ALU/Ripple Adder

- o Embedded Systems
- o Lab Report #2
- o Fareen Pourmoussavian
- o 3/7/19

• Purpose

o The purpose of this lab is to not only create a 4-bit ripple adder but to also design an ALU.

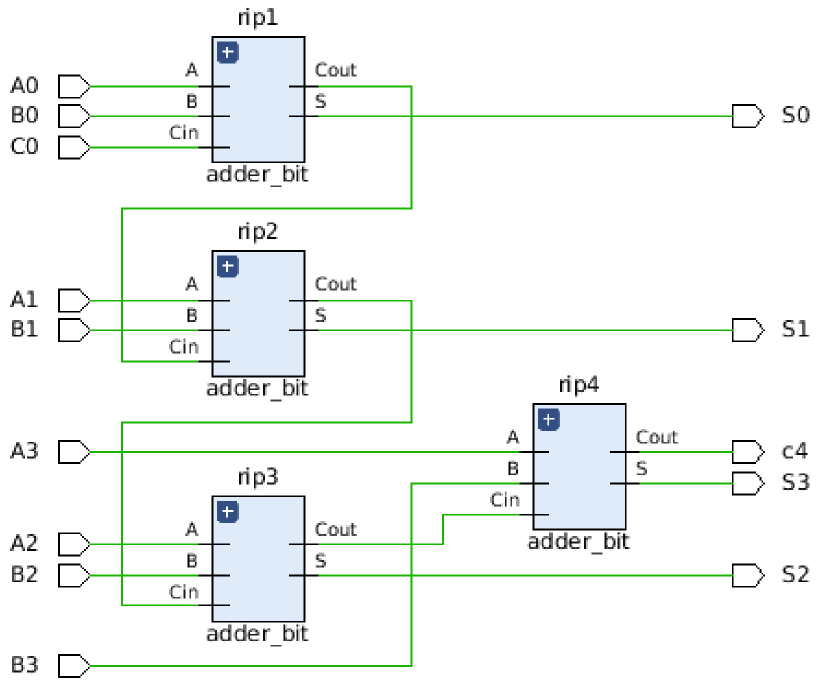
• For Each Part

o Theory of Operation

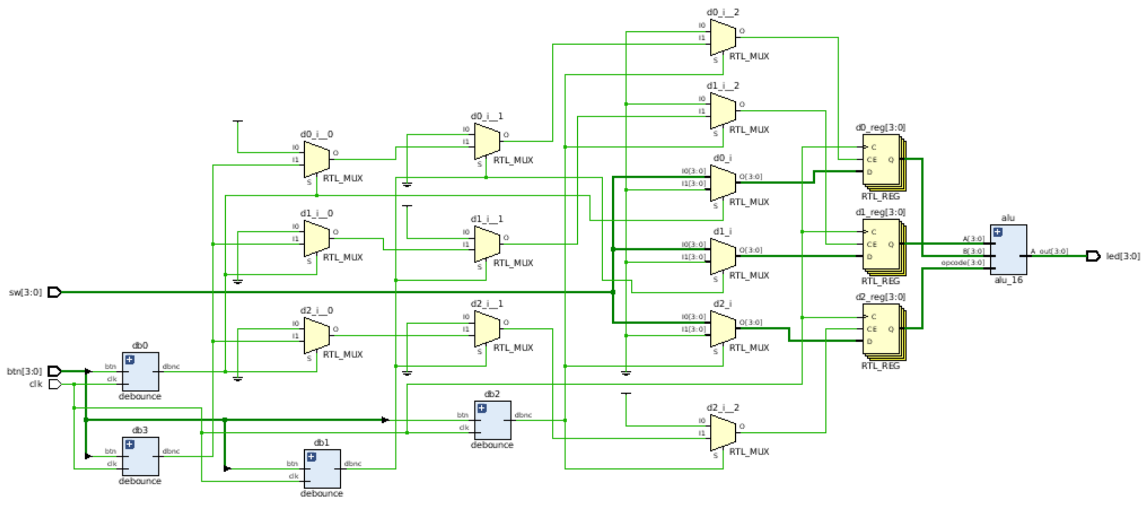
♣ For the ripple adder, given inputs A and B for each bit of the 4-bit adder it should return correct sum and carry-out values. ALU should work

♣ I am expecting the 4-bit adder to work after doing the simulations. For the ALU, I am expecting the ALU to work on the board.

o Schematic Diagram (as desired)



ALU_Tester



Design:

library IEEE;

use ieee.std_logic_1164.all, ieee.numeric_std.all;

```
entity adder_bit is
port(A,B,Cin :in std_logic;
    S,Cout :out std_logic);
end adder_bit;

architecture Adder of adder_bit is
signal F_x, top_an,bot_an :std_logic;

begin

F_x<= A XOR B;
bot_an <= A AND B;
S<=Cin XOR F_x;
COUT<= bot_an OR (Cin and F_x);

end adder;

library IEEE;
use ieee.std_logic_1164.all, ieee.numeric_std.all;

Entity ripple_4 is
port(A0,A1,A2,A3,B0,B1,B2,B3,C0:in std_logic;
    c4,S0,S1,S2,S3:out std_logic);
end ripple_4;

architecture crt4 of ripple_4 is
component adder_bit
```

```
port(A,B,Cin :in std_logic;
    S,Cout :out std_logic);
end component;

signal C1,C2,C3: std_logic;
begin
rip1 : adder_bit
port map(A=>A0,
        B=>B0,
        Cin=>c0,
        S=>S0,
        Cout=>C1);
rip2 : adder_bit
port map(A=>A1,
        B=>B1,
        Cin=>c1,
        S=>S1,
        Cout=>C2);
rip3 : adder_bit
port map(A=>A2,
        B=>B2,
        Cin=>c2,
        S=>S2,
        Cout=>C3);

rip4 : adder_bit
port map(A=>A3,
        B=>B3,
         Cin=>c3,
         S=>S3,
         Cout=>C4);

end crt4;

library ieee;
use ieee.std_logic_1164.all,ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;

entity alu_16 is
port(A,B,opcode :in std_logic_vector(3 downto 0);
     A_out: out std_logic_vector(3 downto 0));
end alu_16;

architecture alu_func of alu_16 is
signal g_t : std_logic_vector(3 downto 0);
begin

    process(opcode)
    begin
       if (A>B) then
            g_t<="0001";
        else
            g_t<="0000";
         end if;
        case(opcode,A,B) is
            when "0000" => A_out <=A+B;
            when "0001"=> A_out <= A-B;
            when "0010"=>A_out<=A+1;
            when "0011" => A_out<=A-1;
            when "0100" => A_out<=0-A;
            when "0101"=>  A_out <= g_t;
            when "0110"=>  A_out <= A(2 downto 0)&"0";
            when "0111"=>  A_out <= "0"&A(3 downto 1);
            when "1000"=>  A_out <= "1"&A(3 downto 1);
            when "1001"=> A_out <= Not(A);
            when "1010"=> A_out <= A AND B;
            when "1011"=>A_out <= A OR B;
            when "1100" => A_out <=A XOR B;
            when "1101" => A_out <=A XNOR B;
            when "1110"=> A_out <=A NAND B;
            when "1111" => A_out <=A NOR B;
            when others => A_Out <= "0000";
        end case;
    end process;
end alu_func;



library ieee;
use ieee.std_logic_1164.all, ieee.numeric_std.all, ieee.std_logic_unsigned.all;

entity alu_top is
port(clk : in std_logic;
     btn,sw : in std_logic_vector(3 downto 0);
     led : out std_logic_vector(3  downto 0));
```

```
end alu_top;

architecture alu_tester of alu_top is
component alu_16
port(A,B,opcode :in std_logic_vector(3 downto 0);
    A_out: out std_logic_vector(3 downto 0));
end component;tianyi.zhang29@rutgers.edu
component debounce
port (clk : in std_logic;
     btn : in std_logic;
     dbnc : out std_logic);
end component;

signal db3_rt,db2_op, db1_A, db0_b :std_logic;
signal d3,d2,d1,d0 : std_logic_vector(3 downto 0);
begin

process(clk)
begin
   if(rising_edge(clk)) then
      if(db2_op='1') then
         d2<=sw;
      elsif(db1_A='1') then
         d1<=sw;
      elsif(db0_B='1') then
         d0<=sw;
      elsif(db3_rt='1') then
         d3<="0000";
```

```
          d2<="0000";
          d1<="0000";
          d0<="0000";
      end if;
   end if;
end process;
```

- o Test
  - ♣ Test bench VHDL

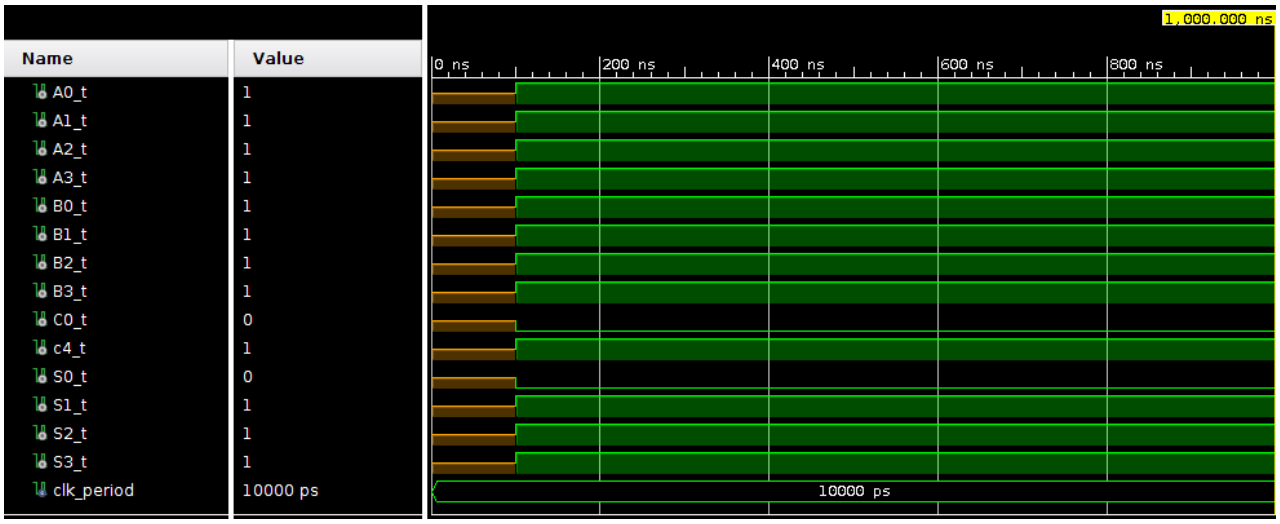
```
Library IEEE;
use IEEE.std_logic_1164.all,IEEE.numeric_std.all;

ENTITY rip_t IS
END rip_t;

architecture rip of rip_t is
Component ripple_4
port(A0,A1,A2,A3,B0,B1,B2,B3,C0:in std_logic;
     c4,S0,S1,S2,S3:out std_logic);
end Component;

signal A0_t,A1_t,A2_t,A3_t,B0_t,B1_t,B2_t,B3_t,C0_t : std_logic ;
signal c4_t,S0_t,S1_t,S2_t,S3_t: std_logic;

constant clk_period : time := 10 ns;

begin
```

```
uut: ripple_4 PORT MAP (

A0 => A0_t,

b0=>b0_t,

C0=>c0_t,

s0=>s0_t,

A1 => A1_t,

b1=>b1_t,

s1=>s1_t,

b2=>b2_t,

A2 => a2_t,

s2=>s2_t,

a3=>a3_t,

b3=>b3_t,

c4=>C4_t,

s3=>s3_t);


--clk_process :process

--begin

--clk_test <= '0';

--wait for clk_period/2;

--clk_test <= '1';

--wait for clk_period/2;


--end process;



stim_proc: process

begin
```

```
wait for 100ns;
A0_t<='1';
B0_t<='1';
c0_t<='0';
A1_t<='1';
B1_t<='1';
A2_t<='1';
B2_t<='1';
A3_t<='1';
B3_t<='1';
wait for 10ns;
wait;
end process;
end;
```

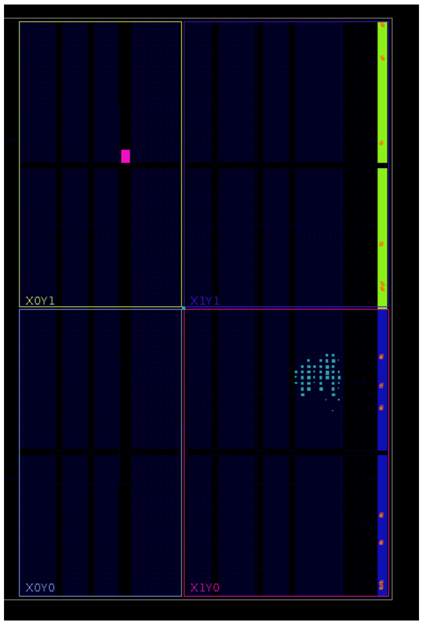
♣ simulation results (screen shots)

4-bit ripple adder

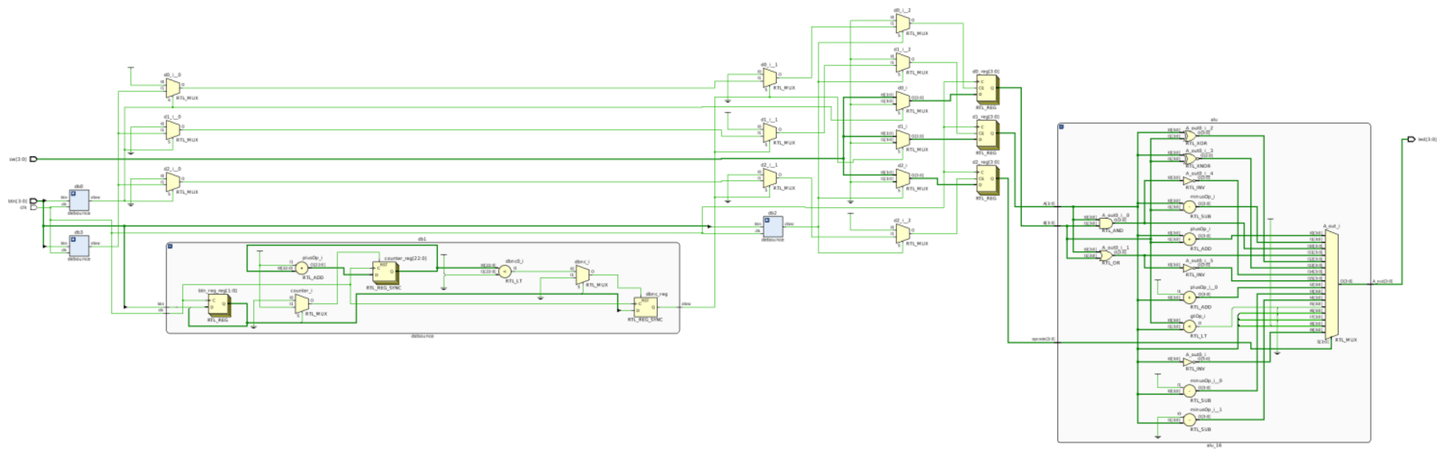


Name	Value
A0_t	1
A1_t	1
A2_t	1
A3_t	1
B0_t	1
B1_t	1
B2_t	1
B3_t	1
C0_t	0
c4_t	1
S0_t	0
S1_t	1
S2_t	1
S3_t	1
clk_period	10000 ps

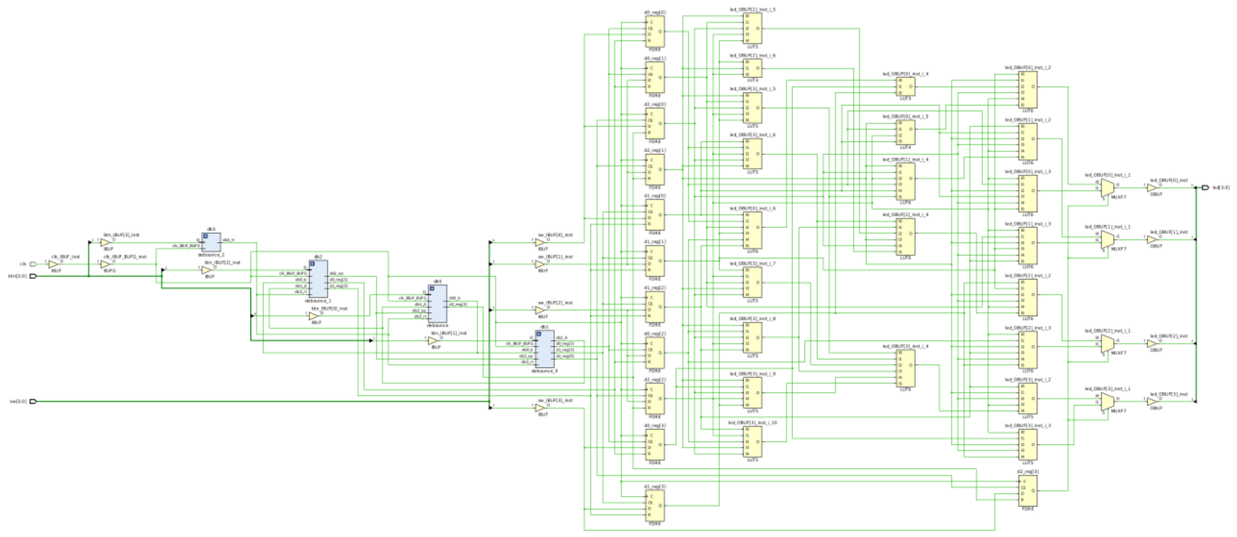
o Implementation (If required to demo on board)



♣ Vivado Elaboration Schematic

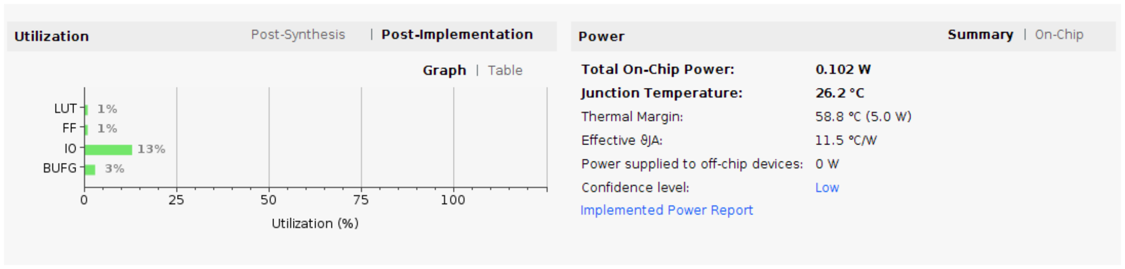


♣ Vivado Synthesis Schematic



♣ Vivado Project Summary Images

- Post-Synthesis Utilization Table



Utilization Post-Synthesis | **Post-Implementation**

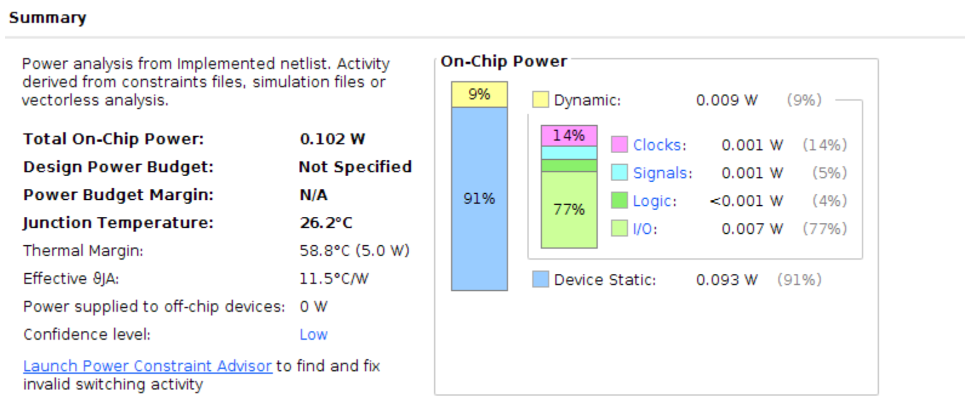
Graph | Table

Power **Summary** | On-Chip

Total On-Chip Power:	**0.102 W**
Junction Temperature:	**26.2 °C**
Thermal Margin:	58.8 °C (5.0 W)
Effective θJA:	11.5 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

- On-chip Power Graphs



Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.102 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26.2°C**
Thermal Margin:	58.8°C (5.0 W)
Effective θJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.009 W	(9%)
Clocks:	0.001 W	(14%)
Signals:	0.001 W	(5%)
Logic:	<0.001 W	(4%)
I/O:	0.007 W	(77%)
Device Static:	0.093 W	(91%)

ZYBO_Master.xdc

- The only things that needed to be changed was the uncommenting of all the LED, Switches, buttons and clock signals so that those ports on the board would work when I try to implement my alu design on the board.

- Discussion

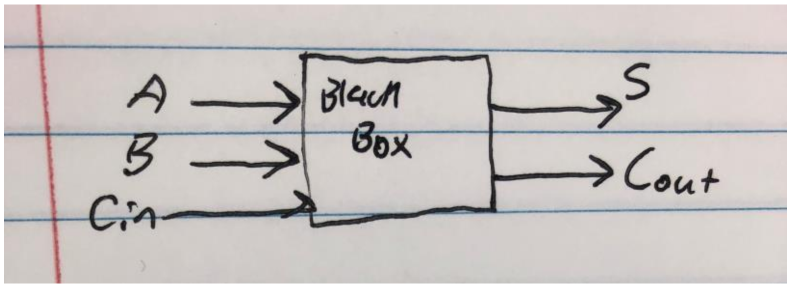
PreLab

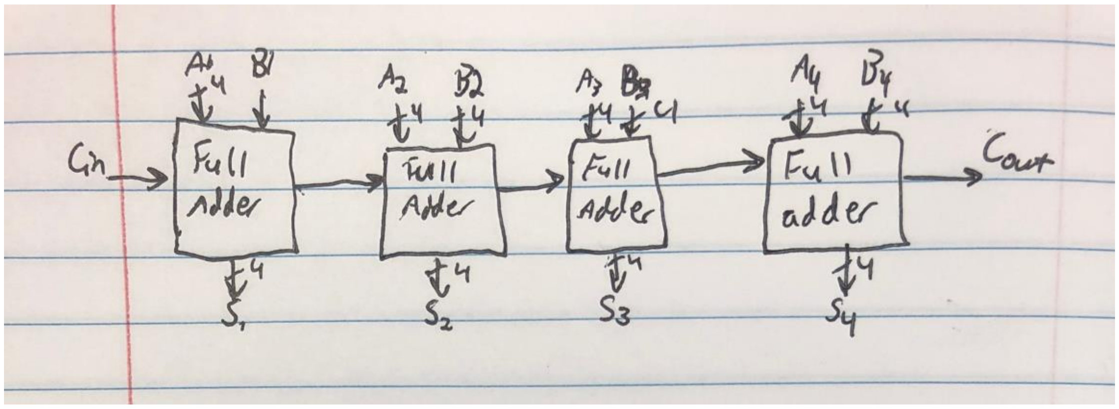
A	B	Cin	Y	Cout
0	0	0	0	0

0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$\oplus \quad \oplus Y = A \;\; B \quad C$

C_out=AB+ACin+BCin=AB + Cin(A+B)





What did you learn?

♣ I learned what the functions of the ALU are and what they do based on a given opcode.

o Questions / Follow Up

♣ This lab manual made everything easy to understand and implement. I understand the ALU executes a function depending on a given op code. I completely understand ripple and full adders and how they add two numbers together by looking at their bits.

♣ I had a little confusion on how to connect switches to buttons to inputs of ALU because debouncers have std_logic outputs while switches and alu are std_logic_vectors