Lab 3: Designing a power efficient architecture for 4-Bit Adders

- a report by

Adithya Anand | A59010781
Gandhar Deshpande | A59005457

CE, Department of
Electrical And Computer Engineering

Prepared in the partial fulfillment of
ECE 260A, in conjunction with taking the final comprehensive examination.

University of California, San Diego

Quarter 1, 2021 - 22

CONTENTS

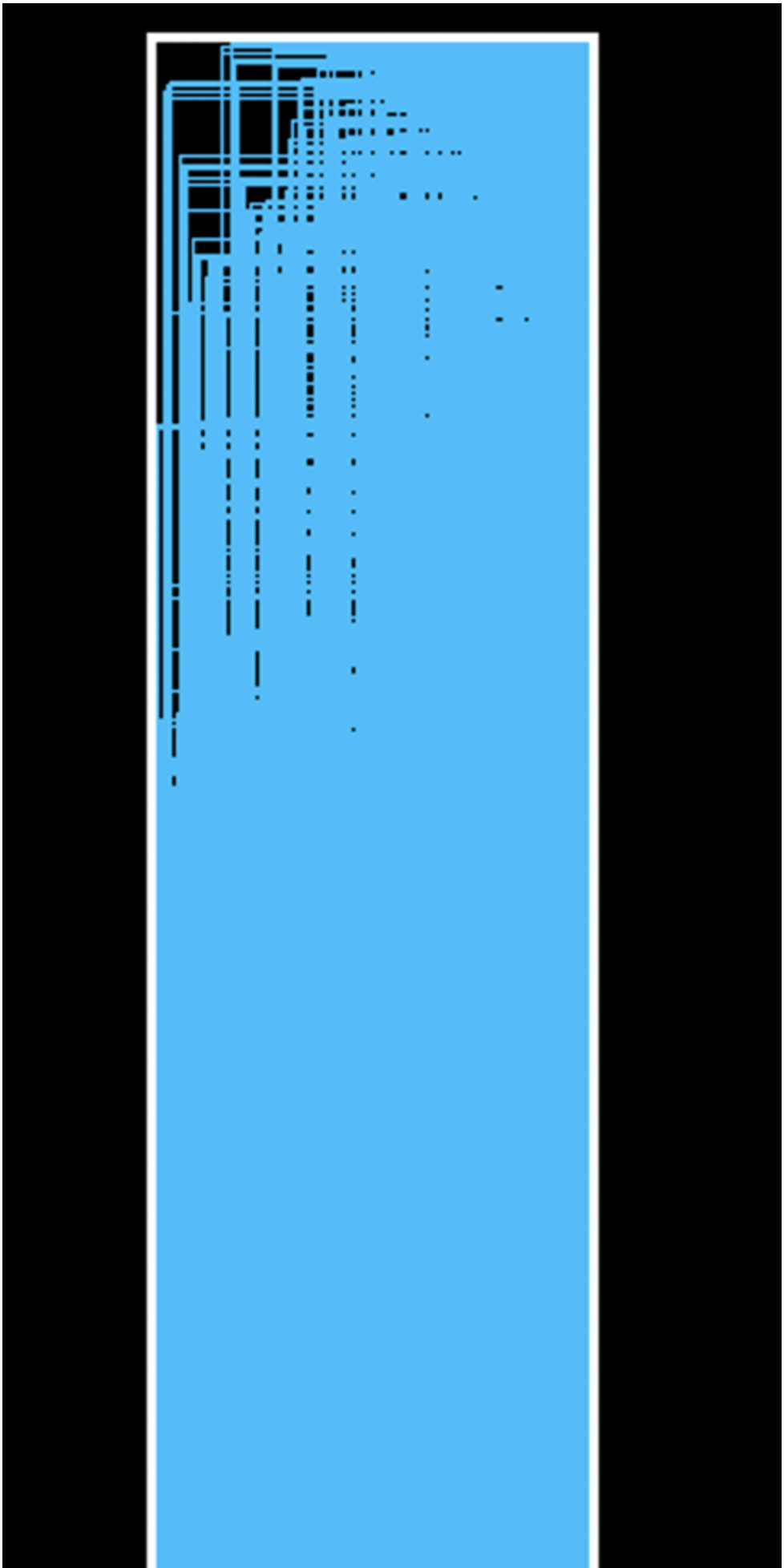
1. Reference Architecture

1.1 Summary

We've taken the reference architecture to be a cascaded ripple carry adder. Implemented in System Verilog, we've introduced a gate-level implementation of a cascade of full adders. The number of full adders is parameterised and introduced by the test bench. It runs the following rubric:

1. ar, br, cr, dr are the four states of the variables.
    a. In each clock cycle, a -> ar, ar -> br, br -> cr, cr -> dr.
2. sum = s3 where s3 = s2 + dr, s2 = s1 + cr, s1 = ar + br
    a. cin[i] = cout[i-1] (**for each stage**)
    b. s1 = ar[i] ^ br[i] ^ cin[i] at any clock cycle.
        i. cout[i] = ((ar[i] ^ br[i]) & cin[i]) | (ar[i] & br[i])
    c. s2 = s1[i] ^ cr[i] ^ cin[i] at any clock cycle
        i. cout[i] = ((s1[i] ^ cr[i]) & cin[i]) | (s1[i] & cr[i])
    d. s3 = s2[i] ^ dr[i] ^ cin[i] at any clock cycle
        i. cout[i] = ((s2[i] ^ dr[i]) & cin[i]) | (s2[i] & dr[i])

1.2 Netlist Diagram



1.3 Simulation Metrics

Metrics	Reference Architecture
Worst negative slack in ps (setup analysis)	-0.37
Total power consumed	1.96 mW
Leakage power	29.15 uW
Area of combinational logic	2242.44
# of combinational cells	961
Total area	2846.52

2. Proposed Architecture

2.1 Design choices over Reference Architecture

We considered the following architectures to implement the adder.

1. Ripple-carry adder in a tree-based architecture.
2. A modified Ripple carry adder with a carry-lookahead adder
3. Behavioral model of an adder

Ripple Carry Adder

We implemented the ripple carry adder in a tree-based architecture. This does the addition in the same manner as the cascaded reference architecture, but it modifies the way the adders are placed, to give a better performance. The adders are now placed in a manner as shown below:

```
s1 = ar + br;
s2 = cr + dr;
sum = s1 + s2;
```

This gives some improvement in terms of latency as well as power and area numbers.

Carry lookahead Adder

We decided against the carry-lookahead adder for the reasons listed below:

1. The combinational logic for generating and propagating the carry of each stage would be too costly with respect to power and area consumption.
2. Carry lookahead adder was optimised only if the number of bits was precomputed. If we parameterize the number of bits, we have a general expression for generating the carry, which does not improve the efficiency as much. The general expression we wrote has been given below:

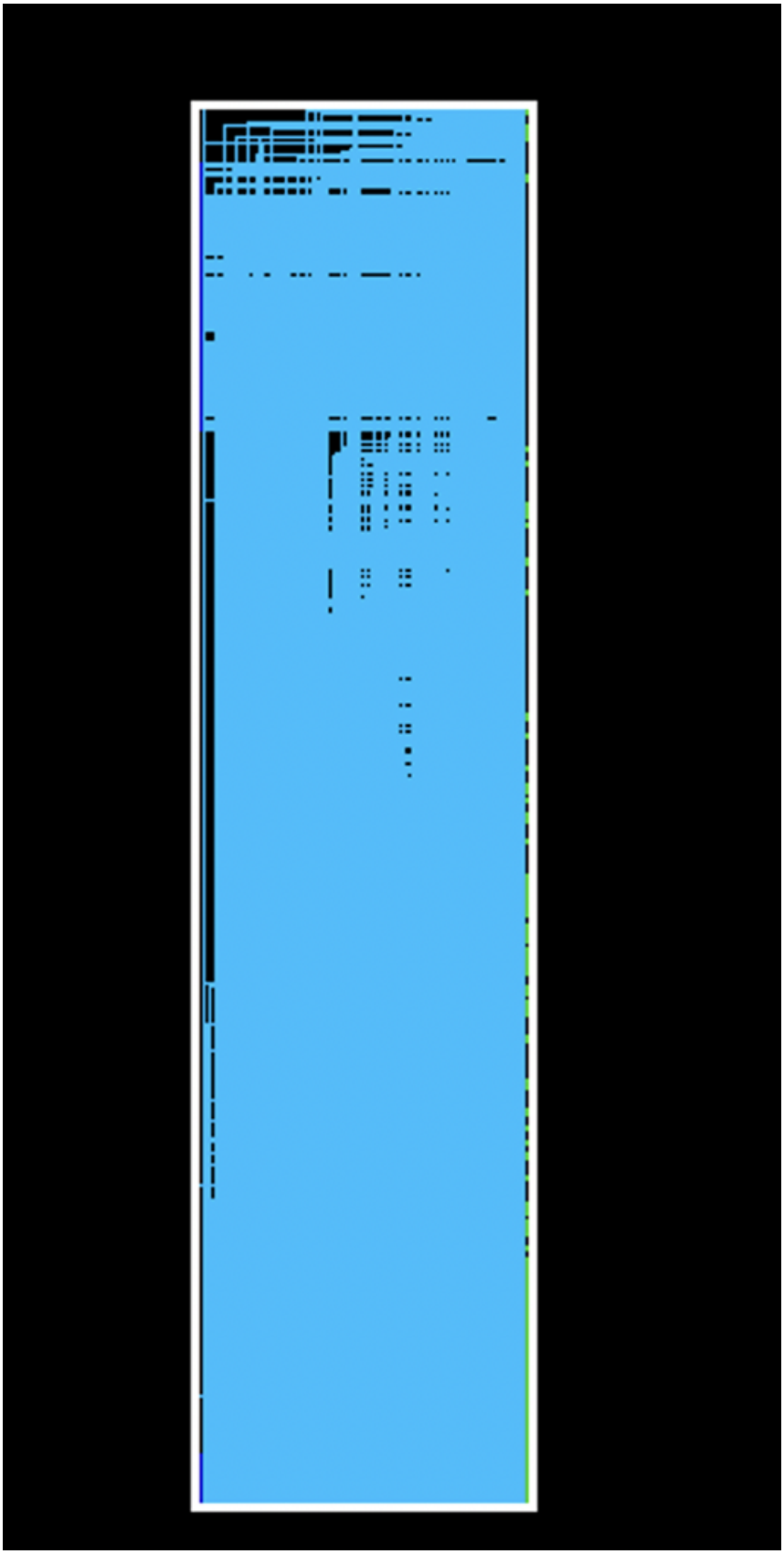
```
assign propagate[i] = ar[i] ^ br[i];
assign generate[i] = ar[i] & br[i];
assign carry[i+1] = generate[i] | (propagate[i] & carry[i]);
assign sum[i] = propagate[i] ^ carry[i];
```

Behavioral model of an Adder

We decided to make complete use of System Verilog's powers, and use a behavioral model of an adder.

1. This would allow the DC Shell to convert our Verilog code into the most optimal RTL schematic due to the number of wires and registers being sparse.
2. We found that the DC Shell does optimization based on resource utilization and power.
3. This reduced the latency overhead compared to the cascaded model, and in turn, the negative slack associated with the adder.
4. On RTL synthesis and inspection of power and area consumption, the behavioural model trumped the reference architecture in all aspects.
5. While the timing is better improved with the ripple carry adder, it comes at a large expense of high resources, which is why we have chosen this model with a little higher latency but much better power and area utilization.

2.2 Netlist Diagram



Netlist for the behavioral model.

2.3 Simulation Metrics

Metrics	Ripple carry with tree
Worst negative slack in ps (setup analysis)	-0.18
Total power consumed	1.84 mW
Leakage power	25.19 uW
Area of combinational logic	1820.52
# of combinational cells	826
Total area	2457.36

Metrics	Carry-Lookahead Model
Worst negative slack in ps (setup analysis)	-0.17
Total power consumed	1.88 mW
Leakage power	26.76 uW
Area of combinational logic	1877.4
# of combinational cells	814
Total area	2516.4

Metrics	**Behavioral Model**
Worst negative slack in ps (setup analysis)	-0.25
Total power consumed	1.61 mW
Leakage power	15.72 uW
Area of combinational logic	925.56
# of combinational cells	358
Total area	1519.2

2.4. What did not work as expected

1. Initially, we tried implementing a carry skip adder. The implementation required choosing between our initial carryIn and each stage's carryOut by multiplexing the inputs at each stage. Our implementation was faulted, and we shifted our focus on the carry-lookahead adder. Upon review, we found that our multiplexer's select was using a single wire which was getting updated over all stages, and we should have used an array instead.
2. The behavioral model optimized the design in terms of area, power and number of combinational units. However, it did not optimize the architecture in terms of latency as much as the tree structure of ripple carry adders.
3. We also considered using carry-select adders; however, carry-select adders have a larger area as compared to carry-lookahead adders, and would not provide as much an improvement in terms of timing due to the lesser number of bits used.

2.5 Future improvements in design

For future improvements, we can create a pipelined version of this adder. With pipelining, the number of combinational blocks between to DFFs would reduce. This allows us to increase the frequency and also gives one output every clock cycle. Another option can be to use the transpose architecture of the filter which would create a directly pipelined version by putting the combinational blocks between the DFFs.

Contributions:

Gandhar: Carry Lookahead implementation, cascaded implementation
Adithya: Carry-skip implementation, behavioral implementation

Report and analysis : Both equal contributions.