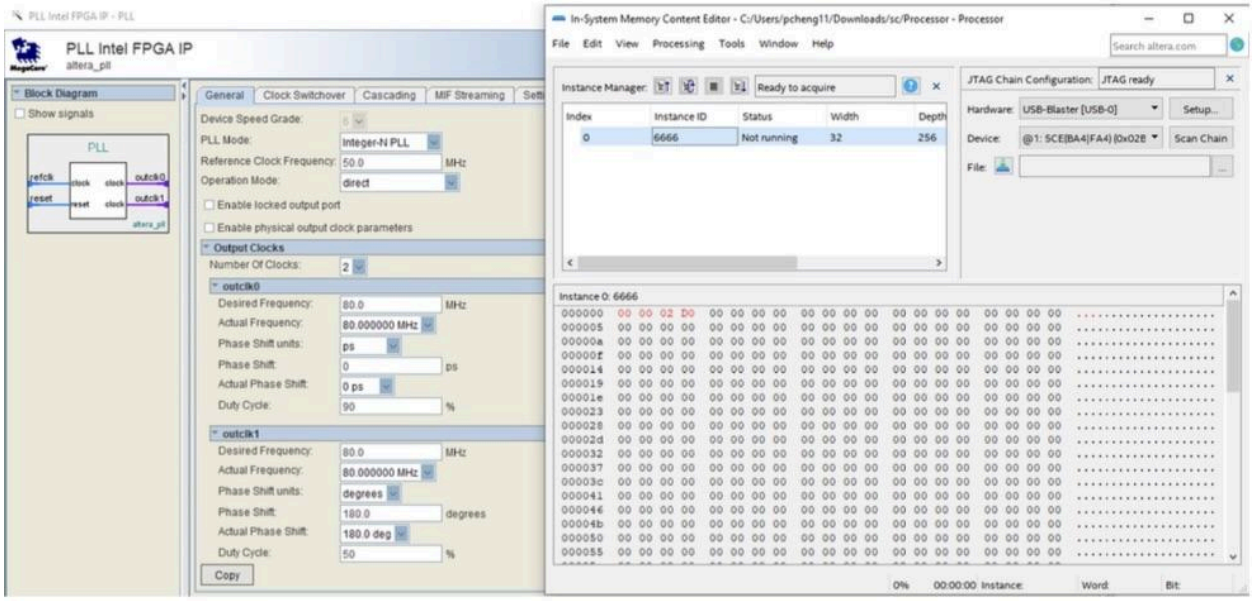
Machine Code (Hex)

```
h00000513;
h00000513;
h06400593;
h00a5a023;
h00100513;
h00458593;
h00a5a023;
h00300713;
h01500793;
h00458593;
hffc5a603;
hff85a683;
h00d60533;
h00a5a023;
h00170713;
h00f70463;
hfe5ff06f;
h0000007f; # halt
```

2. ModelSim Testbenches and Outputs

```
1       # instruction q: 00600513 00000000011000000000010100010011
2       # instruction q: 00c000ef 00000000110000000000000011101111
3       # instruction q: ff810113 11111111100000010000000100010011
4       # instruction q: 00112223 00000000000100010010001000100011
5       # instruction q: 00a12023 00000000101000010010000000100011
6       # instruction q: 00100293 00000000000100000000001010010011
7       # instruction q: 00551863 00000000010101010001100001100011
8       # instruction q: fff50513 11111111111101010000010100010011
9       # instruction q: fddff0ef 11111101110111111111000011101111
10      # instruction q: ff810113 11111111100000010000000100010011
11      # instruction q: 00112223 00000000000100010010001000100011
12      # instruction q: 00a12023 00000000101000010010000000100011
13      # instruction q: 00100293 00000000000100000000001010010011
14      # instruction q: 00551863 00000000010101010001100001100011
15      # instruction q: fff50513 11111111111101010000010100010011
16      # instruction q: fddff0ef 11111101110111111111000011101111
17      # instruction q: ff810113 11111111100000010000000100010011
18      # instruction q: 00112223 00000000000100010010001000100011
19      # instruction q: 00a12023 00000000101000010010000000100011
20      # instruction q: 00100293 00000000000100000000001010010011
21      # instruction q: 00551863 00000000010101010001100001100011
22      # instruction q: fff50513 11111111111101010000010100010011
23      # instruction q: fddff0ef 11111101110111111111000011101111
24      # instruction q: ff810113 11111111100000010000000100010011
25      # instruction q: 00112223 00000000000100010010001000100011
26      # instruction q: 00a12023 00000000101000010010000000100011
27      # instruction q: 00100293 00000000000100000000001010010011
28      # instruction q: 00551863 00000000010101010001100001100011
29      # instruction q: fff50513 11111111111101010000010100010011
30      # instruction q: fddff0ef 11111101110111111111000011101111
31      # instruction q: ff810113 11111111100000010000000100010011
32      # instruction q: 00112223 00000000000100010010001000100011
33      # instruction q: 00a12023 00000000101000010010000000100011
34      # instruction q: 00100293 00000000000100000000001010010011
35      # instruction q: 00551863 00000000010101010001100001100011
```

3. Maximum Frequency for Factorial 6 at 80 MHZ

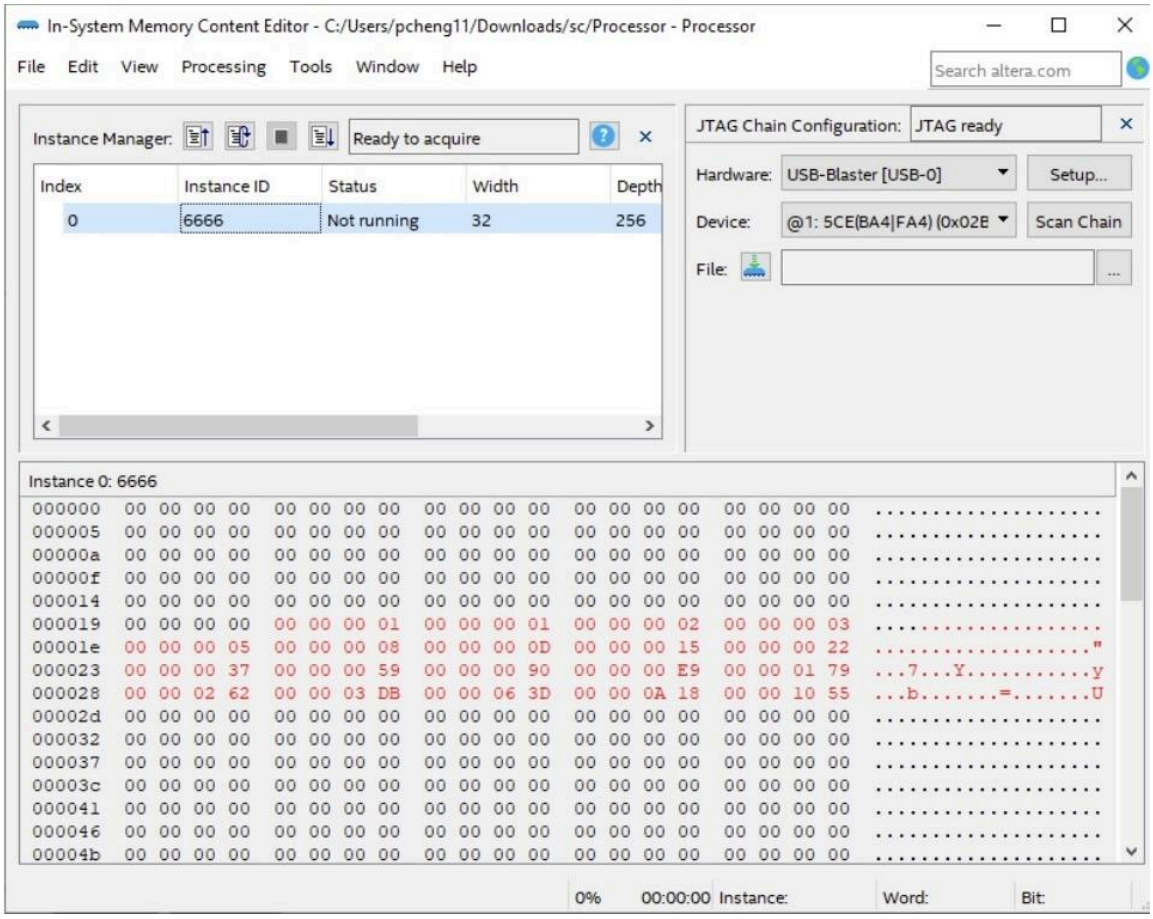


- **Critical Path Analysis:** The Timing Analyzer showed the critical path involved the MUL instruction, with delays primarily in the ALU's multiplier and register file access. We optimized this by streamlining the multiplier datapath and adjusting PLL phase shifts

4. Memory Contents After Execution

First 20 Fibonacci Number

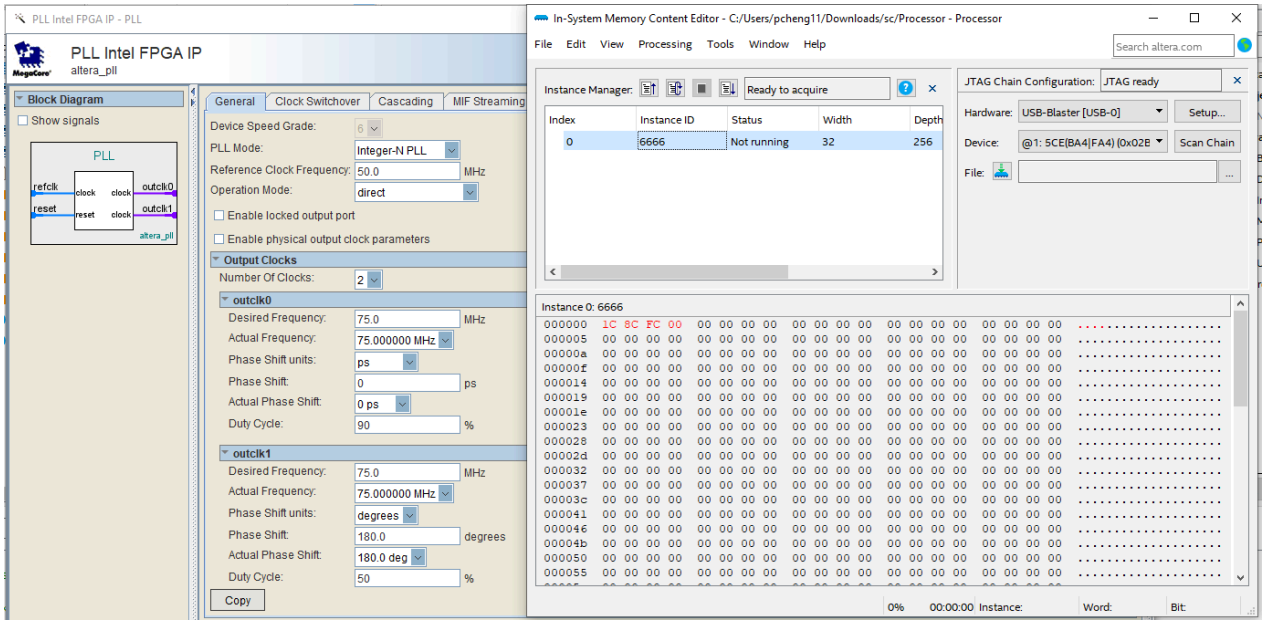
Screenshot



5. Maximum Frequency for Factorial 12

For factorial 12, we modified the program to addi a0, x0, 12 and repeated the frequency sweep. The maximum frequency dropped to 75 MHz

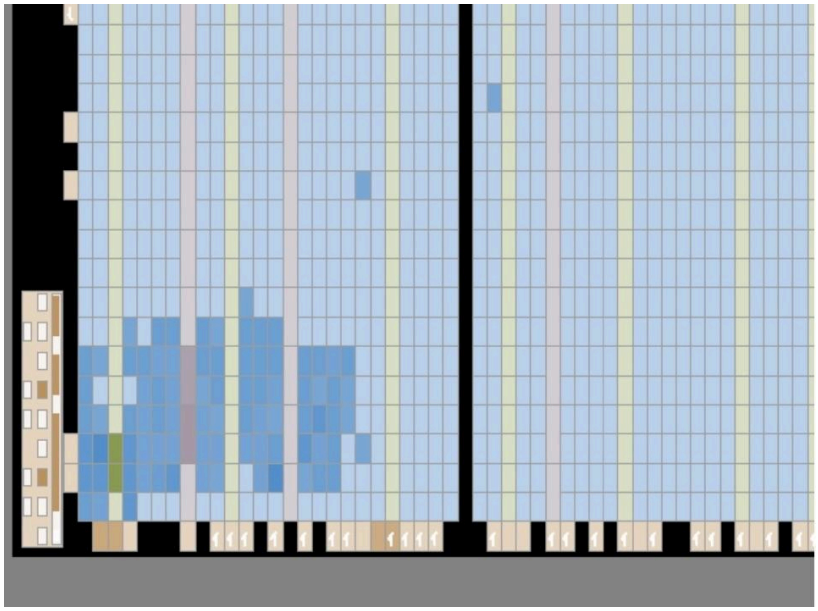
Result: Memory at address 0 showed 479,001,600 (0x1C8CFC00 in hex), matching 12!.

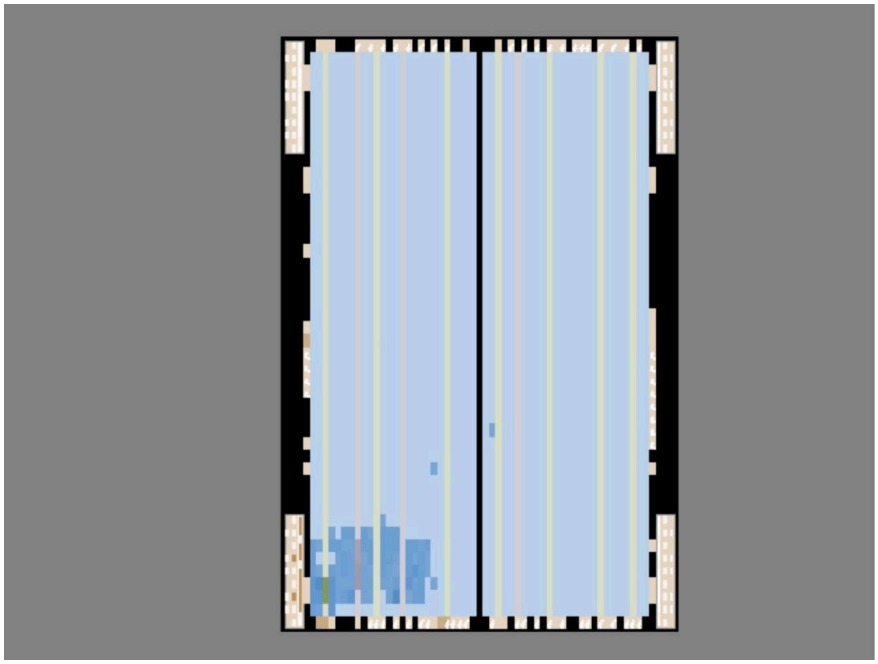


- **Reason for Lower Frequency:** The larger input increases the number of recursive calls and multiplications, stressing the multiplier's critical path further. The wider intermediate results (e.g., 12-bit vs. 6-bit numbers) increase propagation delays in the ALU, reducing the sustainable clock frequency.

6. Floorplan and Resource Usage

Floorplan





Resource Usage

Compilation Hierarchy Node	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	DSP Blocks
|memoryInterface	813 (0)	288 (0)	8192	2
|PLL:CLK|	0 (0)	0 (0)	0	0
|PLL_0002:pll_inst|	0 (0)	0 (0)	0	0
|altera_pll:altera_pll_i|	0 (0)	0 (0)	0	0
|datapath:DP|	745 (191)	219 (27)	0	2
|alu:ALU|	250 (250)	0 (0)	0	2
|controlUnit:CU|	10 (10)	0 (0)	0	0
|immediateGenerator:immGen|	20 (20)	0 (0)	0	0
|registerFile:RF|	274 (274)	192 (192)	0	0
|fakeRom:IM|	21 (21)	5 (5)	0	0
|ram:DM|	47 (0)	64 (0)	8192	0
|altsyncram:altsyncram_component|	47 (0)	64 (0)	8192	0
|altsyncram_96q1:auto_generated|	47 (0)	64 (0)	8192	0
|altsyncram_sjg2:altsyncram1|	0 (0)	0 (0)	8192	0
|sld_mod_ram_rom:mgl_prim2|	47 (30)	64 (55)	0	0
|sld_rom_sr:\ram_rom_logic_gen:name_gen:info_rom_sr|	17 (17)	9 (9)	0	0