ECE 260A – Lab 3

4-tap 16-bit unsigned averaging FIR filter

Team Members

Name	PID	Contribution
Siddharth Thinakaran	A59005429	Implemented Carry Select Adder for different structures and checking functionality of the adders on ModelSIM.
Mihir Pratap Singh	A59011168	Implemented Carry Save Adder for different structures and report writing.
Shreyas Borse	A59009564	Implemented Ripple Carry Adder for different structures and report writing.

Project Overview

The objective is to first design a FIR filter, at register-transfer-level abstraction (in System Verilog) and then synthesize from RTL to a gate-level netlist (in 65nm CMOS). This filter is a **4-tap unsigned magnitude FIR filter with registered I/O,** that needs to be optimized for fast performance. Here, 4-tap indicates that the filter averages the input signal from four different time stamps.

Solution:

- **AIM**

To design a **4-tap unsigned magnitude FIR filter** with weights h[i] =1. We want to add four consecutive signals of width 16-bits. Assume we have a signal *x(n)[15:0]* where n is the time stamp. On the first cycle, we add *x[0] + x[1] + x[2] + x[3]* and in the next cycle we add *x[1] + x[2] + x[3] + x[4]* meaning we do a lot of recomputations. We look at different architectures of adders that are fast while using low area, power and combinational units.

- **ADDER SELECTION**

Here we look at different adder designs as well as the way different adders are used to get a 4-tap FIR filter. We first ran the simulation on the given System Verilog file that used a Ripple Carry adder on the first stage and the tool decides the adder on the second stage. The results showed a slack of -0.18 ns and a power consumption of 1.8422 mW for an area of 2457.36 units. This needs to be improved upon. We then looked into the Carry Save Adder and the Carry Select Adder as well as the canonical, tree and transpose designs to compute slack, area and power for comparison.

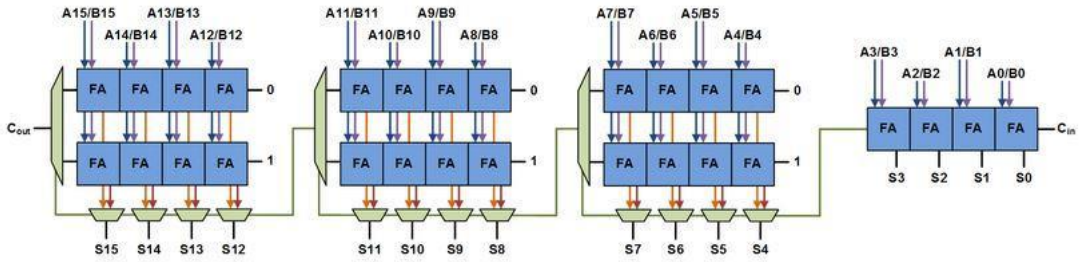

Figure 1: A uniform width 16-bit carry select adder

Carry Select Adder (CSA) was looked at using the design in Fig.1. As can be seen, the obvious disadvantage CSA has is the total area required to implement the adder since two sets of adders are used, one with the $C_{in} = 0$ and one with the $C_{in} = 1$. The delay is reduced from a ripple carry adder since the CSA does not have to wait for the carry from the previous Full Adder within a set. The multiplexers are used to select the sum bits as well as the C_{out} from each set of full adders (here of width 4). The carry select adder can be found in the *c_sel_add.sv* file. We have called the module wherever required in our designs.

Takeaway from using CSAs in design:

- As expected, the total area as well as the combinational cells area increased largely w.r.t. the reference (given) architecture.
- The slack became even worse (-0.19 ns) when all the Ripple-Carry Adders were replaced by Carry Select Adder. Thus, the CSA gave roughly the same performance w.r.t. slack but using more area as well as power.

Carry Save Adder (CSaA) was also coded and used. Since the carry save adder has three inputs, two CSaA were used as can be seen on the right in Figure 2. The sum and carry of the second CSaA is fed into flip flops that are summed later. To implement the CSaA itself, full adders are used. Figure 2 shows the implementation of a CSaA that adds 4-bit signals. The same can be extended for a 16-bit signal, which we have done. Using CSaA should help reduce slack since carry outs are not rippled through each stage like in carry ripple adder. Also, the area should be comparable to a ripple carry adder.

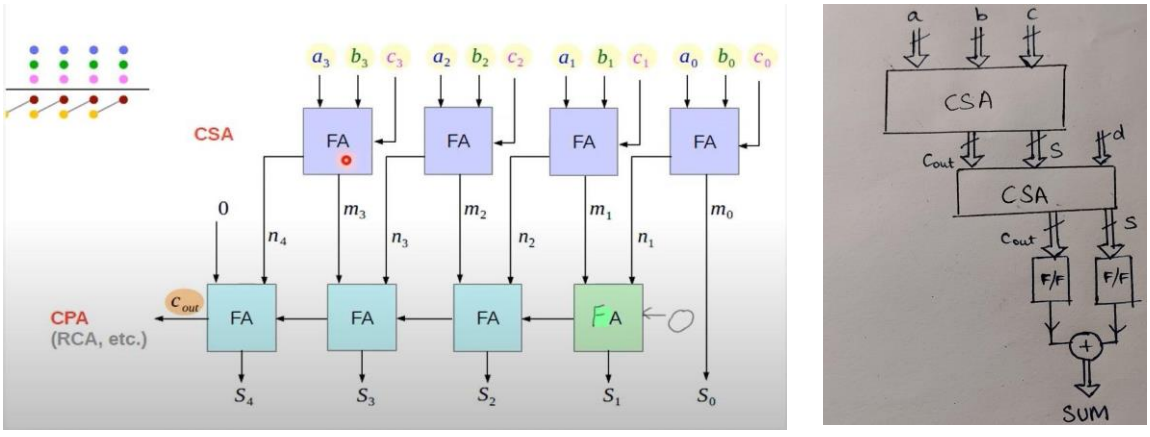

Figure 2: Carry Save Adder implementation

Takeaway from using Carry Save Adders in design:

- The tool gave a larger negative slack than an RCA, which was not what we expected.
- But the area and number of combinational units decreased as compared to the given architecture, which was what we expected.
- Power consumption as well as leakage power both decreased when compared to an RCA implementation.

We further looked into the different architectural structures such as canonical, transpose and simple cascading. We stayed away from pipelining since pipelining adder tree would not give results in the same clock cycle and possibly take up more register memory and delay.

Transpose architecture showed the most promising results since there is only one adder in between any two flip flops. There are also no entirely new additional 16-bit registers like the example of pipelining the canonical form, but only 1 or 2 additional flops to store the carry-out of previous sums. Carry Select adder was initially tried with the transpose architecture, with which slack was met, with lesser area compared to canonical form, which improved further when we let the tool select. The major area advantage comes from the fact that the multiplication factor is 1, so just 16-bit registers are enough instead of 32-bit or more, depending on the multiplier.

- **RESULT TABLE**

The following table summarizes the different architectures proposed for the design. The reference architecture (RCA + Tool selected) is seen in the first row of the table.

Architecture (Stage1 + Stage 2)	Worst negative slack in ns (setup analysis)	Total power consumed (mW)	Leakage power (µW)	Area of combinational logic	# of combinational cells	Total area
RCA + Tool decides	-0.18	1.8422	25.194	1820.52	826	2457.36
All RCA (Canonical cascade)	-0.37	1.9605	29.15	2242.44	961	2846.52
RCA (Tree)	-0.15	1.8958	27.331	1992.6	876	2637.72
All CSA(Canonical)	-0.19	1.9636	29.296	2027.88	845	2679.12
2 CSaA + Tool selects	-0.21	1.7502	20.081	1263.24	588	1905.48
CSA (Transpose)	0	1.7676	16.116	1210.68	600	1829.88
All Tool selects (Transpose)	0	1.7403	14.981	1181.88	569	1799.28

Table 1: Summary of area, timing and power for different proposed architectures

Here: CSA = Carry Select Adder, RCA = Ripple Carry Adder, CSaA = Carry Save Adder

- **CONCLUSION**

The reports for timing, area and power for the best-case result, i.e. transpose architecture while allowing the tool to select the adder can be seen below.

```
Des/Clust/Port     Wire Load Model       Library
------------------------------------------------
fir4rca            ZeroWireload          tcbn65gpluswc

Point                                    Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                    0.00       0.00
clock network delay (ideal)              0.00       0.00
ar_reg_3_/CP (DFQD4)                     0.00       0.00 r
ar_reg_3_/Q (DFQD4)                      0.14       0.14 f
U463/ZN (ND2D2)                          0.03       0.17 r
U184/ZN (OAI21D4)                        0.04       0.21 f
U428/Z (AO31D1)                          0.10       0.30 f
U532/ZN (AOI21D1)                        0.04       0.35 r
U495/Z (CKXOR2D1)                        0.09       0.43 f
U91/ZN (NR2XD0)                          0.03       0.47 r
s2r_reg_7_/D (DFQD2)                     0.00       0.47 r
data arrival time                                   0.47

clock clk (rise edge)                    0.50       0.50
clock network delay (ideal)              0.00       0.50
s2r_reg_7_/CP (DFQD2)                    0.00       0.50 r
library setup time                      -0.03       0.47
data required time                                  0.47
-----------------------------------------------------------
data required time                                  0.47
data arrival time                                  -0.47
-----------------------------------------------------------
slack (MET)                                         0.00
```

Figure 3: Timing Report

```
Library(s) Used:

    tcbn65gpluswc (File: /home/linux/ieng6/ee260afa21/public/lab3_set

Number of ports:                         36
Number of nets:                         672
Number of cells:                        654
Number of combinational cells:          569
Number of sequential cells:              85
Number of macros/black boxes:             0
Number of buf/inv:                      115
Number of references:                    54

Combinational area:              1181.880021
Buf/Inv area:                     131.040005
Noncombinational area:            617.399997
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                 1799.280017
```

Figure 4: Area Report

```
Global Operating Voltage = 0.9
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1nW


  Cell Internal Power  =   1.4370 mW   (83%)
  Net Switching Power  = 288.3095 uW   (17%)
                         ---------
Total Dynamic Power    =   1.7253 mW  (100%)

Cell Leakage Power     =  14.9811 uW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000              0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000              0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000              0.0000            0.0000  (   0.00%)
clock_network      0.0000           0.0000              0.0000            0.0000  (   0.00%)
register           1.2523           0.1737          5.9900e+03            1.4319  (  82.28%)
sequential         0.0000           0.0000              0.0000            0.0000  (   0.00%)
combinational      0.1847           0.1146          8.9911e+03            0.3084  (  17.72%)
--------------------------------------------------------------------------------------------------
Total              1.4370 mW        0.2883 mW       1.4981e+04 nW         1.7403 mW
1
```

Figure 5: Power Report

The best results are obtained with transpose architecture while allowing the tool to select the adder with optimization ON. Thus, the tool is using an optimized adder that is likely different from CSA, RCA or CSaA which gives the least amount of slack, area and power. We obtain a slack of 0 ns with the power consumption and power leakage values of 1.7403mW (5.53% improvement) and 14.98 **µ**W respectively. The total area, combinational area and number of combinational cells requirements are 1799.28 units (26.78% improvement), 1181.88 units and 569 respectively. The improvements over the reference design are summarized in the table below.

	Reference Design	Tool selects all adders (Transpose)	%age Improvement
Worst negative slack in ns (setup analysis)	-0.18	0	Slack Met
Total power consumed (mW)	1.8422	1.7403	5.53%
Leakage power (uW)	25.194	14.981	40.54%
Area of combinational logic	1820.52	1181.88	35.08%
# of combinational cells	826	569	31.11%
Total area	2457.36	1799.28	26.78%

Table 2: Comparison of given (reference) design and best-case design

- **POINTER**

The following folder contains proposed System Verilog files, Design Compiler power, timing and qor reports:

`/home/linux/ieng6/ee260afa21/sborse/lab3/dc_proposed_design/Implemented_SV`

_____END OF REPORT_____