RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

Graded Exercise VLSI Design WS24/25:

Binary Multiplier

Prof. Dr. P. Fischer

Lehrstuhl für Schaltungstechnik und Simulation
Uni Heidelberg

The Task

- In this exercise, you should create a full custom implementation of a digital ***binary multiplier*** circuit
  - The input are two N bit binary numbers A, B, i.e. bus signals A⟨N-1:0⟩ and B⟨N-1:0⟩
  - We restrict ourselves to positive values, i.e. A≥0, B≥0
  - The result is a 2N bit wide SUM vector
  - You may choose N=4…8. If you work systematically, there is nearly no extra effort for larger N.
- We will implement the multiplier as a very simple 'array multiplier'. This is not very efficient, there are much more clever implementations ('Wallace Tree', 'Booth Encoding')
- You should do
  - A hierarchical schematic
  - Analog simulations of basic elements
  - A Mixed Mode Simulation of the full multiplier
  - A nice, compact layout which is DRC and LVS clean.

How Multiplication Works

- Multiplication of integers can be done with the 'school' method.
- One input number (here A=13) is multiplied with all digits of B (here 1234) -> 'partial products'
- The partial products are added, respecting digit position
- The summing can create overflows (carry) from a lower to a high digit position

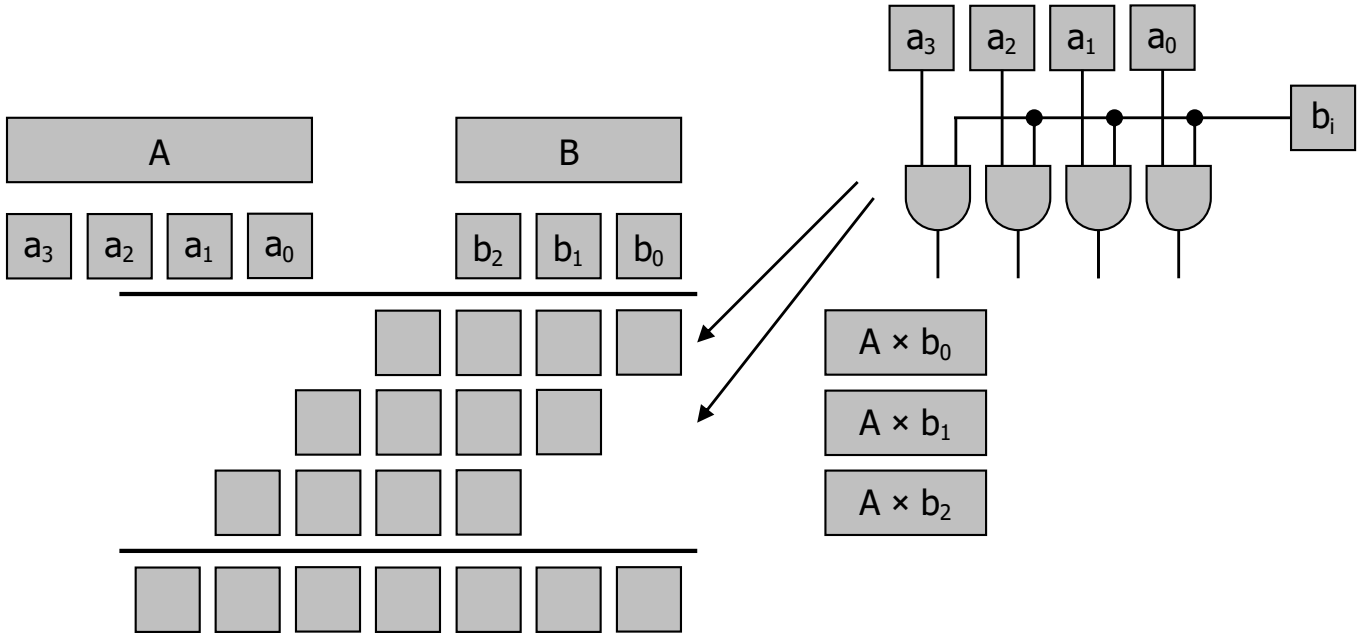
```
   A      B

  13 x 1234
        52  <---  13 x 4
       39   <---  13 x 30   = 13 x 3, shifted 1 to the left
      26    <---  13 x 200  = 13 x 2, shifted 2 to the left
     13     <---  13 x 1000 = 13 x 1, shifted 3 to the left
Carry    1 1
Sum:   16042
```

'partial product'

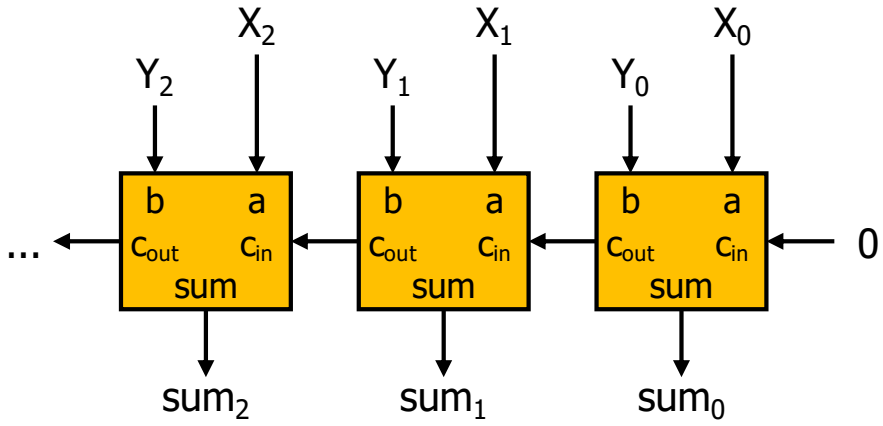
Multiplication With Binary Numbers

- In binary, digits can only take 0/1, so the row multiplication is trivial:
  - A x 0 = 0
  - A x 1 = A
- We need just an AND gate on all bit positions!



Summing Up

- Adding up the upper 2 rows requires a digital adder
- This is made up from full adder circuits:



c_{in}	b	a	c_{out}	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- A carry signals propagates from the least significant digit to higher digits
- The first position has no carry-in, so a Half Adder can (could) be used.
- The sum is then added to the next row etc.
- Overall, N-1 rows of adders are required.

The Full Adder

- You can learn about digital logic in the slides of ‘Digitale Schaltungstechnik’ in WS21/22
  - Slides ‘Kombinatorik und Flipflops’, p15ff.

- There are many ways to make a full adder:
  - You find an efficient transistor level implementation of a full adder in the slides
  - You can also make it from two half adders
  - Or you use basic gates

1. Schematic

- Make schematics+symbols of the AND gate and the Full Adder.
- You may also make a schematic/symbol of an adder bit with the AND in front.
- Make a schematic+symbol of an N bit adder. It may be clever to have a ‘normal’ input and an input which is ‘ANDed’ with a control bit. Use bus notation to keep the schematic simple!
- Assemble the multiplier from the adders and some more stuff. Use bus notation!

(Optimizations)

- To optimize the circuit a bit you could
  - Use a NAND instead of the AND and handle the inverted signal somehow ‘later’
  - Similarly live with inverting Full Adders
  - Use only a Half Adder at the first position

- As mentioned, there are more efficient ways to add up all the bits, see ‘Wallace Tree’, ‘Dada-Tree’,…

2. Analogue Simulation

- Simulate the basic cells.
  - Check that behavior is correct for all bit combinations.
- Simulate the adder for some values of X,Y
- What is the worst case speed (when a carry propagates fully through the chain) ?

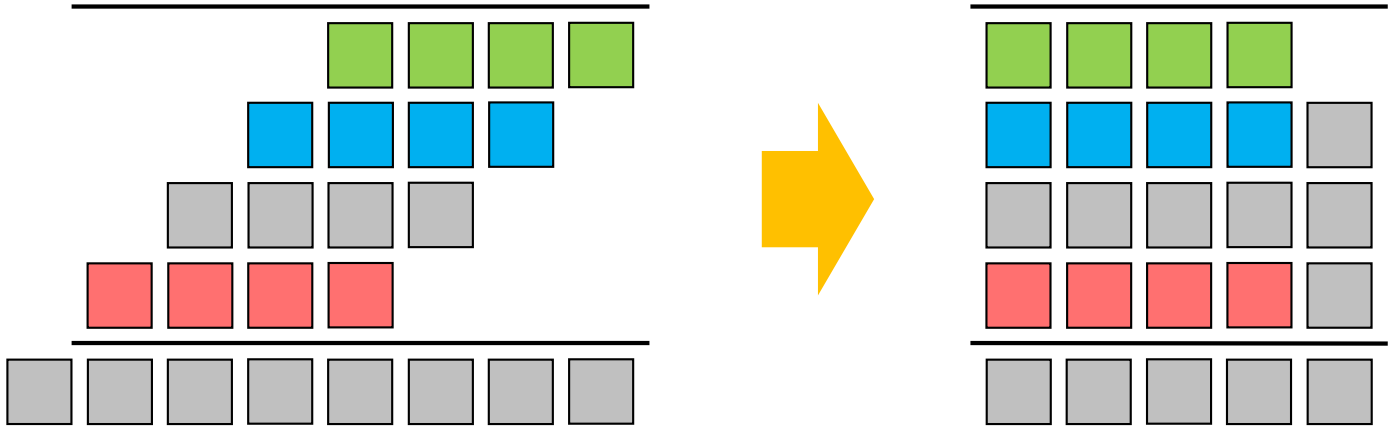
3. Mixed Mode Simulation

- Check with a mixed mode simulation that the adder works correctly for some random numbers.
- Check the full multiplier for some ‘typical’ cases and for random numbers.

- How long does it need in a worst case until the result is complete. What is the ‘critical path’?

4. Layout

- Make a compact layout of the FullAdder. Arrange the pins, power, … such that you can place cells next to each other.
- Make the layouts of all cells in the schematic hierarchy.
- Try to make the top cell rectangular by placing the adders under each other (not shifted, as shown left)



- The layout must be DRC and LVS clean

Report

- Provide a document (10-15 pages) with
  - A summary of the task and how you solved it.
  - Explanations on how the circuits work, why you chose them as they are, and how you set up the hierarchy
  - Schematics (just screen shots), using a decent hierarchy with some explanations
  - Simulation results (function, speed) with explanations
  - Results of the mixed mode simulation + Verilog code
  - A ‘nice’ layout (screenshots). Explain your reasonings.

Bonus

- For a very good grade
  - your description should be complete and decently formatted
  - Schematics should be tidy
  - You may simulate transistor speed corners to see how slow the circuit can get in the worst case.
  - Layout should be rather compact (most space occupied my MOS)

  - You could implement an adder of variable size as a PCELL, similarly for the full multiplier, so that you can instantiate a NxM multiplier.
  - You may have a look at the Wallace Tree concept, explain how that works and why it is clever.