Lab 3 Report
4-bit Accumulator (ECE558)

Luan Vo (30553600), November 9th 2020 (Extended to Thursday 8:00am Approved)

***Purpose & Introduction* — This lab is the continuation of lab 2, which was a 1-bit accumulator.**

I. Design Capture (Step A1):

From lab 2, we have the following design for 1-bit accumulator symbol:

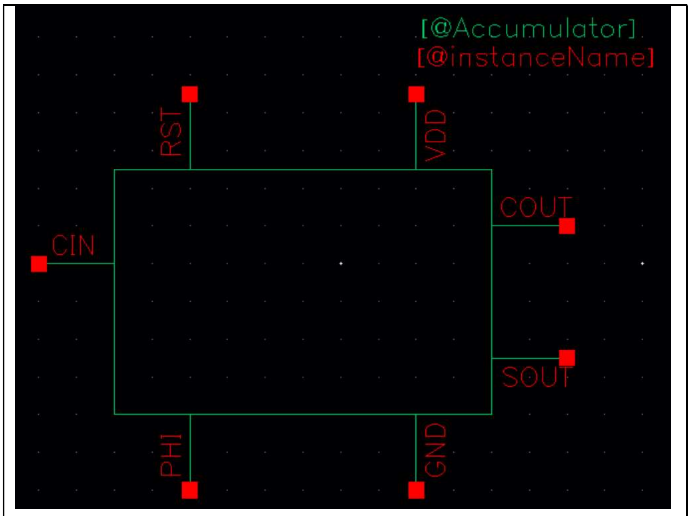

Figure 1. *1-bit Accumulator Symbol*

By using the library cell symbol of the 1-bit accumulator from lab 2, we can make the 4-bit accumulator by appending 4 1-bit accumulators in series. Observe the figure below:

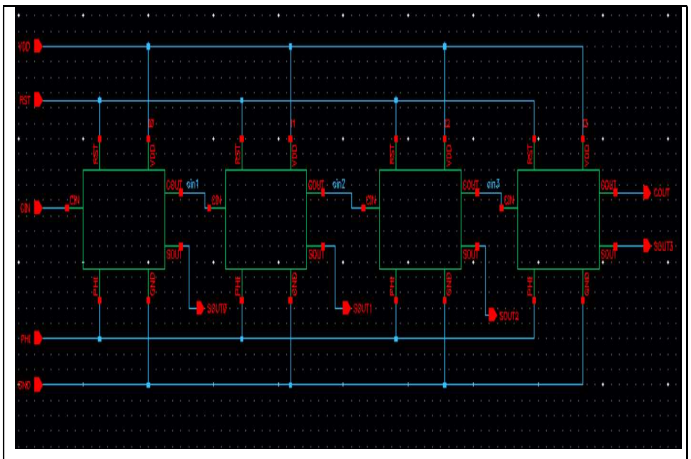

Figure 2. *4-bit Accumulator Schematic*

Since 558 students are using half-adders instead of full-adders, the COUT of HA is wired to the input of next HA. Which means that there are no A inputs in the schematic. (check Appendix A for enlarged version). From Fig. 2, we could then generate the following symbol for a 4-bit accumulator:

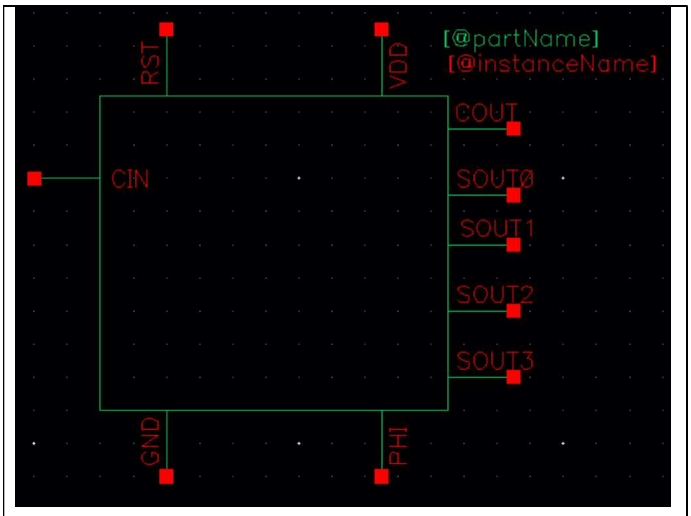

Figure 3. *4-bit Accumulator Symbol*

II. Schematic Validation (Step A2)

A. Input Sequences from ID

Convert ID to 4-bits, we have the following table:

ID digit	4 bits	Least significant
3	0011	1
0	0000	0
5	0101	1
5	0101	1
3	0011	1
6	0110	0
0	0000	0
0	0000	0

Where the least significant column will be our inputs for vec file.

B. Checking with truth table, stable PHI

To test the validity of the design, I will examine the truth table of the 4-bit accumulator stage-by-stage. This will include the SOUT, and COUT of each stage. Firstly, let's look at the first stage. ***However, the first two inputs will be randomized, as we do not know what happen at the very start (since 1 of the inputs to HA is from SOUT).*** Thus, before any calculations are made, we have to reset the circuit by input RST = 0.

The setup for the vec file is as follows: Trise/Tfall = 30ps | Tperiod = 300ps. Where CIN = 111011100010.

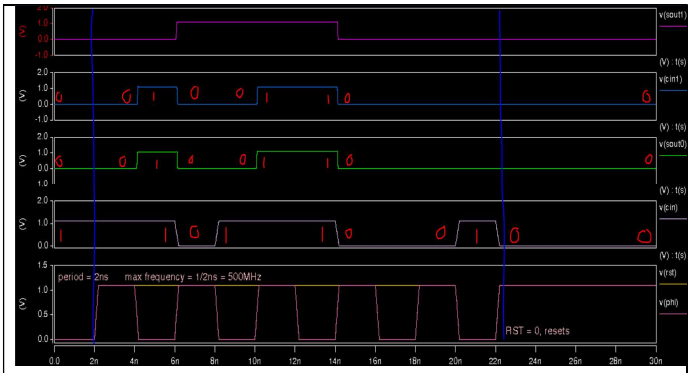

Figure 4. *First Stage Output*

From Fig. 4, we have the following table, where COUT0 = CIN1. (resets at blue line in figure)

Table 1. *4-bit Accumulator First Stage Output*

Line	PHI	RST	CIN	SOUT0	COUT0
1	0	0	1	0	0
2	1	1	1	0	0
3	0	1	1	1	1
4	1	1	0	0	0
5	0	1	1	0	0
6	1	1	1	1	1
7	0	1	1	1	1
8	1	1	0	0	0
9	0	1	0	0	0
10	1	1	0	0	0
11	0	0	1	0	0
12	1	1	0	0	0

The inputs start from line 3, since we wasted 1 clock cycle to start the accumulator (as it is needed to reset in line 1, else the starting state of the accumulator is unpredictable). Thus, at line 3, ***the flipflop started storing the value at rising edge – meaning that the value must be at 0 on a 0 to 1 edge of PHI***. Therefore, it is logical to expect that the stored value will last for 1 clock cycle (i.e: 2 inputs). ***In other words, when flipflop stored CIN in line 3, CIN in line 4 is ignored***. This is correct, as in Fig. 4, we observe that at around 1ns, even though CIN = 0, SOUT does not change, as that value is not stored. However, the value of flop is the inverse of SOUT, as flipflop is followed be an inverter. In table 1, reset is highlighted in yellow. It is worth noting that when RST = 0, SOUT = 0 on the next rising edge of PHI. ***Which means that the value stored in the flop will not change instantaneously, as it needs to be waited until the next rising edge, not the current one***.

Ultimately, the results for SOUT0 and COUT0 of the first stage are correct. Next, we observe the behaviors of the second stage's output:

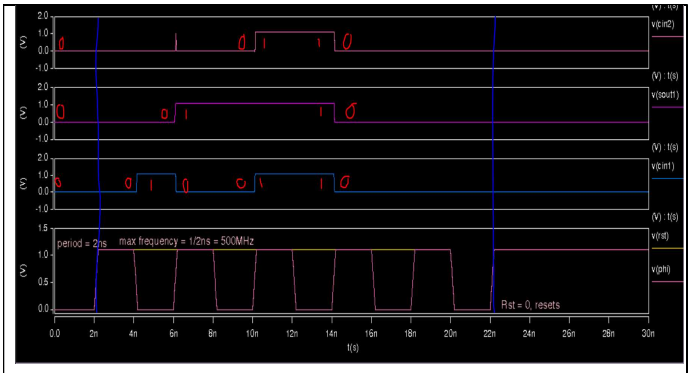

Figure 5. *Second Stage Output*

From Fig. 5, we have the following table, where COUT1 = CIN2. (resets at blue line in figure)

Table 2. *4-bit Accumulator Second Stage Output*

Line	PHI	RST	CIN1	SOUT1	COUT1
1	0	0	0	0	0
2	1	1	0	0	0
3	0	1	1	0	0
4	1	1	0	1	0
5	0	1	0	1	0
6	1	1	1	1	1
7	0	1	1	1	1
8	1	1	0	0	0
9	0	1	0	0	0
10	1	1	0	0	0
11	0	0	0	0	0
12	1	1	0	0	0

Similarly, flipflop only sample at rising edges (0 to 1), and the resets are still placed in the same place at lines 1 and 11. In addition, ***when the flop stores these values at rising edges, SOUT will be flipped of whatever is being stored***. It is evident that the resulting SOUT1 and COUT1 are correct since their addition adheres to the behavior of a normal half-adder. Next, we observe the behavior of the third stage:

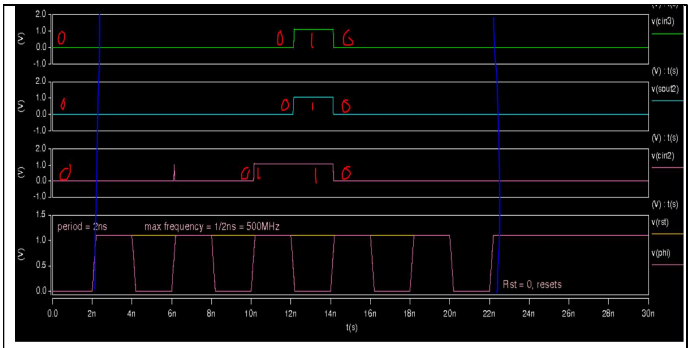

Figure 6. *Third Stage Output*

From Fig. 6, we have the following table, where COUT2 = CIN3.

Table 3. *4-bit Accumulator Third Stage Output*

Line	PHI	RST	CIN2	SOUT2	COUT2
1	0	0	0	0	0
2	1	1	0	0	0
3	0	1	0	0	0
4	1	1	0	0	0
5	0	1	0	0	0
6	1	1	1	0	0
7	0	1	1	1	1
8	1	1	0	0	0
9	0	1	0	0	0
10	1	1	0	0	0
11	0	0	0	0	0
12	1	1	0	0	0

In this stage, we have many 0s output in SOUT, this is because when either RST or S2 is 0, the output will be 1, and when that is inverted, SOUT will be 0. Similarly, the sum of CIN2 and SOUT2 produces the correct COUT2. Lastly, we observe the fourth stage:

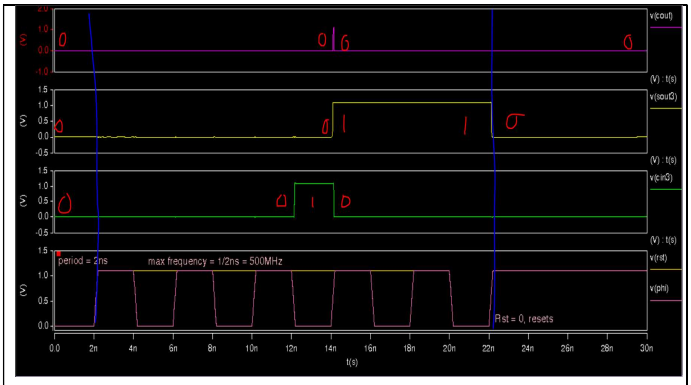

Figure 7. *Fourth Stage Output*

From Fig. 7, we have the following table, where COUT3 = COUT.

Table 4. *4-bit Accumulator Third Stage Output*

Line	PHI	RST	CIN3	SOUT3	COUT3
1	0	0	0	0	0
2	1	1	0	0	0
3	0	1	0	0	0
4	1	1	0	0	0
5	0	1	0	0	0
6	1	1	0	0	0
7	0	1	1	0	0
8	1	1	0	1	0
9	0	1	0	1	0
10	1	1	0	1	0
11	0	0	0	1	0
12	1	1	0	0	0

In this stage, we have the similar behaviors of SOUT3, where the NAND of S3 and RST both resulted in a 1, and the inverse of that is SOUT3 = 0. We also have the correct addition between CIN3 and SOUT3 that produces COUT3.

Therefore, this means that the 4-bit accumulator is working correctly, as observed in the above graphs and tables.

III. Determine Maximum Clock Frequency (Step A5):

Since frequency is 1/time. Therefore, the higher the frequency, the smaller the clock time. Also, to avoid triangle-shaped waveforms, we want the rise and fall time to be 10% of the period. Input sequences are the same as STEP 2. Below is the image of the original measurement where:

Maximum: Rise/Fall = 0.2ns, Period = 2ns => f = 500MHz

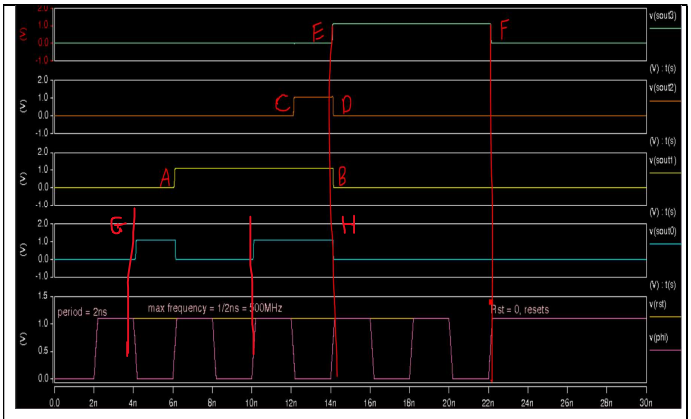

Figure 8. *Accumulator with clock speed of 500MHz (2ns)*

A. Maximum clock

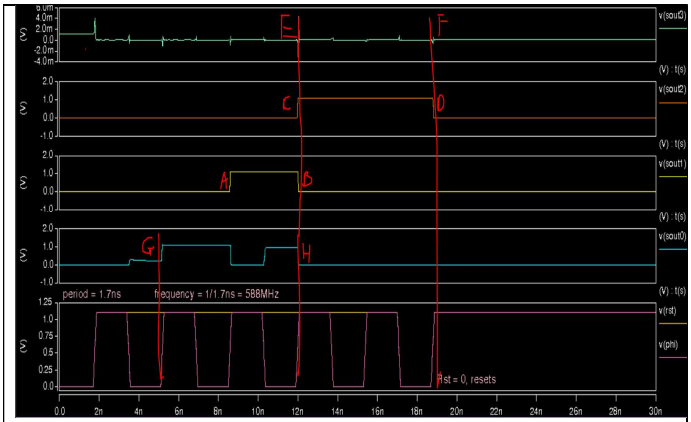

Figure 9. *Accumulator with clock speed of 588MHz (1.7ns)*

The maximum operating frequency is defined as the frequency in which all 4 of accumulators output the correct bits, and not just the first one. It is worth noting that, if the first accumulator failed, then the ones that come after it will also fail. However, with the given input sequence, we can conclude that the system failed below 500MHz. Since at 1.8ns, the output at the fourth accumulator failed. Thus, making the circuit's logic becomes incorrect. (where all 4 failed in Fig. 9)

CIN = 111011100010 (same with Step A2) (1)

The correct results are captured between the labeled points, which serve as the reference point to compare with between Fig. 8 and 9. While my inputs do not have any values followed SOUT and COUT after point D and F, nonetheless ***so long that the values detected at the rising edge are either inconsistent or delayed from what we would expect in normal circumstances (step A5, Fig. 8, 9), then it is safe to say that such behaviors mean that the circuit does not behave as expected*** (since the clock is faster than what the switching could handle).

B. When the system is unstable and 20% above

The system is, in my opinion, considered unstable at 588MHz. We see that, while the change is very small, nonetheless, at 588MHz (1.7ns) in Fig. 10, we see that the output between E and F is 0 – where it should have been 1. It means that the circuit is lagging and could not catch up with the clock. Fig. 11 is where the period is 10%-20% above the unstable value at 1.8ns or 555MHz. ***(all other SOUT are incorrect as well)***

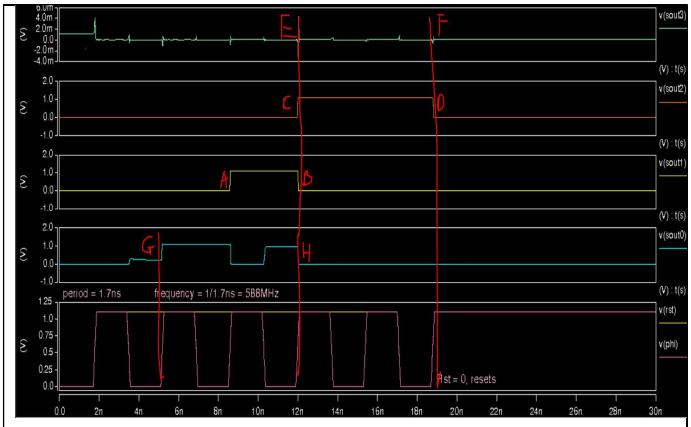

Figure 10. *Accumulator with clock speed of 15.38 GHz (65ps)*

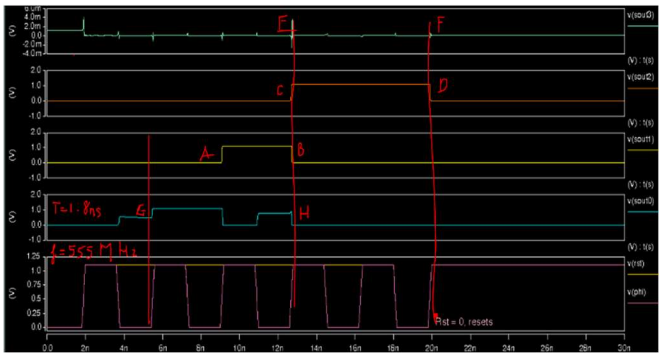

Figure 11. *Accumulator with clock speed of 12.5 GHz (80ps)*

From Fig. 10 and 11, we can see that the point at which the output seems to differ is when the period is at around 65ps because, at this point, we were expecting SOUT1 to be 1 between points E and F. however, between those points, SOUT1 of 1.7ns outputs 0. ***With this, I concluded that given the input sequence (1) from Step A2, after 1.7ns the output no longer behaves as expected, thus making around 2ns the maximum period of the system – this corresponds to around 500MHz in frequency.***

Besides, we see that Fig. 11 shows an extra 20% of 1.7ns, which shows that the system outputs as expected at 80ps. Which means that, any time larger than 1.7ns, the system behaves as expected.

C. Critical Path

Albeit that 558 design is not a traditional RCA, however, it does not mean that it doesn't act like one. With the 4-bit accumulator, ***the critical path will be the path that has the longest delay, and in this case, it is the path from CIN to the output of the 4th accumulator***. This is because it takes the longest time for all the previous COUT to reach it, this is also known as the propagation delay.

The path starting from input CIN to COUT is the critical path because the adder at the 4th position has to wait for the CIN from all previous adders before it can perform the adding logic. ***Due to this waiting time, each adder must wait for the carry bit from previous adders before any calculations can be made***. With that in mind, we then have the following formula for RCA critical path delay:

RCA critical delay=number of accumulator * gate delay (2)

IV. Power Calculations (Step A6)

Before we perform the analysis of this section, here are some useful knowledge and information regarding the power consumption modes of a circuit. As we have learned, dynamic power can be represented as:

$$P_{dynamic} = P_{switching} + P_{short} \quad (1)$$

Whereas, static power is defined as the power leakage when the circuit is not in operation and can be represented as:

$$P_{static} = (I_{sub} + I_{gate} + I_{junc} + I_{contention}) * VDD \quad (2)$$

A. Dynamic Power Simulation

To perform this, I am assuming that $P_{dynamic} >> P_{static}$. Which means that $P_{total} \sim= P_{dynamic}$. ***Therefore, the approximation of the circuit's dynamic power consumption (through the entire input sequence) can be represented as the average power consumption***. Since the average power can give us a good approximation of the average dynamic power (i.e: approximate power of all the switching within the circuit). I will be taking measurements of power with the command below in HSPICE:

.measure tran avgpower AVG power from=1ps to= period (3)

Where the from and to will be our entire sequence operational time, as adding more than necessary will interfere with the average power calculation (since hspice assumes that the last bit will drag on until we put a stop to it). I will be making my power measurements from 300ps down to 70ps with several

measurements. Following the measurements yield the following table: (used ps for better measurement accuracy)

Note: I have 12 inputs, so the total period for hspice simulation is equal to: period = 12 * (one period).

Table 5. *Frequency vs AVG power*

One period (ps)	Frequency (GHz)	$P_{average}$ (uW)
300	3.33	28.01
270	3.7	30.84
240	4.167	35.21
210	4.76	36.91
180	5.55	39.64
150	6.67	47.4
120	8.33	59.84
90	11.11	84.97
70	14.28	111.7

In the end, I wanted to make the graph as accurate as possible, so I made a few more measurements. Below is the graph generated from this measurement set:

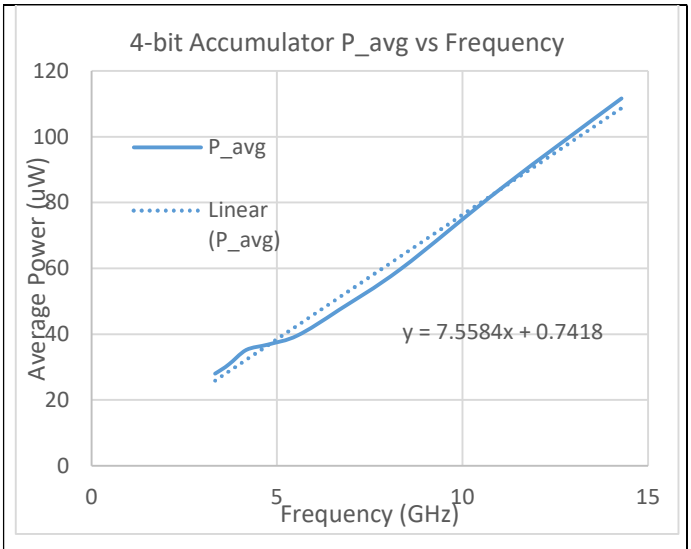

Figure 12. *4-bit Accumulator Power vs Frequency best fit line*

From this graph, we can see that power consumption in terms of uW/GHz is approximately the slope of the line, which is shown through the trend line in the graph. The reason being that the slope represents the possible increment of uW per GHz. For example, at 11.11GHz: $P_{average\sim}$= 7.56*11.11= 83.996uW, which is almost equal to what we have in the table. However, this result yields an interesting observation: the intercept variable of the trendline. (Measurement stops near maximum operating frequency).

Ultimately, the average switching power in terms of uW/GHz is the slope of the trend line (since it is dependent of frequency), which is 7.55 uW/GHz. Which is the equivalent of:

7.55 uW/(MHz*10^-3) = 7.55mW/MHz

B. Static Power

To calculate the average static power, one interesting observation from this diagram is the y-intercept, in this case, is 0.7418uW. Since static power is defined as the power leakage of FETs and devices when the circuit is not in operation or steady states, which means that this consumption is being consumed when transistors are consistent. ***Which means that, this consumption could be estimate to be consistent throughout the process – regardless of frequency (i.e: average)***. Which means that, ideally, the static power should be the y-intercept value. While I noticed that adding more points to the graph will cause a grow in inconsistency of the circuit trend line. Meaning that the best option should be making the trendline as fit as possible to the uW line, that way, we can get the most accurate readings of the static power, or y-intercept. The reason behind this is that, ***static power is extremely small when compared to dynamic power*** (In normal assumptions and ideals). However, as transistors keep on shrinking and can operate at a faster frequency, the contribution of static power to the total power consumption increases. Ideally, in a circuit architecture, we would only want dynamic power consumption – as static power is the lost of power without achieving any activities within the circuit [1].

Fig. 12 ***we can make an approximation that, ideally, average static power will be approximately 0.7418uW***. It makes sense, as the static power should – ideally – be very small when compared to switching power.

C. Why P_avg ~= P_avg_switching

From part A and B, we can then conclude the following:

Table 6. *Static vs Switching Power Comparision*

Static Power (nW)	Switching Power (uW/GHz)	Static/Switching at 1GHz (%)
742	7.558	9.8

From table 6, we can see that static power is only equal to roughly 10% (or less) of switching power. With this in mind, by setting average power to be the average switching power, we can roughly say that static power is around 10% of the total average power (or less). This means that, static power can generally be ignored. For example:

At f = 1GHz =>P_avg = 7.558 meaning that static power ~=10% of P_avg.

At f = 11.11GHz=>P_avg= 84 meaning that static power~=0.9% of P_avg.

This concludes that the faster the switching is, the less likely the circuit is going "static" and thus less leakage. This makes sense, since the shorter the "idle" time of the circuit becomes, the less the circuit becomes susceptible to power leakage. ***Due to this, when we measure the average power, we can ignore static power, and conclude that***:

$$P\_switching >> P\_static$$

$$P\_avg = P\_switching + P\_static \sim= P\_switching \quad (2)$$

V. SYNTHESIS USING SYPNOSIS (STEP B1):

A. Setting up

```
luanvo@vlsicad1:~$dc_shell

                        Design Compiler Graphical
                             DC Ultra (TM)
                              DFTMAX (TM)
                         Power Compiler (TM)
                           DesignWare (R)
                            DC Expert (TM)
                        Design Vision (TM)
                         HDL Compiler (TM)
                        VHDL Compiler (TM)
                            DFT Compiler
                      Library Compiler (TM)
                        Design Compiler(R)

          Version E-2010.12-SP5-2 for linux -- Oct 29, 2011
               Copyright (c) 1988-2011 Synopsys, Inc.

This software and the associated documentation are confidential and
proprietary to Synopsys, Inc. Your use or disclosure of this software
is subject to the terms and conditions of a written license agreement
between you, or your company, and Synopsys, Inc.

Initializing...
dc_shell> source setup.tcl
Information: Defining new variable 'type'. (CMD-041)
Information: Defining new variable 'RTL_DIR'. (CMD-041)
Information: Defining new variable 'GATE_DIR'. (CMD-041)
Information: Defining new variable 'modname'. (CMD-041)
Information: Defining new variable 'clkname'. (CMD-041)
```

Figure 13. *Setup TCL*

Refer to the table below for the meaning of each command and the modifications made to each variable:

Table 7. *Setting up design compiler parameters*

Command	Purpose	Modification
Dc_shell	Enter design compiler	N/A
Set modname	Change the current name of the design	Accumulator
Set clkname	Change the current name of the clock	PHI

The reason why we are setting modname to accumulator is because ***the design classified in lab3_partB folder is named "accumulator.v" which means that the design is named accumulator***. The specification clkname is necessary because depends on each design, the clock being applied to the system would change, ***I picked PHI for the clock name seeing that it is better than leaving it CMD-041 as the setup automatically generated***. In addition, the DFF is clocked by PHI in our design. If no clkname is specified, an error will occur, as the system the circuit design needed a clock.

```
dc_shell> source read.tcl
Loading verilog file '/home/ece558_2020/luanvo/accumulator.v'
Detecting input file type automatically (-rtl or -netlist).
Running DC verilog reader
Warning: Overwriting design file '/home/ece558_2020/luanvo/accumulator'. (DDB-24)
Reading with Presto HDL Compiler (equivalent to -rtl option).
Running PRESTO HDLC
Compiling source file /home/ece558_2020/luanvo/accumulator.v

Inferred memory devices in process
        in routine accumulator line 17 in file
                '/home/ece558_2020/luanvo/accumulator.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|      SOUT_reg       | Flip-flop |   4   |  Y  | N  | N  | N  | N  | N  | N  |
===============================================================================
Presto compilation completed successfully.
Current design is now '/home/ece558_2020/luanvo/accumulator.db:accumulator'
Loaded 1 design.
Current design is 'accumulator'.
accumulator
dc_shell> source Constraints.tcl
Current design is 'accumulator'.
  Allocating blocks in 'accumulator'

  Linking design 'accumulator'
  Using the following designs and libraries:
  --------------------------------------------------------------------------
  NangateOpenCellLibrary (library)
                          /home/ece558_2020/luanvo/NangateOpenCellLibrary_slow_ccs.db
  dw_foundation.sldb (library)
                          /usr/synopsys/E-2010.12-SP5-2/libraries/syn/dw_foundation.sldb

1
dc_shell>
```

Figure 14. *Read TCL and Constraints*

If an optimal area is needed, in the constraints file, we have to set the following statement: set_max_area 0. This is to let the system design a layout that will output the optimal output (i.e: as close to 0 as possible). ***To specify the frequency, we need to change the CLK_PER parameter in constraints.tcl*** file. Since the file is presented in ns, we must convert from ps to ns before we change the parameter. ***From netlist, there are 4 D flipflops***.

B. Validation

The following table describe the validity and properties of the circuit when using automated method of synthesis:

Table 8. *Synthesis design validation and properties of accumulator*

Parameter	Value in Constraints	Validity
Timing violation	20 ns	MET
Estimated Area	0 um^2	39.9

This means that, with the given values in the constraints file, there is ***no violation in the circuit***. In addition, the ***area is estimated to be (maximized) at 39.9 um^2***.

By looking at the netlist file, we see that the file is called a module. ***This means that the file contains information about the desired behaviors of the given circuit/design***. Which means that ***RTL is similar to an abstraction of the design***. With the given abstraction and the desired behaviors***, the circuit is then formed by using a program that automatically create the circuit for the user***. In our case that software is called "Synthesis". ***Hence, RTL synthesis is technology dependent***, as it only requires the user to write an abstraction and the circuit will be automatically formed by synthesis tool. This is to makes the job of create larger scale circuit become easier, since a modern circuit has millions to billions of transistors.

C. Upper and Lower Frequencies using Batch

The maximum clock frequency according to synthesis is recorded when there exists violation in "timing_max_slow_holdfixed_tut1.rpt". Given this condition, the maximum frequency in which synthesis report to have a violation is at: (Note: changes in setup.tlc is needed for this)

```
 Operating Conditions: slow   Library: NangateOpenCellLibrary
 Wire Load Model Mode: top

  Startpoint: CIN (input port clocked by PHI)
  Endpoint: COUT (output port clocked by PHI)
  Path Group: PHI
  Path Type: max

  Des/Clust/Port     Wire Load Model       Library
  ------------------------------------------------
  accumulator        5K_hvratio_1_1        NangateOpenCellLibrary

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock PHI (rise edge)                   0.0000     0.0000
  clock network delay (ideal)             0.0000     0.0000
  input external delay                    0.6580     0.6580 r
  CIN (in)                                0.0513     0.7093 r
  U43/ZN (NAND2_X1)                       0.0760     0.7853 f
  U36/ZN (OAI211_X1)                      0.1494     0.9348 r
  U35/ZN (OAI211_X1)                      0.0873     1.0221 f
  U38/ZN (NAND2_X1)                       0.0958     1.1178 r
  COUT (out)                              0.0293     1.1472 r
  data arrival time                                  1.1472

  clock PHI (rise edge)                   1.7000     1.7000
  clock network delay (ideal)             0.0000     1.7000
  clock uncertainty                      -0.0500     1.6500
  output external delay                  -0.5660     1.0840
  data required time                                 1.0840
  -----------------------------------------------------------
  data required time                                 1.0840
  data arrival time                                 -1.1472
  -----------------------------------------------------------
  slack (VIOLATED)                                  -0.0632
```

Figure 15. *CLK_PER for violation*

As shown in Fig. 15, the CLK_PER that resulted in positive timing violation is at 1.7ns. This is an equivalent of 588MHz. Lower frequency is equivalent to higher CLK_PER value (since frequency and time has an inverse relationship). Let's pick the lower frequency to be around 20% less than the maximum frequency, which is around 370MHz (i.e: 2.7ns), we have the following table:

Table 9. *Synthesis implementations at different frequencies*

CLK_PER (ns)	Frequency (MHz)	Area (um^2)	Cells
1.7	588	60.65	47
2.7	370	39.9	13

From table 9, we ca see that the closer we approach the positive timing violation of the circuit, the more cells synthesis generated. ***At maximum frequency, the area is about 1.5 times larger than that of the regular area***. Ultimately, ***at 1.7ns and 2.7ns, we have 2 different synthesis implementations*** as described in table 9.

VI. PLACE AND ROUTE USING CADANCE (STEP B2):

A. Initial Setup for PR

Originally, the aspect ratio for this design is 1 and density of 0.9. which means that W/L = 1, creating a square floor plan. With such configuration, we have the following design:

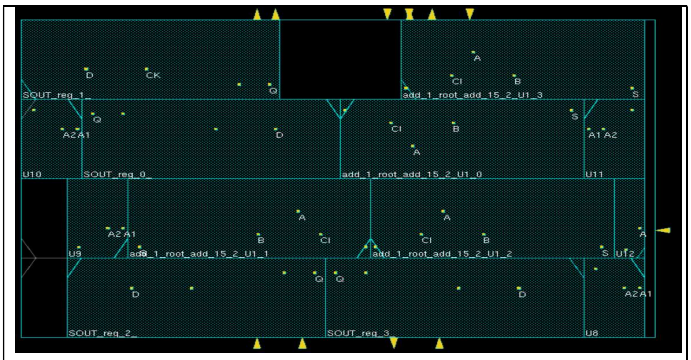

Figure 16. *Accumulator's physical view (without wire, layer)*

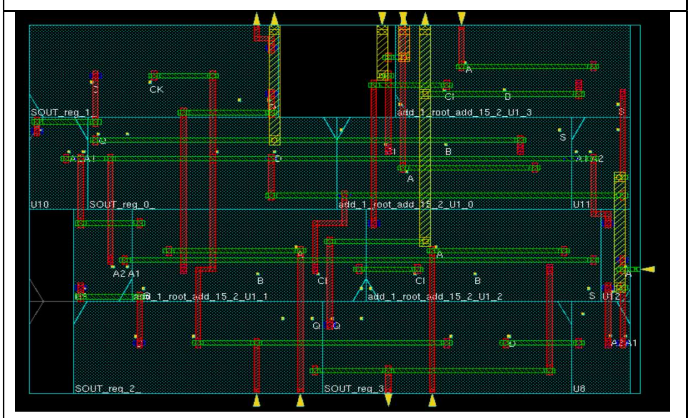

Figure 17. *Accumulator's physical view (with wire, layer)*

B. Changing ratio to 0.5 to create a rectangular floorplan

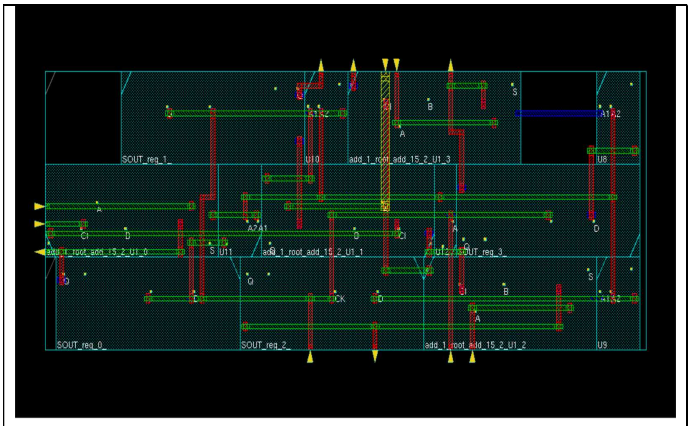

Figure 18. *Accumulator's physical view (with wire, layer)*

```
106 Total standard cell length = 0.0285 (mm), area = 0.0000 (mm^2)
107 Average module density = 0.909.
108 Density for the design = 0.909.
109       = stdcell_area 150 (40 um^2) / alloc_area 165 (44 um^2).
110 Pin Density = 0.387.
111           = total # of pins 58 / total Instance area 150.
112 Checking spec file integrity...
```

Figure 19. *Accumulator's allocated area for silicon*

From Fig. 18, we can see that with 0.5 aspect ratio, we have created a rectangular floorplan. In addition, this aspect ratio does not create any hazards nor violations in the layout. Also, from Fig. 19, we can see that the silicon area is around 44 um^2.

It is worth noting that, in within the config file, there is a spec called density, which is set at 0.9. This value stands for the specified density that synthesis will strive to achieve. Which means that from line 109 and 110 in Fig. 19, we can see that the density of standard cell over our design is 0.9. Hence, that is why our allocated design area is 165 (44 um^2).

VII. SYNTHESIS FOR DIFFERENT BIT-WIDTHS (STEP B3):

A. Design different bit-widths accumulators

Below is the table that detailed the maximum operational frequencies and the area at said frequency.

Table 10. *Synthesis implementations at different frequencies*

Bit-width	CLK_PER (ns)	Frequency (MHz)	Area (um^2)	Cells
4	1.7	588	60.65	47
8	2.2	454	112.518	81
16	2.7	370.37	236.74	175
24	3.2	312.5	346.86	252
32	3.7	270.27	474.544	350

From table 9, we have the following graphs:

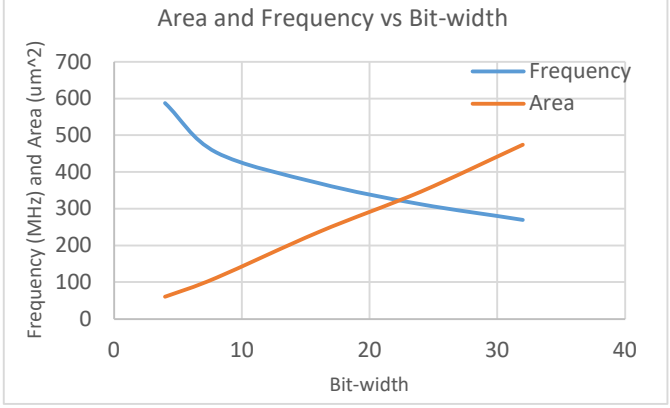

Figure 20. *Area and Frequency vs Bit-width*

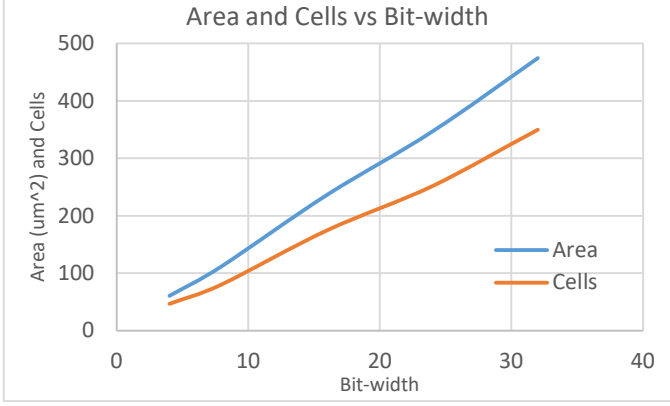

Figure 21. *Area and Cells vs Bit-width*

B. Discussion and Analysis

From table 10 and Fig. 20, we see that the more stages (i.e: stages increases as bit-width increases) the accumulator maximum operational frequency drops. Which makes sense, as the ***critical path has now increased, leading to longer delay between input CIN and output COUT*** (or SOUT). ***Due to this longer delay, the maximum operational frequency drops (due to inverse relationship with time)***. Which was why we observed an decrease in maximum frequency as more bit-width are added to the accumulator.

Interestingly, as we increase the stages of the accumulator, the area also increased. ***This is the synthesis software attempted to come up with the design that adheres to the constraints specified*** as well as the striving to meet the CLK_PER requirements. This makes sense, as the more demanding the requirements are the more complicated and sophisticated a design is. This can also be seen through Fig. 21, as the bit-width increases, so does the area and the cells that are needed to complete the design.

The design specification or model resides in – for our lab – "accumulator.v" by modifying this file, we can change the number of bit-width for our accumulator. To modify the number of bits, I change the array indices of each parameter (such as SOUT, S, and CIN) within the accumulator design file. Below is the sample code for a 32-bit accumulator:

```
module accumulator( PHI, RST, A, CIN, SOUT, COUT);

input PHI;
input RST;
input [31:0] A;
input CIN;
output reg[31:0] SOUT;
output COUT;

wire [31:0] B;
wire [32:0] S;

assign B = SOUT;
assign COUT=S[31];
assign S = A+B+CIN;

always @(posedge PHI)
begin
   if(RST)
     SOUT <= 32'h0;
   else
     SOUT <= S[31:0];
end

endmodule
```

Figure 22. *32-bit accumulator sample code*

As shown in Fig. 22, the number of bit-widths are dictated by the array's length specified in each variable. This script seems to be correct, as when running the design compiler, there were no errors detected. For example:

```
Loading verilog file '/home/ece558_2020/luanvo/lab3_partB/accumulator.v'
Detecting input file type automatically (-rtl or -netlist).
Running DC verilog reader
Reading with Presto HDL Compiler (equivalent to -rtl option).
Running PRESTO HDLC
Warning: Can't read link_library file 'your_library.db'. (UID-3)
Compiling source file /home/ece558_2020/luanvo/lab3_partB/accumulator.v

Inferred memory devices in process
        in routine accumulator line 17 in file
                '/home/ece558_2020/luanvo/lab3_partB/accumulator.v'.
===============================================================================
|    Register Name    |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|     SOUT_reg        | Flip-flop |  32   |  Y  | N  | N  | N  | N  | N  | N  |
===============================================================================
Presto compilation completed successfully.
Current design is now '/home/ece558_2020/luanvo/lab3_partB/accumulator.db:accumulator'
Loaded 1 design.
Current design is 'accumulator'.
accumulator
source Constraints.tcl
Current design is 'accumulator'.
Loading db file '/home/ece558_2020/luanvo/lab3_partB/NangateOpenCellLibrary_slow_ccs.db'
Loading db file '/usr/synopsys/E-2010.12-SP5-2/libraries/syn/dw_foundation.sldb'
Information: Checking out the license 'DesignWare'. (SEC-104)
Information: Evaluating DesignWare library utilization. (UISN-27)

============================================================================
| DesignWare Building Block Library  |         Version         | Available |
============================================================================
| Basic DW Building Blocks           | E-2010.12-DWBB_201012.5 |     *     |
| Licensed DW Building Blocks        | E-2010.12-DWBB_201012.5 |     *     |
============================================================================

  Allocating blocks in 'accumulator'
```

Figure 23. *32-bit successfully compiled*

With this, we can conclude that, Fig. 22 script works, therefore, the results obtained in earlier parts (i.e: steps B1 and B2) must be correct as well. ***Ultimately, the synthesis tool is technology dependent, and it is an automated process to design a circuit and layout that will give the desired performance***. However, this is not flawless, for example, we observe some empty space in Fig. 17 and 18 which represented that nothing is being placed there. Nonetheless, such space is undesirable in the final product. ***Which is why synthesis and encounter are used as supplements to ease the design process of a very large scale circuits and integrations***.

VIII. COMPARE RESULTS (STEP C1):

A. Comparision table

In this section, we will compare the result of the 4-bit accumulator generated by synthesis and encounter to our design

in part A. From the results gain in the above sections, we have the table below:

Table 11. *Setting up design compiler parameters*

Category	Part A (Manual)	Part B (Automatic)
Bit-width	4	4
Area	Chose option 2, so no layout	60.6 um^2
Max time	2ns	1.7ns
Max frequency	500 MHz	588 MHz
Static Power at max frequency (refer to table 6 for manual)	741.18 nW	790.5392 nW
Switching Power (refer to table 6 for manual)	7.558 uW	7.5391 uW
Average Power at maximum frequency	111.7 uW	26.05 uW
Metal layers	Chose option 2, so no layout	10 layers

B. How measurements for part B is taken

```
Operating Conditions: slow   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

Design        Wire Load Model            Library
------------------------------------------------
accumulator            5K_hvratio_1_1    NangateOpenCellLibrary


Global Operating Voltage = 0.95
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000ff
    Time Units = 1ns
    Dynamic Power Units = 1uW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =  18.5198 uW   (71%)
  Net Switching Power  =   7.5391 uW   (29%)
                         ---------
Total Dynamic Power    =  26.0589 uW  (100%)

Cell Leakage Power     = 790.5392 nW
```

Figure 24. *Synthesis Power Report*

```
#Analyzing routing resource...
#Routing resource analysis is done on Tue Nov 10 19:51:04 2020
#
#  Resource Analysis:
#
#               Routing  #Avail      #Track     #Total     %Gcell
#  Layer      Direction   Track     Blocked      Gcell    Blocked
#  ---------------------------------------------------------------
#  Metal 1        H          40           0           6   100.00%
#  Metal 2        V          42           0           6     0.00%
#  Metal 3        H          40           0           6     0.00%
#  Metal 4        V          26           0           6     0.00%
#  Metal 5        H          19           0           6     0.00%
#  Metal 6        V          26           0           6     0.00%
#  Metal 7        H           6           0           6     0.00%
#  Metal 8        V           9           0           6     0.00%
#  Metal 9        H           3           0           6     0.00%
#  Metal 10       V           4           0           6     0.00%
#  ---------------------------------------------------------------
#  Total                    215       0.00%  60    10.00%
#
#
```

Figure 25. *Encounter cell design spec*

To makes synthesis output a power report, a command must be added to CompileAnalyze.tcl file as below:

Report_power > power.rpt

This command instructs synthesis to perform a power analysis on the "accumulator.v" circuit. Thus, resulted in Fig. 24 report. From this, we can tell that synthesis software is mostly used for circuit analysis, whereas encounter is used to actually build a sample vlsi layout using given schematic.

C. Why there exist the difference between part A and B?

One of the most obvious reason lie in the fact that ECE558 students are using half adder instead of the full adder this could contribute to the difference in maximum operational frequency between part A and that of part B.

Secondly, I do not know how to manipulate the inputs using synthesis (which is why the voltage stays at 1V in Fig. 24). ***This contributed to the difference, since, in part A, since CIN in synthesis is unknown to me***. ***In addition, the input sequence for RST and PHI is also unknown***. Which means that the maximum operational frequency could have been failed earlier than 2ns (from manual). Nonetheless, the difference between 2ns and 1.7ns is relatively small, and thus is tolerable.

While the static and switching power are very closely equal between the two method. However, the total average power is off by a large margin. ***This is most likely due to the fact that I used the frequency above maximum operational frequency for my measurements***. Thus, since the system is switching at a faster pace (in GHz – refer to table 5 and Fig. 12), causing the average power to be more than that of synthesis.

Overall, the two methods yield very similar results, except for the difference in average power and the reasons are as stated above.

IX. CRITICAL PATH (STEP C2):

In synthesis, the critical path data can be found in the the following files:

- Timing_max_slow_holdfixed
- Timing_max_typ_holdfixed
- Timing_maxfast_holdfixed

Usually, we would use slow holfixed, however, it does not matter in this case, as the path remains the same throughout these 3 files. For this test, I picked typical holfixed because it is more general. Observe the figure below:

```
Operating Conditions: typical   Library: NangateOpenCellLibrary
Wire Load Model Mode: top

  Startpoint: A[0] (input port clocked by PHI)
  Endpoint: COUT (output port clocked by PHI)
  Path Group: PHI
  Path Type: max
```

Figure 26. *Critical path of 4-bit accumulator*

The critical path starting point is from A[0] (i.e: first input) and end at COUT. This is exactly the same with what we had in step A5 (or Section 3, part C in this report). In part C, it is stated that the ***critical path starts from CIN instead of A[0] (because of half-adder)***, and ends at COUT. This agrees with what synthesis outputs. ***Which makes sense, as a ripple carry adder (4-bit accumulator) has a propagation delay between COUT of each stage, so the more stages a ripple carry adder has, the slower it will become***. And interestingly, since it is an RCA, the rate which RCA slowed down is linear. Which can be observed in table 10.

This idea is important because this means that RCA is a series implementation, and each accumulator has certain delay and that will add up as more "ripples" or stages are added to the RCA. Therefore, the longest path would naturally be the starting input (CIN) towards the last carryout (COUT).

X. SUMMARY

In conclusion, this lab teaches us how to better use SPICE tools and combining more gates than that of lab 1. In addition, we also got to see the importance of sizing and their effects on the circuit's delay and behaviors.

REFERENCES

[1] J. Butts and G. Sohi, "A static power model for architects," Proceedings 33rd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO-33 2000, Dec. 2000.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.