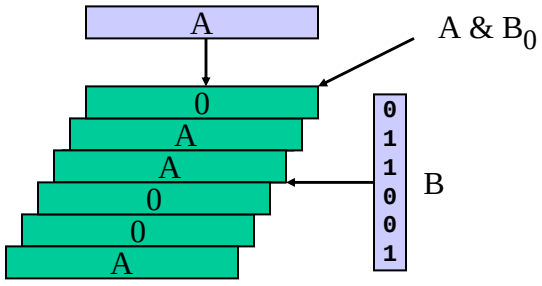
FIXED-INPUT MULTS

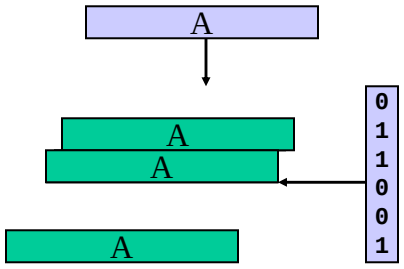
Fixed-input Multiplier

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



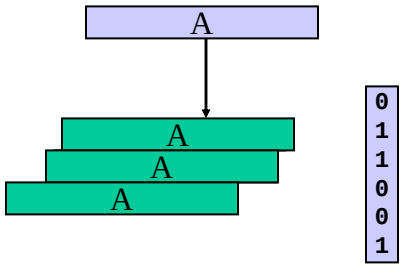
Fixed-input Multiplier

- Remove partial products that are always zero



Fixed-input Multiplier

- Reduce size by half on average, often more if you can pick the "multiplier" carefully



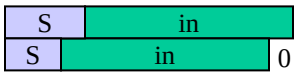
Fixed-input Multiplier

- The goal is to find the minimum number of power-of-2 numbers to add together to equal the fixed multiplier input
- Ex: multiply by 3

  ×3 = (×2) + (×1)

Multiply by 3

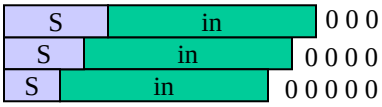
- Ex: multiply by 3
  - x3 = (x2) + (x1)
  - Verilog:




```
input [7:0] in;
wire  [9:0] product;
// multiply by 3
assign product = {in[7], in, 1'b0}
               + {in[7], in[7], in};
```

Multiply by 56

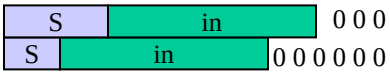
- Ex: multiply by 56
  - x56 = (x32) + (x16) + (x8)
  - Verilog:




```
input [7:0] in;
wire  [13:0] product;
// multiply by 56
assign product =
   {in[7], in, 5'b00000}
 + {in[7], in[7], in, 4'b0000}
 + {in[7], in[7], in[7], in, 3'b000};
```


Multiply by 56 (better)

- Ex: multiply by 56 (better)
  - x56 = (x64) – (x8)
  - Verilog:

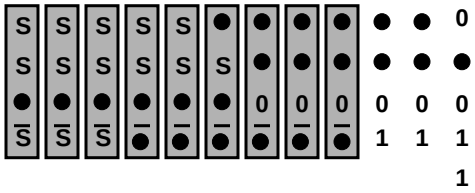



```
input [7:0] in;
wire  [13:0] product;
// multiply by 56
assign product =
   {in, 6'b00000}
 - {in[7], in[7], in[7], in, 3'b000};


assign product_same =
   {in, 6'b00001}
 + {~in[7], ~in[7], ~in[7], ~in, 3'b111};  // one way of many
```

Dot Diagram Example

- out = X*3 + Y*56
  - Inputs: 6-bit 2's complement
- Procedure
  - Input range: [-32, +31]
  - Decompose ×3 = ×2 + ×1
  - Decompose ×56 = ×64 – ×8
  - Output range: [-32, +31] × [59] = [-1888, +1829]
  - Output width: 12 bits
  - Fill out dot diagram
    - S = sign extension bit
    - invert bits when necessary
    - show zeros if dot alignment is not obvious
- Two approaches of thinking of negative partial products (with identical product of course)
  1) Shift PP then invert entire word
  2) Invert PP then shift



- OR -