FIXED-INPUT MULTS
Fixed-input Multiplier

• This topic deals with a single multiply operation where one of the inputs is fixed—by this we mean one of the multiplier’s inputs is known at the time of hardware design, will never change, and is therefore designed into the hardware

• Simplifications will generally place the fixed input onto the multiplier input rather than the multiplicand input
Fixed-input Multiplier

- Sometimes, one input is fixed
  - So remove partial products that are always zero

![Diagram of a fixed-input multiplier]
Fixed-input Multiplier

- Remove partial products that are always zero
Fixed-input Multiplier

- Reduce the overall size by half on average, and even more if you can pick the *multiplier* carefully.
Fixed-input Multiplier

• An efficient design would instantiate hardware for only the necessary (non-zero) partial products— the minimum number of power-of-2 numbers which add together to equal the fixed multiplier input
• Secondly, we look for cases when a negative partial product will simplify the overall circuit
• A very simple example: multiply by 3
  \[ \times 3 = (\times 2) + (\times 1) \]
Multiply by 3

- Ex: multiply by 3
  - $\times 3 = (\times 2) + (\times 1)$
  - Verilog:
    ```verilog
    input [7:0] in;
    wire [9:0] product;
    // multiply by 3
    assign product = {in[7], in[7], in} + {in[7], in, 1'b0};
    ```
Handling Negative Partial Products

- There are two ways to handle negative partial products; the dot diagram is different but both give the same result and eventually simplify to the same hardware.
  1. Multiply the *multiplicand*, then invert it (Ex: 6-bit multiplicand $\times -8$)

```
  S S ● ● ● ● ● ● 0 0 0
  S S ● ● ● ● ● ● 1 1 1
  S S 1

  S S ● ● ● ● ● ● 1 1 1
```

2. Invert the *multiplicand*, then multiply it

```
  S S S S S S ● ● ● ● ● ● 1

  S S ● ● ● ● ● ● 0 0 0
  S S 1
```
Multiply by 56

- Ex: 8-bit 2’s complement number multiplied by +56
  - Three partial products
  - $\times 56 = (\times 32) + (\times 16) + (\times 8)$
  - Verilog:

```verilog
input [7:0] in;
wire [13:0] product;

// multiply by 56
assign product =
  {in[7], in[7], in[7], in, 3'b000} +
  {in[7], in[7], in, 4'b0000} +
  {in[7], in, 5'b00000};
```

```plaintext

\[
\begin{array}{c}
\text{S} \\
\text{S} \\
\text{S} \\
\text{S} \\
\text{in} \\
\text{in} \\
\text{in} \\
\text{in} \\
\text{in} \\
\text{in} \\
000000
\end{array}
\]
```
Multiply by 56 (better)

- Ex: 8-bit 2’s complement number multiplied by +56 (better)
  - Implement with 2 partial products
  - \( \times 56 = (\times 64) - (\times 8) \)
  - Verilog:

```verilog
input [7:0] in;
wire [13:0] product;

// multiply by 56
assign product =
{in, 6'b000000}
- {in[7], in[7], in[7], in, 3'b000};

// one way of many possible ways to write it
assign product_same =
{in, 6'b000000}
+ {~in[7], ~in[7], ~in[7], ~in, 3'b111};
+ {14'b00000000000001};
```
Dot Diagram Example

- out = X*3 + Y*56
  - Inputs: 6-bit 2’s complement
- Procedure:
  - Input range of X and Y: [-32, +31]
  - Decompose X*3 = X*2 + X*1
  - Decompose X*56 = X*64 - X*8
  - Output range: [-32, +31] × [59=3+56]
    = [-1888, +1829]
  - Output width: 12 bits, [-2048, +2047]
  - Fill out dot diagram
    - S = sign extension bit
    - invert bits when necessary
    - show zeros if dot alignment is not obvious