ECSE 325 Lab Final Report

Group 06

Adrian Wang 260769387
Irene Ma 260776283
Xiangyun Wang 260771591

1. Circuit function description

In this lab, the SHA 256 Hash code generating system is completed based on the work of previous labs. Each hash code output needs a 512-bit block input and 64 clock cycles. At the beginning the hash core logic is modified so that ***Kt_i*** becomes an input that changes following a snippet, and ***Wt_i*** becomes the input called message schedule. The message schedule logic mixes up and transforms the words in the 512-bit block input, so that the hashing process becomes very hard to be reverse engineered. This component of the circuit is created first and named ***g06_MS.vhd*** in the code.

The input and output listing of the component is:

Input	**Output**
M_i	***Wt_i***
ld_i	
CLK	

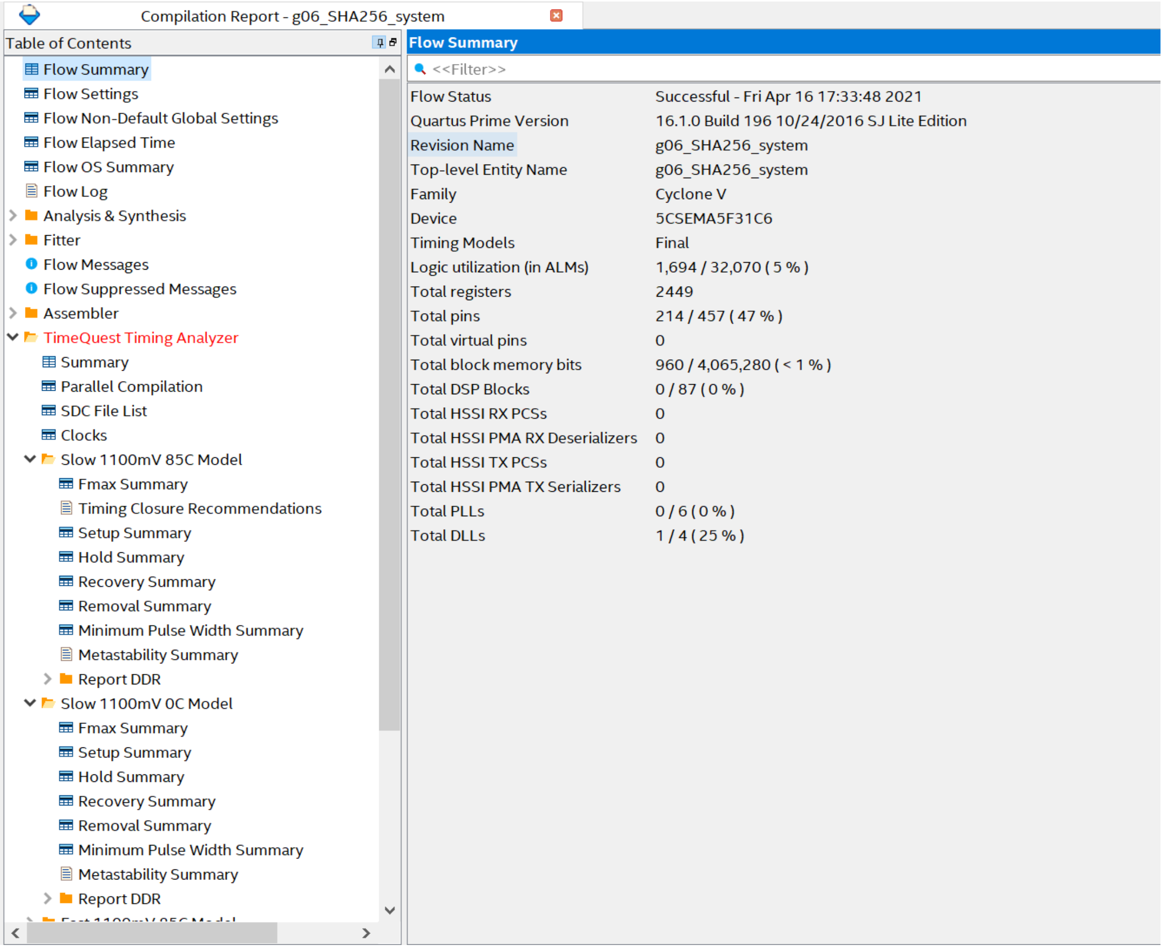
and then the component is connected to the ***Hash_Core*** so that both ***Wt_i*** and ***Kt_i*** change during the Hashing rounds. Finally the ***SHA_256*** logic is modified with initial hash values that will be updated with ***A_o*** to ***H_o*** signals every clock cycle.

In the second part, the overall system is defined with Qsys following the instructions, and the Avalon port is connected and configured with the g06_SHA256_custom_component.vhd. After the system is set up correctly, the full compilation of the system is implemented and takes approximately 7-10 minutes. The flow summary, timing analysis, and functional simulation are shown in pictures below. From the flow summary we can see that because of the total pin limit, only two blocks could be implemented onto this FPGA board.

2. Combinational block implementation

2.1. Flow Summary

As shown in the figure, the circuit logic utilization is 1694 in ALMs (5%).



Flow Status	Successful - Fri Apr 16 17:33:48 2021
Quartus Prime Version	16.1.0 Build 196 10/24/2016 SJ Lite Edition
Revision Name	g06_SHA256_system
Top-level Entity Name	g06_SHA256_system
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	1,694 / 32,070 (5 %)
Total registers	2449
Total pins	214 / 457 (47 %)
Total virtual pins	0
Total block memory bits	960 / 4,065,280 (< 1 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	1 / 4 (25 %)

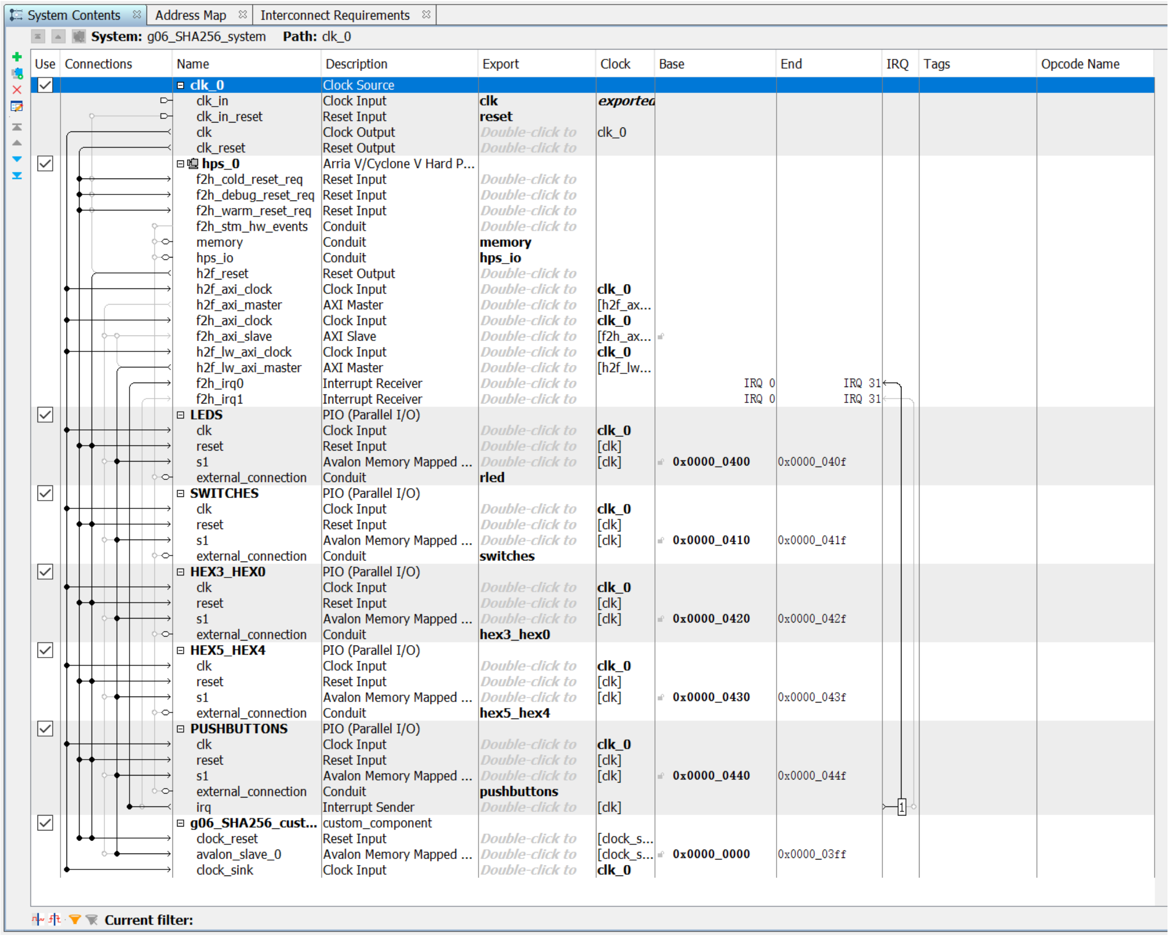
2.2. Timing Analysis

For the main clock in the top level system, the maximum frequency is at 99 MHz, which perfectly falls into the order of 50-150 MHz.

	Clock name	Fmax	Restricted Fmax	Setup slack
Slow 0C Setup	clk_clk	99.04 MHz	99.04 MHz	39.903
Slow 85C Setup	clk_clk	99.58 MHz	99.58 MHz	39.958

The original screenshots are attached to the end of the report (APPENDIX A)

2.3. Final Qsys system contents pane



System Contents | Address Map | Interconnect Requirements

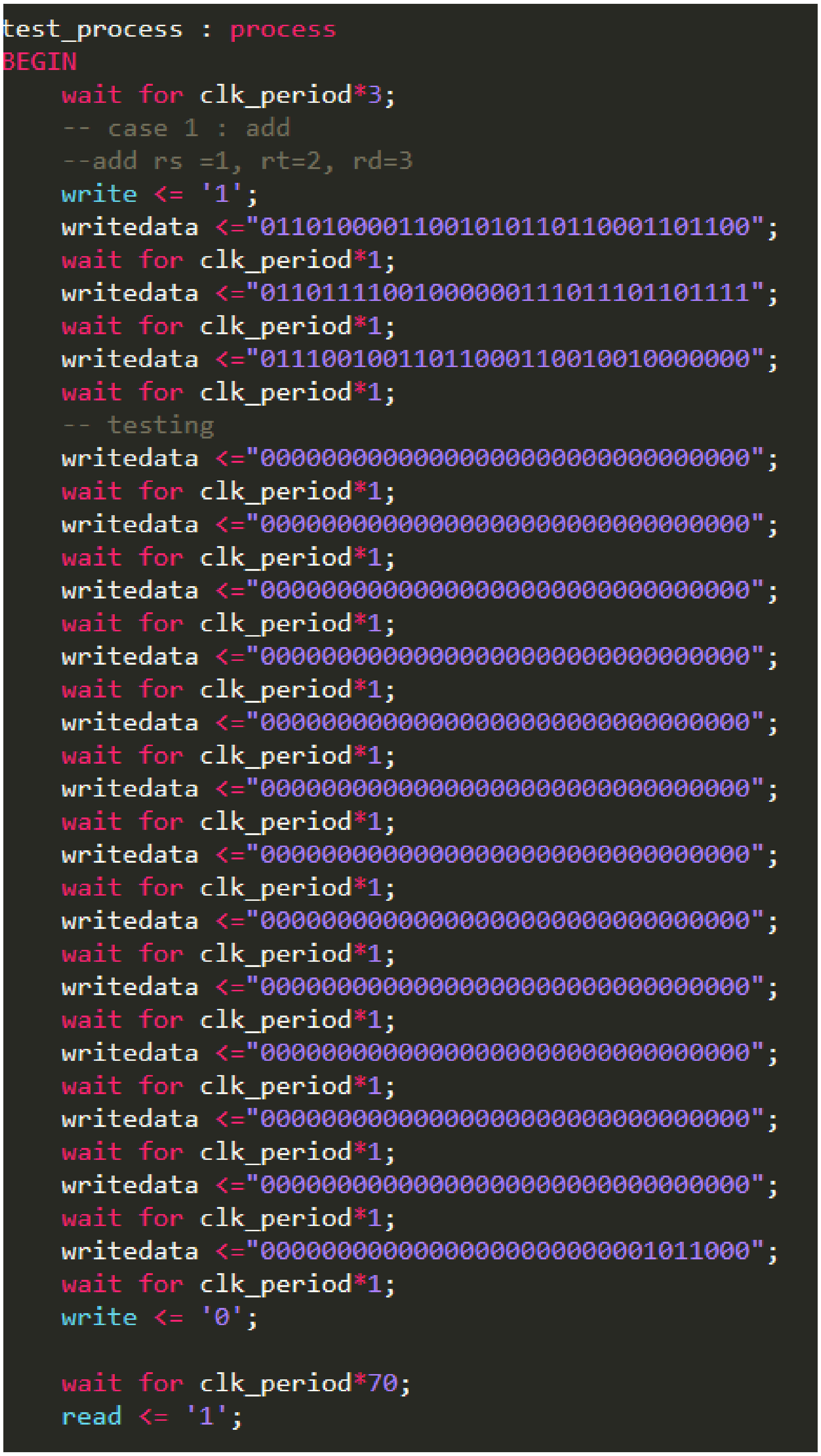
System: g06_SHA256_system **Path:** clk_0

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
☑		⊟ **clk_0**	Clock Source							
		clk_in	Clock Input	**clk**	*exported*					
		clk_in_reset	Reset Input	**reset**						
		clk	Clock Output	*Double-click to*	clk_0					
		clk_reset	Reset Output	*Double-click to*						
☑		⊟ **hps_0**	Arria V/Cyclone V Hard P...							
		f2h_cold_reset_req	Reset Input	*Double-click to*						
		f2h_debug_reset_req	Reset Input	*Double-click to*						
		f2h_warm_reset_req	Reset Input	*Double-click to*						
		f2h_stm_hw_events	Conduit	*Double-click to*						
		memory	Conduit	**memory**						
		hps_io	Conduit	**hps_io**						
		h2f_reset	Reset Output	*Double-click to*						
		h2f_axi_clock	Clock Input	*Double-click to*	**clk_0**					
		h2f_axi_master	AXI Master	*Double-click to*	[h2f_ax...					
		f2h_axi_clock	Clock Input	*Double-click to*	**clk_0**					
		f2h_axi_slave	AXI Slave	*Double-click to*	[f2h_ax...					
		h2f_lw_axi_clock	Clock Input	*Double-click to*	**clk_0**					
		h2f_lw_axi_master	AXI Master	*Double-click to*	[h2f_lw...					
		f2h_irq0	Interrupt Receiver	*Double-click to*		IRQ 0	IRQ 31			
		f2h_irq1	Interrupt Receiver	*Double-click to*		IRQ 0	IRQ 31			
☑		⊟ **LEDS**	PIO (Parallel I/O)							
		clk	Clock Input	*Double-click to*	**clk_0**					
		reset	Reset Input	*Double-click to*	[clk]					
		s1	Avalon Memory Mapped ...	*Double-click to*	[clk]	**0x0000_0400**	0x0000_040f			
		external_connection	Conduit	**rled**						
☑		⊟ **SWITCHES**	PIO (Parallel I/O)							
		clk	Clock Input	*Double-click to*	**clk_0**					
		reset	Reset Input	*Double-click to*	[clk]					
		s1	Avalon Memory Mapped ...	*Double-click to*	[clk]	**0x0000_0410**	0x0000_041f			
		external_connection	Conduit	**switches**						
☑		⊟ **HEX3_HEX0**	PIO (Parallel I/O)							
		clk	Clock Input	*Double-click to*	**clk_0**					
		reset	Reset Input	*Double-click to*	[clk]					
		s1	Avalon Memory Mapped ...	*Double-click to*	[clk]	**0x0000_0420**	0x0000_042f			
		external_connection	Conduit	**hex3_hex0**						
☑		⊟ **HEX5_HEX4**	PIO (Parallel I/O)							
		clk	Clock Input	*Double-click to*	**clk_0**					
		reset	Reset Input	*Double-click to*	[clk]					
		s1	Avalon Memory Mapped ...	*Double-click to*	[clk]	**0x0000_0430**	0x0000_043f			
		external_connection	Conduit	**hex5_hex4**						
☑		⊟ **PUSHBUTTONS**	PIO (Parallel I/O)							
		clk	Clock Input	*Double-click to*	**clk_0**					
		reset	Reset Input	*Double-click to*	[clk]					
		s1	Avalon Memory Mapped ...	*Double-click to*	[clk]	**0x0000_0440**	0x0000_044f			
		external_connection	Conduit	**pushbuttons**						
		irq	Interrupt Sender	*Double-click to*	[clk]			1		
☑		⊟ **g06_SHA256_cust...**	custom_component							
		clock_reset	Reset Input	*Double-click to*	[clock_s...					
		avalon_slave_0	Avalon Memory Mapped ...	*Double-click to*	[clock_s...	**0x0000_0000**	0x0000_03ff			
		clock_sink	Clock Input	*Double-click to*	**clk_0**					

Current filter:

2.4. Functional Simulation

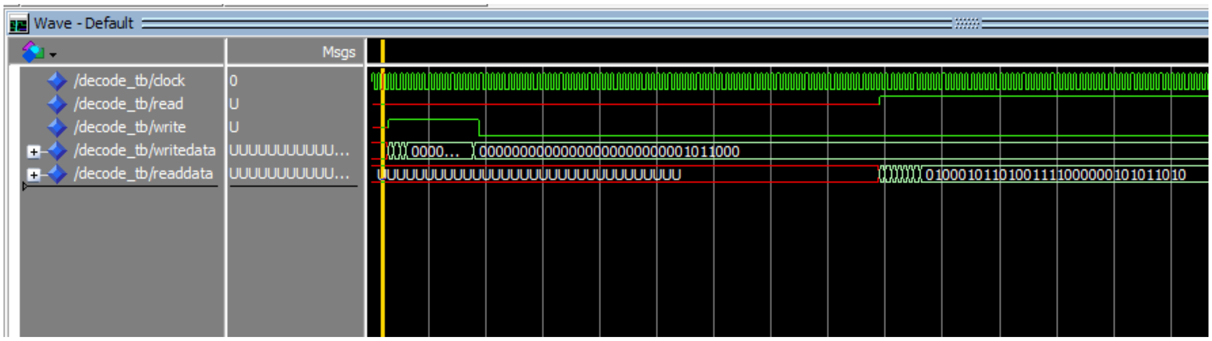
As shown in the figure below, the "Hello World" encoded binary is pipelined written into the circuit by 32 bits and after 64 clock cycles the result is outputted to the port named "readdata". To ensure the signal has enough space to output, the interval to wait in the testbench is slightly larger than 64 bits.

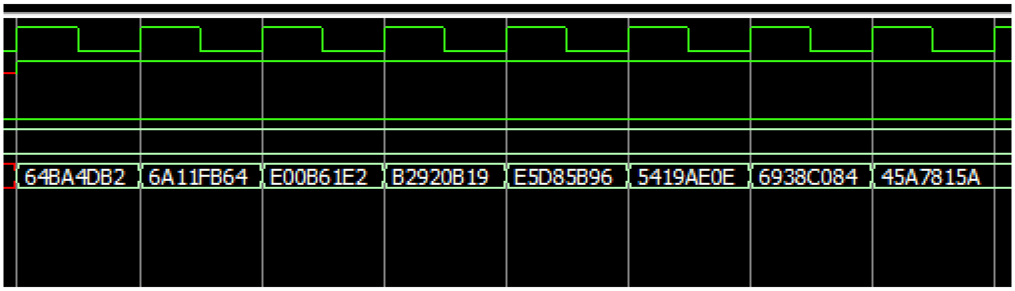


```
test_process : process
BEGIN
    wait for clk_period*3;
    -- case 1 : add
    --add rs =1, rt=2, rd=3
    write <= '1';
    writedata <="01101000011001010110110001101100";
    wait for clk_period*1;
    writedata <="01101111001000000111011101101111";
    wait for clk_period*1;
    writedata <="01110010011011000110010010000000";
    wait for clk_period*1;
    -- testing
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000000000000";
    wait for clk_period*1;
    writedata <="00000000000000000000000001011000";
    wait for clk_period*1;
    write <= '0';

    wait for clk_period*70;
    read <= '1';
```

The output is in the correct format with eight 32 bit signals. The complete hashcode in hexadecimal is: "64BA4DB26A11FB64E00B61E2B2920B19E5D85B965419AE0E6938C08445A7815A". The result is different from the one provided in the online tutorial, but in the correct format. This could be the result of slight differences in the circuit, and due to the limited time, debugging for the issue is not worth. The algorithm has ensured that the hash code achieved is unique to each of the input and the computation core is a great representation of cryptographic operation.



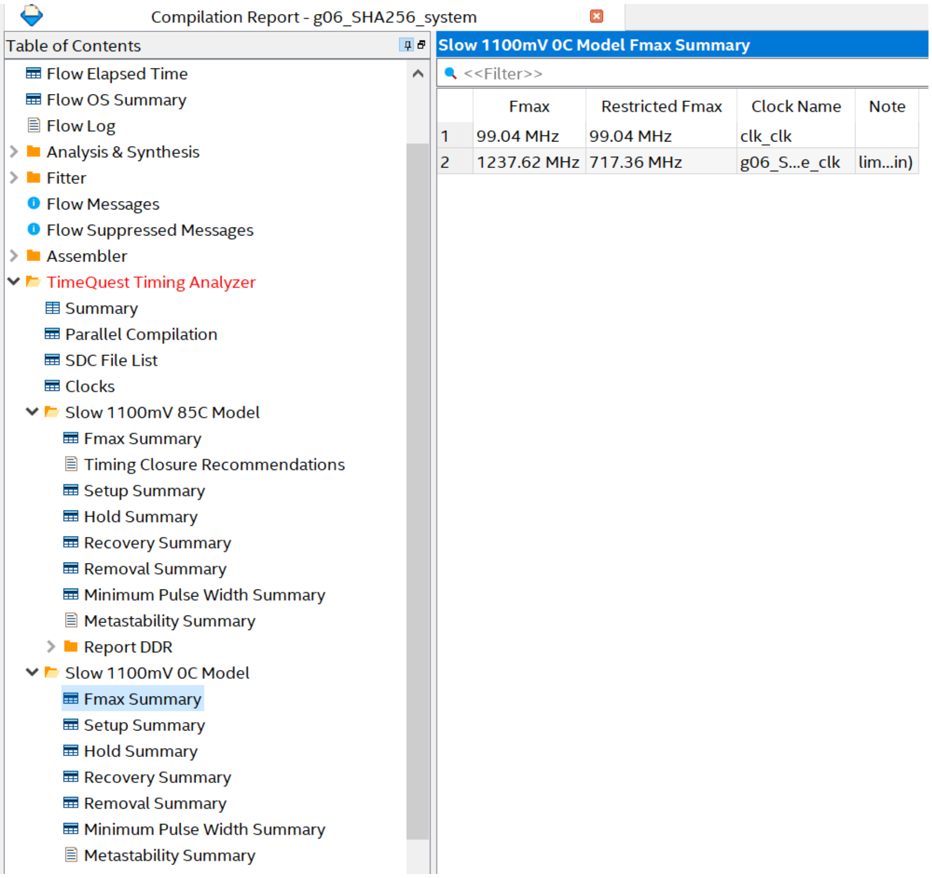


3. Discussion

The program is implemented following the lab instructions, and tiny mistakes in the given code are solved. For example, on page 14 of the lab instruction, the component port list has an address port missing, and in the SHA_256 code provided in the lab 3, the H_o port was not connected, which resulted in an output of 7*32 bits at the beginning. Once these are detected and solved, the program runs as expected in timing analysis as well as in output form. If more time is available, the program would be reviewed to try to output the same hash code as provided.

APPENDIX A

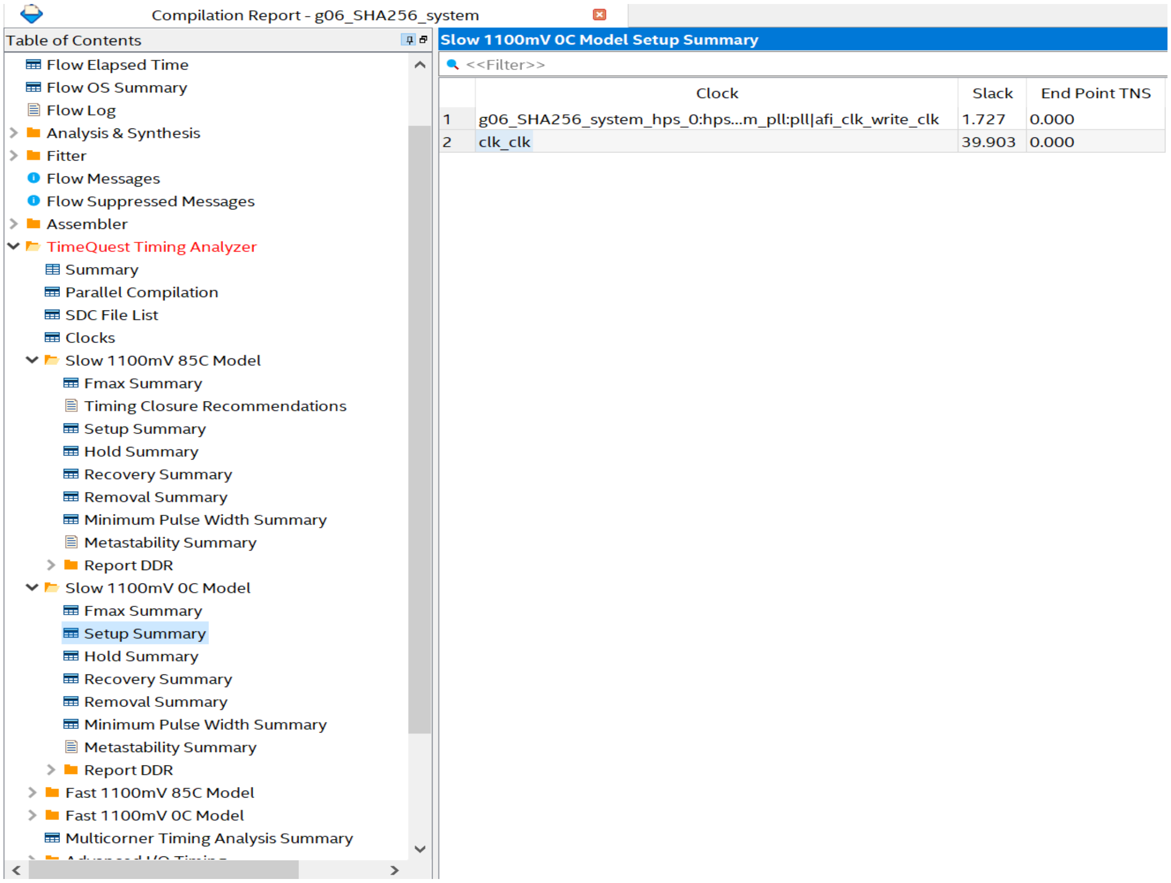
0C Fmax



Slow 1100mV 0C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	99.04 MHz	99.04 MHz	clk_clk	
2	1237.62 MHz	717.36 MHz	g06_S...e_clk	lim...in)

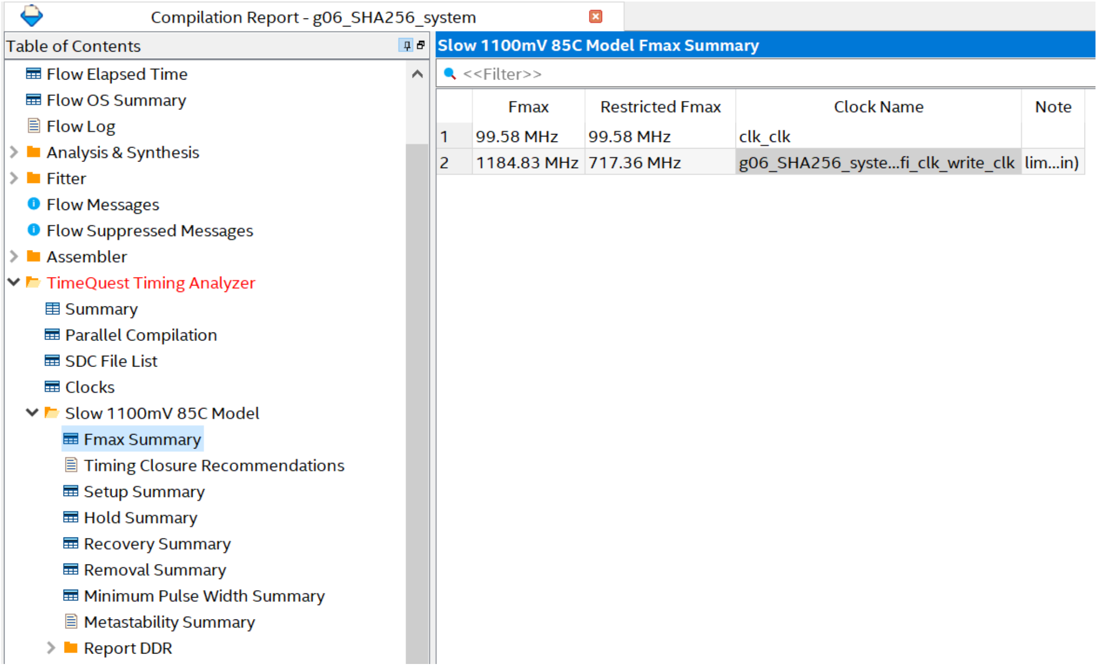
0C Setup



Slow 1100mV 0C Model Setup Summary

	Clock	Slack	End Point TNS
1	g06_SHA256_system_hps_0:hps...m_pll:pll|afi_clk_write_clk	1.727	0.000
2	clk_clk	39.903	0.000

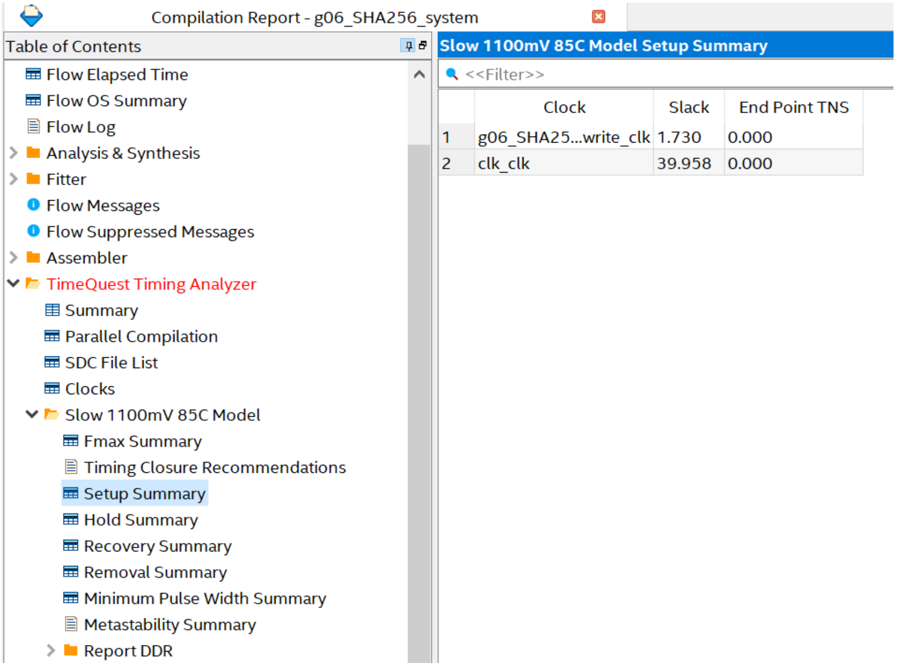
85C Fmax



Slow 1100mV 85C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	99.58 MHz	99.58 MHz	clk_clk	
2	1184.83 MHz	717.36 MHz	g06_SHA256_syste...fi_clk_write_clk	lim...in)

85C Setup



Compilation Report - g06_SHA256_system

Table of Contents

- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Flow Messages
- Flow Suppressed Messages
- Assembler
- TimeQuest Timing Analyzer
  - Summary
  - Parallel Compilation
  - SDC File List
  - Clocks
  - Slow 1100mV 85C Model
    - Fmax Summary
    - Timing Closure Recommendations
    - Setup Summary
    - Hold Summary
    - Recovery Summary
    - Removal Summary
    - Minimum Pulse Width Summary
    - Metastability Summary
    - Report DDR

Slow 1100mV 85C Model Setup Summary

<<Filter>>

	Clock	Slack	End Point TNS
1	g06_SHA25...write_clk	1.730	0.000
2	clk_clk	39.958	0.000