FINAL PROJECT: REAL TIME DIGITAL STOPWATCH DESIGN AND FPGA IMPLEMENTATION

Project Posted: 3 April 2023 (M) Demo Due: 28 April 2023 Report Due: 1 May 2023
Points: 100

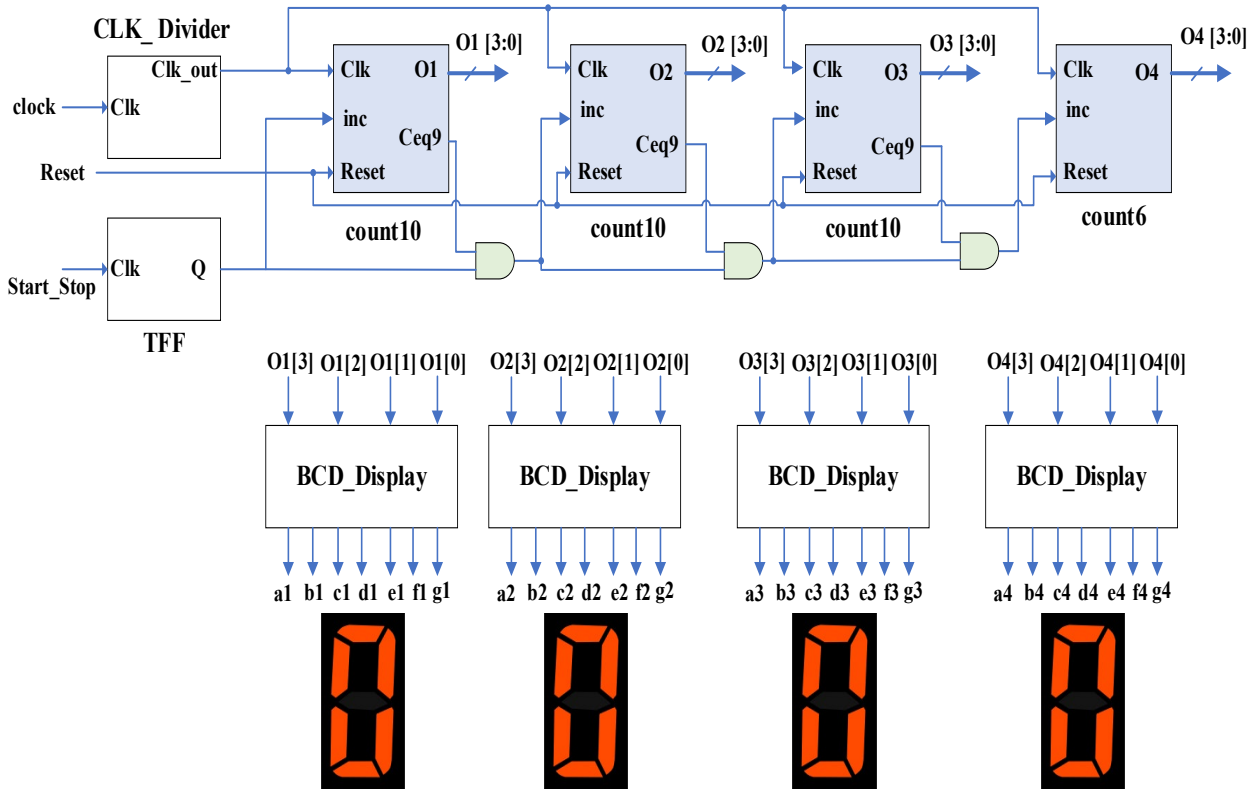
Objectives

1. Combining all the concepts learned throughout the semester to implement a **Real Time Digital Stopwatch.**
2. The stopwatch should have the following features:
   - a ***Start/Stop*** button (single button) that switches the watch on/off when depressed.
   - It should start by displaying 00:00 when powered on.

GOAL

An opportunity to 'connect the dots' between theoretical knowledge and practical implementations. The project may require time commitment; however, I can assure you that the outcome will be extremely satisfying! After the successful completion of this project, you can proudly say that you have designed something **"Cool"**!

Fig. 1. Stopwatch Block Diagram



1. **Designing a 4-bit Ripple Carry Adder that Adds 1-bit Cin to 4-bit A (10 points)**
   - First design a **half-adder (HA)** that can add two one-bit values, **X** and Y**,** and produces one-bit outputs **Sum** and **Carry**.
   - Use the **HA** modules together to form a 4-bit Ripple Carry Adder (RCA) that performs the function A [3:0] +Cin, as shown in the RCA block in Fig. 2.
   - **Include the RCA module code in your project report.**

2. **Designing a BCD-to-Seven Segment Display Controller (10 points)**
   - Create a BCD-to-Seven Segment Display controller that displays a 4-bit number on a 7-segment display. You have already derived the equations for the seven segment outputs to control a common-anode display in LAB#5. Use those equations to model the **BCD_display module in Verilog using Boolean Expressions.**
   - **Include the BCD_display module code in your report.**

3. **Designing Counter Modulus 10 Component (20 point)**
   - You will need to use a **D-Flip Flop (DFF) with active high synchronous reset.** This DFF can store one-bit of information.
     - When **reset**=1: At the **rising edge** of the clock the DFF is reset-to-0, i.e., outputs logic 0.
     - When **reset**=0: at the **rising edge** of the clock, whatever is at the input of the DFF is passed to the output, i.e., **input=output**.

     The Verilog code for the **DFF** is given to you. You **do NOT** need to design that.
   - Design a counter modulus 10 (i.e., 0,1,2,3,4,5,6,7,8,9,0,1…), and make it into a **module** called **Count10.**
     - Use 4 **DFF** components to store the 4-bit counter output, the **RCA** to increment the counter, and combinational logic that resets the **DFF** components when incrementing and the count output is 9 (i.e., 1001_2) or the global reset is logic 1. Refer to the figure below and use that to model the **count10 module.**
   - **Include the count10 module code in your report.**

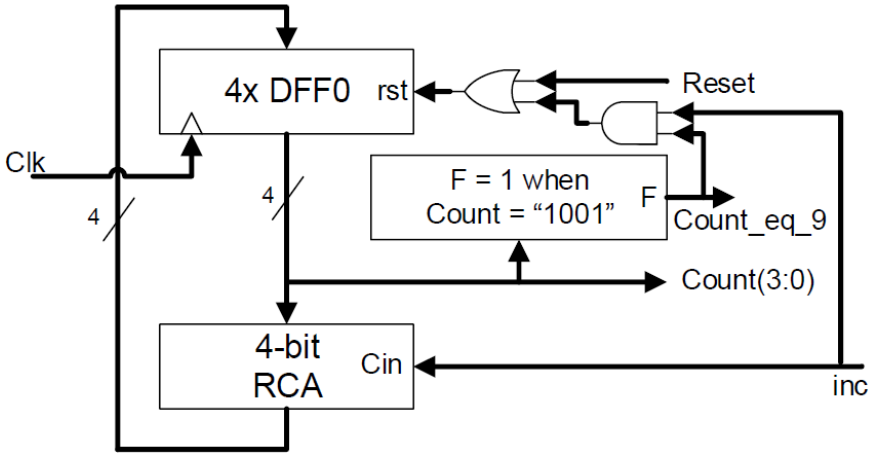


Fig. 2. Count10 Module Block Diagram

4. Designing Counter Modulus 6 Component (10 points)

- Design a counter modulus 6 (i.e., 0,1,2,3,4,5 ,0,1…), and make it into a **module** called **Count6.**
  - Use similar procedure as **Count10** design **in Part 3.** Use 4 **DFF** components to store the 4-bit counter output, the **RCA** to increment the counter, and combinational logic that resets the **DFF** components when incrementing and the count output is 5 (i.e., 0101_2) or the global reset is logic 1. Refer to the same figure (Fig. 2) and use that to model the **count6 module.**
- **Include the count6 module code in your report.**

5. Designing a Frequency Divider (code is provided):

- The internal clock of the FPGA is 50MHz. However, that is too fast and the changes in the display will not be visible at that rate. A frequency divider using D-Flip Flops is designed that will give you a period of approximately 1/100s given a 50MHz clock input. Make it into a symbol called **CLK_divider**, with output *clk_out*, and input *clk*.
- The code is already provided. You **do NOT** need to design this component.

PROCEDURE

1. Complete the **Stopwatch module** as shown in Fig. 1. You will need one instance of the **CLK_divider**, 3 instances of the **count10** module, and one instance of the **count6** module, as shown in Fig. 1.
2. Add 1 T-Flip Flop component and connect its *clk* input to the stopwatch's *Start/Stop* input, such that its output will toggle each time the *Start/Stop* button is depressed. T-Flip Flop code is already provided.
3. Connect the modules together to form the stopwatch described at the beginning of the lab. Hint: the *clock* input of all the counters should be connected to the *clk_out* output of **CLK_divider**; you'll also need three additional AND gates to properly increment the higher order counters.
4. Connect the outputs of your 4 counters to the 4 **BCD_display** components.
5. Follow the Altera Quartus II Tutorial to program the Altera FPGA with your Verilog design.
   - assign ***Reset*** to **SW0**, ***Start/Stop*** to **Pushbutton**[0], ***clock*** to the internal 50MHz clock, `CLOCK_50`, and the least significant **count10** output to Seven Segment Digit 4 (HEX4[0]-HEX4[6]), the next to Seven Segment Digit 5 (HEX5[0]-HEX5[6]), the most significant **count10** output to Seven Segment Digit 6 (HEX6[0]-HEX6[6]), and the **count6** output to Seven

Segment Digit 7 (HEX7[0]-HEX7[6]) (refer to Sections 4.2 – 4.5 of the DE2-115 User Manual; the user manual PDF is provided under the project module on CANVAS).

Deliverables

1. **(20 points) Demonstration to the instructor/TA.**
2. **(80 points)** PDF Report including:
   - **a.** Cover Page (Project Name, Student Name, Project Objective) (2 points)
   - **b.** Project Description (2 points)
   - **c.** Complete Verilog code of the following module: **RCA, BCD_Display, count10, and count6 (50 points)**
   - **d.** **Verilog code of the complete STOPWATCH module (20) points.**
   - **e.** Project Comment: a paragraph on how this project combined many different topics learned as a part of the course, what have you learned, what type of competency you have achieved, etc. (5 points)
   - **f.** Conclusions (1 point)

ACADEMIC INTEGRITY AND HONESTY

I would like to emphasis on one point: **you MUST maintain the ACDEMIC INTEGRITY AND HONESTY while completing and submitting your project. Do NOT copy-paste your codes from your peers, friends, and/or any other sources!**

The whole idea of a project is to help you gather more expertise in an area, and you should try your best to succeed! Remember, your honest effort will be appreciated even if you fail to achieve all the goals!

However, if you take any dishonest means/paths and get caught then I will have to take strict actions as per FL Poly academic integrity policy as mentioned in syllabus (Academic Policy)