

## Laboratory 2

### Lab Objective:

The aims of this laboratory session are:

- (1) To implement a digital piano using Xilinx Virtex-5 Evaluation Platform
- (2) To perform Static Timing Analysis (STA) to look at the delay values
- (3) To perform simulation using a testbench

### Learning Outcomes:

At the end of this lab session, you would have learned the following:

1. Performing STA and understanding reported delay values
2. Looking at critical paths in Technology Viewer in ISE
3. Performing simulation using a testbench in Xilinx ISim
4. Concepts of clock dividing and pulse width modulation

### Laboratory Exercises:

In the exercises in Lab2, you will control 3 LEDs and a speaker on the board using the 7 DIP switches and two push buttons.

Notice the location of the LEDs, DIP switches and push buttons on Fig.1. Number 6 refers to the 8 general purpose DIP switches, number 8 refers to the 5 LEDs and 5 push buttons (E, S, W, N, C). To learn about the pin locations for DIP switches, LEDs and push buttons, see Table 1-5 (page 20), Table 1-6 (page 21) and Table 1-7 (page 22) in UG 347 or the Appendix in Lab 1 manual.

In your lab exercise 2A you will use 7 DIP switches (1 to 7) and drive only one of them high at a time.

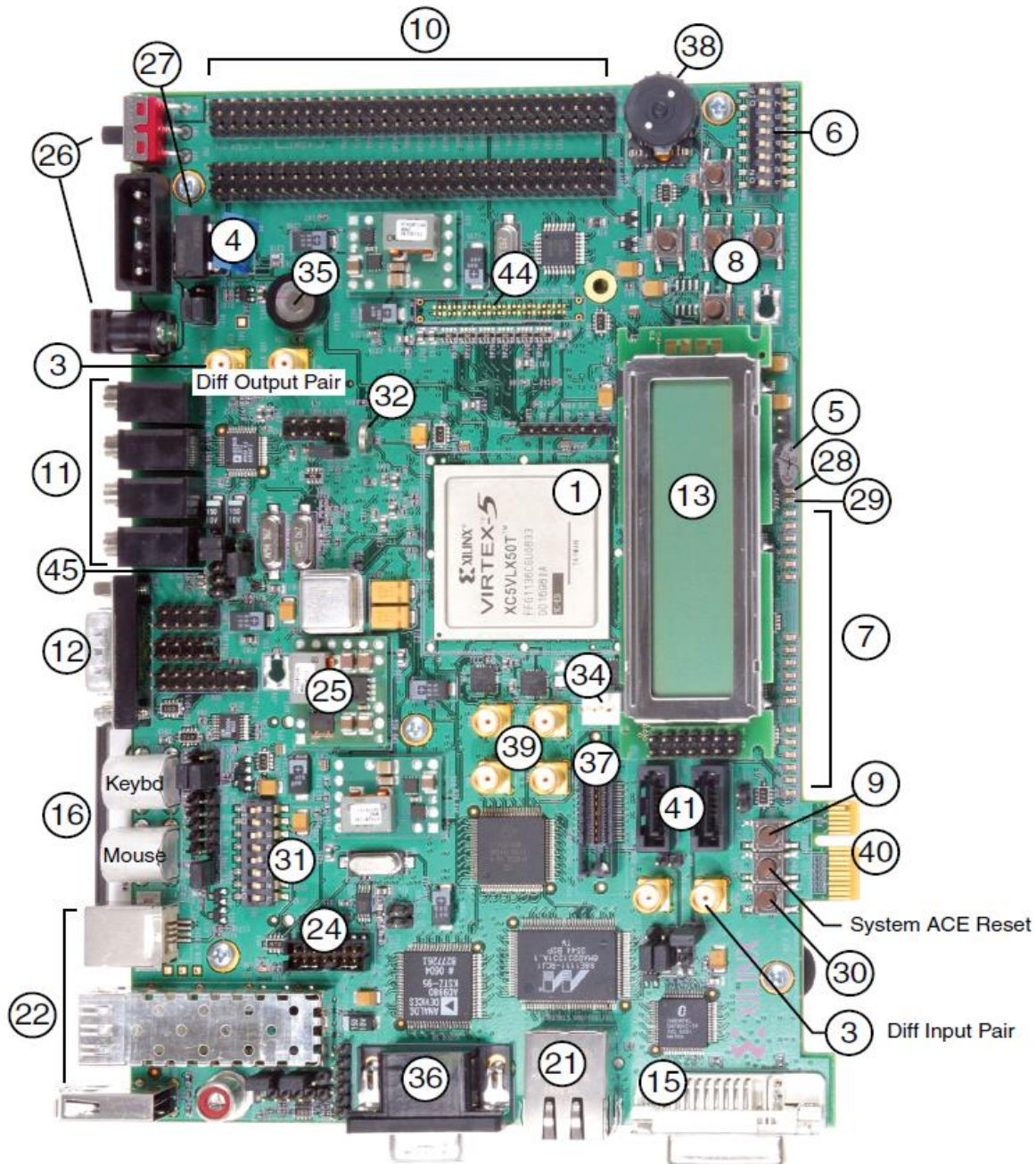


Fig.1: Front part of XUPV5-LX110T Evaluation Platform

## Exercise 2A:

### Piano music notes:

Use Xilinx XUPV5-LX110T to derive the music note of (Do, Re, Mi, Fa, So, La, Ti) in three different frequency ranges (low, high, medium) through the DIP switches 1 -7. Speaker should play corresponding music note when a DIP switch is switched to high. Only one of the seven switches should be set to high at one time.

The default music note frequencies range is **medium** when any switch of 1-7 is set to high. But when **South** push button is pressed and any switch of 1-7 is set to high, a **low** frequency range of music notes should be played. If **North** push button is pressed and any switch of 1-7 is set to high, corresponding **high** frequencies range of music notes should be played. At the same time, the low, medium, high range of frequencies should be denoted by the **South**, **Center** and **North** LED.

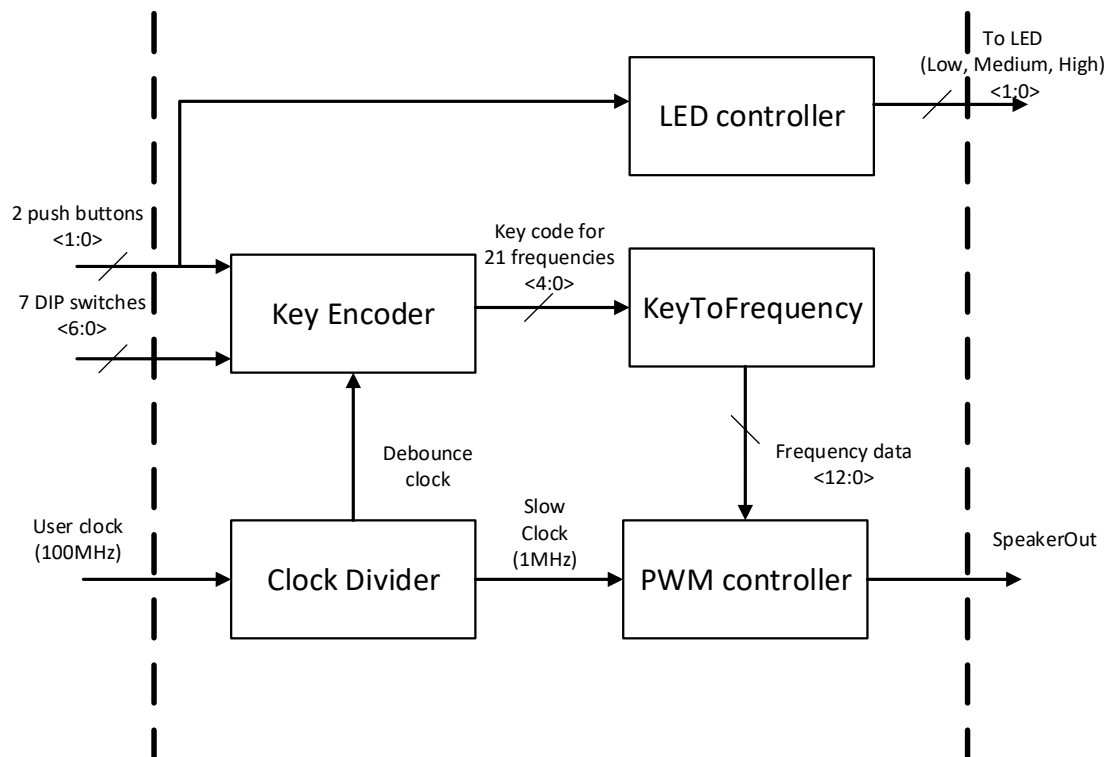


Fig.2 Piano sub-modules to be implemented

Fig. 2 shows the design flow and the sub-modules of the digital piano. Here is a simple module specification describing the function of each sub-module and the connections between them.

**Key Encoder:** Take 7 DIP switches and 2 push buttons as input. Each of 7 DIP switches is a piano key note, “Do Ra Me Fa So La Ti” respectively. The output of this module is a key code number from 0 to 21. 1-7 will correspond to 7 music note at low frequency, 8-14 at medium frequency and 15-21 represents high frequency music notes. 2 push buttons are used to specify the frequency range, one for low frequency and the other one for high frequency. Default frequency range is medium (None of the buttons pushed). If both buttons are pushed, the module will regard the key code number as 0.

**LED controller:** It actually controls 5 LEDs on the board: West, East, South, North and Center. West and East LEDs are always off. South, Center and North LEDs represent for low, medium and high frequency respectively. If both buttons are pushed, all LEDs are off.

**KeyToFrequency:** This will search for the corresponding music note frequency in a hard coded table with the input key code number. The frequency related information will be used to modulate the speaker in PWM controller.

**Clock Divider:** The clock divider will slow down the user clock to 1MHz and feed to the PWM controller.

**PWM controller:** This controller will accept frequency related information and generate corresponding pulse-width modulated signal.

The following is the decimal slow\_rate value of 21 music notes

	Slow_rate(low)	Slow_rate(medium)	Slow_rate(high)
Do	1909	956	478
Ri	1701	852	426
Mi	1515	759	379
Fa	1433	716	358
So	1276	638	319
La	1137	568	284
Ti	1012	506	253

**After you have done Exercise 2A above, proceed to do exercises 2B and 2C.**

#### **Exercise 2A:**

##### **Steps:**

1. Create a Xilinx ISE project as you did in Lab 1.
2. When adding all the source and testbench files, select the association of files as shown in Fig. 3. Modify the source files (key\_encoder.v, keytofrequency.v, clk\_divider.v) by the tips in comments.
3. Repeat the steps for behavioral simulation as you did in Lab 1.
4. In the Isim window, type **run all** at Isim prompt i.e. **Isim > run all**
  - a. You will see the simulation run, print some messages in the console and then stop after some time.
  - b. Verify the waveform to see if **key\_input\_w** is changing or not as shown in Fig. 4

- c. Verify if the **speaker\_out\_w** is changing or not as shown in Fig. 4  
 (click on the arrow in the red boxes to find out the instance when these signals transition)

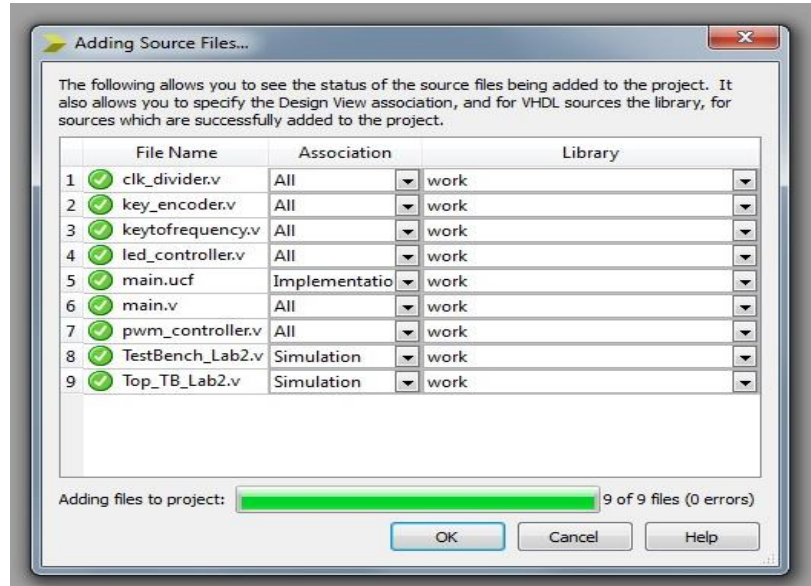


Fig. 3: Associating testbench files with Simulation

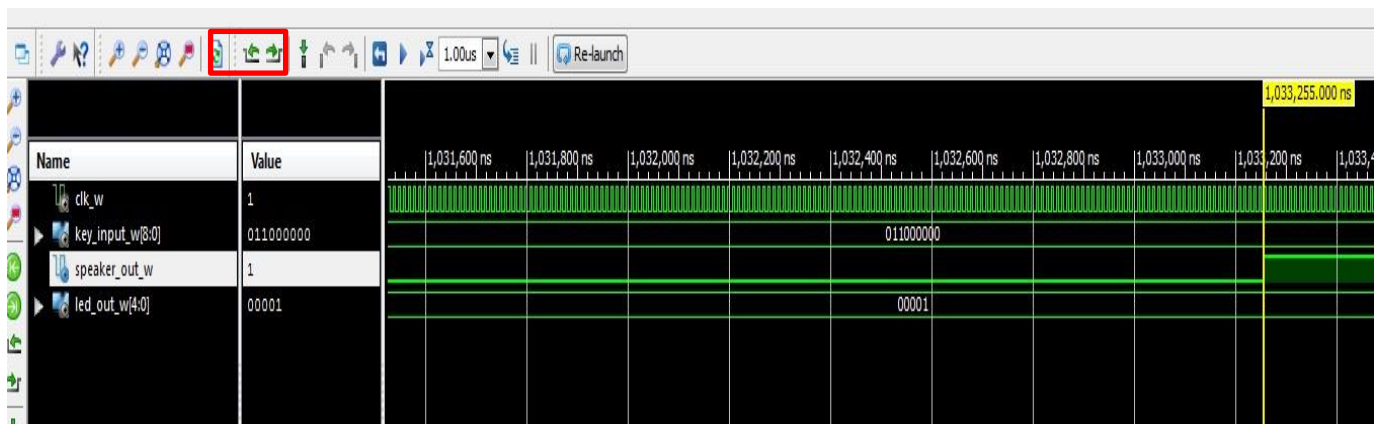


Fig.4: Observing the change in speaker\_out\_w and key\_input\_w signals

5. Close the simulation and go back to **implementation** view.
6. **Implement** the design as you did in Lab 1.
7. You will now learn to do Static Timing Analysis (STA) using Timing Analyzer. Go to **Tools -> Timing Analyzer** and click on **Post-Place & Route** as shown in Fig. 5
8. Click **OK** on Timing Report Tips pop-up window.
  - a. You will see a Timing Report as shown in Fig. 6
  - b. Click on timing summary in "**Report Navigation**" and note the "**score**" as shown in Fig. 7.



- c. A “score” of zero means that there is no timing error and your design met the 10 ns (100 MHz) clock period constraint

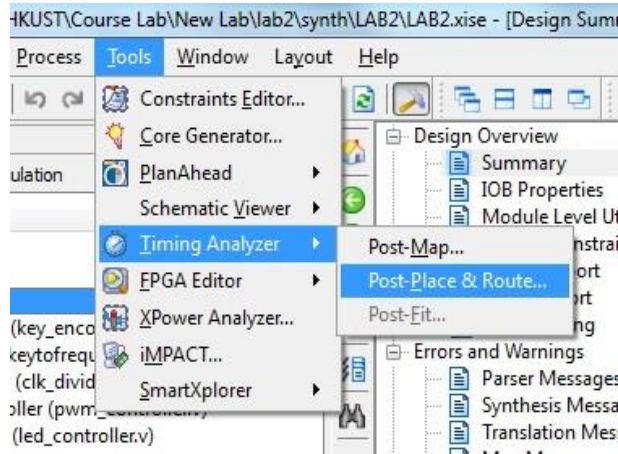


Fig. 5: Performing Static Timing Analysis-1

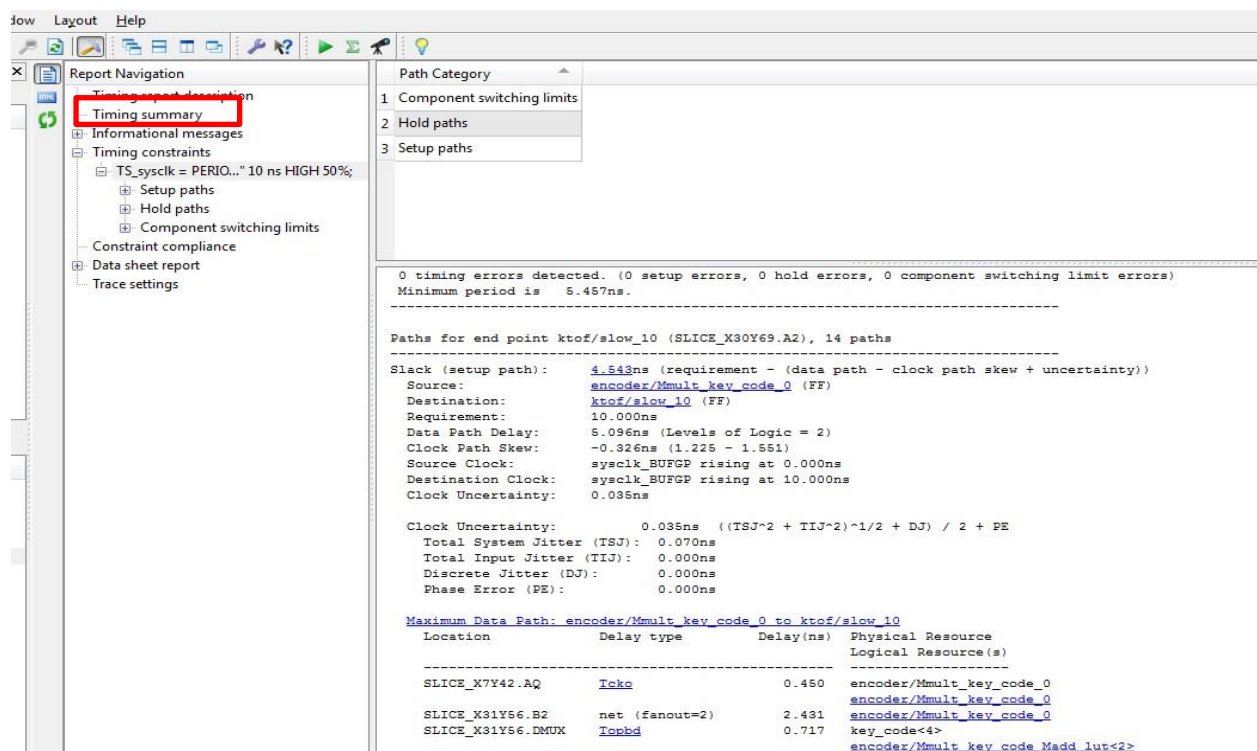
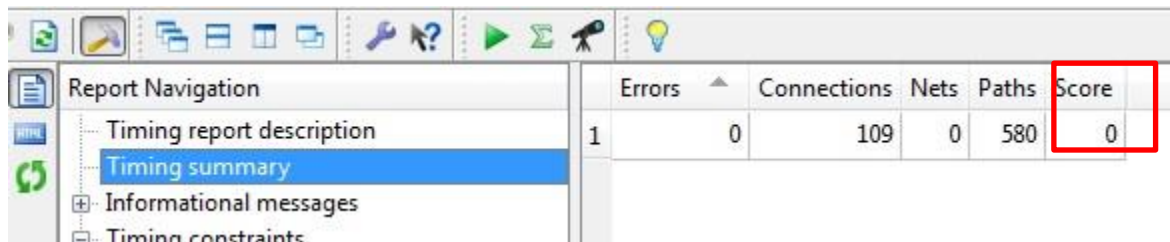


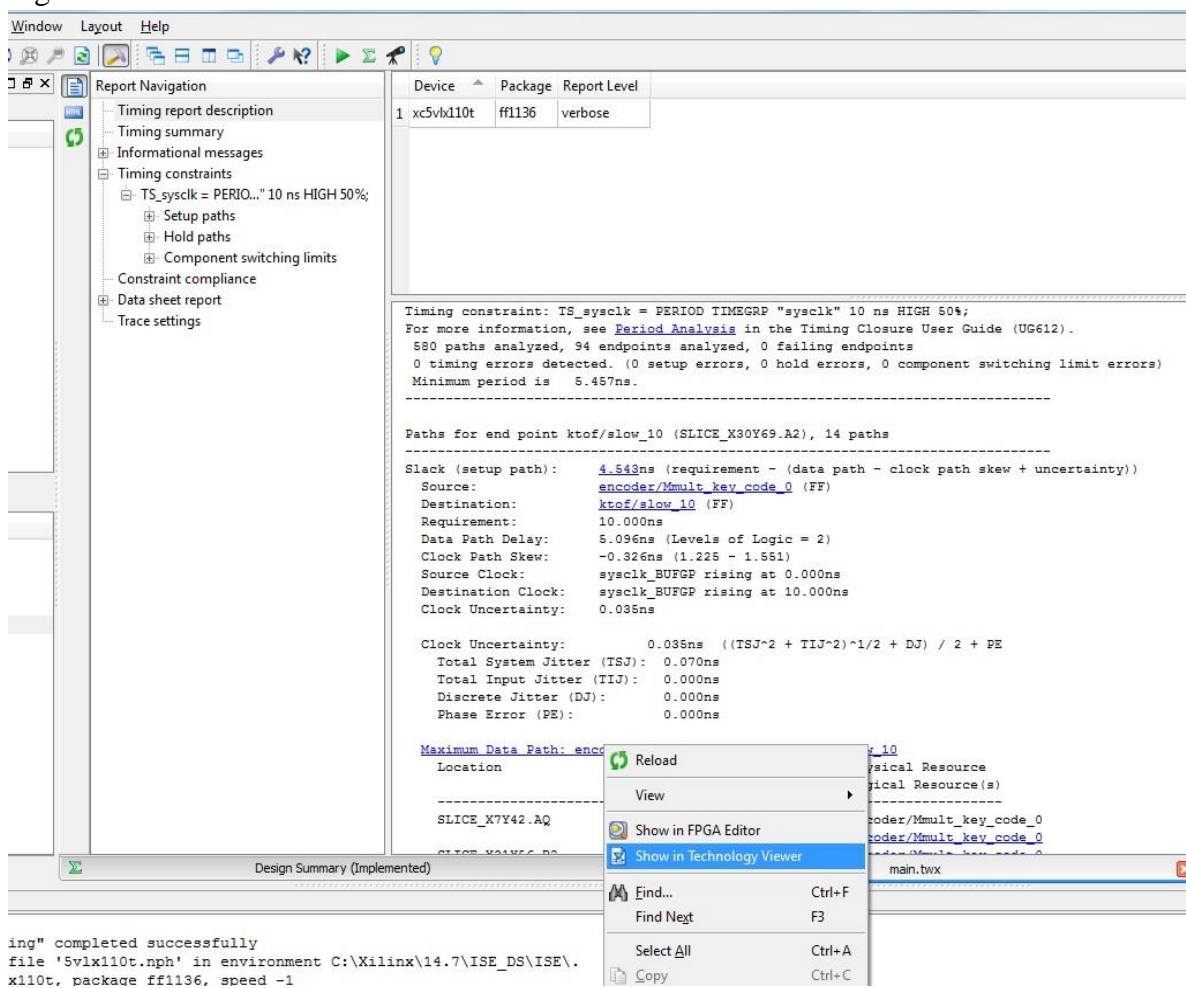
Fig. 6: Static Timing Analysis report



Errors	Connections	Nets	Paths	Score
1	0	109	0	580
				0

Fig. 7: Timing summary and score

- Click on a path highlighted in blue in the timing report and see the options as shown in Fig. 8.



Window Layout Help

Report Navigation

- Timing report description
- Timing summary
- Informational messages
- Timing constraints
  - TS\_sysclk = PERIOD TIMEGRP "sysclk" 10 ns HIGH 50%;
    - Setup paths
    - Hold paths
    - Component switching limits
- Constraint compliance
- Data sheet report
- Trace settings

Device Package Report Level

1 xc5vkl10t ffl136 verbose

Timing constraint: TS\_sysclk = PERIOD TIMEGRP "sysclk" 10 ns HIGH 50%;  
 For more information, see [Period Analysis](#) in the Timing Closure User Guide (UG612).  
 580 paths analyzed, 94 endpoints analyzed, 0 failing endpoints  
 0 timing errors detected. (0 setup errors, 0 hold errors, 0 component switching limit errors)  
 Minimum period is 5.457ns.

Paths for end point ktot/slow\_10 (SLICE\_X30Y69.A2), 14 paths

Slack (setup path): 4.543ns (requirement - (data path - clock path skew + uncertainty))

Source: encoder/Mmult\_key\_code\_0 (FF)  
 Destination: ktot/slow\_10 (FF)  
 Requirement: 10.000ns  
 Data Path Delay: 5.096ns (Levels of Logic = 2)  
 Clock Path Skew: -0.326ns (1.225 - 1.551)  
 Source Clock: sysclk\_BUFGR rising at 0.000ns  
 Destination Clock: sysclk\_BUFGR rising at 10.000ns  
 Clock Uncertainty: 0.035ns

Clock Uncertainty: 0.035ns ((TSJ<sup>2</sup> + TIJ<sup>2</sup>)<sup>1/2</sup> + DJ) / 2 + PE  
 Total System Jitter (TSJ): 0.070ns  
 Total Input Jitter (TIJ): 0.000ns  
 Discrete Jitter (DJ): 0.000ns  
 Phase Error (PE): 0.000ns

Maximum Data Path: encoder/Mmult\_key\_code\_0

Location: SLICE\_X7Y42.A2

Design Summary (Implemented)

ing" completed successfully  
 file '5vxl10t.nph' in environment C:\Xilinx\14.7\ISE\_DS\ISE\  
 x110t, package ffl136, speed -1

Reload View Show in FPGA Editor Show in Technology Viewer Find... Find Next Select All Copy

Fig. 8: Options to see a timing path

- Click on **Show in Technology Viewer** to see the selected timing path. You will see something similar to Fig. 9. The red lines show the selected path after implementation.



Fig. 9: A selected path shown in Technology Viewer

11. Go to **Tools -> FPGA Editor -> Post-Place and Route**. This will open up FPGA Editor.
12. Go to timing report, and select **Show in FPGA Editor** in Fig. 8. You will see something similar to Fig. 10 which shows the selected path (in red) on the actual FPGA. Play with the zoom buttons in the tool bar at the top to see more details.

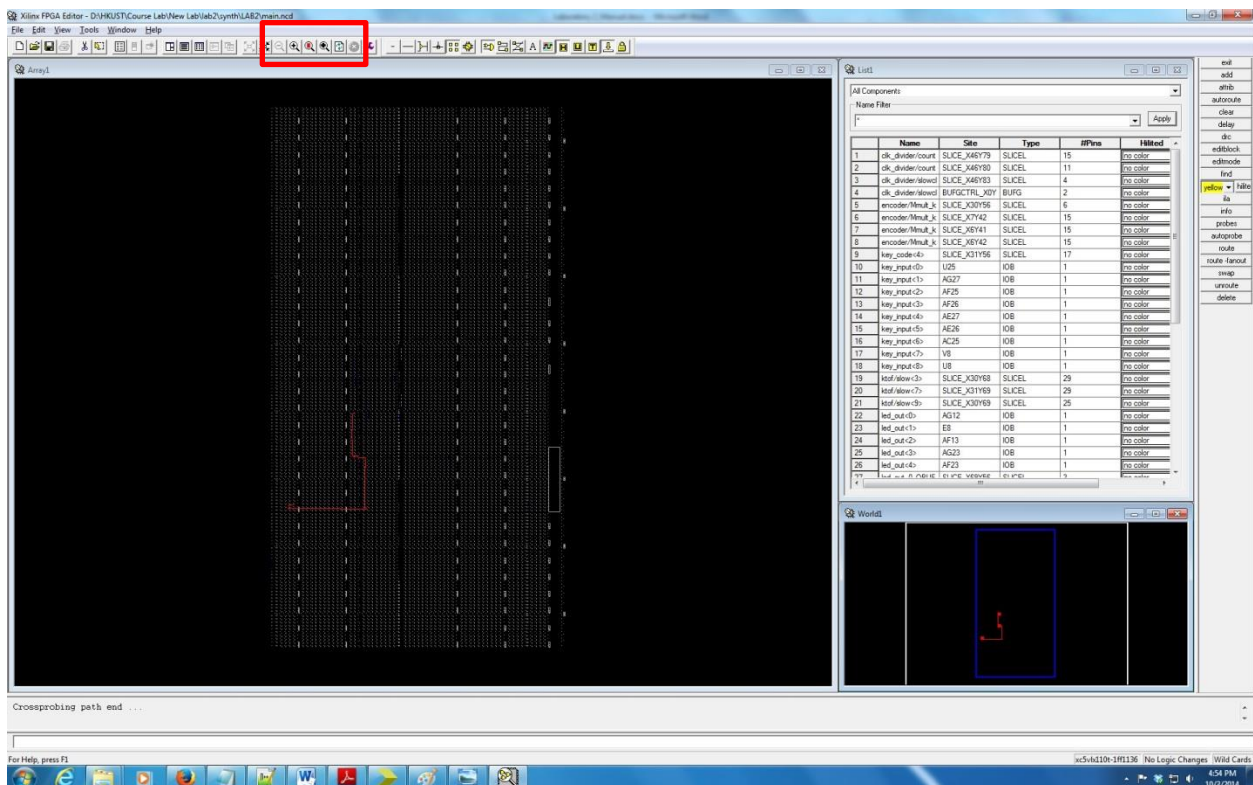


Fig.10: Seeing a timing path in FPGA editor



13. Close FPGA Editor

14. Generate post place and route simulation model and perform post route simulation as shown in Fig. 11. In the Isim window, type **run all** at Isim prompt i.e. **Isim > run all**

- You will see the simulation run, print some messages in the console and then stop after some time.
- Verify the waveform to see if **key\_input\_w** is changing or not.
- Verify if the **speaker\_out\_w** is changing or not.

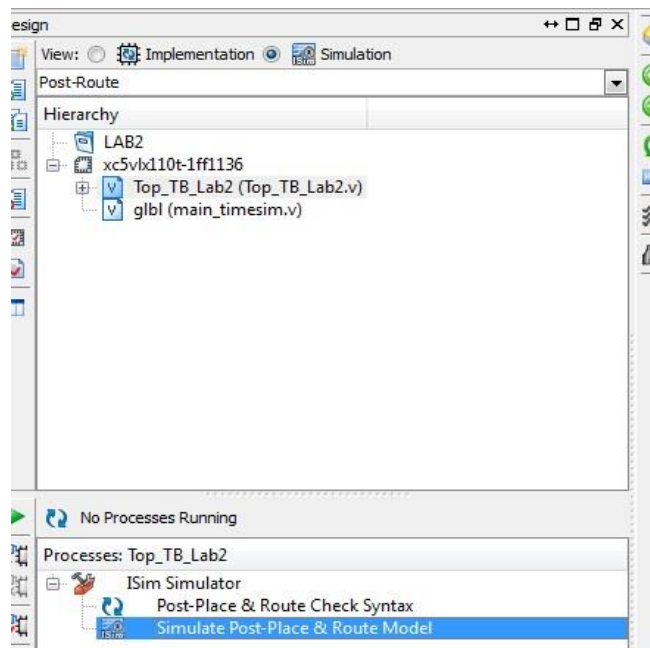


Fig. 11: Post place and route timing simulation

15. Quite post place and route simulation after verifying the waveforms

16. Generate the programming file and program the FPGA and demonstrate to TA.

### Exercise 2B:

- Modify the UCF file by changing the line "TIMESPEC TS\_sysclk = PERIOD "sysclk" 10 ns HIGH 50%" to set a tighter clock period constraint. Change the clock period to 2 ns instead of 10 ns.
  - Did your design meet timing?
  - Can you perform STA and report on the timing score?
  - Can you show the failing paths? Can you show a failing path in technology viewer?
  - What type of timing error is reported (set up or hold)?
  - Can you fix this error by only changing the clock period constraint?

- (2) Modify the TestBench\_Lab.2 file to provide more inputs to your design. Repeat behavioral simulation and show to TA.

### Exercise 2C:

You are required to store frequency information in BRAM instead of hard coding them in Verilog, and achieve the same digital piano functionality. The BRAM will function as a random access memory (RAM) and you will initialize it with the frequency values after programming the code in FPGA. In your design, you will instantiate this BRAM in the *keytofrequency* module and feed the key code to the address bus as a result of which the frequency value will be read out.

Hints:

1. Replace all codes in *keytofrequency.v* with *ram\_keytofrequency.txt*.
2. Create RAM IP: New Source->IP (Core Generator & Architecture Wizard)->Memories & Storage Elements->RAMs & ROMs->Block Memory Generator. Then try to generate a RAM IP. Make sure the name of this RAM module is the same as that in *ram\_keytofrequency*.
3. Fill in the blank part to finish the function. Run on board and demonstrate to TA.