Hierarchical Design

Introduction

In this lab you'll practice instantiating modules by building a digital combination lock. You'll be supplied five modules. You'll also design a sequence detector based on a resetable 4-digit shift register.

Components

You will need the CPLD board, the four-digit 7-segment LED display and the 4×4 matrix keypad used in previous labs.

Requirements

Each press of a key should shift that digit into the 4-digit 7-segment LED display from the right. The red LED on the CPLD board should be on only when the display shows the last four digits of your BCIT ID (this shows when the lock is open). Pressing the * key should reset the display to 0000 and close the lock (turn off the LED).

Your design must instantiate the modules described below.

Module Descriptions

You will be supplied with a file, **lab6modules.sv** containing definitions of the following modules.

keypad This module outputs a 4-bit binary value, **digit**, corresponding to the key that is being pressed. Keys A through D output their hexadecimal values, * outputs value 14 (hex **E**) and # outputs 15 (hex **F**). The **valid** output is true when the value of **digit** is valid. **digit** should be ignored otherwise.

```
module keypad      // 4x4 matrix keypad decoder
  ( output logic [3:0] row,// matrix row output
    input logic [3:0] col, // matrix column input
    input logic clk,       // clock
    output logic [3:0] digit, // value of key
    output logic valid ) ; // high if digit is valid
```

display The 4-digit LED display displays the four 4-bit digits on the input **digits**.

```
module display        // 4-digit 7-segment display
  ( input logic [3:0][3:0] digits, // digits L-to-R
    input logic clk,           // clock
    output logic [3:0] en,     // active-low enables
    output logic a, b, c, d,
                 e, f, g ) ; // active-high segments
```

clkdiv The clock divider generates a square-wave clock. It has parameters for the input and output frequency.

```
module clkdiv                  // clock divider
  #( fin = 50000000,           // input frequency
     fout = 200 )              // output frequency
   ( input logic clk_in,       // input clock
     output logic clk );       // clock out (50%)
```

debounce The switch debouncer outputs the input value, **sw_in** on the output **sw**, when the input has been stable for parameter **N** clock cycles.

```
module debounce                // switch debouncer
  #(parameter N=16'hffff,      // delay (clocks)
    w=1)                       // switch width
   ( input logic [w-1:0] sw_in, // switch input
     input logic clk,          // clock
     output logic [w-1:0] sw ) ; // switch output
```

rising The rising-edge detector sets **out** high for one clock cycle when the **in**put transitions from low to high between rising edges of the clock.

```
module rising            // rising-edge detector
  ( input logic in, clk, // input & clock
    output logic out ) ; // high one clock on edge
```

Suggested Solution

The diagram in Figure 1 shows a possible solution.

You'll need to design the logic in the block marked "shift register and sequence detector." This is a 4-element 4-bit shift register that shifts the **digit** from the **keyboard** module into a shift register when the debounced **valid** signal indicates a new digit. It also resets the shift register to zero (**16'h0000**) if * is pressed (decimal value 14).

Use the **clk** signal as a clock. Do *not* use the outputs of the **rising** or **debounce** modules as a clock.

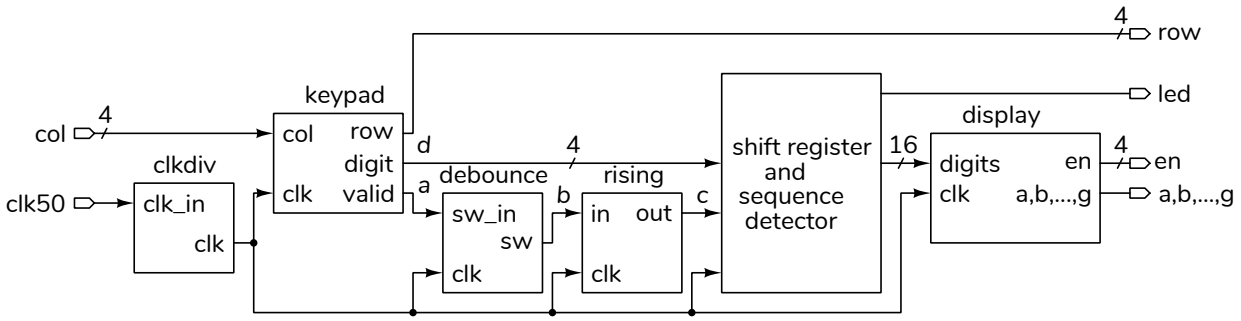


Figure 1: Example solution schematic.

You'll need to set the parameters of the **`clkdiv`** (e.g. **`fout=2000`** Hz) and **`debounce`** (e.g. **`N=20`**, 10 ms).

Procedure

Create a project as in previous labs. Download the **`lab6modules.sv`** file from the course web site and add it to your project. Pin assignments can be imported from previous labs that used the keypad and LED display if you are using the same pins. Remember to enable the pull-ups on the **`col`** inputs.

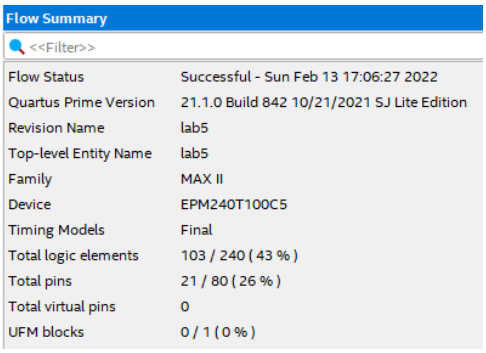
Create a **`lab6.sv`** file containing a module (e.g. **`lab6`**) that has the necessary inputs and outputs as shown in the diagram and described in previous labs. Your module should also instantiate the modules described above and declare the signals you'll need to connect the modules (those labelled *a* through *d* in the diagram – but use your own names). You'll need to add some code to implement the shift register and to turn on the **`led`** when the shift register has the correct combination (the last four digits of *your* BCIT ID).

Program the CPLD and connect it to the keypad and LED display as in previous labs. Test your design.

Submission

To get credit for completing this lab, submit the following to the Assignment folder for this lab on the course website:

1. A PDF document containing:
   - A Verilog listing of your design (e.g. **`lab6.sv`**). Do *not* include a listing of **`lab6modules.sv`**.
   - A screen capture of your compilation report as in previous labs.



   - The schematic created by **Tools > Netlist Viewers > RTL Viewer** and using **File > Export...** It may look like Figure **??**.

2. If you do not demonstrate your design in the lab, a video of the display, keypad and CPLD board showing:
   - the display being reset to all-zeros when [*] is pressed
   - entering the last four digits of your BCIT ID and the LED lighting when the correct digits are entered
   - the display being reset to all-zeros when [*] is pressed
   - entering in [1] [2] [3] [4] and showing that the LED does not light

A sample video is available on the course website.

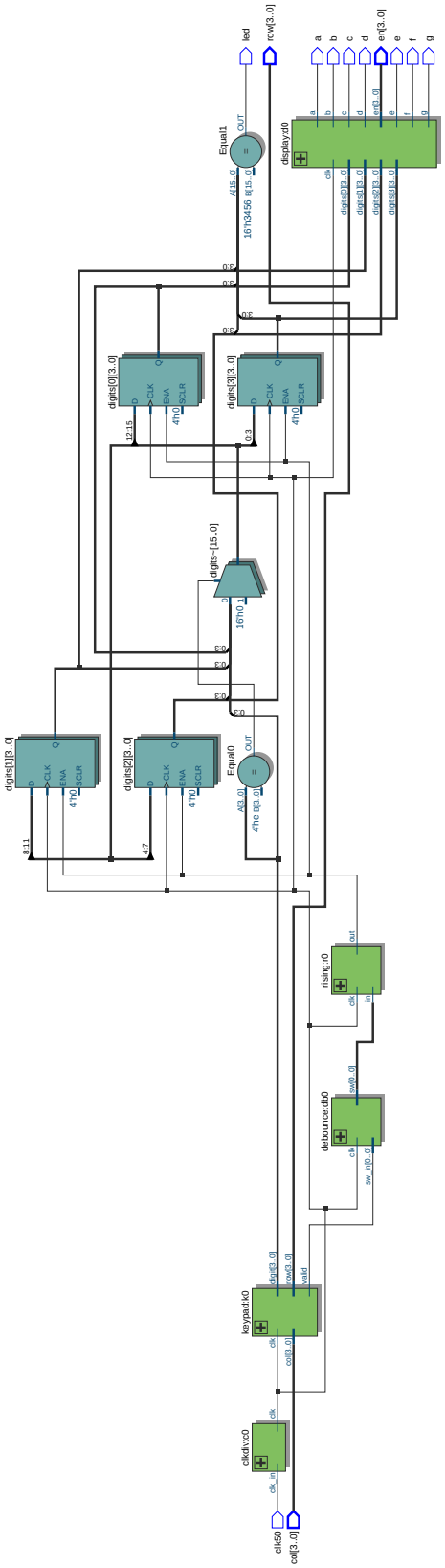


Figure 2: Example RTL Netlist output.