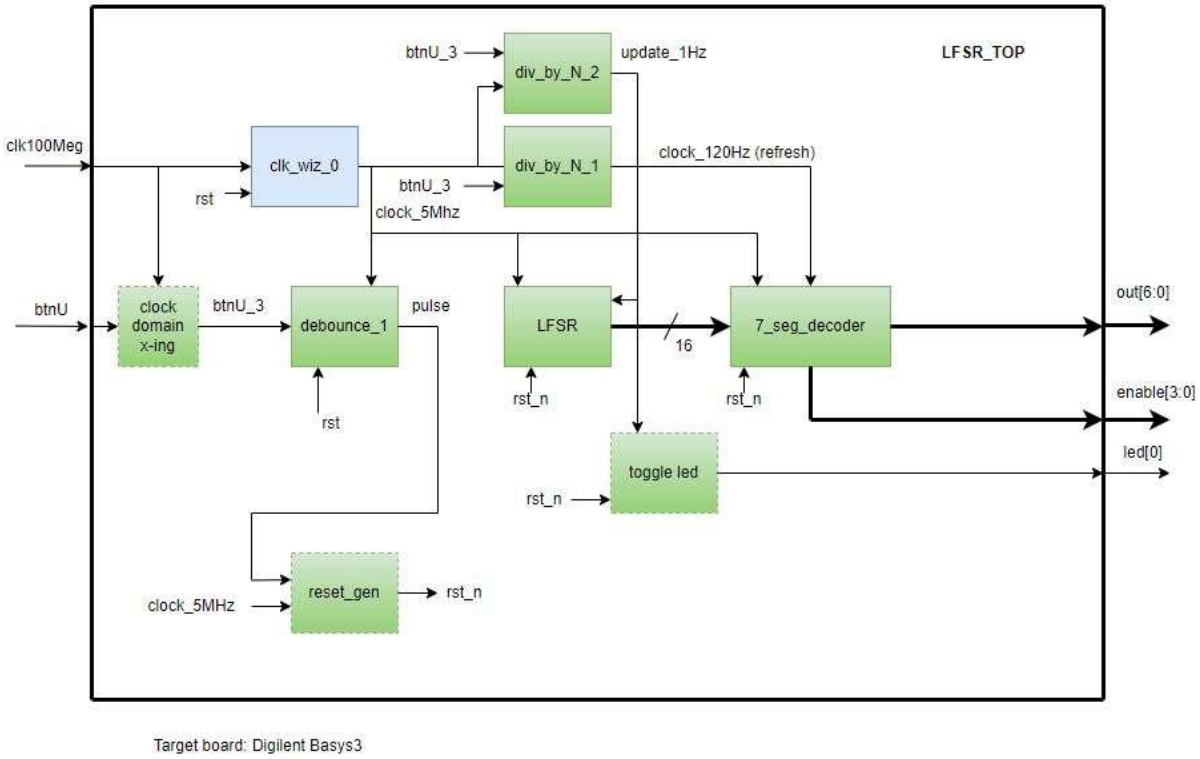
This tutorial revisits my Basys 3 FPGA Project #1 by covering a different **tool-flow**. My first project Vivado 2022.2 and used RTL modules for the entire project. The approach used on this revision is to use Vivado Block Design. Like project #1, this project uses the Digilent Basys 3 FPGA development 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 used a top level RTL module and instantiated all other modules inside the top level module. This is depicted in Figure 1. That project is available on github. https://github.com/Btremaine/Basys_3_LFSR. The repository includes a PDF of the first tutorial.

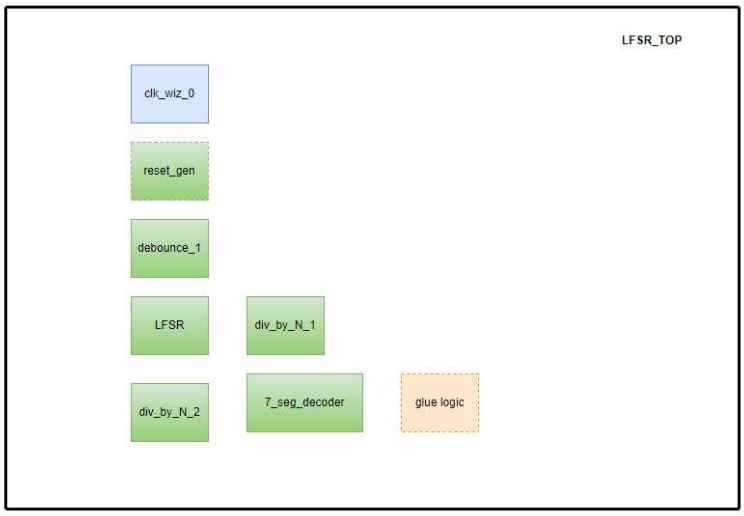
**Figure 1 – FPGA Project #1**




## RTL Approach

Project #2 has the same objectives as project #1 but the tool-flow used is Block Design instead of strictly RTL modules. The RTL approach is shown in figure 2 in the top level module *lfsr_top.v.* Each one of the green blocks shown is an RTL module substantiation. The *clk_wiz_0* is the substantiation of the clock wizard IP. The red block is glue logic implemented in Verilog in the top level module.

**Figure 2 – Project #1 RTL Approach**

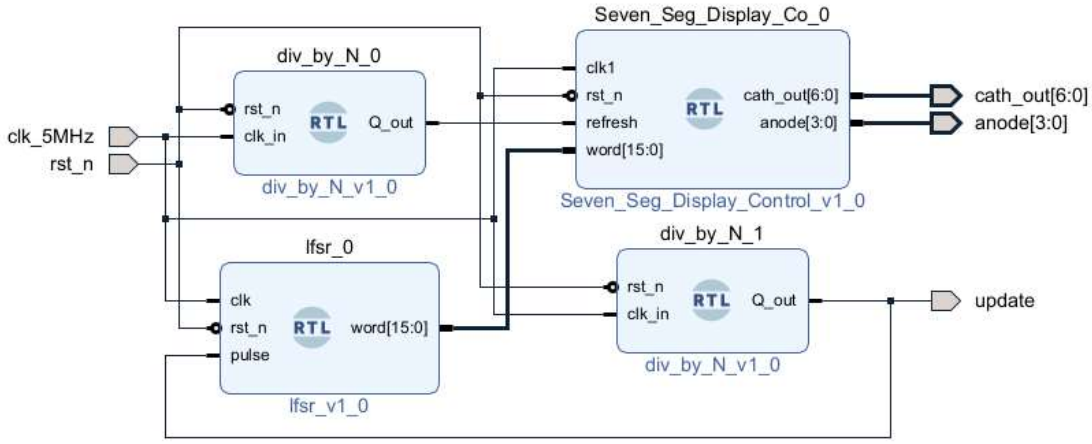



## Block Design

The Vivado project has a tool called Block Design that allows opening a drawing palette that allows creating a design. Under *Project Manager*, choose *IP Integrator → Create Block Design*. Vivado will prompt for the name of the design. The default is *design_1*.

The Block Design palette allows adding IP cores as well as adding RTL modules that have already been added to the project. Figure 3 shows the resulting BD for project #2.

**Figure 3 – Project #2 BD**



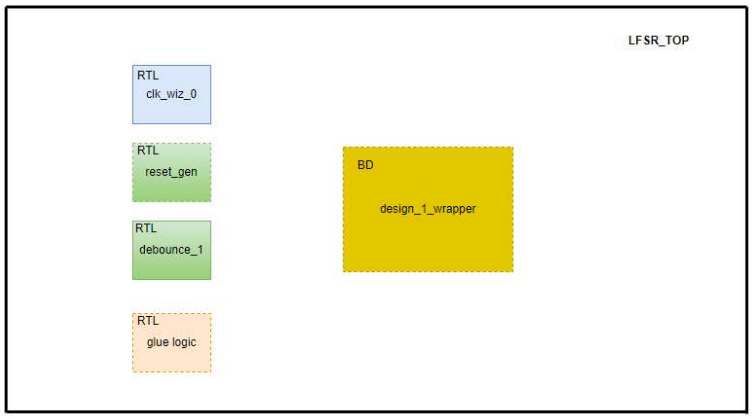
The interconnection of the modules is done within the BD editor. By right clicking on the palette and choosing ‘Add Module’ or ‘Add IP’ the appropriate block can be entered.

If there are parameters in the RTL module or IP core these can be changed by clicking on the module and updating the parameters. Input and output ports are added as shown to the BD.

After the BD is configured the design can be checked by right clicking on the palette and choosing ‘validate design’.

Once the BD is completed a *wrapper* can be generated by right clicking on the design_1.bd file in the sources → Hierarchy tab and choosing ‘*create HDL wrapper*’. This will create a module *design_1_wrapper*, that can be instantiated at the top level as *design_1_wrapper_1*. This is shown in figure 4. The interconnection is done at the top level as you would do any modules.

**Figure 4 – top level using BD**




## Synthesis & Implementation

The use of BD does not effect synthesis or implementation. The results of static timing are the same as before.

**Figure 5 – 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.**

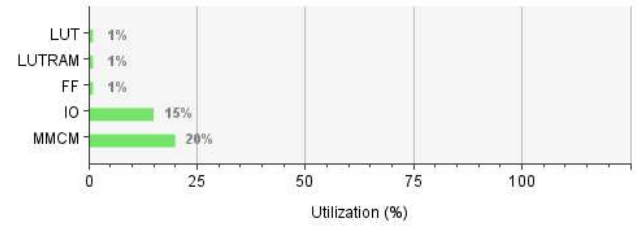
The system clock is 5Mhz (200ns period), so the WNS of +194.506ns represents a 5.494ns shift from ideal. There were no Methodology violations reported and no DRC errors reported.
In the timing check there were 13 warnings but these were as follows:

- 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 6 – 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 |



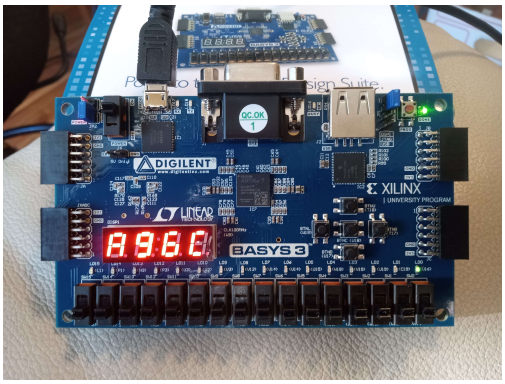

## Conclusion

The use of strictly RTL modules versus Block Design is a personal preference. For me, the BD offers more structure and hides some detail in the BD *objects*. The resulting BD diagram offers some documentation that is not otherwise available.

## Source Code

The Verilog source (Vivado project) code for this project is available at:
https://github.com/Btremaine/Basys_3_LFSR_BD

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