

ELEX 7660: Lab 2

Matrix Keypad Decoder

|  |  |  |
| --- | --- | --- |
| Nicholas Huttemann | 25/01/2019 | Set: T |

Table of Contents

[1 Modules 3](#_Toc536198249)

[1.1 Top level lab2.sv 3](#_Toc536198250)

[1.2 Colseq 5](#_Toc536198251)

[1.3 Kpdecode 5](#_Toc536198252)

[1.4 Decode7 6](#_Toc536198253)

[1.5 Simulation Results 7](#_Toc536198254)

[2 RTL Diagrams 7](#_Toc536198255)

[2.1 Top Level 7](#_Toc536198256)

[2.2 Colseq 8](#_Toc536198257)

[2.3 Kpdecode 8](#_Toc536198258)

[2.4 Decode7 9](#_Toc536198259)

[3 Compilation Summary 9](#_Toc536198260)

Table of Figures

[Figure 1: Simulation Results 7](#_Toc536198261)

[Figure 2: Top level diagram lab2.sv 7](#_Toc536198262)

[Figure 3: colseq.sv diagram 8](#_Toc536198263)

[Figure 4: kpdecode.sv diagram 8](#_Toc536198264)

[Figure 5: decode7.sv diagram 9](#_Toc536198265)

[Figure 6: Compilation Summary 9](#_Toc536198266)

# Modules

## Top level lab2.sv

// ELEX 7660 ELEX 7660 201710 Lab2

// 4x4 keypad decoder and LED display

// Ed.Casas 2017-1-11

**module** lab2 **(** **output** **logic** **[**3**:**0**]** kpc**,** // column select, active-low

**(\*** altera\_attribute **=** "-name WEAK\_PULL\_UP\_RESISTOR ON" **\*)**

**input** **logic** **[**3**:**0**]** kpr**,** // rows, active-low w/ pull-ups

**output** **logic** **[**7**:**0**]** leds**,** // active-low LED segments

**output** **logic** **[**3**:**0**]** ct**,** // " digit enables

**input** **logic** reset\_n**,** CLOCK\_50 **)** **;**

**logic** clk **;** // 2kHz clock for keypad scanning

**logic** kphit **;** // a key is pressed

**logic** **[**3**:**0**]** num **;** // value of pressed key

**assign** ct **=** **{** **{**3**{**1'b0**}},** kphit **}** **;**

pll pll0 **(** **.**inclk0**(**CLOCK\_50**),** **.**c0**(**clk**)** **)** **;**

colseq colseq\_0 **(.\*)** **;**

kpdecode kpdecode\_0 **(.\*);**

decode7 decode7\_0 **(.\*)** **;**

**endmodule**

// megafunction wizard: %ALTPLL%

// ...

// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!

// ...

**module** pll **(** inclk0**,** c0**);**

**input** inclk0**;**

**output** c0**;**

**wire** **[**0**:**0**]** sub\_wire2 **=** 1'h0**;**

**wire** **[**4**:**0**]** sub\_wire3**;**

**wire** sub\_wire0 **=** inclk0**;**

**wire** **[**1**:**0**]** sub\_wire1 **=** **{**sub\_wire2**,** sub\_wire0**};**

**wire** **[**0**:**0**]** sub\_wire4 **=** sub\_wire3**[**0**:**0**];**

**wire** c0 **=** sub\_wire4**;**

altpll altpll\_component **(** **.**inclk **(**sub\_wire1**),** **.**clk

**(**sub\_wire3**),** **.**activeclock **(),** **.**areset **(**1'b0**),** **.**clkbad

**(),** **.**clkena **({**6**{**1'b1**}}),** **.**clkloss **(),** **.**clkswitch

**(**1'b0**),** **.**configupdate **(**1'b0**),** **.**enable0 **(),** **.**enable1 **(),**

**.**extclk **(),** **.**extclkena **({**4**{**1'b1**}}),** **.**fbin **(**1'b1**),**

**.**fbmimicbidir **(),** **.**fbout **(),** **.**fref **(),** **.**icdrclk **(),**

**.**locked **(),** **.**pfdena **(**1'b1**),** **.**phasecounterselect

**({**4**{**1'b1**}}),** **.**phasedone **(),** **.**phasestep **(**1'b1**),**

**.**phaseupdown **(**1'b1**),** **.**pllena **(**1'b1**),** **.**scanaclr **(**1'b0**),**

**.**scanclk **(**1'b0**),** **.**scanclkena **(**1'b1**),** **.**scandata **(**1'b0**),**

**.**scandataout **(),** **.**scandone **(),** **.**scanread **(**1'b0**),**

**.**scanwrite **(**1'b0**),** **.**sclkout0 **(),** **.**sclkout1 **(),**

**.**vcooverrange **(),** **.**vcounderrange **());**

**defparam**

altpll\_component**.**bandwidth\_type **=** "AUTO"**,**

altpll\_component**.**clk0\_divide\_by **=** 25000**,**

altpll\_component**.**clk0\_duty\_cycle **=** 50**,**

altpll\_component**.**clk0\_multiply\_by **=** 1**,**

altpll\_component**.**clk0\_phase\_shift **=** "0"**,**

altpll\_component**.**compensate\_clock **=** "CLK0"**,**

altpll\_component**.**inclk0\_input\_frequency **=** 20000**,**

altpll\_component**.**intended\_device\_family **=** "Cyclone IV E"**,**

altpll\_component**.**lpm\_hint **=** "CBX\_MODULE\_PREFIX=lab1clk"**,**

altpll\_component**.**lpm\_type **=** "altpll"**,**

altpll\_component**.**operation\_mode **=** "NORMAL"**,**

altpll\_component**.**pll\_type **=** "AUTO"**,**

altpll\_component**.**port\_activeclock **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_areset **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkbad0 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkbad1 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkloss **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkswitch **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_configupdate **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_fbin **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_inclk0 **=** "PORT\_USED"**,**

altpll\_component**.**port\_inclk1 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_locked **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_pfdena **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_phasecounterselect **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_phasedone **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_phasestep **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_phaseupdown **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_pllena **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_scanaclr **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_scanclk **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_scanclkena **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_scandata **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_scandataout **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_scandone **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_scanread **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_scanwrite **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clk0 **=** "PORT\_USED"**,**

altpll\_component**.**port\_clk1 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clk2 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clk3 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clk4 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clk5 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkena0 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkena1 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkena2 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkena3 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkena4 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_clkena5 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_extclk0 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_extclk1 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_extclk2 **=** "PORT\_UNUSED"**,**

altpll\_component**.**port\_extclk3 **=** "PORT\_UNUSED"**,**

altpll\_component**.**width\_clock **=** 5**;**

**endmodule**

## Colseq

// colseq.sv - ELEX 7660 – Sequentially pulls each column of the keypad low until a matching keypress is detected.

// Nicholas Huttemann 2018-01-22

**module** colseq **(input** **logic** **[**3**:**0**]** kpr **,**

**input** **logic** clk**,** reset\_n**,**

**output** **logic** **[**3**:**0**]** kpc **=** 'b0111**);**

// The four states corresponding to each column

**int** states **[**4**]** **=** '**{**'b0111**,** 'b1011**,** 'b1101**,** 'b1110**};**

**logic** **[**1**:**0**]** state **=** 0**;**

**logic** hold**;**

**always\_comb**

**if** **(**kpr **==** 'b1111**)** // If there was a keypress, hold

hold **<=** 0**;**

**else**

hold **<=** 1**;**

**always\_ff** **@** **(posedge** clk**,** **negedge** reset\_n**)** **begin**

**if** **(~**reset\_n**)**

kpc **<=** 'b0111**;**

**else**

**if** **(~**hold**)** **begin** // If not holding, cycle through the columns

kpc **=** states**[**state**];**

**if** **(**state **==** 3**)**

state **=** 0**;**

**else**

state**++;**

**end**

**end**

**endmodule**

## Kpdecode

// kpdecode.sv - ELEX 7660 – Given the detected row ('kpr') and column ('kpc') location of a press on the keypad,

// set 'num' to the decoded key value and drive the 7-segment with 'kphit'.

// Nicholas Huttemann 2018-01-25

**module** kpdecode **(input** **logic** **[**3**:**0**]** kpc**,**

**input** **logic** **[**3**:**0**]** kpr**,**

**output** **logic** **[**3**:**0**]** num**,**

**output** **logic** kphit**);**

// Define a virtual keypad as a 2D array. Changed from int to logic.

**logic** **[**3**:**0**]** keypad **[**4**][**4**]** **=** '**{**'**{**1**,**2**,**3**,**10**},**

'**{**4**,**5**,**6**,**11**},**

'**{**7**,**8**,**9**,**12**},**

'**{**14**,**0**,**15**,**13**}};**

**logic** **[**1**:**0**]** row**;**

**logic** **[**1**:**0**]** col**;**

**always\_comb** **begin**

**if** **(**kpr **!=** 'b1111**)** **begin** // A key has been pressed, determine its value

kphit **=** 1**;**

**case** **(**kpc**)** // Which column was detected?

'b0111 **:** col **=** 0**;**

'b1011 **:** col **=** 1**;**

'b1101 **:** col **=** 2**;**

'b1110 **:** col **=** 3**;**

**default:** col **=** 0**;**

**endcase**

**case** **(**kpr**)** // Which row was detected?

'b0111 **:** row **=** 0**;**

'b1011 **:** row **=** 1**;**

'b1101 **:** row **=** 2**;**

'b1110 **:** row **=** 3**;**

**default:** row **=** 0**;**

**endcase**

**end**

**else** **begin** // No key was pressed

kphit **=** 0**;**

row **=** 0**;**

col **=** 0**;**

**end**

num **=** keypad**[**row**][**col**];** // The decoded value is located in 'keypad' at the detected row and column index.

**end**

**endmodule**

## Decode7

// decode7.sv - ELEX 7660 – Converts ‘num’, an integer 0 – 15, into an 8-bit vector ‘leds’ that will light up corresponding segments on a 7-segment LED.

// Nicholas Huttemann 2018-01-22

**module** decode7 **(** **input** **logic** **[**3**:**0**]** num**,**

**output** **logic** **[**7**:**0**]** leds**);**

**always\_comb**

**case(**num**)**

0 **:** leds **=** 8'b11000000**;**

1 **:** leds **=** 8'b11111001**;**

2 **:** leds **=** 8'b10100100**;**

3 **:** leds **=** 8'b10110000**;**

4 **:** leds **=** 8'b10011001**;**

5 **:** leds **=** 8'b10010010**;**

6 **:** leds **=** 8'b10000010**;**

7 **:** leds **=** 8'b11111000**;**

8 **:** leds **=** 8'b10000000**;**

9 **:** leds **=** 8'b10010000**;**

4'ha**:** leds **=** 8'b10001000**;**

4'hb**:** leds **=** 8'b10000011**;**

4'hc**:** leds **=** 8'b11000110**;**

4'hd**:** leds **=** 8'b10100001**;**

4'he**:** leds **=** 8'b10000110**;**

4'hf**:** leds **=** 8'b10001110**;**

**default:** leds **=** 8'b11000000**;**

**endcase**

**endmodule**

## Simulation Results



Figure 1: Simulation Results

# RTL Diagrams

## Top Level

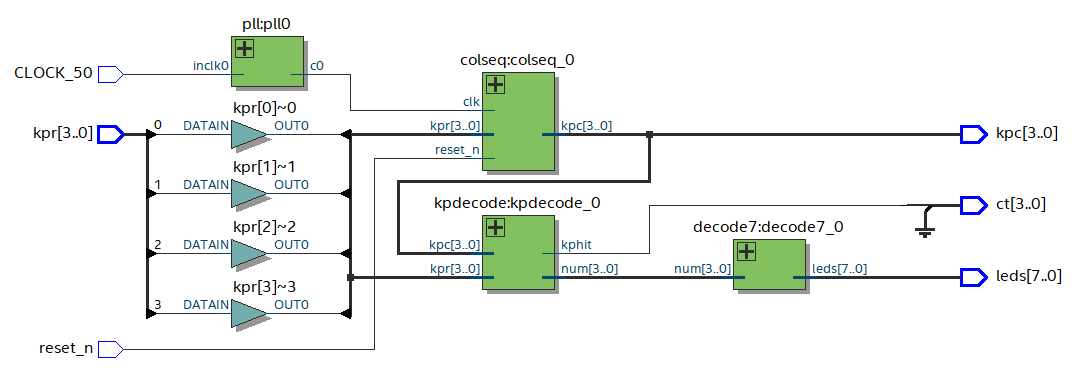


Figure 2: Top level diagram lab2.sv

## Colseq

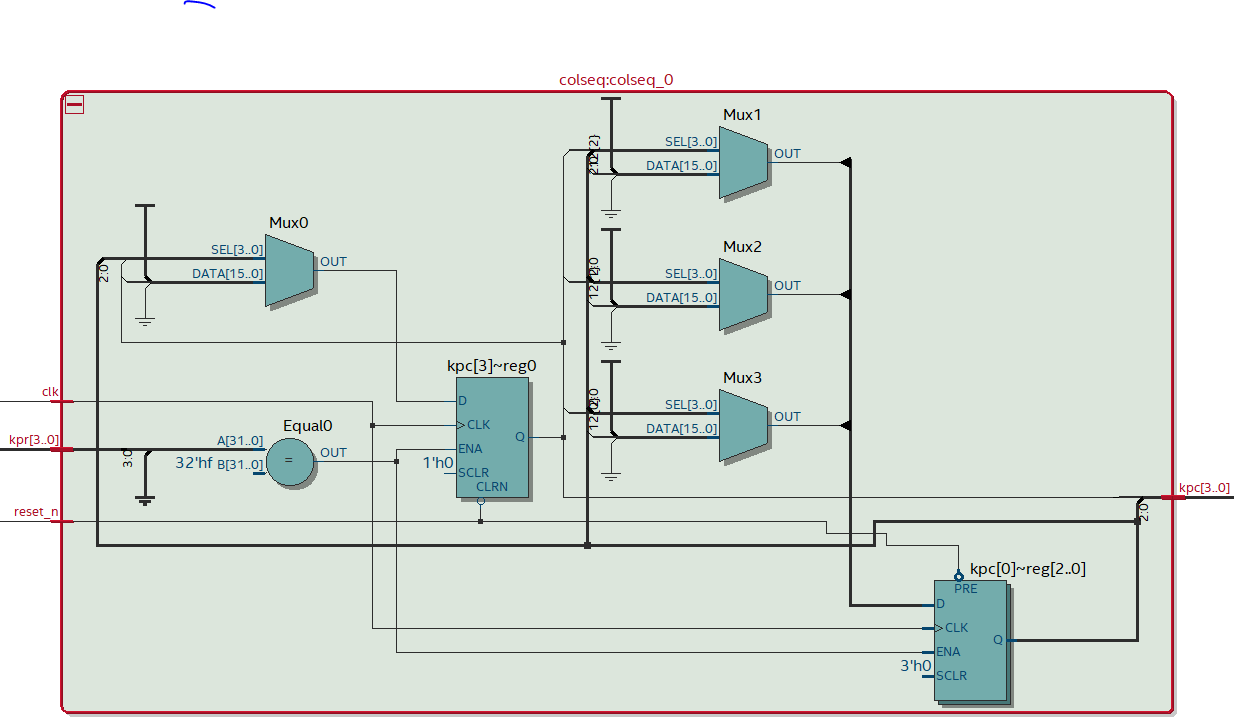


Figure 3: colseq.sv diagram

## Kpdecode

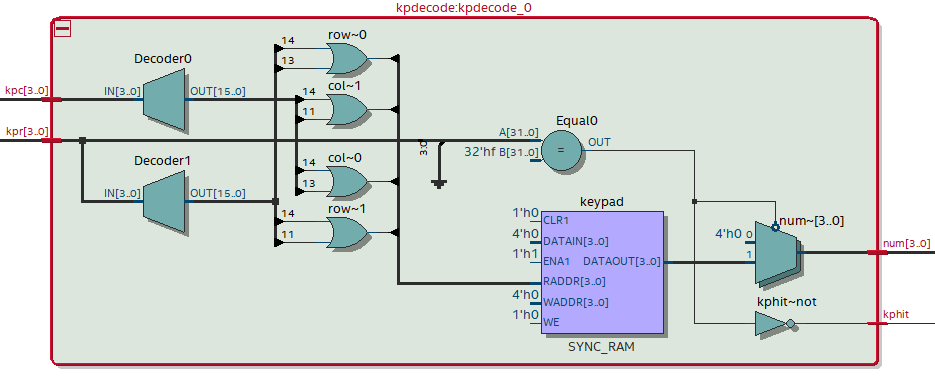


Figure 4: kpdecode.sv diagram

## Decode7

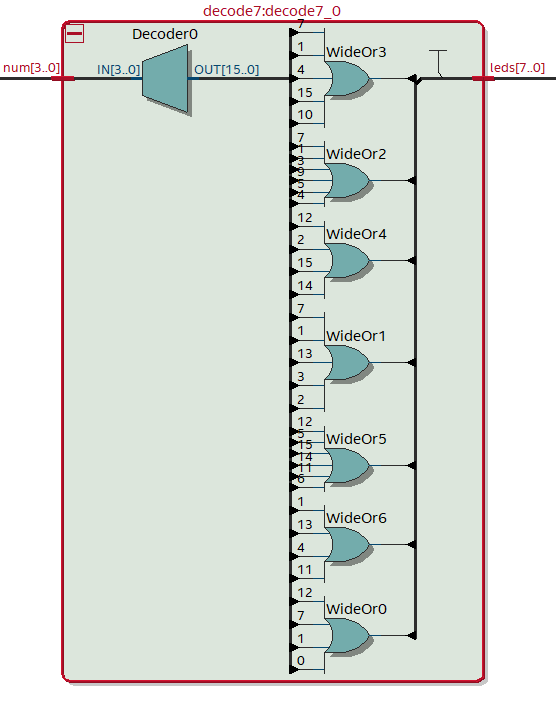


Figure 5: decode7.sv diagram

# Compilation Summary

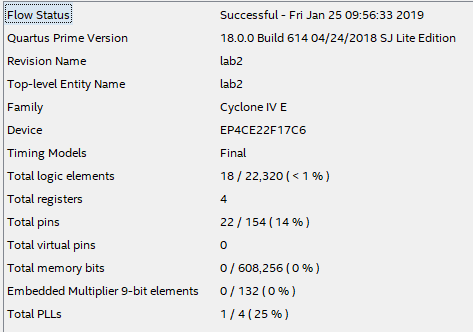


Figure 6: Compilation Summary