數位電路實驗
Lab 1
亂數產生器

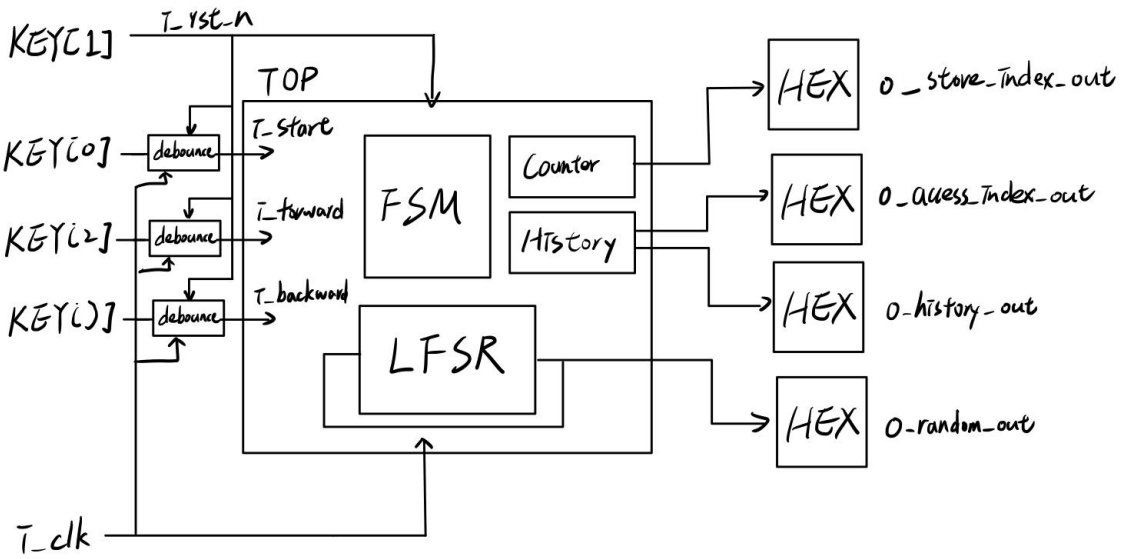
Team 02

金家逸 B10502076
王維勤 B10502010
林桓鈺 B10502013

1. Introduction:

In this experiment, our team implemented a simple random number generator with the added functionality of storing the results of each random number generated. Additionally, we enabled the display of the stored random numbers based on the order using forward and backward buttons.

2. System Architecture:



(1) Debounce.sv :

Stabilize key press glitch, and provide 1-clock-pulse keydown/keyup signal.

(2) SevenHexDecoder.sv :

To display a number on DE2-115, using 7-segment displays where each of the 7 segments is controlled by a 0 or 1 to determine whether it should be lit (0 for lit and 1 for unlit). If nothing is displayed, it would be represented as 7'b1111111.

(3) Top.sv :

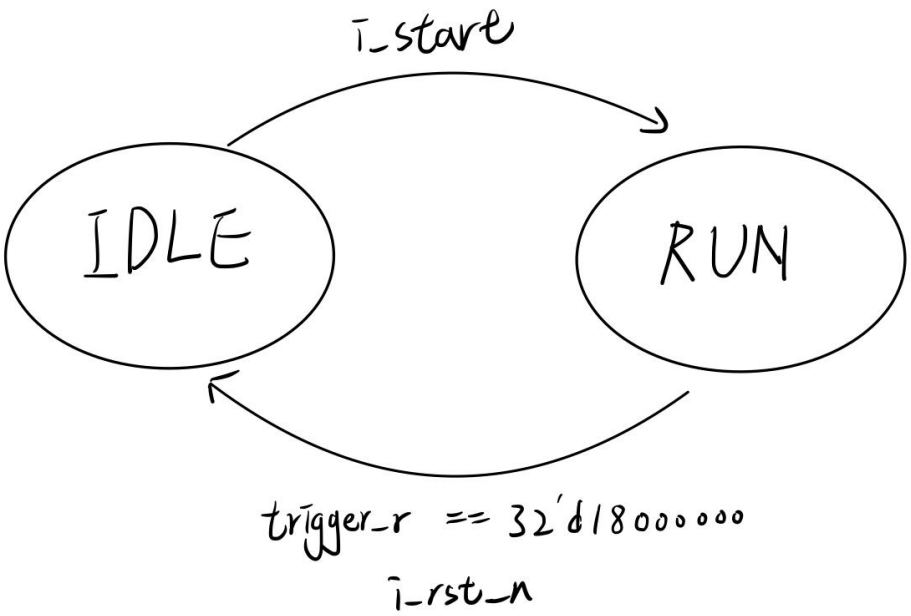
This is the main design module where the Finite State Machine (FSM) is defined and the functionality for storing random number results is implemented.

(a) LFSR : Implementing random number functionality.

(b) History : Storing previous random number results, with a capacity of storing up to eight sets, and concurrently displaying the current result set number.

(c) Counter : Calculating the current number of generated random number sets.

3. Finite state machine:



(1) IDLE:

The initial state, it awaits user input. Press 'i_start' to initiate random number generation and enter the 'RUN' mode. Alternatively, use 'forward' and 'backward' to browse the random number records (history).

(2) RUN:

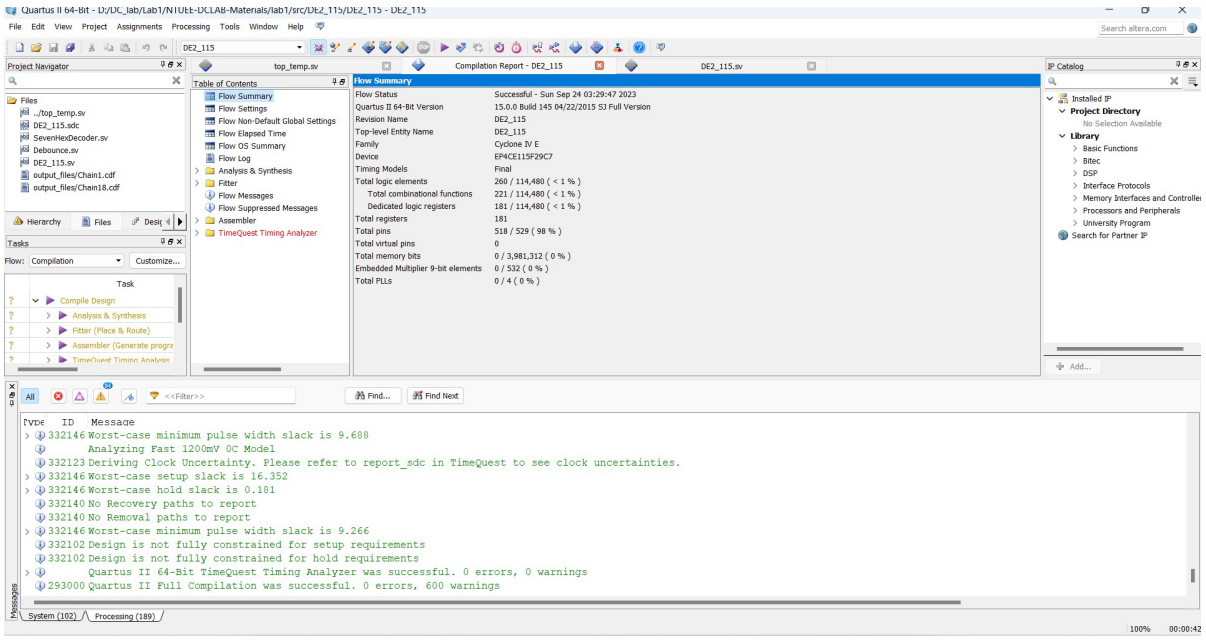
This is the random number generation process. When 'trigger_r' reaches the value of 32'd18000000, it generates the result and returns to the 'IDLE' state. Alternatively, you can interrupt the process and return to 'IDLE' through the asynchronous 'i_rst_n' signal."

4. File Structure:

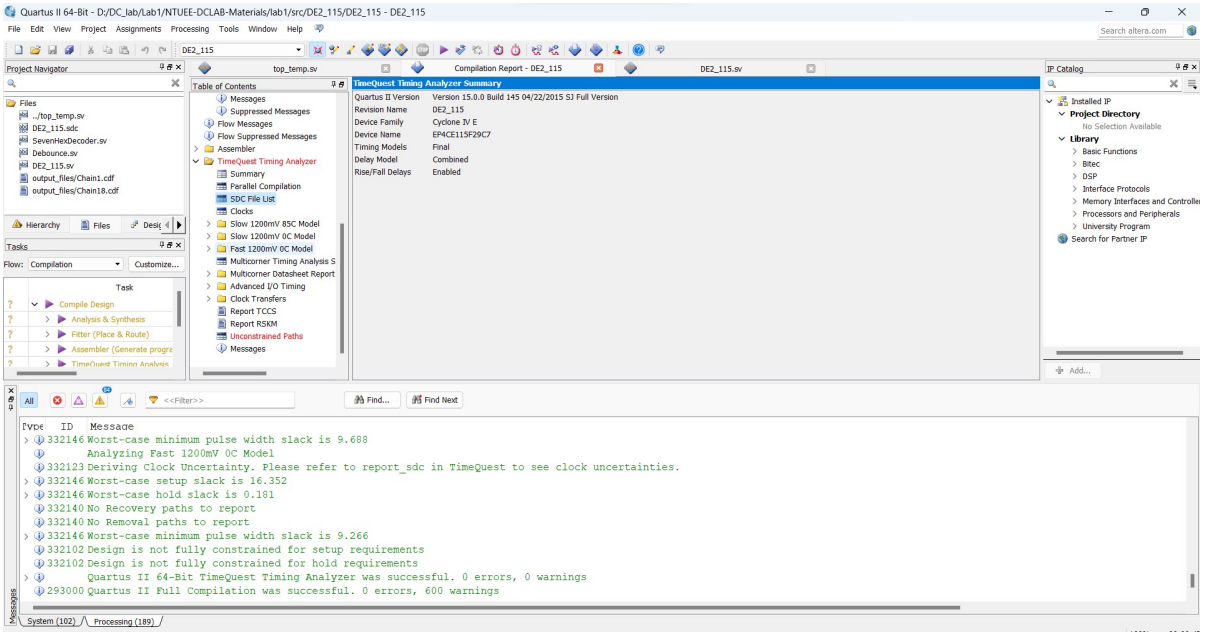
```
team02_lab1/
└── team02_lab1_report/
└── src/
    ├── Top.sv
    └── DE2_115/
        ├── DE2_115.sv
        ├── DE2_115.qsf
        ├── DE2_115.sdc
        ├── SevenHexDecoder.sv
        └── Debounce.sv
```

5. Screenshot:

(1) Fitter Summary



(2) Timing Analyzer



6. Problems Encountered and Solutions:

The initial challenges we faced included our unfamiliarity with Verilog syntax. Before the semester began, we had to learn the basics of Verilog syntax on a website called HDLbit. During the implementation of lab1, we spent a significant amount of time struggling due to our lack of understanding of debugging tools. This resulted in extensive debugging efforts later, where we had to meticulously debug line by line and burn the code onto the board each time to verify if it was running correctly and if any improvements were needed.

Regarding team collaboration, since lab1 only had one top module, it was challenging to divide responsibilities effectively among team members. As a result, we could only work on the experiment when all three of us were available, rather than utilizing individual time effectively. Although team members could discuss issues with each other, it was not possible to plan the experiment's progress more efficiently.

In conclusion, our lack of familiarity with the tools consumed a lot of time for us to adapt and become proficient. The lack of clear division of tasks in our teamwork resulted in delays in our experiment progress. For lab2, if we can become more proficient in using

the tools and establish a more precise division of tasks, we can undoubtedly work more efficiently.