How to use Verilog and Basys 3 to do stop watch

I have done this project for an online class. The project is written by Verilog. The clock generator, enable\_sr(enable digit) and ssd (seven segment display) modules were provided. My task was to write the top module and counter modules to make a stop watch on the 7 segment display. Originally, the project was implemented in Basys 2. I also used Xilinx ISE Webpack. Now, I modified the project and implemented it on Basys 3. In addition, I used Vivado Webpack instead of ISE. I have also written up a getting started guide for Vivado.

**Step 1: Create a new project**

If you haven’t got Vivado, please look at the [getting started guide](http://www.instructables.com/id/Simple-Logic-Design-w-Digilent-Nexys-4-Field-Progr/) and get one for free. You can follow the [How to use Verilog and Basys 3 to do 3 bit counter instructable project](http://www.instructables.com/id/How-to-use-Verilog-and-Basys-3-to-do-3-bit-binary-/) step 2 to create projects and files.

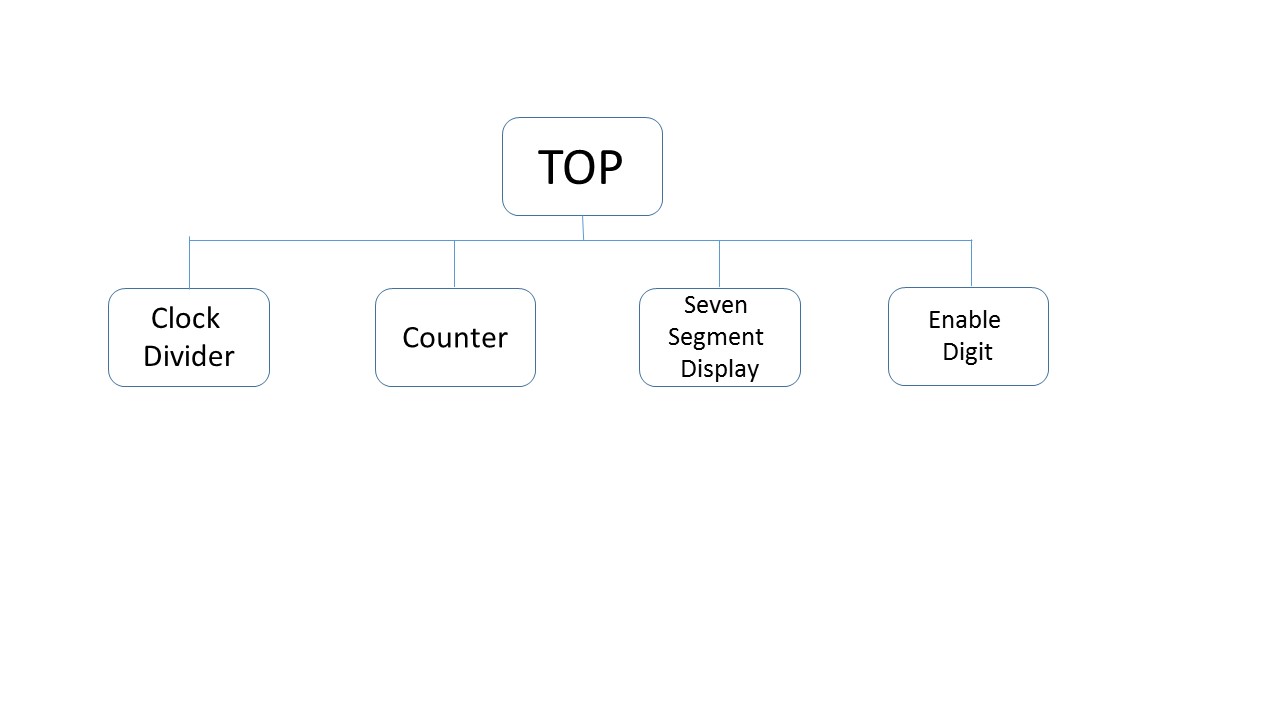
**Step 2: Design the block diagram**

How does stop watch work?

The stop watch keeps time down to 0.1 second. It starts when a push button is pressed. It stops if the same button is pushed. The count will resume if this button is pushed after stop. There is another reset button. The stop watch will be reset if the count stop and reset button is pushed. The count will be displayed in the 4 digits seven segment display.

Block Diagram

Based on the functional requirement, there should be four modules – Top, counter, seven segment display, clock divider and digit enable



**Step 3: Create Clock Divider module**

I have used the original provided file called clkgen. We use delay to create two different clocks – stop watch counter clock and 4 digit seven segment display refresh clock. The Global Clock Buffer primitive is also used to maintain low skew. You can find out the primitive details at [Xilinx Vivado Tutorial](http://www.xilinx.com/support/documentation/sw_manuals/xilinx14_1/7series_scm.pdf). The Basys 3 clock speed is 100MHz, so I modified the for loop count to make sure to keep time down to 0.1 second.

I have referred to the example from [Learn Verilog by Example](http://simplefpga.blogspot.com/2012/07/to-code-stopwatch-in-verilog.html) to calculate the for loop count

How to calculate the for loop count?

Basys 3 has a 100 MHz clock which means that the clock cycle is repeated 100M times in one second. To to create a 0.1 second delay, we multiply the clock with the required time: 100MHz \* 0.1 sec = 10000000. So the clock cycle is repeated 10M times in 0.1 second. Then, we calculate the size of the register that will hold this count. Suppose,

2exp(x) = 10000000

log 2exp(x) = log 10000000 (x)

x ~ 26, so a 27 bit wide vector register variable can hold the count.

**Step 4: Create counter modules**

First we need to design the logic to detect the edge of the button signal because for debounce purpose. There are two buttons – start\_stop button and reset button. We need to create a register variable to store the value - button\_n\_ff and reset\_n\_ff. In addition, we need to create two signals - start\_stop and reset. I used the event trigger procedural clock and non-blocking assignment as well as the conditional statement. The button is active low. When there is difference between value of button and its register variable, there will be change for the start\_stop or reset signal.

I have referred to the example from [Learn Verilog by Example](http://simplefpga.blogspot.com/2012/07/to-code-stopwatch-in-verilog.html) to design the counter logic. The right most seven segment display digit is the least significant digit while the left most is the most significant digit. There are three conditions:

Condition one

Both start\_stop and reset signals are asserted. All digits are zero value

Condition two:

Only start\_stop signal is asserted. Store the previous count. When stop button is pressed again, resume from the previous count.

Condition three:

start\_stop signal is not asserted. I used the nested if else if statement to count from right most to left most digit. Whenever the count for a digit is up to 9, it will be reset back to 0 and then move to the next digit.

**Step 5: Create seven segment display module**

Create a vector input (hex) so that 0-9 can be shown in the seven segment display. The output is the seven LED segments, so I create the vector register output. Use the event triggered procedure block and case statement to assign seven segments into 1 or 0 in each input (hex). The default for case statement is 0. You can refer to the [How to use Verilog and Basys 3 to do 3 bit counter instructable project](http://www.instructables.com/id/How-to-use-Verilog-and-Basys-3-to-do-3-bit-binary-/) to understand how you turn on the seven segment displays and how it works in Basys 3.

**Step 6: Create digit enable module**

Input is the refresh clock created by clock divider module and outputs are all four digits on Basys 3. Create the vector register variable “pattern” to have the initial pattern for digit. Remember, the digit is active low logic (refer to the [How to use Verilog and Basys 3 to do 3 bit counter instructable project](http://www.instructables.com/id/How-to-use-Verilog-and-Basys-3-to-do-3-bit-binary-/) ). Create a shift register with the use of clock edge triggered procedure block to shift “0” in each digit.

**Step 7: Create top module**

Inputs are Basys 3 master clock and buttons (stop and reset). Outputs are 4 digits and seven segment LEDs. I also created the variable for counter clock, refresh clock, hex number (0-9) and count for each digits. I then instantiate the following previously created modules

1. Clock divider (clkgen)
2. Counter (counter)
3. Seven segment display (ssd)
4. Digit enable (enable\_sr)

I then needed to link the hex to reg by using a case statement in the event triggered procedure block.

**Step 8: Synthesis, Implement and Program the project**

You can refer to [How to use Verilog and Basys 3 to do 3 bit counter instructable project](http://www.instructables.com/id/How-to-use-Verilog-and-Basys-3-to-do-3-bit-binary-/) to learn how to synthesis, implement and program the design on the Basys 3.