ECE 474a/574a Assignment 1

Assignment Points: 100 (see rubric for details)
Instruction: Work in groups of two to five. Individual groups allowed (but not encouraged) for online students.

DUE DATE: Tuesday, September 25, 11:59PM

Overview

Create a library of parameterized datapath components and use synthesis results from that library to estimate the critical path for various circuits specified using a behavioral netlist. Finally, compare the estimated critical path to actual critical path by implementing the circuits using Verilog and synthesizing the circuit.

Part 1: Verilog implementation of Datapath Component Library

- Create parameterized Verilog implementations of the following datapath components.
- Each component should be modeled using a single Verilog module.
- Each module should include a Verilog parameter named `DATAWIDTH` that specifies the number of bits for the data inputs and outputs.
- The following provides an overview of the required components for ***all*** students:

Name	Data Inputs	Control Inputs	Data Outputs	Control Outputs	Description
REG	`d`	`Clk, Rst`	`q`		Register
ADD	`a, b`		`sum`		Adder
SUB	`a, b`		`diff`		Subtractor
MUL	`a, b`		`prod`		Multiplier
COMP	`a, b`			`gt, lt, eq`	Determines if a > b, a < b, and a == b
MUX2x1	`a, b`	`sel`	`d`		Multiplexor
SHR	`a`	`sh_amt`	`d`		Logically shifts input `sh_amt` positions to the right
SHL	`a`	`sh_amt`	`d`		Logically shifts input `sh_amt` positions to the left

- ***ECE 574a:*** The following provides an overview of the ***additional*** required components for ECE 574a students:

Name	Data Inputs	Control Inputs	Data Outputs	Control Outputs	Description
`DIV`	`a, b`		`quot`		Returns quotient of a / d.
`MOD`	`a, b`		`rem`		Returns remainder of a / d.
`INC`	`a`		`d`		Returns a + 1
`DEC`	`a`		`d`		Returns a - 1

Part 2: Latency Estimation of Datapath Component Library

- Using Xilinx Vivado, synthesize each component using different settings for the `DATAWIDTH` parameter, including sizes of 2, 8, 16, 32, and 64.
- Select the Xilinx Artix-7 FPGA (XC7A100T) as the target device.
- Determine the critical path for each component for each parameter settings by viewing the synthesis report.
- Create a text file named `DPCL_LAT.txt` that includes a table describing the resulting critical path for times using the following format:

```
ComponentName : Latency2bit Latency8bit Latency16bit Latency32bit
Latency64bit
```

Part 3: Critical Path for Behavioral Netlists

- Using the latency estimates for individual datapath components, calculate an ***estimated critical path*** for the provided behavioral netlists.
- Create a Verilog implementation for each behavioral netlist. Use parameter mapping during the component instantiation to specify the size of each datapath component.
- Synthesize the Verilog implementation for each behavioral netlist to determine the ***actual critical path*** by viewing the synthesis report.
- Create a text file named `NETLISTS_LAT.txt` that summarizes the estimated critical path and actual critical path for each of the behavioral netlists using the following format:

```
NetlistName : EstimatedCriticalPath ActualCriticalPath
```

Part 4: Annotated Behavioral Netlist Schematic

- Choose one of the behavioral netlist.
- Create a schematic showing all inputs, outputs, components, registers, and wires connecting the components. Be sure to include bit widths for all inputs and outputs from components. You may draw the schematic by hand or using Vivado's Schematic feature (recommended).
- Annotate the estimated critical path at the output of every component.
- Annotate the actual critical path within the circuit.
- Save the annotated schematic as a PDF file named `NETLIST.pdf` where NETLIST is the name of the

selected behavioral netlist.

Behavioral Netlist Specification

A behavioral netlist file provides an acyclic connection of components, where:

- All empty lines should be ignored
- All lines beginning with "//" are considered comments and should be ignored
- The netlist file can be assumed to be fully space/tab delimited, i.e. one or more space or tab characters should appear between each token that needs to be parsed, including colons.
- Circuit inputs and outputs can be declared on a line using the formats:

  ```
  input dataType inputName1, inputName2
  output dataType outputName1, outputName2
  ```

- Valid data types include `Int2`, `Int8`, `Int16`, `Int32`, and `Int64`, where the number specifies the width of the data input, output, register. All integer types are unsigned.
- All outputs are implicitly associated with a register (REG) component
- Internal registers must be explicitly declared using the format:

  ```
  register dataType regName1, regName2
  ```

- Wires (internal connections between components) must be explicitly declared using the format:

  ```
  wire dataType wireName1, wireName2
  ```

- Component instantiations are declared on a single line using the following formats for specific components. (Note: comments are provided to indicate the format for specific components defined within the technology library and will not appear within the sample behavioral netlist, nor will you see comments at the end of a line.)

  ```
  o = b               // REG
  o = a + b           // ADD
  o = a - b           // SUB
  o = a * b           // MUL
  o = a > b           // COMP (lt output)
  o = a < b           // COMP (gt output)
  o = a == b          // COMP (eq output)
  o = sel ? i1 : i0   // MUX2x1
  o = a >> sh         // SHR
  o = a << sh         // SHL
  o = a / b           // DIV
  o = a % b           // MOD
  o = a + 1           // INC
  o = a - 1           // DEC
  ```

- The width of a datapath component should be determined as the maximum number of bits for any of the data inputs for that component. Inputs with fewer bits should be padded with 0's in the most significant bits.
- All names for components, inputs, outputs, wires, registers, and instances should be unique.
- All names for inputs, outputs, wires, registers, and instances are case sensitive and can consists of any number of letters or digits

- Input, output, wires, and reg declarations should come before component instantiations.

Submission Requirements

- All assignments must be submitted as a single TAR/GZIP (.tgz) or ZIP (.zip) using the naming convention `NetID1_NetID2.tgz` or `NetID1_NetID2.zip`, where *`NetID1`* and *`NetID1`* are the UA NetIDs for each group member.

- Include a README.txt in your submission that specifies: 1) Names and NetIDs of group members; 2) Xilinx synthesis tool version, 2) target FPGA and speed grade, and 3) brief description method used to calculate critical path.
- Submission should include the following
  - README.txt
  - One .v for each datapath component implementation. The name of the files should match the name of the component.
  - One .v for each Verilog implementation of the behavioral netlists. The name of the files should match the name of the circuit.
  - NETLISTS_LAT.txt
  - DPCL_LAT.txt
  - `NETLIST.pdf` where NETLIST is the name of the selected behavioral netlist for the annotated schematic.

Note: The grading rubric is provided to help you know how you will be graded. Make sure to closely follow the instructions for file and submission report format, since your work will be graded using scripts.