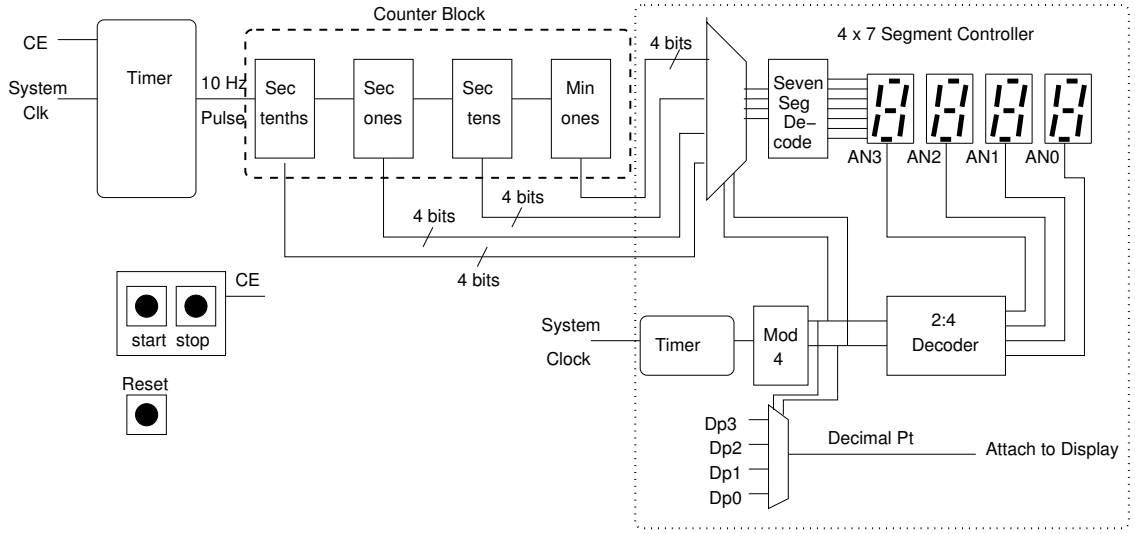
Lab 5 – Fully Functional Stop Watch

1 Objective

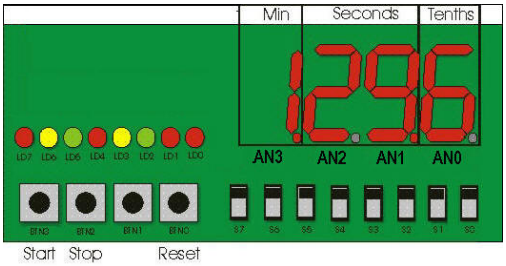
Build a fully functional stopwatch.

2 Introduction

The overall goal of this lab is to build a fully functional stopwatch. This system will involve many smaller parts and putting them together in an overall design. This should not be new, but this will be completed on a large scale than has been done so far. Be sure to save your work often and make backup copies (e.g., USB flash drive, Network drive, etc.) Also, all of your designs should be done with basic logic operations (and, or, not, xor, ...) and with the provided flip-flops. All labs will be completed with structural designs and not behavioral design.



The above diagram shows the basic overall design of the stop watch. The timers and counters will be built in Part II of the lab. Part I will concentrate on the four seven segment display controller. The controller will allow unique data to be displayed on each of the digits of the seven-segment display. The control part of the seven segment controller is the Mod 4 state machine which you will design as part of this lab.



3 Part I

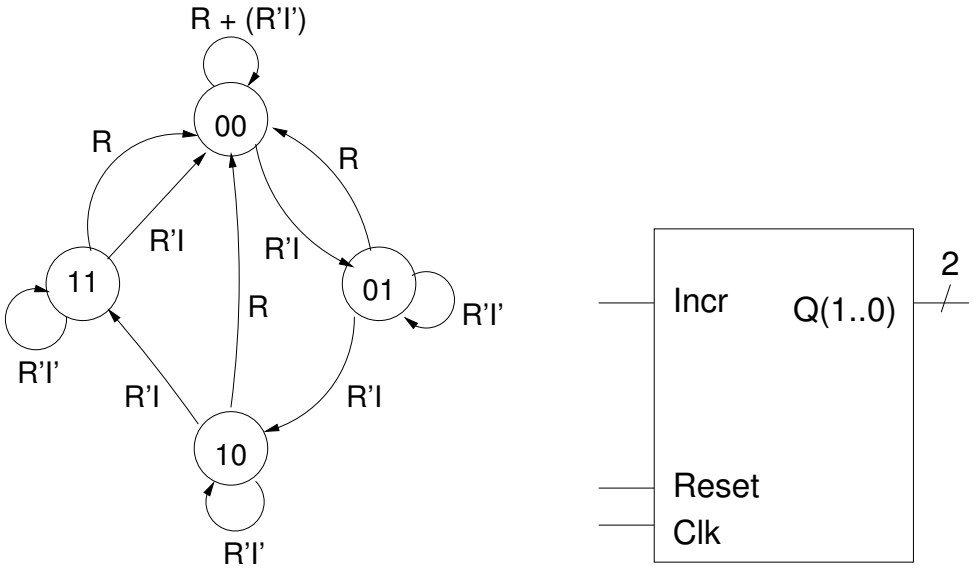
3.1 Preparation

1. Design a 2 to 4 Decoder. This circuit will have one input A(1..0) and produce a single output D(3..0). It will require four separate functions for each output bit. A truth table, K-map, equations, test bench and wave forms should be included in your final lab write up.

2. Copy your trusty 4 to 1 MUX from the previous lab, since you will need it for this one. (Be sure to include all the files).

3. In the previous lab, you also designed a 16 to 4 MUX, although it may not be in a separate file or as a separate entity. Take time to separate this as its own circuit element in preparation for use in this lab.

4. Design the Mod4 State Machine. You should include the state graph, truth table, k-maps and equations in your lab write-up. (Reset has priority.)



After simplifying the Mod4 State Machine with K-Maps, you will have two next state equations for Q(1) and Q(0). Draw these as D-type flip-flops and logic gates. The flip-flops you will be using are simple D-type flip-flops with a synchronous "clear" input. The VHDL code for the flip-flop is provided in the lab05files.zip file on the lab website.

Since the "reset" input to the state machine has priority, nd since the flip-flop has a "clear" input, we can wire the "reset" signal directly to the clr input port of the flip-flops. This automatically gives the "reset" signal priority since the flip flops will always go to "0" on the next clock edge when this signal is asserted . This enables you to simplify the transition table since you no longer have to design circuitry to reset the state machine.

5. 4 × 7 Segment Controller

   The 4 × 7 Segment Controller will allow you to display different data on each of the four seven-segment displays. The operation is simple. It will read the first input and display it on the first digit. It will then read the second input and display it on the second digit and so forth. When it gets to the last digit it will start reading the first digit again. When it cycles through this sequence fast enough the display appears to show all four digits simultaneously.

   The parts you will need for the 4 × 7 segment controller are:

   (a) Seven Segment Decoder (This was used in Lab4, it will be provided for you.) ssg.vhd

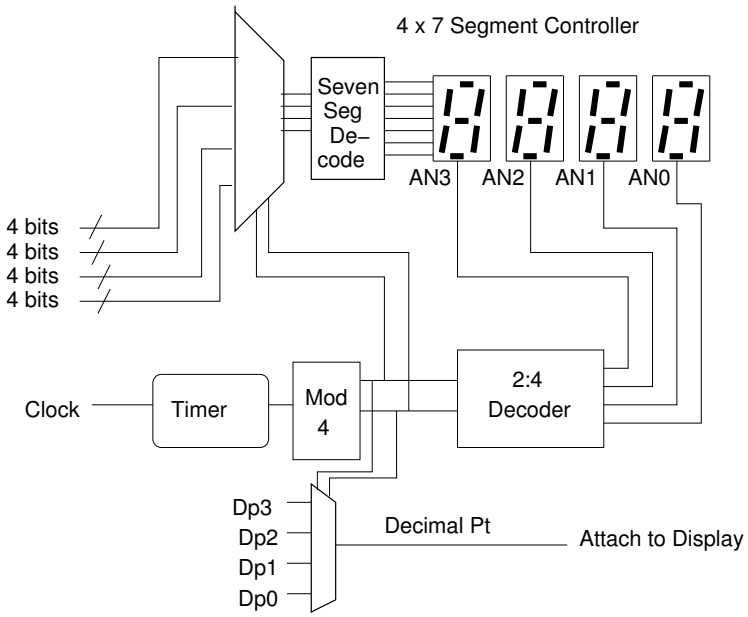
   (b) 16 to 4 Mux (Reuse some code and design as described above.)

   (c) Mod4 (make in this lab with provided flip-flops)

   (d) 2 to 4 Decoder (Design as described above).

   (e) 4 to 1 Mux (Reuse design)

   (f) Programmable Timer (supplied) prog_timer.vhd



3.2 Procedure

1. Build the Mod4 state machine using VHDL code. Show the work you did to obtain the next state equations. Compile the circuit then simulate it to verify correct operation. Don't forget to initialize the flip-flops by asserting the "reset" signal.

2. Build a Test Bench for the programmable timer. You are building this to become familiar with the programmable timer. Program the timer to give a 200 Hz signal.

This will later be hooked up to the increment signal for the Mod4 counter. Determine the value to load into the programmable timer to achieve this result. Document your calculations.

(a) Download the Programmable Timer from the lab 5 web site.

(b) See the Programming the Timer tutorial for details on how to program the timer.

3. Build the 4x7 Segment Controller using the parts listed in the preparation. When you copy files from your old labs, be sure to have unique copies for this new lab. Also put your own copies of the provided VHDL in the same location.

4. Your new entity for the 4x7 Segment Controller should have the inputs and outputs shown as well as a few test signals including the two outputs from the Mod4 and the zero or tp clock (part of the programmable timer)

Inputs	Outputs
Digit1 (4 bits)	Ca
Digit2 (4 bits)	Cb
Digit3 (4 bits)	Cc
Digit4 (4 bits)	Cd
System Clock	Ce
Reset	Cf
Dp0	Cg
Dp1	An0
Dp2	An1
Dp3	An2
	An3
	DP
	Q(1) (Mod4 bit)
	Q(0) (Mod4 bit)
	tp
	zero

5. Testbench

- Create a test bench for the Seven Segment Controller part.
- The inputs for the digits should be as follows
  - 0001 on digit 1 - leftmost 7-segment display
  - 1010 on digit 2
  - 1011 on digit 3
  - 1000 on digit 4 - rightmost 7-segment display

  Dp0 through Dp3 should be mapped to switches. This will allow you to control the decimal points during lab sign-off. Please note that dp is low asserted.

- Attach the rest of the inputs and outputs as appropriate for correct operation and testing.

Test the results of your design. Have a TA verify the operation of your working circuit.

4 Part II

4.1 Overall Design

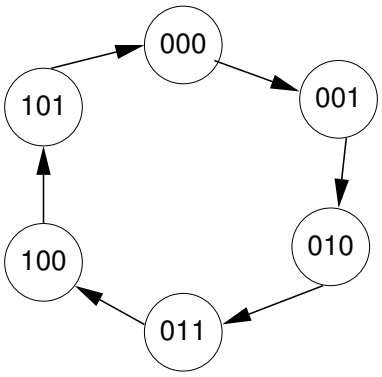
The stopwatch will be the culmination of previous designs. It is important to have valid working parts for each of the previous parts before you start this lab. It is VERY important to have a complete 4x7 segment controller working when you come into this lab. It will SIGNIFICANTLY reduce the amount of time you spend working on this lab
Also note the graphic of the digilab from Part I. This indicates where the different buttons, switches, and displays should be mapped.

4.2 Preparation

1. The start and stop buttons for the control of this design will be used in a way similar to an SR Latch. You should use a flip-flop with the start and stop signals to control the clock enable (ce) used within the circuit. Like the SR latch, if you press start the output of the flip-flop should be '1' and when you press stop, the output of the flip-flop should be '0'.

2. Using conventional state machine design - design the Mod6 and Mod10 counters. You should use the Flip-flops with Clear and Clock enable functionality - specifically the DFF_CE flip-flop given in the lab05files.zip. This will reduce the number of inputs required in the design of the state machines.

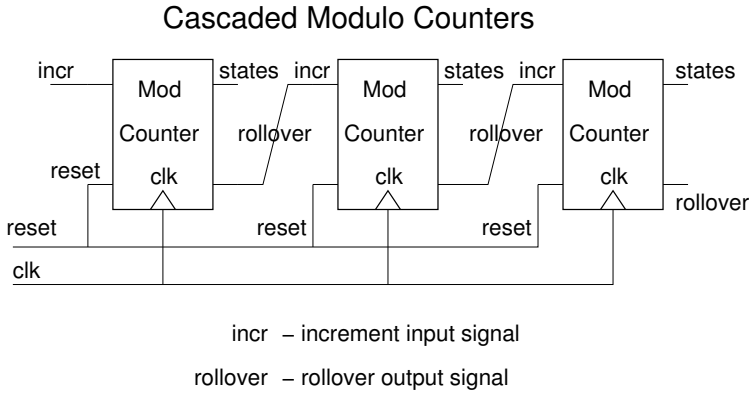
   Document the state graph, state table, k-maps, and equations in your lab write-up. Also document your cascade design (see below).



Mod 6 Binary Counter Design the circuit for a Mod6 counter. It will have 6 total states from 0-5. In addition to your 3 state bits, you should generate a clock enable

output since the counters will be cascaded. Cascading means that the clock enable output will be the clock enable input for the next counter module. Determine exactly when the carry out should be asserted.

Mod 10 Binary Counter Design the circuit for a Mod10 counter. It will have 10 total states from 0-9. Again, you should generate a clock enable output. Determine exactly when the carry out should be asserted.



The rollover outputs on the counter modules should be designed such that the signal is asserted for the length of only ONE clock period.

3. The stopwatch requires a periodic pulse of 10 times per second. You must determine what preload value to put into the programmable timer to generate the 10 Hz pulse. Please refer to the previous lab if you do not remember how to setup the programmable timer. Document your calculations and the load number in your lab write-up.

4.3 Procedure

The stopwatch will consist of

- 1 4x7 Segment Controller (Part I - be sure to include all of the building blocks)
- 3 Mod10 Binary Up Counters (This lab)
- 1 Mod6 Binary Up Counter (This lab)
- 1 Programmable Timer (Provided)
- 1 Start/Stop clock enable circuit (This lab)

1. Design and simulate your Start/Stop Clock Enable circuit.
2. Build the Mod6 and Mod10 counters. Simulate each separately to verify their correct operation.

3. Build the Counter Block using the Mod10 and Mod6 counters. Simulate this block to make sure that your counter will count correctly and that the digits will overflow at the correct time. Any clock speed will work for the simulation because we are only interested in the sequence of numbers at this time. When you hook up the programmable timer later, you will have to plug in the correct value to get the correct timing.

4. Add a programmable timer and program it to generate 10Hz.

   (a) Add a Programmable Timer to your design. You can copy the VHDL source from part I.

   (b) See the Programming the Timer tutorial for details on how to program the timer.

   (c) You may need to adjust the timer preload value in order to get the 10Hz frequency.

5. Complete the entire stopwatch. Create your test bench using:

   (a) the Counter Block

   (b) the 10Hz timer

   (c) the start/stop clock enable (for controlling the start/stop functionality)

   (d) the 4x7 segment controller

   (e) 3 buttons: start, stop and reset

   The reset should clear the values in the counter block but it should NOT stop the 4x7 segment display from operating.

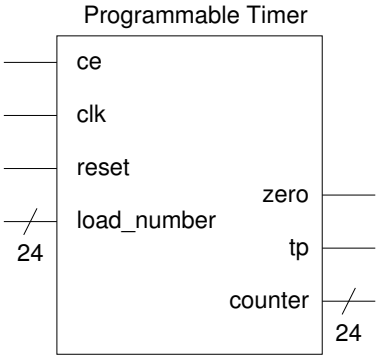
   Fully document your design.

Pass-off the circuit as defined in the Preparation Section. This includes a download of the entire circuit using the top-level circuit provided in an additional zip file.

5 Programming the Programmable Timer Tutorial

5.1 Understanding the Programmable Timer I/O

The timer is really just a count down counter (i.e. it counts down to 0). When Reset is a "1" or if the counter value reaches 0, the counter gets reloaded with the preload value (supplied on the load_number input port) and then begins counting back down to 0.



ce is the clock enable. Tying this to '1' or Vcc, will cause the timer to run continuously.

clk is the clock input. It is connected to the system MCLK signal.

reset is the reset signal. It is an active high reset and is asynchronous to the clock input.

load_number is a 24-bit input bus which supplies a preload value (see below) for the timer.

zero is the zero output signal which is asserted when the count reaches 0. The zero signal is only asserted for one clock period of clk.

tp is a test point for the programmable timer. The output is a square wave at 1/2 the frequency of the zero output.

counter is a 24-bit output bus which gives the state of the 24 bits of the counter. It is only used for debugging purposes.

The zero output will be connected to the Incr input of the Mod4 state machine you design in part I of this lab.

5.2 Specifying the Preload Value

It is up to you to determine the value for the preload and provide it as an input to the load_number port..
To calculate the desired value for the preload you need the following data:

1. 50MHz system clock in
2. 200Hz increment clock out
3. Count down counter

As an example, suppose you wanted the ceo output port to pulse once every 12 clock periods. You would specify the load_number input to be:

`conv_std_logic_vector(12, 24)` – the decimal number 12 specified as a 24-bit number. This conversion function requires that you import `STD_LOGIC_ARITH` in the library block found at the top of your code. For example:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; -- this line is required for std_logic, etc.
use IEEE.STD_LOGIC_ARITH.ALL; -- this gives us the conv_std_logic_vector ...
```

You could also use:

`"x00000C"` – This is 12 in hexadecimal. In VHDL, hexadecimal is assumed to be 4bits wide per digit.

In either case, you would simply provide that number as input to the timer on the load_number port.