Due Date: Sunday November 10, 11:59pm– Submit this editable **Word** document NOT pdf

0 / 28

EEDG/CE 6370
Design and Analysis of Reconfigurable Systems
Homework 7
High-Level Synthesis Optimizations

Name: Prathamesh Sanjay Gadad

Email: psg220003@utdallas.edu

PART 1 – sobel.c Synthesis and Verification – Use part1 source sobel.zip

a.) Synthesize the sobel.c design. Report from the QoR file the size of the circuit in terms of number of LUTs and registers. Report also the latency of the synthesized circuit and the critical path and maximum frequency from CWB. Include screenshots from CWB here.

Marks
4

Module Name	Basic Library Name
sobel	CWBSTDBLIB

FPGA Family	FPGA Device	FPGA Package	FPGA Speed
cycloneV	-	-	-

Resource Utilization			
ALUTs [1]	Registers	Block Memory Bits	DSPs
175	32	0	0

Latency Index	Clock Period	Net	Port	
1	20ns	204	34	
Total States	Critical Path Delay	Pin Pair	In	Out
1	6.289ns	344	26	8

Resource Utilization

Module Name	Count	ALUTs [1]	Registers	Block Memory Bits	DSPs
Total	-	175	32	0	0

Maximum Frequency = 1/6.289ns = 159 MHz

Delay

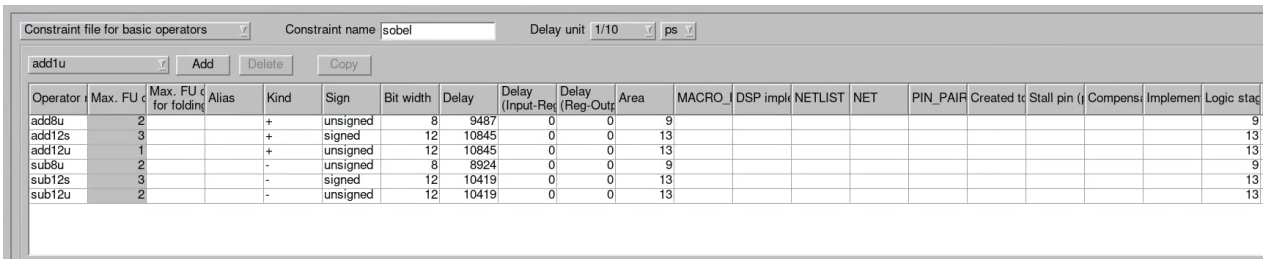
Clock Period	Critical Path Delay
20ns	6.289ns

TOTAL (ns)	IN	FU	MUX	DEC	MISC	MEM
6.29	0.00 (0%)	3.55 (56%)	2.17 (34%)	0.00 (0%)	0.57 (9%)	0.00 (0%)

Name	Pin	Signal	Delay (ns)	Arrival Time (ns)	Logic Stage	Type
input_row_a01	o1	-	-	0.00	0	-
sub12u_10@1	o1	sub12u_101ot	0.93	0.93	10	-
add12s_11_11_1@1	o1	add12s_11_11_11ot	0.43	1.36	14	-
add12s_11@1	o1	add12s_111ot	0.52	1.88	19	-
sub12s_11_10@1	o1	sub12s_11_101ot	0.73	2.61	27	-
_NMUX_308	o1	sumX1_t1	0.85	3.45	28	-
_ROR_1469	o1	C_02	0.57	4.02	29	-
_NOT_1473	o1	-	0.00	4.02	29	-
_NMUX_310	o1	add8u1i1	0.66	4.68	30	-
add8u@1	o1	add8u1ot	0.95	5.63	39	-
_NOT_1460	o1	-	0.00	5.63	39	-
_NMUX_312	o1	SUM2_t	0.66	6.29	40	-
_NOT_1375	o1	-	0.00	6.29	40	-
sobel_ret_r	din	-	-	6.29	40	-

b.) Annotate from Resource constraint file (FCNT) the number and type of Functional Units (FUs) needed to fully parallelize the description (include screenshots from the reports). Explain why these FUs are needed and why no multipliers are needed

Marks
4



Operator	Max. FU	Max. FU for folding	Alias	Kind	Sign	Bit width	Delay	Delay (Input-Reg	Delay (Reg-Outp	Area	MACRO_	DSP imple	NETLIST	NET	PIN_PAIR	Created to	Stall pin (	Compens	Implemen	Logic stag
add8u	2			+	unsigned	8	9487	0	0	9										9
add12s	3			+	signed	12	10845	0	0	13										13
add12u	1			+	unsigned	12	10845	0	0	13										13
sub8u	2			-	unsigned	8	8924	0	0	9										9
sub12s	3			-	signed	12	10419	0	0	13										13
sub12u	2			-	unsigned	12	10419	0	0	13										13

These functional units are needed for different addition and subtraction operations with various bit widths, signs, and types in the Sobel filter, which finds edges by calculating image gradients. The filter uses convolution with a small 3x3 matrix, and instead of actual multipliers, we use adders to repeat additions for multiplication. So, only adders and subtractors are required.

c.) Perform Logic synthesis using Quartus Prime and compare the area results in terms of ALMS/ALUTs. You might call Quartus from within CWB as shown in the lab sheet or manually create a project in Quartus and include the RTL file generated by CWB, and an SDC file. Discuss if they match or not and why they do/don't. Add screenshot from Quartus here. Fill out the table given below.

Marks
6

	CWB	Quartus
ALMS/ALUTs	175	127
Registers	32	32
IOS	34	34
DSP Macros	0	0

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	71
2		
3	Combinational ALUT usage for logic	127
1	-- 7 input functions	0
2	-- 6 input functions	8
3	-- 5 input functions	9
4	-- 4 input functions	0
5	-- <=3 input functions	110
4		
5	Dedicated logic registers	32
6		
7	I/O pins	34
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	CLOCK~input
12	Maximum fan-out	32
13	Total fan-out	527
14	Average fan-out	2.32

The difference in ALMs/ALUTs between CWB and Quartus is due to Quartus's optimization techniques, which reduce logic usage. CWB relies on a pre-characterized technology library with specific functional units (FUs) that match the generated RTL, then sums up all the RTL components. In contrast, Quartus converts the RTL into a gate netlist and maps it to a lookup table. The register, I/O pin, and DSP macro

counts remain the same, as these are core design requirements. This indicates that Quartus Prime's synthesis is more resource-efficient, improving performance and lowering power consumption.

sobel.c Verification

a.) Perform a pure SW simulation, a cycle-accurate simulation and an RTL simulation using the untimed test vectors used for the software simulation and make sure that the simulation outputs match for the two versions with HLS. Show the result (paste console window). Past also the pure SW simulation results (.bmp file).

Marks
6

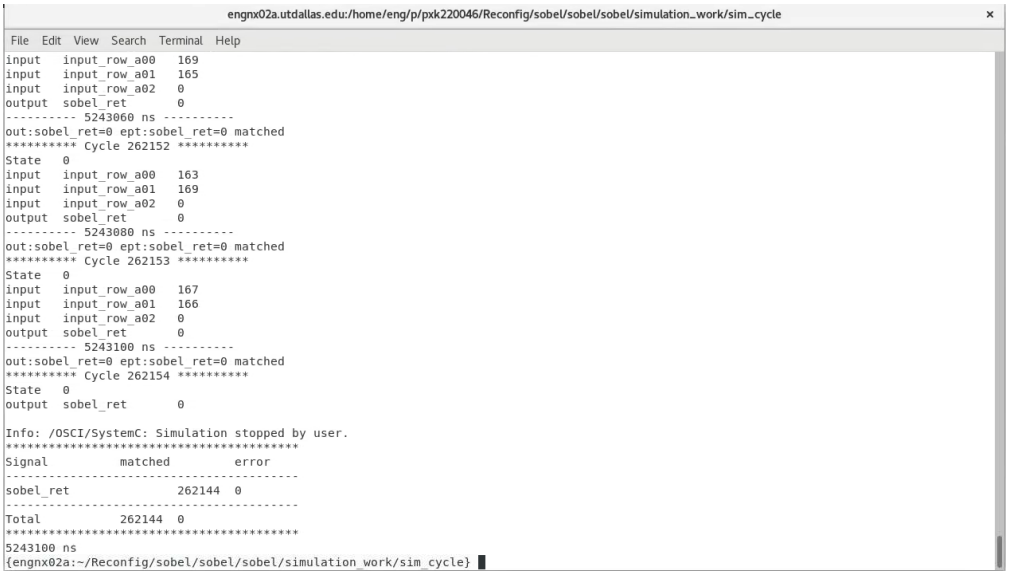
Cycle Accurate Simulation –

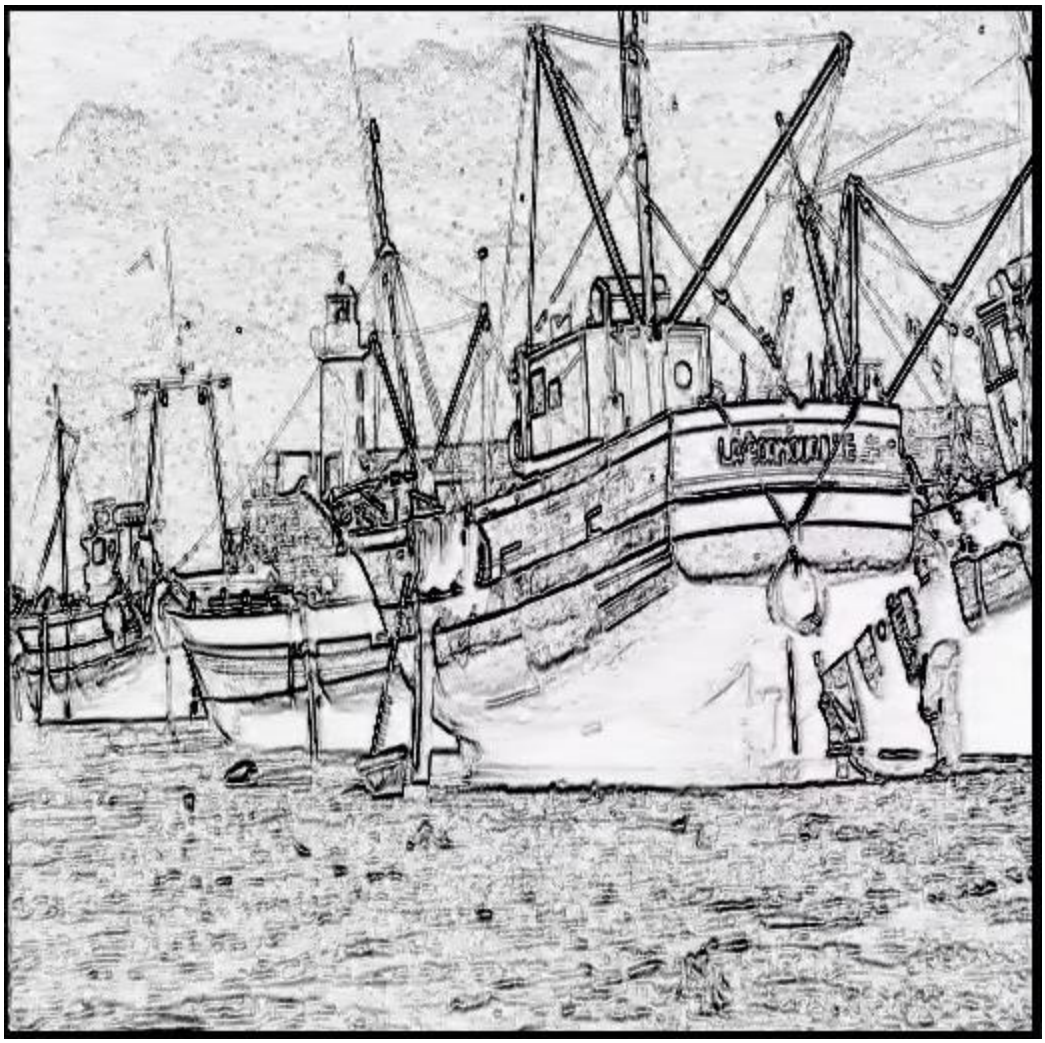
```
***************
Signal          matched         error
-----------------------------------------
sobel_ret       262144     0
-----------------------------------------
Total      262144     0
***************
5243100 ns
>>>>>>>>>> END Simulation [sim_cycle.exe]
>>>>>>>>>> START Vcdconv [vcdconv]
2023/10/30 22:47:10 Execute:
     vcdconv -i sobel_test.vcd
>>>>>>>>>> END Vcdconv [vcdconv]
```

RTL Simulation –

```
#  Simulation Summary
#  ------------------------------------------------------------------------
----------
#  Comparison result(s):
#
#   Signal Name       compare    matched      error  x not matched  z
not matched
#  ------------------------------------------------------------------------
----------
#   sobel_ret          262144     262144          0              0
0
#  ------------------------------------------------------------------------
----------
```

```
#    Total                262144      262144           0                0
0
#   ------------------------------------------------------------------------
----------
#
#   Simulation time :
#    Elapsed time :           5243110000 (x1ps)
#
#   Warnings :
#   *************************
#
# ** Note: Data structure takes 25330240 bytes of memory
#          Process time 13.81 seconds
#          $finish    : sobel_E_tb.v(40)
#    Time: 5243110 ns  Iteration: 0  Instance: /T_sobel_00
# End time: 00:24:13 on Oct 31,2023, Elapsed time: 0:09:24
# Errors: 0, Warnings: 0
>>>>>>>>>> END Simulation [vr_vsim.bat]
```





PART 2 sobel.c Design Space Exploration – Use part2 source sobel explorationl.zip

a.) Open source_sobel_exploration folder and you will find three files:

- Sobel.c
- attrs.h
- lib_sobel.info
- cycloneV.FLIB cycloneV.BLIB

Sobel.c: This new sobel file version has synthesis directives specified as comments in the code ranging from ATTR1 to ATTR2. The actual attributes are specified in attrs.h

Attrs.h: A sample file with the synthesis directives that CWB needs to synthesize the new sobel are given here. E.g.:

#define ATTR1 Cyber array=REG

Substitutes the ATTR1 in the comment in sobel.c by Cyber array=REG

lib_sobel.info: The library with all of the possible synthesis directives for each of the individual operations

cycloneV.FLIB/.BLIB: CWB technology libraries

Create a script that reads in the lib_sobel.info file and creates for each possible attribute combination a unique attrs.h. Every new combination has to be parsed (cpars), and synthesized (bdltran) as follows:

Step 1: Read lib_sobel.info with all attributes for each operation.

Step2: Generate new attrs.h header file with unique attributes combination.

Step3: Parse the new description. Make sure attrs.h is in the same folder as sobel.c:
%cpars sobel.c

Step4: Synthesize design calling HLS (bdltran)
%bdltran -c2000 -s sobel.IFF -lfl cycloneV.FLIB -lb cycloneV.BLIB

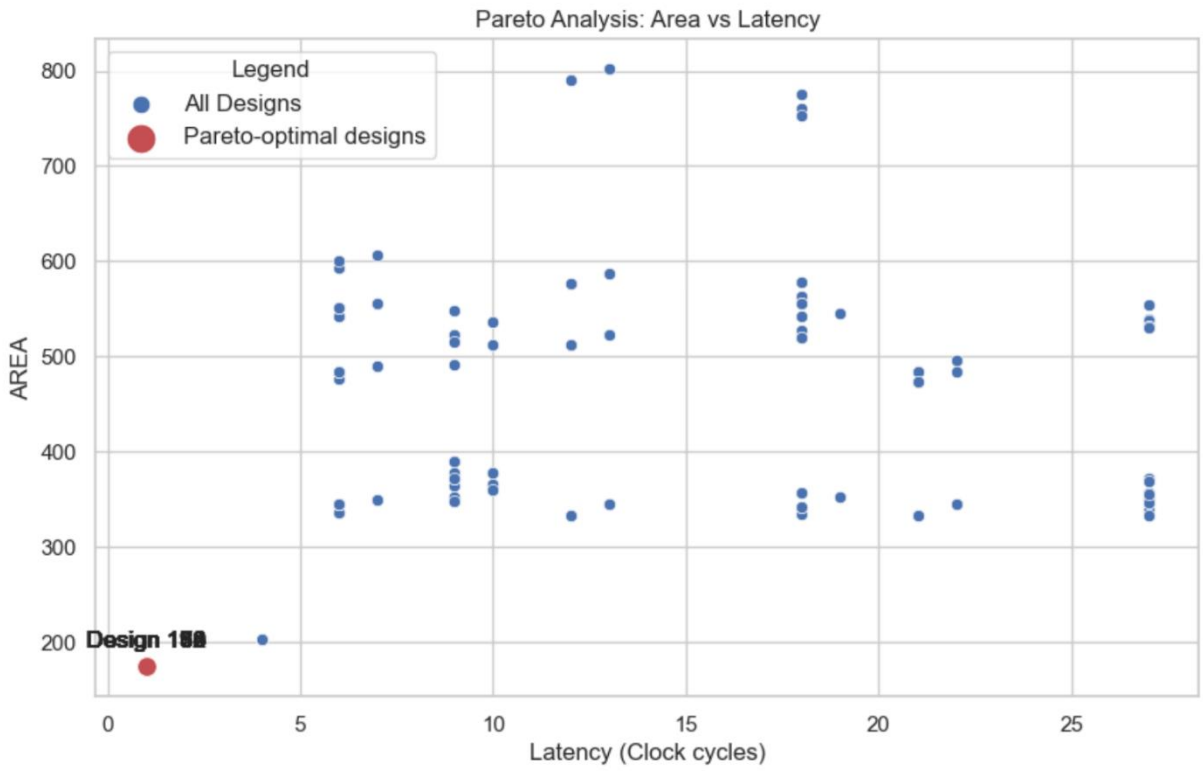
Step5: Read the area and latency of the new design and store sobel.QOR file or sobel.csv file and attrs.h file by moving it to either another or renaming the files.

Repeat step2 until all combinations are generated.

Step6: Generate report file with all the results and report optimal designs (area, latency and pragmas that lead to them).

Plot the graph of **area vs. latency** of all of the designs generated (y-axis=-area, x-axis=latency). Report the pragmas that lead to the Pareto-optimal designs. Discuss the results. Are they what you expected? (Yes/No). Create a short YouTube video showing how your explorer works.

Marks
8



The graph shows that the Pareto-optimal designs (in red) provide the best balance between area and latency, with no other designs being better in both. Lower latency designs usually need more area, while higher latency designs use less, which matches expected hardware trade-offs. These optimal configurations meet

typical expectations, offering efficient options for either faster performance (low latency) or saving area (higher latency).

YouTube link: https://youtu.be/Yz7e_Eootog