EEL 4712C: Digital Design
Spring 2024

LAB 01: Basic RTL Components & Adders
(50 Points)

Pre-Lab Objectives

- In the first part of this lab, implement and simulate each of the following RTL components in Modelsim: 8 Bit Encoder, 8 Bit Decoder, 4x1 Multiplexer, 1x4 Demultiplexer, Register.
- When implementing the mux, demux, and register, utilize generics to make your code reusable in the synthesis of multiple different components. Creating good designs here will reduce your work in later assignments.
- For the second part of this lab, implement a generic ripple carry adder and a generic carry look ahead adder. You should use generate statements for each of these to meet the design requirements. Once the model works in simulation, it should be tested on the FPGA.
- Your lab submission should include the .vhd files for these components along with the annotated simulations of the components. Simulations that are not annotated to show understanding and functionality of components will not receive full credit.
- If a testbench is not provided for an entity, you are expected to create your own. Any testbenches created should demonstrate functionality of your design.

Pre-Lab Requirements

Part 1: RTL Component Design & Simulation (10 points)

In the first part of this lab, you will implement, simulate, and synthesize common RTL components.

1. 8 Bit Encoder
    - Implement an encoder with 8 bit input and the appropriate size output.
2. 8 Bit Decoder
    - Implement a decoder with 8 bit output and the appropriate size input.
3. 4x1 Multiplexer
    - Implement a generic width input 4x1 multiplexer with generic width output
4. 1x4 Demultiplexer
    - Implement a generic width input 4x1 multiplexer with generic width output
5. Register
    - Implement a generic width input register with generic width output

Part 2: Ripple Carry Adder (5 points)

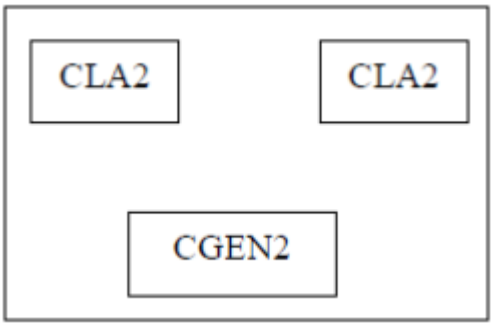
For the second part of this lab, you will implement a full adder and a generic width ripple carry adder. The ripple carry adder will be demoed on an FPGA.

1. Design a full adder architecture in VHDL. You are encouraged to use any information included in the lecture slides.
2. Using the provided entity, create a structural ripple-carry adder architecture called RIPPLE_CARRY. To do this, use a for-generate to create a chain of full adders WIDTH long.

Part 3: Carry Look Ahead Adder (15 points) + Demo (20 points)

In the final portion of this lab, you will use the formulas presented in the lecture slides to create a 4 bit carry look ahead adder. Use the provided entity declaration.

1. Review the block propagate and generate formulas from the lecture slides and consider what these reduce to in terms of the inputs x and y.
2. Design a 2 bit carry look ahead adder using a behavioral architecture.
3. Design a carry generator that calculates the carries, block generates, and block propagates from two instances of the 2-bit CLA. Then connect the three entities together in a structural architecture named CARRY_LOOKAHEAD to form a 4-bit CLA.



Verification

To test your RTL components and adders, limited testbenches have been included. In order to test the adder make sure to change the name of the architecture being tested in adder_tb.vhd to the desired architecture. These testbenches are not comprehensive and should be modified if you wish to do comprehensive tests of your design.

Demo Procedure

1. Using the provided top_level.vhd and the 7-segment display entity you created in Lab 0, connect your adder to create a synthesizable circuit.
2. In Quartus, assign pins to the inputs (switches & buttons) and outputs (LEDs) such that the correct outputs are displayed on the 7-segment display.
3. Once the final product is working, show a TA the correct outputs on the FPGA.

Deliverables & Submission Guidelines

- Summarize your work in a report. Template for lab report is provide on Canvas
- Zip your lab solutions and report in a file name <Your_UFID>.zip with a folder structure as shown below.
    - Make sure to use the file names shown below in your submission

```
Your_UFID/
├── report.pdf
├── P1/
│   ├── encoder.vhd
│   ├── encoder_sim.jpgRe
│   ├── decoder.vhd
│   ├── decoder_sim.jpg
│   ├── 4x1mux.vhd
│   ├── 4x1mux_sim.jpg
│   ├── 1x4demux.vhd
│   ├── 1x4demux_sim.jpg
│   ├── reg.vhd
│   └── register_sim.jpg
├── P2/
│   ├── fa.vhd
│   └── fa_sim.jpg
├── P3/
│   ├── adder.vhd
│   ├── adder_sim.jpg
│   ├── cla2.vhd
│   ├── cla2_sim.jpg
│   ├── cgen2.vhd
│   └── cgen2.jpg
```

Provided Entities

Adder Entity (for use with both Ripple Carry and Carry Look Ahead)

```
entity adder is
    generic (
        WIDTH : positive := 8
    );
    port (
        x, y      : in  std_logic_vector(WIDTH-1 downto 0);
        carry_in  : in  std_logic;
        s         : out std_logic_vector(WIDTH-1 downto 0);
        carry_out : out std_logic
    );
end adder;
```