# Foundations of Computer Systems Design Lab (CS2310)
## Lab 6: Multipliers

1st October 2024

In this lab, we will learn how to implement multiplier using carry save adders and wallace tree techniques in Verilog.

**Note:** You are free to use any style of coding (not just structural) for this assignment.

## 1    Carry Save Multiplier (60 - Points)

Design a 4x4-bit unsigned carry save multiplier. The following figure represents carry save multiplication:

| | | | | $a_3$ | $a_2$ | $a_1$ | $a_0$ |
|---|---|---|---|---|---|---|---|
| | | | x | $b_3$ | $b_2$ | $b_1$ | $b_0$ |
| $p_{70}$ | $p_{60}$ | $p_{50}$ | $p_{40}$ | $p_{30}$ | $p_{20}$ | $p_{10}$ | $p_{00}$ |
| $p_{61}$ | $p_{51}$ | $p_{41}$ | $p_{31}$ | $p_{21}$ | $p_{11}$ | $p_{01}$ | x |
| $p_{52}$ | $p_{42}$ | $p_{32}$ | $p_{22}$ | $p_{12}$ | $p_{02}$ | x | x |
| $p_{43}$ | $p_{33}$ | $p_{23}$ | $p_{13}$ | $p_{03}$ | x | x | x |
| $z_7$ | $z_6$ | $z_5$ | $z_4$ | $z_3$ | $z_2$ | $z_1$ | $z_0$ |

Figure 1: Carry Save Multiplication

Write Verilog code that implements an unsigned multiplier using the carry save adder. The name of the file with the multiplier should be "*carry_save.v*". The module signature should be:
```
module carry_save(input [3:0] op1,op2, output [7:0] res);
```

## 2   Wallace Tree Multiplier (40 - Points)

Write Verilog code that implements a 4x4-bit unsigned wallace tree multiplier.
The name of the file with the multiplier should be "*wallace_tree.v*".

The module signature should be:
```
module wallace_tree(input [3:0] op1,op2, output [7:0] res);
```

- Use minimal number of carry-save adders and half-adders.

- Use a carry look-ahead adder for the last stage addition.

**Hint:** Draw out the design to get a better sense of the implementation

## 3   Bonus question

Write Verilog code that implements a 6x6-bit unsigned wallace tree multiplier.
The name of the file with the multiplier should be "*wallace_tree_6.v*".

The module signature should be:
```
module wallace_tree_6(input [5:0] op1,op2, output [7:0] res);
```

## *********** Take home bonus question *************

Implement a generalized unsigned multiplier to handle any operand size. The
name of the file with the multiplier should be "*bonus.v*".

The module signature should be:
```
module N_bit_mul #(parameter N) (input[N-1:0] op1,op2, output[2*N-1:0] res);
```

- To evaluate your design use the file "*test_bonus.sh*" provided along with
  "*tb_bonus.v*".

- Run "`chmod +x test_bonus.sh`" on the terminal to make the file executable

- Now you may run "`test_bonus.sh testbench_file.v`" to evaluate your
  design further for different sizes of the input

**NOTE:** This bonus question has more weightage than other bonus question
(including the ones from the previous labs)

**************************************************