**ECE 6213 – Design of VLSI Circuits**

**Fall 2022 – The George Washington University – Dr. Jerry Wu**

**Osama Yousuf – HW 6**

**Problem 1 – 20 pts**

**Determine whether a synthesis tool detects and removes equivalent states from the sequential machine described by the STG in Fig P6-18. (you may use synthesis tools; in your DC script, find out the way to turn-off optimization; hint: set\_max\_area 1 or 0); you can find these two options in the documentation. Or, you can try them in the tools. (See my tutorial for more detail)**

Diagram, engineering drawing

Description automatically generated

When a STG is submitted to a synthesis tool, it will attempt to detect and delete equivalent states that are present. The tool then creates a sequential machine that is equivalent to the original STG without the now deleted equivalent states.

To test with DC, I first designed a simple Moore State Machine in Verilog for the STG (State Transition Graph) described above.

**Code:** stg\_moore\_fsm.v

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

After reviewing DC documentation for area optimization, I found that the “set\_max\_area” command can be used to control the degree of optimization that is done on the design during the synthesis process.

* If the set\_max\_area is not defined, DC does minimal area optimization. This is useful if the area is not important since it reduces the compile time.
* If “set\_max\_area 0” is defined, DC attempts to reduce the area as much as possible without increasing any timing violation. In this case, I found that equivalent states of the STG were indeed getting detected and removed.

The area synthesis report with “set\_max\_area 0”, and the corresponding constraint file “stg\_optimized.tcl” is provided below:

**Area Report (optimized)**

Table

Description automatically generated

**Synthesized Schematic (optimized)**

A screenshot of a computer

Description automatically generated with medium confidence

**Constraint File (optimized)**

Graphical user interface, text, application, email

Description automatically generated

The area synthesis report for the unoptimized case, the schematic, and the corresponding constraint file “stg\_unoptimized.tcl” is provided below:

**Area Report (unoptimized)**

Table

Description automatically generated

**Synthesized Schematic (unoptimized)**

Graphical user interface

Description automatically generated

**Constraint File (unoptimized)**

Graphical user interface, text, application, email

Description automatically generated

As can be seen by comparing the two area reports and schematics, DC has aggressively detected and gotten rid of nets and combinational cells. In the optimized case, we see a total of only 20 combinational cells compared to the 41 combinational cells in the optimized case. This is evident in the schematic as well, as the unoptimized schematic is significantly larger, and in the final cell areas as well (10350 units in the optimized case, against the 14085 units with minimal area optimization).

Synthesis reports are provided separately as part of the submitted files in appropriate folder inside the “p1” directly, along with Verilog scripts.

**Problem 2 – 20 pts**

**Modify the STA script example and run the STA for your FIFO design. Make sure you included your discussion based on what we teach in the constraints and STA class.**

Firstly, I had to edit my FIFO code as the DC tool was giving synthesis errors due to a lot of my assignments leading to the “double-drive” error. I restructured my code to make it synthesizable, ensuring that the testbench behavior is maintained (from my homework 4).

All files for this problem are present in the p2 directory. The Verilog files for the FIFO design are fifo.v, the testbench is fifo\_tb.v, the constraint file is fifo.tcl, and DC output files and synthesis reports are in the Output Reports folder. A summary of these is provided below:

**Verilog code (fifo.v)**

Text

Description automatically generated

Text

Description automatically generated

**Top Level Schematic**

Text

Description automatically generated

Since I have two clocks in my design (one for reader, one for writer), I adjusted the .tcl file accordingly. Useful comments are provided explaining my choices.

**Constraint File (fifo.tcl)**

Text

Description automatically generated

Text

Description automatically generated

The .tcl commands used to constraint a certain path depended on the type of the path. To constrain:

* **All register to register paths:** I simply defined the reader and writer clocks.
* **All input to register paths:** I set the input delay for all inputs relative to both, reader and writer clocks, as shown in the constraint file above. I made sure that the list of inputs for the reader doesn’t include the writer clock pin, and the list of inputs for the writer clock doesn’t include the reader clock pin.
* **All register to output paths:** I set the output delay for all outputs relative to both, reader and writer clocks.
* **Input to Output paths:** These paths were implicitly constrained by specifying the input and output delays directly.

I wanted DC to minimize the final design’s area, so I used set\_max\_area 0 as a constraint as well.

**Timing Reports**

A picture containing text

Description automatically generated

Graphical user interface, text

Description automatically generated

As can be seen, under the given reader, writer clock parameters, my FIFO code does not pass setup timing requirements. There is a violated slack of -22.06 units for the writer path, and -13.03 for the reader path.

There would be a few ways to address these violations:

* Lower the clock frequencies such that:

is satisfied (for both reader and writer)

* Optimize the FIFO code to be faster
* Use a potentially different synthesis library (vendor-specific)
* Minimize clock non-idealities (uncertainty, latency, etc.)
* Manually edit the cell and gate file to see if there’s room for further optimization in the layout generated by the DC.

**Area Report**

Text

Description automatically generated

**Constraint Report**

Text

Description automatically generated

**Problem 3 – 5 points**

1. **Latch vs. DFF**

* A DFF is an edge-triggered type of memory circuit while the latch is a level-triggered type of memory circuit.
* The output of a latch changes whenever the input changes, whereas the output of a DFF additionally depends on a clock signal.
* A latch can be designed using logic gates. A DFF on the other hand can be designed using a clock and latches.

1. **Determine if the following statements generate latches, if it does, how can you fix it.**

Text, letter

Description automatically generated

1. No latch generates here, since the value of y is assigned in all cases on a.
2. Two latches can generate here. Assuming both x and y have been set once, then if a is not 3, then the previous value of x will be latched on. Similarly, if a is 3, then the previous value of y will be latched on. We can fix this by adding an assignment to y in the if block, and x in the else block.
3. Two latches generate here in the case when a does not change since we do not have an else block. We can fix this by adding an else block with assignments to both x and y.
4. No latch generates here since the value of both x and y are assigned in all cases.
5. No latches will generate here. If the if block executes, x = 0 and y = 1, otherwise x = 1 and y = 0. The previous value for both x and y need not be remembered.