TIMER PERIPHERIAL

This lab has you create a timer peripheral such as that typically found in a microconroller.

BACKGROUND

Review the notes on creating bus-based peripherals and parameterized Verilog.

TASK DESCRIPTION

You are to implement a timer peripheral with the following features:

- 32-bit timer, 32-bit period register
- Status, Control register

The module interface is given below (note the parameter definitions and default values):

```
module timer32(clk, reset, din, dout, wren, rden,addr)
    );
input clk, reset, wren, rden;
input [31:0] din;
output [31:0] dout;
input addr[2:0];

parameter PERIOD = 32'h0000000F;  //must have this initial value
parameter ENBIT = 1'b0;
```

The input/output definition is:

- *clk* – clock input
- *reset* – high true reset
- *rden* – read enable for dout bus
- *din* – data input bus
- *dout* – data output bus, reflects the contents addressed by the *addr* input when *rden* is high, else *dout* is 0.
- *addr* – address bus for internal registers
- *wren* – write line for a register. The register selected by *addr* is written on the rising clock edge when *wren* is high.

Register addressing and reset behavior:

Register	Address	Value at reset
Timer	0b00	0
Period	0b01	Set to parameter `PERIOD`
Status/Control	0b10	All bits 0, except for enable bit, which is set to

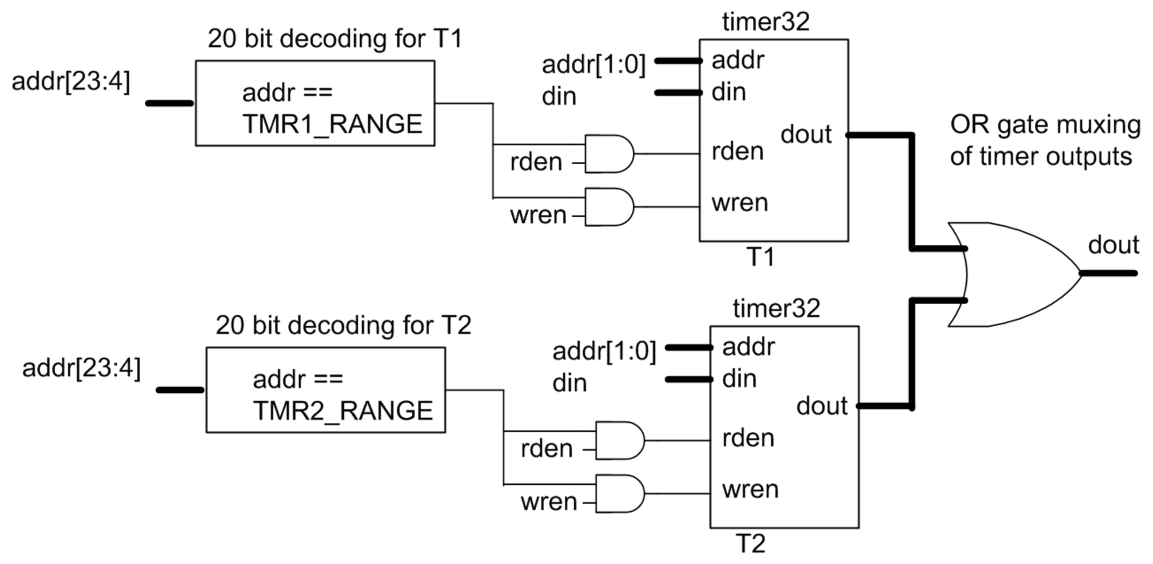
		parameter ENBIT
(dout = 0 for this)	0b11	

Status/Control register definition:

- Bit 0: Enable bit, timer counts up when '1', frozen when '0'.
- Bit 1: TMR Flag – this is set to a '1' when the period register matches the timer contents. This bit is cleared when a read is done of this register – it can also be cleared by a write to the register
- Bit 2: Toggle bit – this bit is complemented each time the period register matches.

For the control register, a write operation takes precedence over any other action.
For the TMR flag, if a read operation (clears the TMR flag) and a period match (sets the TMR flag) occur in the same clock cycle, then the period match takes precedence.

After you have the timer module completed and tested, you are to implement two of these modules with address bus decoding as shown below.



The module interface for this logic in is shown below, the parameters TMR1_RANGE and TMR2_RANGE define the address ranges for Timer1, Timer2 respectively.

```
module timer32bus(clk, reset, din, dout, wren,rden, addr);
input clk, reset, wren,rden;
input [31:0] din;
output [31:0] dout;
input [23:0] addr;    //24 bit address

//20-bit decode, compare against addr[23:4]
parameter TMR1_RANGE = 20'h9250A;   //20 bit decode
parameter TMR2_RANGE = 20'h3C74D;   //20 bit decode
```

Associated files

The ZIP archive associated with this lab contains the following files:

- *timer32.v* -- complete this module
- *tb_timer32.v* – test bench for timer 32
- *timer32bus.v* – complete this module for part 2
- *tb_timer32bus.v* – testbench for part 2
- *report.doc* – report file that needs to be filled out

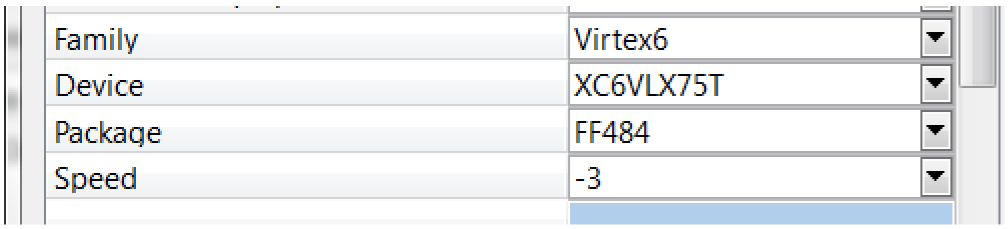
Part 1: timer32.v

Create a project named *lab6_timer32* and finish implementing the *timer32.v* module (use the Spartan Family Xilinx device you have used for previous labs). Verify both behavioral simulation and Post-route simulation using the *tb_timer32.v* testbench.

Fill out the requested information in the report.doc file.

Part 2: timer32bus.v

Create a project named *lab6_timer32bus* and finish implementing the *timer32bus.v* module that requires instantiating two of the timer32 modules you completed in Part1. For this project, use the device Family, Device, Package and Speed grade shown below – this is necessary as the Spartan family FPGA does not have enough pins



Family	Virtex6
Device	XC6VLX75T
Package	FF484
Speed	-3

Verify both behavioral simulation and Post-route simulation using the *tb_timer32bus.v* testbench.

Fill out the requested information in the report.doc file.

Part 3 (bonus): timer32bus.v (+20 pts)

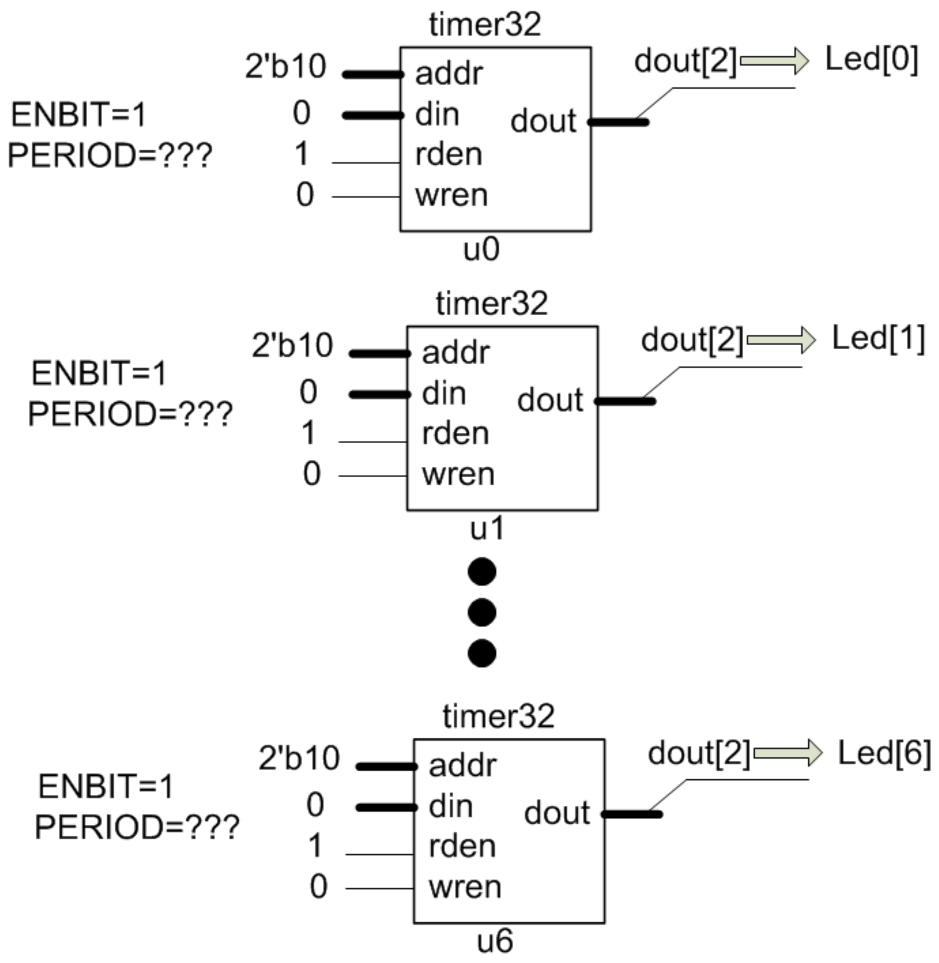
This bonus requires you to create a design to be loaded into the Basys2 board. Create a project named *lab6_bonus* with a top-level Verilog module that has the following interface:

```
module timertop(Led,swt,mclk);
output [6:0] Led;
```

```
input [0:0] swt;
input mclk;
```

Within this module, instantiate seven copies of the timer32 module as shown below. The address bus is hardwired to 2b'10 and the *rden* input to '1', thus outputting the status register on the output. Bit 2 of the output (the toggle bit) is tied to an LED. The ENBIT parameter of each timer module is '1', so the timers are running at power up. Set the PERIOD parameter such that the timer periods are 0.125 s, 0.25 s, 0.5 s, 1.0 s, 2.0 s, 4 s, and 8 s. Because the LED is tied to the toggle bit, it will be ON for the timer period and then OFF for the timer period (e.g., ON for 8 seconds, then OFF for 8 seconds). Visually, the LEDs will appear to count in a binary pattern. The *mclk* input is 50 MHz.

The clock input of the each timer is tied to *mclk*, and the reset input to the switch 0 input (*swt[0]*). Remember that you need to add the appropriate UCF file to your project as was done in the first lab exercise in order to map this to the Basys board. Also remember to follow the directions from the first lab that sets the FPGA Start-up Clock to be the JTAG clock.



To get credit for this part, you need to demo this to the TA as well as submitting this project.

Submission

For submission, create a directory named 'lab6_*netid*', i.e., (lab6_rbr6).

Copy your *lab6_timer32* project directory to this directory.
Copy your *lab6_timer32bus* project directory to this directory.
Copy your *lab6_bonus* project directory to this directory.
Copy your completed *report.doc* to this directory.

Create a ZIP archive of this directory and submit it.