CS220: Introduction to Computer Organization

Lab #2

Digital Clock

11 August 2011

1 Introduction

We will design a digital clock module that has 4 different modes. The clock gives the time as an output, and can also perform some other utility functions (alarm clock, stopwatch).

2 Concept

We use counters to keep track of the time in a digital clock. A counter is a device which stores the number of times a particular event or process has occurred, often in relationship to a clock signal. On exceeding its maximum value, the counter reverts back to its original value.

Three counters will be used in the basic "time" mode - a `Seconds` counter, a `Minutes` counter and an `Hours` counter. The `Seconds` counter ideally receives a 1 Hz signal as input for its clock. When its value is 59, on receiving the next clock pulse, it resets itself to 0 and sends a clock signal to the `Minutes` counter. The functioning of the `Minutes` counter as well as the `Hours` counter is likewise similar, except the `Hours` counter resets on receiving a clock pulse when its value is 23 instead of 59.

Some more counters will be needed for the other modes of the clock.

3 Implementation

First we will look at the design of a simple module that gives binary output. Later, we describe the detailed design that drives LED and 7-segment displays. For this lab, You have to implement the second design.

We use BSV `rule ... endrule` with conditions to control the behaviour of the clock.

4 Simple Digital Clock

In this example, we shall synthesize a `DigitalClock` module, which takes a 50 MHz clock input on the input wire `ClockCrystal` and gives the desired outputs `Seconds[5:0]`, `Minutes[5:0]` and `Hours[4:0]`.

Three concurrent behaviour of `rule ... endrule` blocks can be used to update the outputs using non-blocking assignment operators (`<=`). To supply a 1 Hz signal to the concurrent `rule ... endrule` blocks, one can make use of a counter that receives a 50 MHz clock from the onboard clock source. The `rule ... endrule` blocks should execute only when the counter reaches an appropriate value so that the total time taken to reach that value is one second, after which the counter should reset itself to 0.

5 Digital Clock with LED/7 Segment Display

This module requires considerable amount of control signals due to the requirement of displaying the output on Xilinx Spartan-3 LED/7 Segment Displays.

A major difference in this example, apart from the obvious inclusion of components required to drive the displays, is the splitting of the `Seconds`, `Minutes` and `Hours` counters into two counters each. Each counter in the units place works as a decimal counter, counting up to 9, after which it resets on the next clock pulse. Counters in the tens place resent on achieving a value of 5, 5 and 2 (with the units counter on 3) for seconds, minutes and hours respectively. This is because the 7 segment displays accept only one character at a time, and converting a binary number into a two digit decimal number is complicated.

The six counters are updated using the same method of concurrent `rule ... endrule` blocks and a counter generating a 1Hz signal as in the previous example.

When the circuit is initially loaded on the FPGA, the display on the LED's must show `1 2 0 0` (12 Hrs, 00 Min), and the `Seconds` counter (not displayed on LED) must be initialized to the value of `0 0`. The clock must start "ticking" immediately.

6 Adding "modes" to the Clock

Final, we enhance the design of the clock, so that it has the following utility modes:

1. HH:MM mode
2. MM:SS mode
3. Alarm mode
4. Stopwatch mode

The push button **BTN0** will be used to rotate among these four modes. Next we describe these modes in details.

6.1 HH:MM mode

This is the mode where the clock will display Hour and Minute components of the time. This is the initial mode of the clock, with `1 2 0 0` as the initial display.

This mode is also activated when the clock is in Stopwatch mode and the user presses input **BTN0** once.

In this mode, the user can set the time (Hours and Minutes) using the following sequence:

1. Press push button **BTN1** to start loading the hour value.
2. Use slider switches **SW0 ... SW4** to input a valid hour value (0 ... 23).
3. Press push button **BTN2** to load the hour value.
4. Press push button **BTN1** to start loading the minute value.
5. Use slider switches **SW0 ... SW5** to input a valid minute value (0 ... 59).
6. Press push button **BTN2** to load the minute value.
7. Press push button **BTN1** to stop loading.

Activate LED **LD6** when hour value is being updated, and LED **LD5** when minute value is being updated. Blink LED **LD4** if the input value is invalid.

6.2 MM:SS mode

This is the mode where the clock will display Minute and Second components of the time.

This mode is activated when the clock is in HH:MM mode and the user presses input **BTN0** once.

In this mode, the user can set the time (Minutes and Seconds) using the following sequence:

1. Press push button **BTN1** to start loading the minutes value.
2. Use slider switches **SW0 . . . SW4** to input a valid minutes value (0 . . . 59).
3. Press push button **BTN2** to load the minute value.
4. Press push button **BTN1** to start loading the second value.
5. Use slider switches **SW0 . . . SW5** to input a valid second value (0 . . . 59).
6. Press push button **BTN2** to load the second value.
7. Press push button **BTN1** to stop loading.

Activate LED **LD6** when minute value is being updated, and LED **LD5** when seconds value is being updated. Blink LED **LD4** if the input value is invalid.

6.3 Alarm mode

This mode is activated when the clock is in MM:SS mode and the user presses input **BTN0** once.

This mode shows the alarm time in HH:MM format. When the clock time is same as alarm time (alarm "activated"), LED **LD7** on the FPGA must blink.

The alarm time is initialized to `0 7 0 0` (7 AM).

The alarm time can be changed using the following sequence:

1. Press push button **BTN1** to start loading the hour value.
2. Use slider switches **SW0 . . . SW4** to input a valid hour value (0 . . . 23).
3. Press push button **BTN2** to load the hour value.
4. Press push button **BTN1** to start loading the minute value.
5. Use slider switches **SW0 . . . SW5** to input a valid minute value (0 . . . 59).
6. Press push button **BTN2** to load the minute value.
7. Press push button **BTN1** to stop loading.

Activate LED **LD6** when hour value is being updated, and LED **LD5** when minute value is being updated. Blink LED **LD4** if the input value is invalid.

6.4 Stopwatch mode

This mode is activated when the clock is in Alarm mode and the user presses input **BTN0** once. The clock behaves like a stopwatch with START, STOP, RESET buttons.

To simplify the design you may use 4 counters, one each for unit's, ten's , hundred's and thousand's place.

The stopwatch is initialized to `0 0 0 0`.

In this mode,

1. Push button **BTN1** to toggle between start and stop stopwatch. The counting starts from the last value at which the watch was stopped.

2. Push button **BTN2** to reset stopwatch display to **0 0 0 0**.

7 Exercise

1. Implement the digital clock in BSV. Make sure that you create separate interfaces/modules for each of the mode, and connect them together to create final design. You may also need Seven Segment Decoder from Lab #1.

2. Load and run the digital clock module on Xilinx Spartan-3 FPGA. USe the pin constraints as described in the document.

For this exercise to work properly on FPGA, you might need to write a **debouncer** circuit. Search the internet or ask your TAs for the details in lab.

Also, do not leave RST_N pin unconnected, as it may result in undesired behaviour.