California Polytechnic State University, Pomona

ECE 4304.01
Lab 6

Adrian Alarcon
Isaac Elizarraz
Jacob Swenke

03/24/21

Architecture

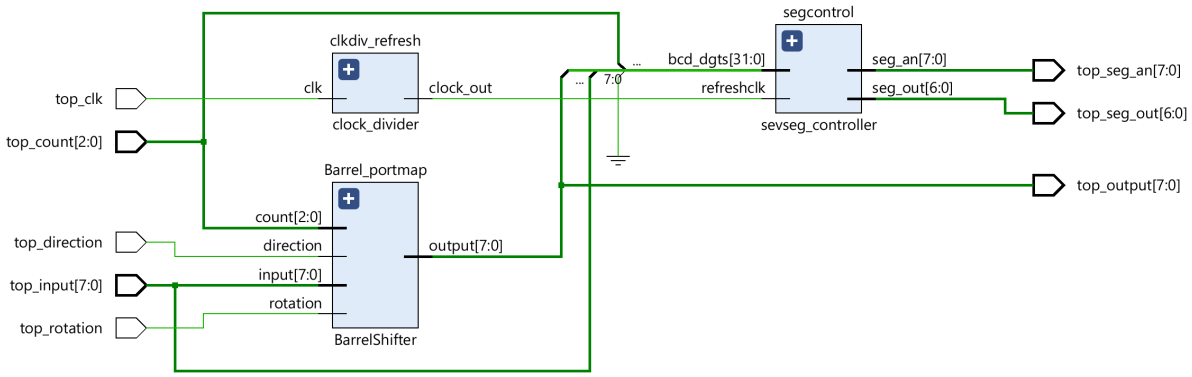


Fig. 6.1 - Elaborated Design

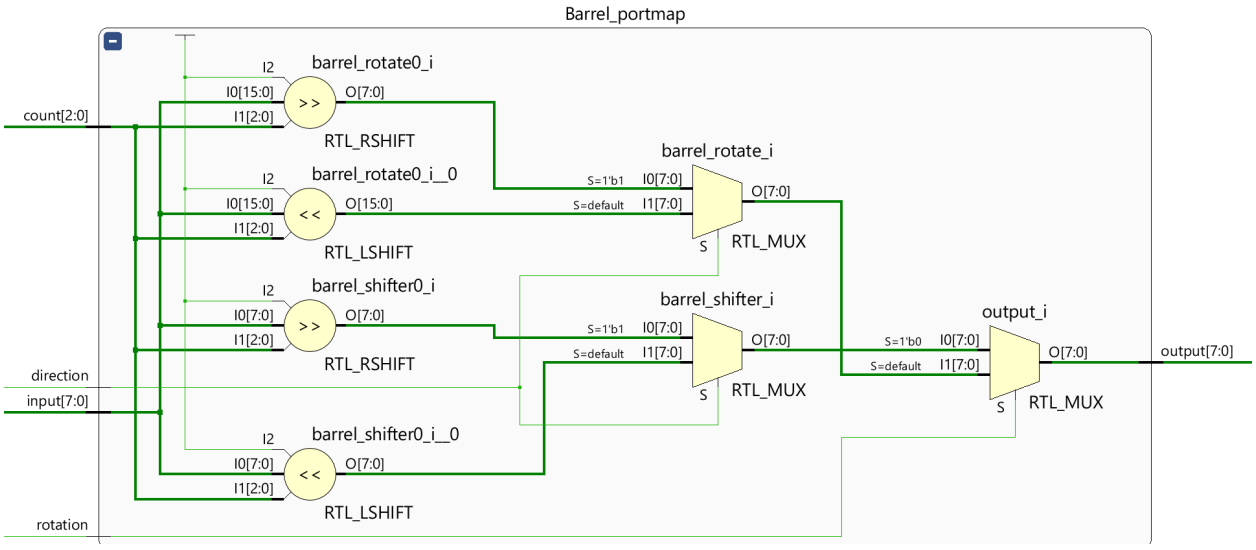


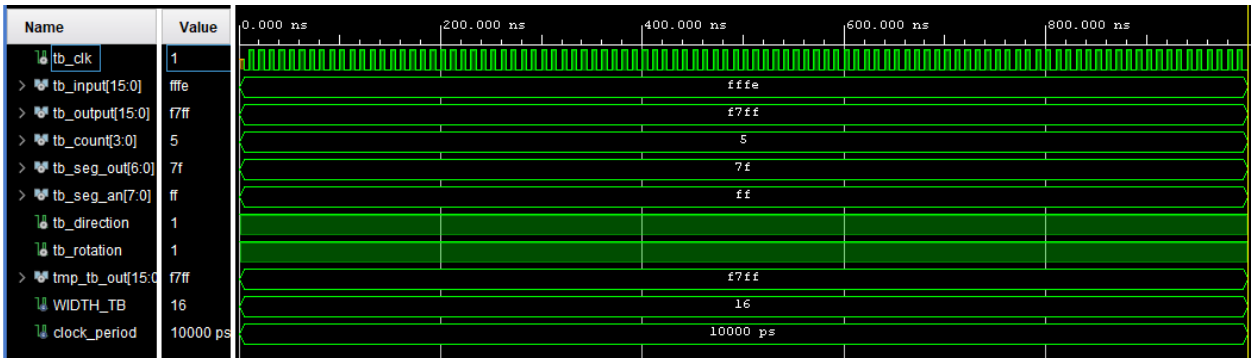
Fig 6 .2 - Elaborated Design for Barrel Shifter/Rotator

Trick of the Codes

The main trick to the code for our lab is the functions implemented within the architecture of the *barrel_shifter*; within the architecture there are two functions, one for the actual shifting and one for rotating. The shifting function takes in three inputs, *din*, *dir, and cnt;* Din is representative of the manual inputs from the board, dir, is what direction to shift in, and cnt is that amount of bits that are being shifted. The rotate function works in a similar fashion taking in the same inputs as the shift function. The operation of the two work differently, within the shift function, the function only returns a shifted bit. Within the rotate function we have a case statement as there is more that is being worked on behind-the-scenes. In the architecture we

call upon the function depending if the rotation input is high or low. If the rotation input is low, then the system works as a shifter; if high, the system rotates teh bits.

For the testbench, it was a little tricky getting the syntax correct at first for textio. After figuring that out, it was relatively easy to test different widths after remapping the generic width within the test bench. When starting a testbench, you must create a file within the project file location titled “bshift_input.txt” and input the direction, rotation, count, and numeric input. An example of this would be “1 1 5 FFFE” where the input is 16 bits (“1111 1111 1111 1110”) , rotated to the left (1) by five (5) bits. The 2nd “1” specifies whether the operation is a shift or rotate and in this case we did a rotation.




```
file_open(file_vectors, "C:/Users/Jacob/Desktop/Desktop/CPP FILES/2021 SPRING/4304L VHDL/LAB6/barrelshifty/barrelshifty.srcs/sim_1/new/bshift_input.txt", read_mode);
file_open(file_results, "C:/Users/Jacob/Desktop/Desktop/CPP FILES/2021 SPRING/4304L VHDL/LAB6/barrelshifty/barrelshifty.srcs/sim_1/new/bshift_output.txt", write_mode);

while not endfile(file_vectors) loop
    readline(file_vectors, v_ILINE);
    read(v_ILINE, v_direc);
    read(v_ILINE, v_space);
    read(v_ILINE, v_rotat);
    read(v_ILINE, v_space);
    hread(v_ILINE, v_count);
    read(v_ILINE, v_space);
    hread(v_ILINE, v_input);

    tb_direction    <= v_direc;
    tb_rotation     <= v_rotat;
    tb_count        <= v_count;
    tb_input        <= v_input;

    wait for 5 * clock_period;

    write(v_OLINE, tb_input);
    write(v_OLINE, v_space);
    write(v_OLINE, tmp_tb_out);
    writeline(file_results, v_OLINE);

end loop;
    file_close(file_vectors);
    file_close(file_results);

    wait;
process;
```

bshift_input - Notepad

File Edit Format View Help

1 1 5 FFFE1

bshift_output - Notepad

File Edit Format View Help

1111111111111110 1111011111111111

Corner Cases

The only corner cases that are present are those that occur in the instance of changing the generic in the testbench to a number that is not a power of 2. In this case, you just have to rewrite the way the textio reads the bshift_input.txt file by changing the last “hread(v_ILINE, v_input)” to either read character by character, or hex and a character (so it takes in all the digits properly). Other than that, there weren’t any other corner cases we encountered when developing and testing our code for the barrel shifter.

Resource Information

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM
✓ synth_1	constrs_1	synth_design Complete!								67	36	0.0
✓ impl_1	constrs_1	route_design Complete!	5.739	0.000	0.263	0.000	0.000	0.140	0	67	36	0.0

Fig. 6.3 - Design Runs

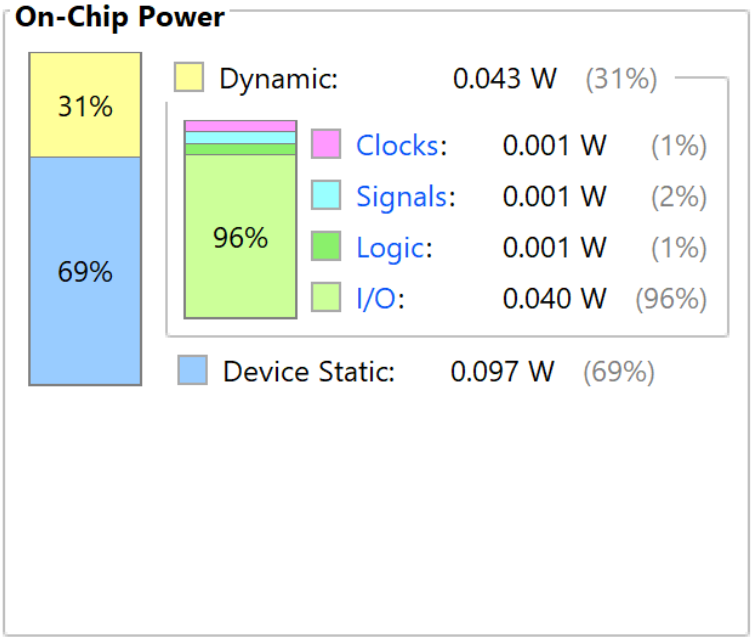


Fig. 6.4 - On-Chip Power