Logo

Description automatically generated**EEDG/CE 6370**

0

22

**Design and Analysis of Reconfigurable Systems**

**Homework 7**

**High-Level Synthesis Optimizations**

**Students Name: Khoa Diep, Dismas Ezechukwu**

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

1. Synthesize the sobel.c design. 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. Include screenshots from CWB here.

|  |
| --- |
| Marks |
| 4 |
|  |

A screenshot of a computer

Description automatically generated

A screenshot of a table

Description automatically generated

The multipliers are extremely expensive in terms of area and time, thus the synthesizer avoids these Fus.

These FUs (adders and subtracters) are needed to perform the circuit.

1. 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.

|  |
| --- |
| Marks |
| 2 |
|  |

A screenshot of a computer program

Description automatically generated

A blue and white sign

Description automatically generated

ALUTs: 175

Registers: 32

Critical Path Delay: 6.289 ns

Fmax: 159.007 Mhz

Latency: 1 clock cycle

1. Perform Logic synthesis using Quartus Prime and compare the area results in terms of LUTs. 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.

|  |
| --- |
| Marks |
| 4 |
|  |

A white background with black text

Description automatically generated

ALUTs: 128

There are less ALUTs from Quartus compared to CWB and do not match. This is due to CWB using the generated library for synthesis to create the circuit, versus the Quartus Prime physically routes the circuit into the board, which is more accurate.

**sobel.c Verification**

1. Perform 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).

|  |
| --- |
| Marks |
| 4 |
|  |

Cycle-Accurate Simulation:

A screenshot of a computer program

Description automatically generated

NOTE: Added a random test vector to sodel\_ret.clv

**A screenshot of a computer

Description automatically generated**

Zoomed in to random spot of the wave.

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

1. 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 -lf 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)

|  |
| --- |
| Marks |
| 8 |
|  |

A screen shot of a graph

Description automatically generated

Pragmas:

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=REG

ATTR3 Cyberarray=REG

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=REG

ATTR3 Cyberarray=EXPAND,array\_index=const

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=REG

ATTR3 Cyberarray=ROM

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2Cyberarray=REG

ATTR3 Cyberarray=LOGIC

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=EXPAND,array\_index=const

ATTR3 Cyberarray=REG

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=EXPAND,array\_index=const

ATTR3 Cyberarray=EXPAND,array\_index=const

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=EXPAND,array\_index=const

ATTR3 Cyberarray=ROM

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=EXPAND,array\_index=const

ATTR3 Cyberarray=LOGIC

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=ROM

ATTR3 Cyberarray=REG

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=ROM

ATTR3 Cyberarray=EXPAND,array\_index=const

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=ROM

ATTR3 Cyberarray=ROM

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=ROM

ATTR3 Cyberarray=LOGIC

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=LOGIC

ATTR3 Cyberarray=REG

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=LOGIC

ATTR3 Cyberarray=EXPAND,array\_index=const

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=LOGIC

ATTR3 Cyberarray=ROM

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

ATTR1 Cyberarray=EXPAND,array\_index=const

ATTR2 Cyberarray=LOGIC

ATTR3 Cyberarray=LOGIC

ATTR4 Cyberunroll\_times=all

ATTR5 Cyberunroll\_times=all

The Pareto-optimal designs are when Area (LUTS) = 175 and Latency (clock cycles) = 1. There are some designs close to these designs (with Area = 176 and Latency = 1). Our expectation was that there would be more designs with tradeoffs between area and latency (i.e., some designs with Area = 175 and Latency =1 and some designs with Area =100 and Latency = 2). However, there are several designs with the same optimum Area and same optimum Latency. This means that there are no tradeoffs in choosing between designs (the most optimum design is **always** with an area of 175 LUTs and latency of 1 clock cycle) which is contrary to our expectations.