EXAMPLE MULTIPLIER

Example 20 bit x 20 bit Multiplier

- 20 bit × 20 bit signed 2’s complement multiplier
- “Full” non-iterative datapath (one multiply per cycle)
- Four pipeline stages
  - Four cycles of latency
- “Booth-2” *multiplier* encoding
- Complex sign extension techniques eliminates sign extension bits with Booth-2 staggered partial products

Example 20 bit x 20 bit Multiplier

- Full output is 20 + 20 = 40 bits
  - Will need to round down to 20 bits later
  - Choose to keep a 24-bit output for later calculations
  - The motivation can be illustrated with a base-10 example
    - The lowest 4 digits of the product have little meaning

```
     0.5213
   × 0.8392
 ----------
 0.43747496
```

  - In this example multiplier, the partial product array is truncated, and the output is rounded to keep the mean of the rounding error equal to zero. This array truncation results in a 27% reduction of partial product array hardware. Although the savings are substantial, this method is generally not recommended due to greatly increased difficulty in writing a matching golden reference, and increased irregularity in the hardware.

Guideline #1 for Placing Carry-Save Adders

- Each "dot" that exists at the end of a pipeline stage must be registered (one high-power large-area clock-requiring flip-flop)
  - Of course constant "1"s can be placed anywhere and are never registered
  - There is no such thing as a "0" in a dot diagram
- Therefore, the best design will minimize the number of dots in each stage by placing as many full adders (eliminates one dot) and 4:2 adders (eliminates two dots)

Guideline #2 for Placing Carry-Save Adders

- Each pipeline stage requires a large number of flip-flops
- Each pipeline stage adds one cycle of latency to the datapath
  - This often causes a penalty at a higher level of the architecture. In some cases the penalty is significant, in some cases it is negligible.
- Therefore, place carry-save adders to minimize the number of pipeline stages

Example 20 bit x 20 bit Multiplier

- Notation for upcoming dot diagrams:

```
. = input_bit, can be either a 0 or 1

, = NOT(.)

0 = always zero

1 = always one

S = the partial product sign bit

E = bit to clear out sign_extension bits

e = NOT(E)

- = carry_out bit from (4,2) or (3,2) adder in adjacent
    column to the right

x = throw this bit away
```

Example 20 bit x 20 bit Multiplier

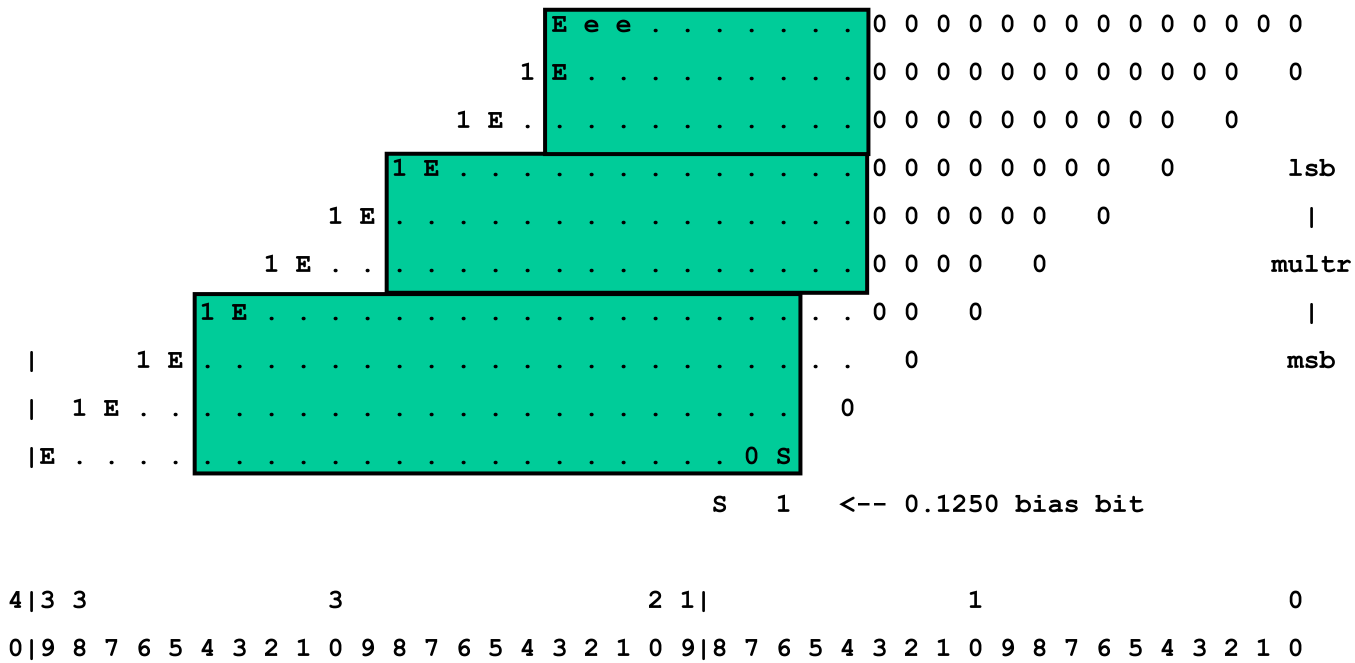
- Booth-encoded partial product array

```
                                  E e e . . . . . . . 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                                1 E . . . . . . . . . 0 0 0 0 0 0 0 0 0 0 0 0   0
                            1 E . . . . . . . . . . . 0 0 0 0 0 0 0 0 0 0   0
                        1 E . . . . . . . . . . . . . 0 0 0 0 0 0 0 0   0      lsb
                    1 E . . . . . . . . . . . . . . . 0 0 0 0 0 0   0            |
                1 E . . . . . . . . . . . . . . . . . 0 0 0 0   0             multr
            1 E . . . . . . . . . . . . . . . . . . . 0 0   0                    |
 |      1 E . . . . . . . . . . . . . . . . . . . . .   0                      msb
 |  1 E . . . . . . . . . . . . . . . . . . . . .   0
 |E . . . . . . . . . . . . . . . . . . . . . 0 S
                                            S   1   <-- 0.1250 bias bit

4|3 3               3                   2 1|                1                   0
0|9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9|8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

Example 20 bit x 20 bit Multiplier

- 3:2 and 4:2 adders reduce partial products



Example 20 bit x 20 bit Multiplier

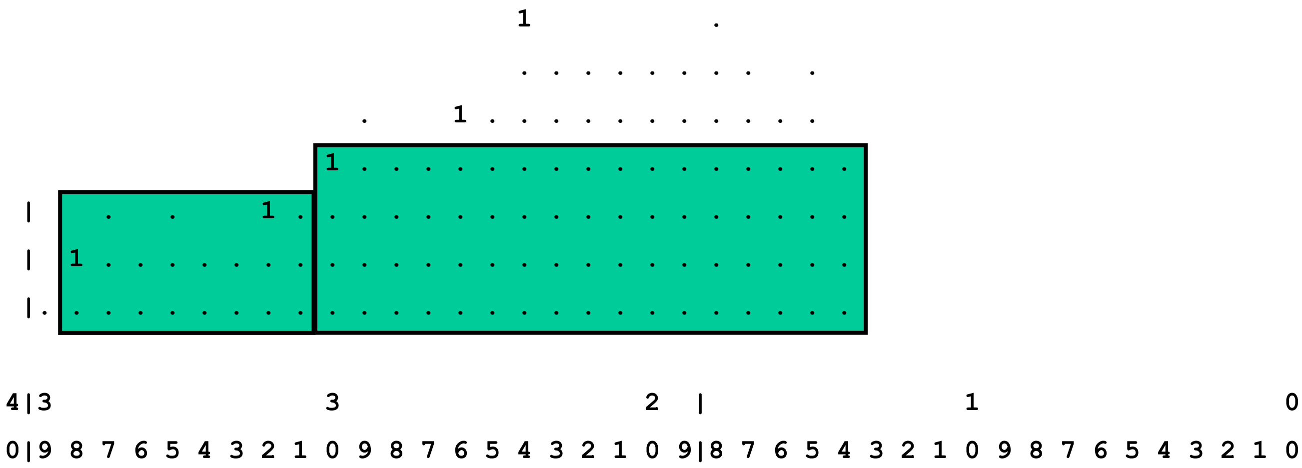
- After one level of adders

```
                                1           .
                                . . . . . . . .   .
                      .     1 . . . . . . . . . . .
                    1 . . . . . . . . . . . . . . . .
 |    .   .     1 . . . . . . . . . . . . . . . . . .
 |  1 . . . . . . . . . . . . . . . . . . . . . . . .
 |. . . . . . . . . . . . . . . . . . . . . . . . . .

4|3                 3                   2  |                1                   0
0|9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9|8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

Example 20 bit x 20 bit Multiplier

- Another pass of 3:2 and 4:2 adders



Example 20 bit x 20 bit Multiplier

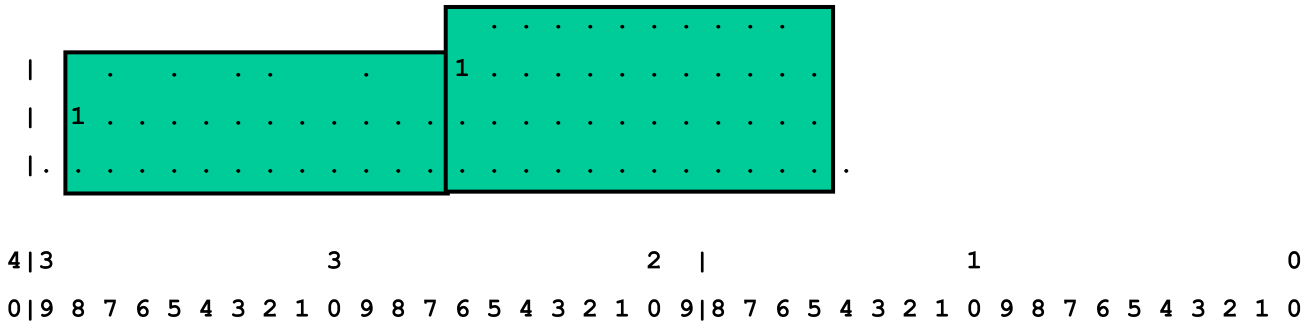
- After two levels of adders

```
                                  . . . . . . . . . .
 |    .   .   . .    .    1 . . . . . . . . . . .
 |  1 . . . . . . . . . . . . . . . . . . . . . . .
 |. . . . . . . . . . . . . . . . . . . . . . . . . .

4|3                 3                   2  |                1                   0
0|9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9|8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

Example 20 bit x 20 bit Multiplier

- Another pass of 3:2 and 4:2 adders

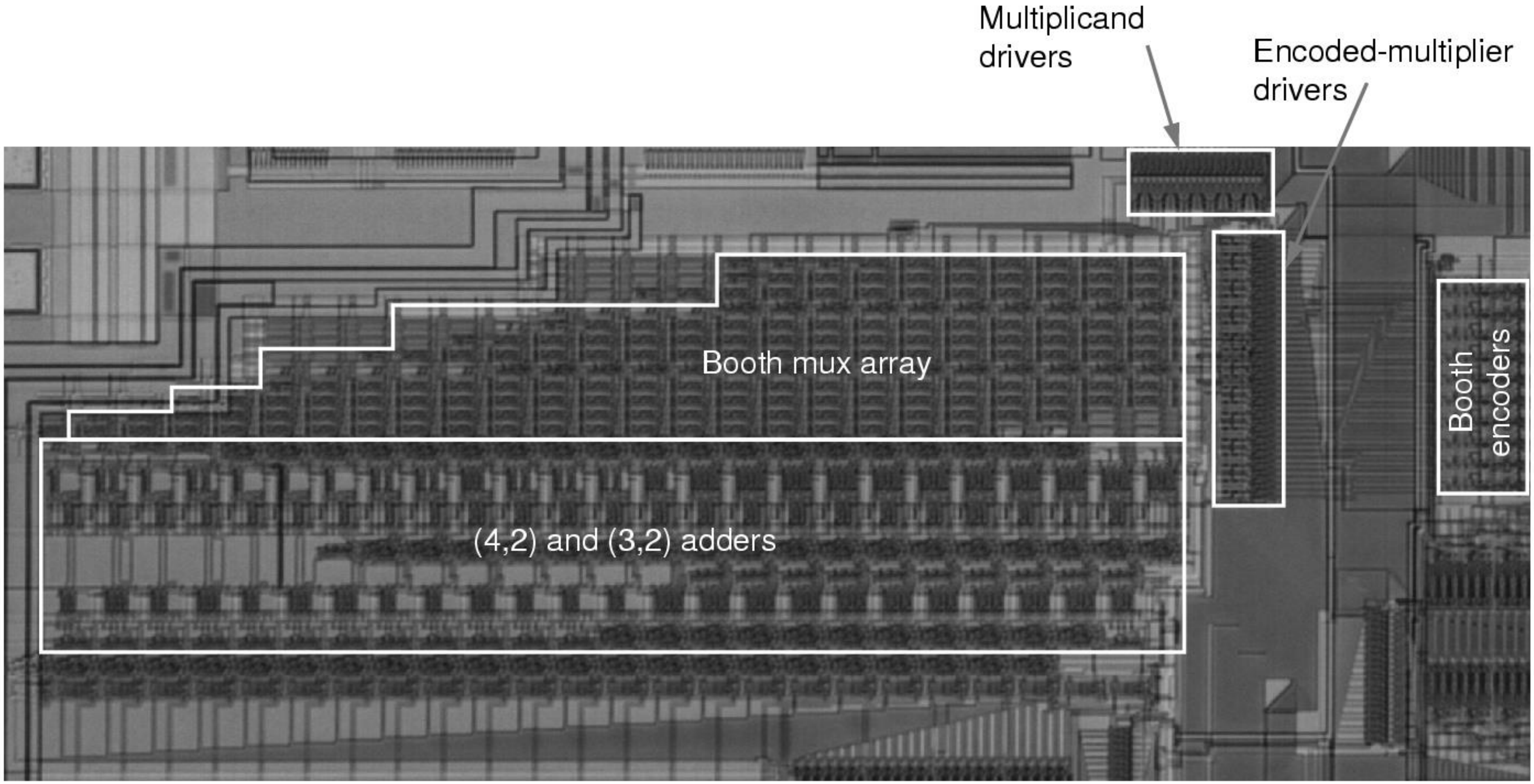


Example 20 bit x 20 bit Multiplier

- After three levels of adders, it’s ready for a carry-propagate adder!

```
 |. . . . . . . . . . . . . . . . . . . . . . .
 |. . . . . . . . . . . . . . . . . . . . . . . . . x
                                                    0 1
4|3                 3                   2  |                1                   0
0|9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9|8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

Die Photo


Multiplicand drivers
Encoded-multiplier drivers
Booth mux array
Booth encoders
(4,2) and (3,2) adders