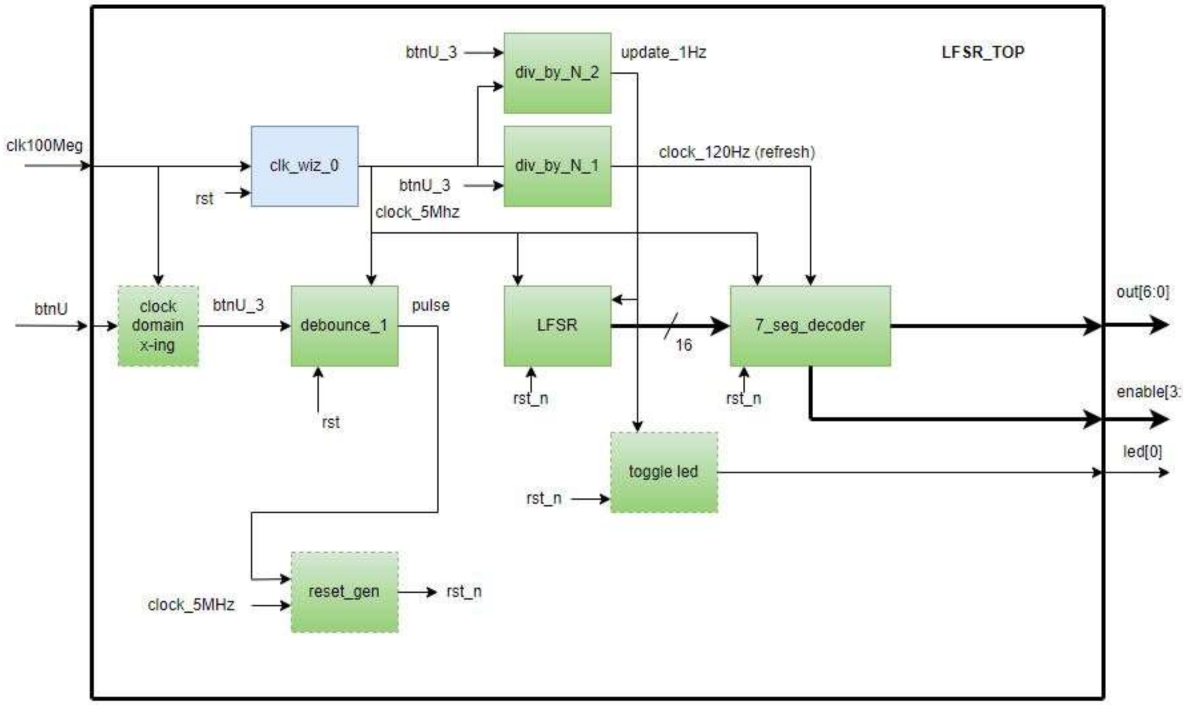
Tutorial on Basys 3 FPGA Project #1
Tremaine Consulting Group 1/29/2023

This tutorial covers my first experience using the Digilent Basys 3 FPGA development board. This board uses an Artix 7 Xilinx chip, specifically the xc7a35tcpg236-1 part. The tools used are Vivado 2022.2 with the WebPACK license.

My first project on this board is diagrammed in figure 1.

Figure 1 – FPGA Project



Target board: Digilent Basys3

What the Project Does

The project displays a pseudo-random sequence on the 4-digit 7-segment display. The pseudo-random sequence is generated by a linear feedback shift register (LFSR). Information on this topic is available at: https://en.wikipedia.org/wiki/Linear-feedback_shift_register#Fibonacci_LFSRs . In particular the LFSR type used in this project is a Fibonacci configuration that cycles through through 2^{m} -1 patterns before repeating, and never outputs 0.

Decisions to be made in the design

One of the first decisions to be made in the design was whether the 4-digit 7-segment display would be used to display decimal or hex values. Using decimal values the largest display value is 9,999 and with hex it is FFFF (or 65,535). This choice determines the size of the LFSR and determines whether the FPGA needs to convert the LFSR register output to hex or decimal digits. I made the arbitrary decision to display hex values. This decision was made to simplify the nibble to hex display conversion. The LFSR chosen was 16-bits.

The 7-segment display has four anodes, one for each digit, and seven cathodes, shared for all digits. I am not using the decimal point segment. Because of this arrangement of the display the digits must be displayed one digit at a time. To give a ‘look’ that it is a continuous display the digits are updated at a refresh rate of 120Hz. The persistence of the eye makes it look as if all digits are illuminated at the same time. In addition the LFSR register is advanced every 1 sec. An LED is also toggled at the LFSR update rate of 1 sec. As a side note, with the LFSR updated every 1 sec the pseudo-random pattern will repeat every 65,535 sec or 18.24 hrs.

Clock Design

Based on previous experience it was decided to use an MMCM IP to generate a 5MHz clock source from the 100MHz pin input. The advantage in using an MMCM is that it routes the clocks in special clock fabric that optimizes the timing to minimize clock skew and delay.

The *refresh* ‘clock’ and *1Hz* ‘clock’ are not really clocks. They are clocked by the 5MHz clock to achieve a synchronized design. The refresh ‘clock’ and 1Hz ‘clock’ are generated in a synchronous divide by N logic using 5MHz as the clock and a parameter to set the divide value. The outputs are synchronized to the 5MHz MMCM clock.

Other Items

The external button, btn_U, is used for a manual active high reset. The button input is synchronized to the 5MHz clock using three cascaded DFF and the output goes to debounce logic ‘*debounce.v*’. After the debounce module the clean pulse then goes to a reset generator, ‘reset_gen.v’ to generate an active low reset pulse for every module.

The design flow was to use the Vivado project mode and work entirely with RTL modules. The IP Generator was used to create an RTL module for the MMCM clock that was instantiated in the top level RTL file. The Verilog modules used, and their hierarchy, are shown in the diagram below:

top_lfsr.v

- *clk_wiz_0.v*
- *div_by_N.v*
- *reset_gen.v*
- *lfsr.v*
- *Seven_Seg_Display_Control.v*
- *debounce.v*
- *toggle_Led.v*

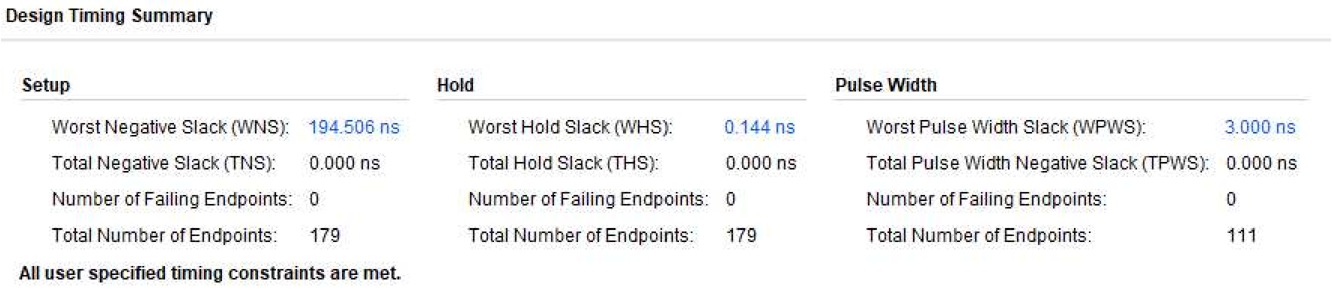
The design was first debugged in the *Vivado 2022.2* Behavioral simulation before running synthesis and implementation. The test bench used was *TB_top_lfsr.v*.

Static Timing

The static timing was studied after the synthesis run. My original implementation of the LFSR was not using a valid clock on the shift register so I changed the design to use a DFF with asynchronous reset and an enable.

The timing summary after the modification is shown in Figure 2.

Figure 2 – Post Implementation Static Timing Report



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	194.506 ns	Worst Hold Slack (WHS):	0.144 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	179	Total Number of Endpoints:	179	Total Number of Endpoints:	111

All user specified timing constraints are met.

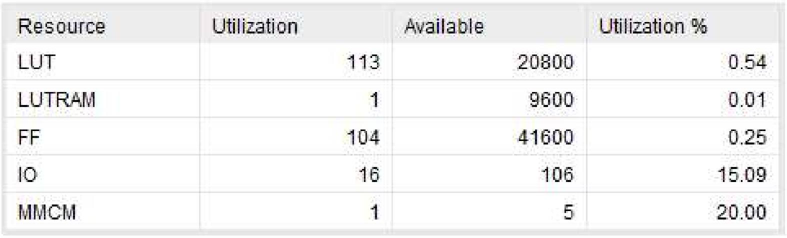
The system clock is 5Mhz (200ns period), so the WNS of +194.506ns represents a 5.494ns shift from ideal. There were no Methodology violations reported and no DRC errors reported.

In the timing check there were 13 warnings but these were as follows:

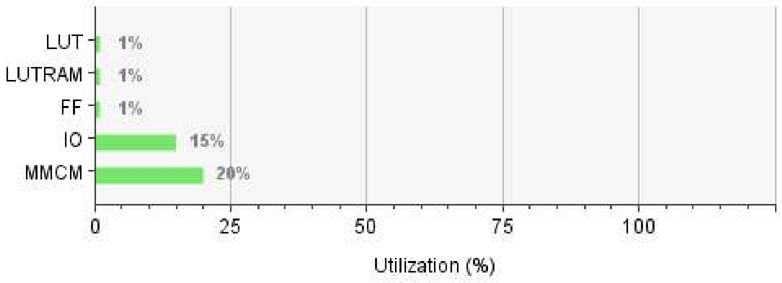
- One warning of no input delay specified on input pin btnU.
- Twelve warnings of no output delay on outputs cath_out[7:0], enable[3:0] and led[0].

The utilization is very low, as expected for this simple project.

Figure 3 – Utilization



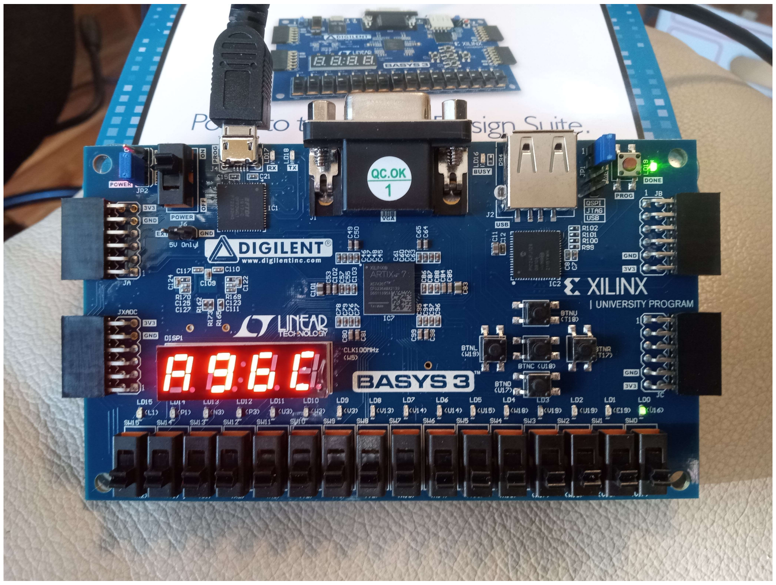
Resource	Utilization	Available	Utilization %
LUT	113	20800	0.54
LUTRAM	1	9600	0.01
FF	104	41600	0.25
IO	16	106	15.09
MMCM	1	5	20.00



Source Code

The Verilog source (Vivado project) code for this project is available at: github

A display of the board is shown here:



The 7-segment display is showing a random hexadecimal number and the green LED in the lower right is on. A video is shown at this link: https://photos.app.goo.gl/t1uitjeTHdBKdg227