Arithmetic Circuits-2

- Multipliers
  - Array multipliers
- Shifters
  - Barrel shifter
  - Logarithmic shifter

Binary Multiplication

$$X = \sum_{i=0}^{M-1} X_i \, 2^i \qquad Y = \sum_{i=0}^{N-1} Y_i \, 2^i$$

Multiplicand — Multiplier

Product $Z = X * Y$

$$= \sum_{i=0}^{N-1} \left(\sum_{j=0}^{M-1} X_i Y_j 2^{i+j} \right)$$

Partial products

- Product = Sum of partial products

Multiplication

- Example:

$$\begin{array}{rl} 1100 & : \ 12_{10} \\ \underline{0101} & : \ 5_{10} \end{array}$$

Multiplication

- Example:

$$\begin{array}{rl} 1100 & : \ 12_{10} \\ \underline{0101} & : \ 5_{10} \\ 1100 & \end{array}$$

Multiplication

- Example:

$$\begin{array}{rl} 1100 & : \ 12_{10} \\ \underline{0101} & : \ 5_{10} \\ 1100 & \\ 0000 & \end{array}$$

Multiplication

- Example:

```
  1100  :  12(10)
  0101  :  5(10)
  ----
  1100
 0000
1100

--------
```

Multiplication

- Example:

$$\begin{array}{rl} 1100 & : \ 12_{10} \\ \underline{0101} & : \ 5_{10} \\ 1100 & \\ 0000 & \\ 1100 & \\ \underline{0000} & \end{array}$$

Multiplication

- Example:

$$
\begin{array}{rl}
1100 & : 12_{10} \\
\underline{0101} & : 5_{10} \\
1100 & \\
0000 & \\
1100 & \\
\underline{0000} & \\
00111100 & : 60_{10}
\end{array}
$$

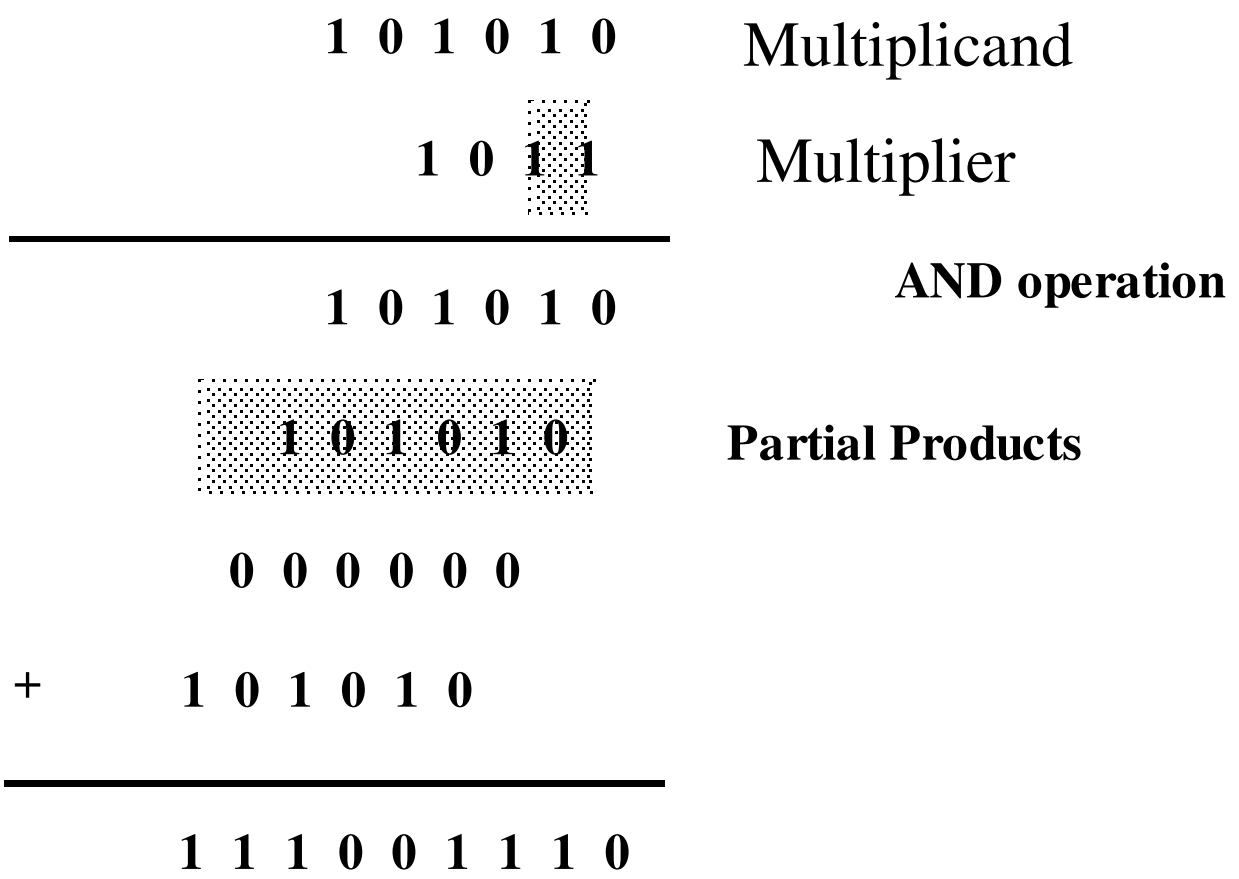
Multiplication

- Example:

```
   1100  :  12_10          multiplicand
   0101  :  5_10           multiplier
   1100                    ┐
  0000                     │ partial
 1100                      │ products
 0000                      ┘
--------
00111100 :  60_10          product
```

- M x N-bit multiplication
  - Produce N M-bit partial products
  - Sum these to produce M+N-bit product

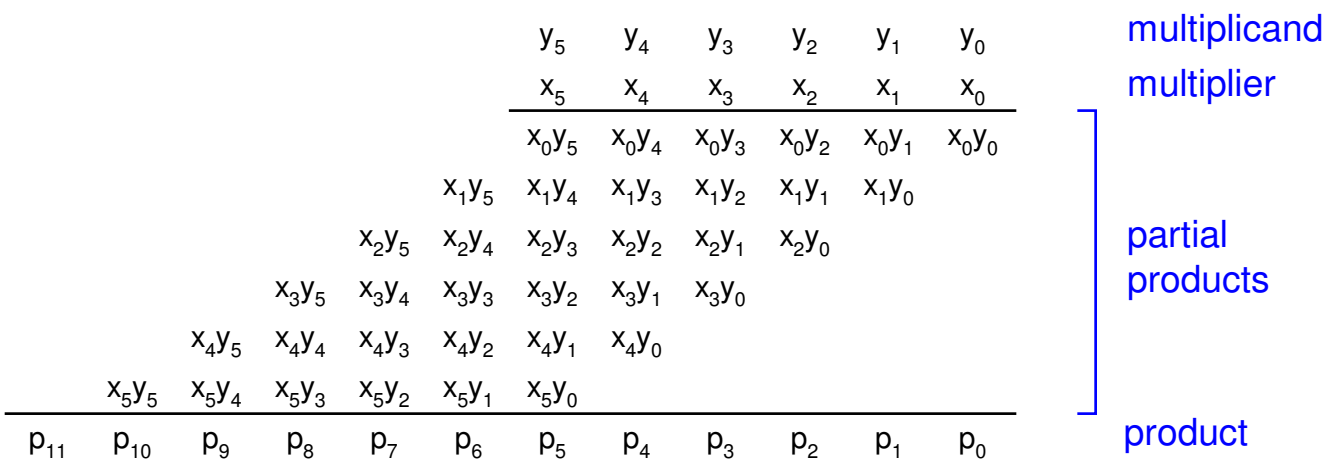
The Binary Multiplication



General Form

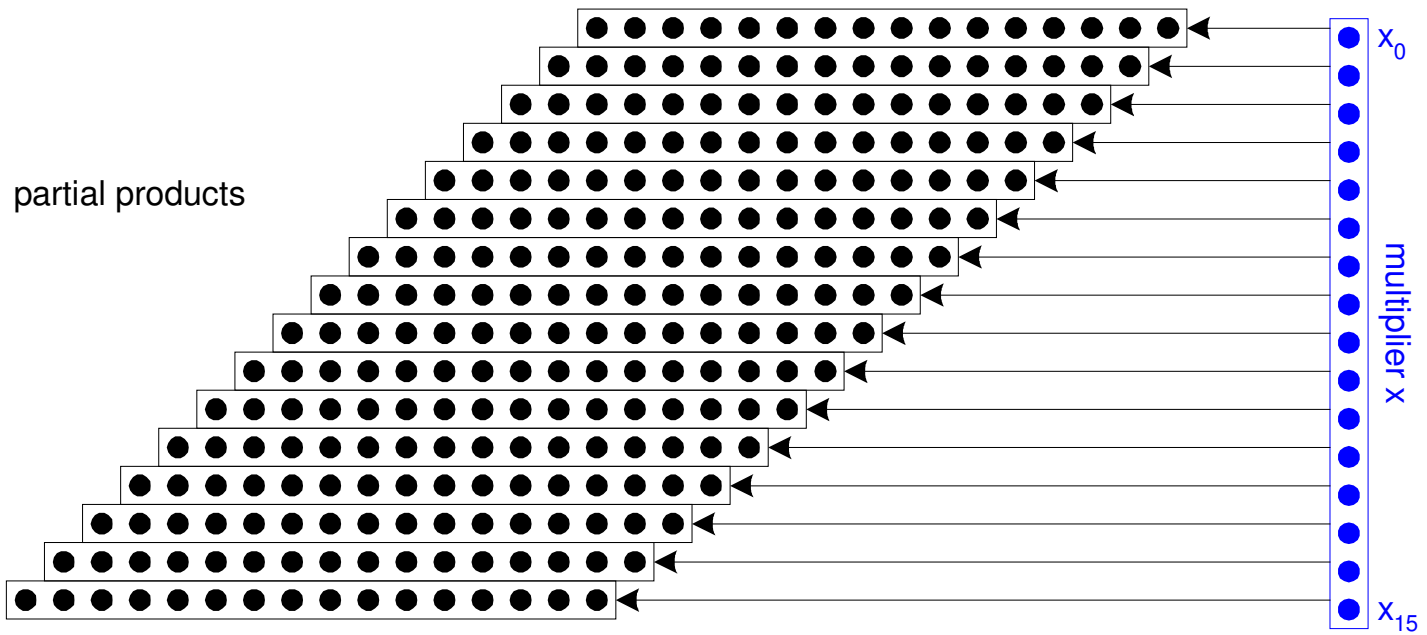
- Multiplicand: $Y = (y_{M-1}, y_{M-2}, \ldots, y_1, y_0)$
- Multiplier: $X = (x_{N-1}, x_{N-2}, \ldots, x_1, x_0)$
- Product:

$$P = \left(\sum_{j=0}^{M-1} y_j 2^j \right) \left(\sum_{i=0}^{N-1} x_i 2^i \right) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_i y_j 2^{i+j}$$

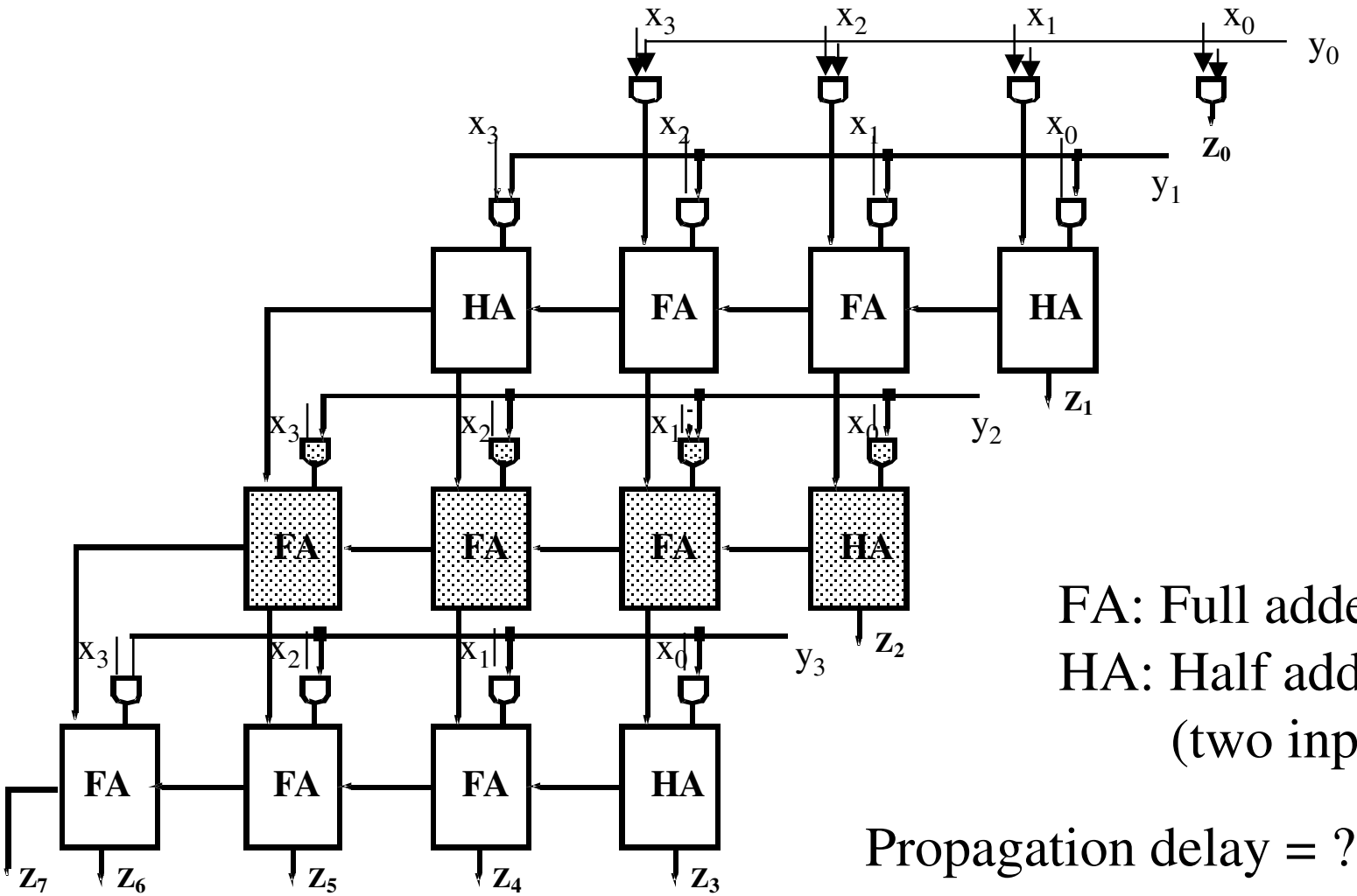


Dot Diagram

- Each dot represents a bit



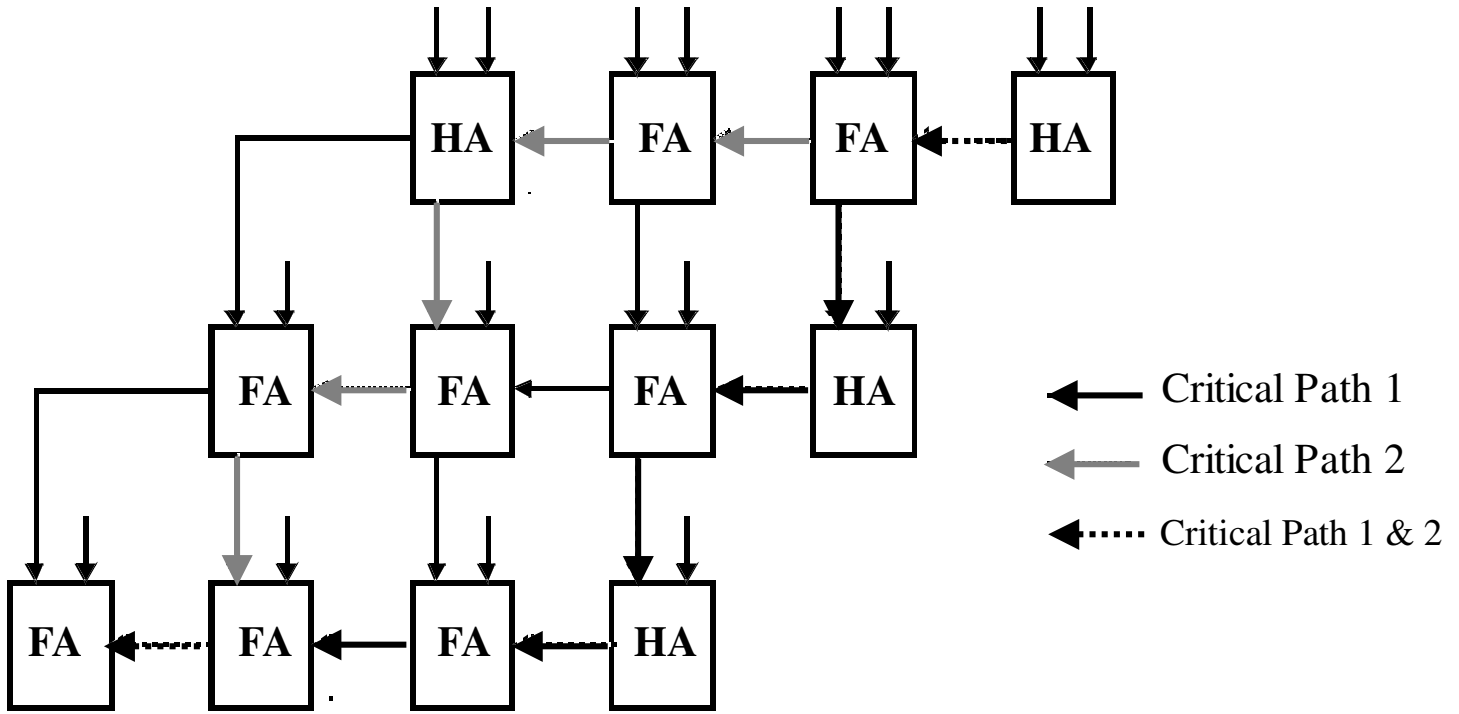
The Array Multiplier



FA: Full adder
HA: Half adder
(two inputs)

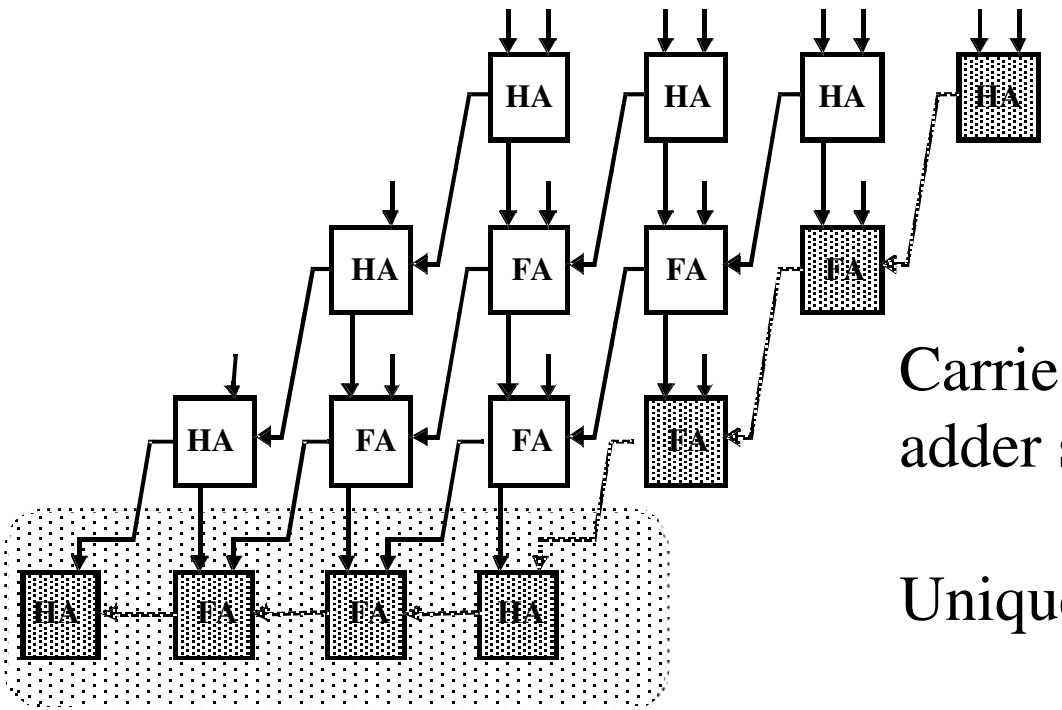
Propagation delay = ?

The MxN Array Multiplier — Critical Path



$$t_{mult} \approx [(M-1)+(N-2)]t_{carry}+(N-1)t_{sum}+(N-1)t_{and}$$

Carry-Save Multiplier



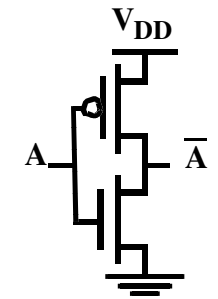
Carries saved for next adder stage

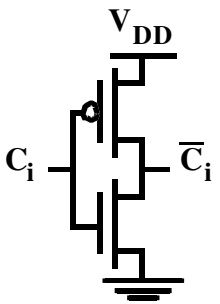
Unique critical path

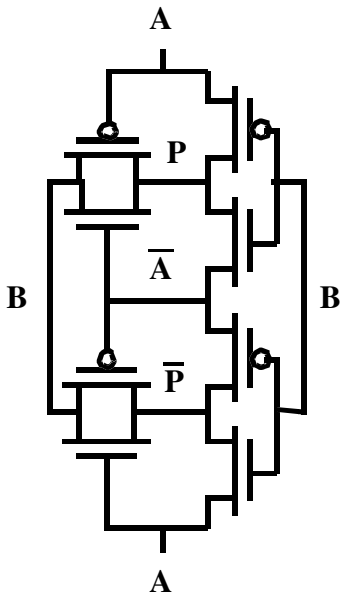
Trade offs?

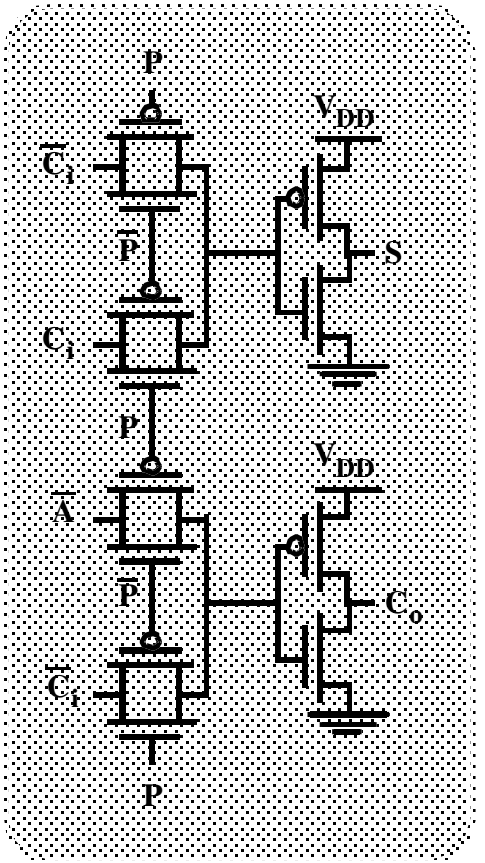
$$t_{mult} = (N-1)t_{carry} + (N-1)t_{and} + t_{merge}]$$

Adder Cells in Array Multiplier



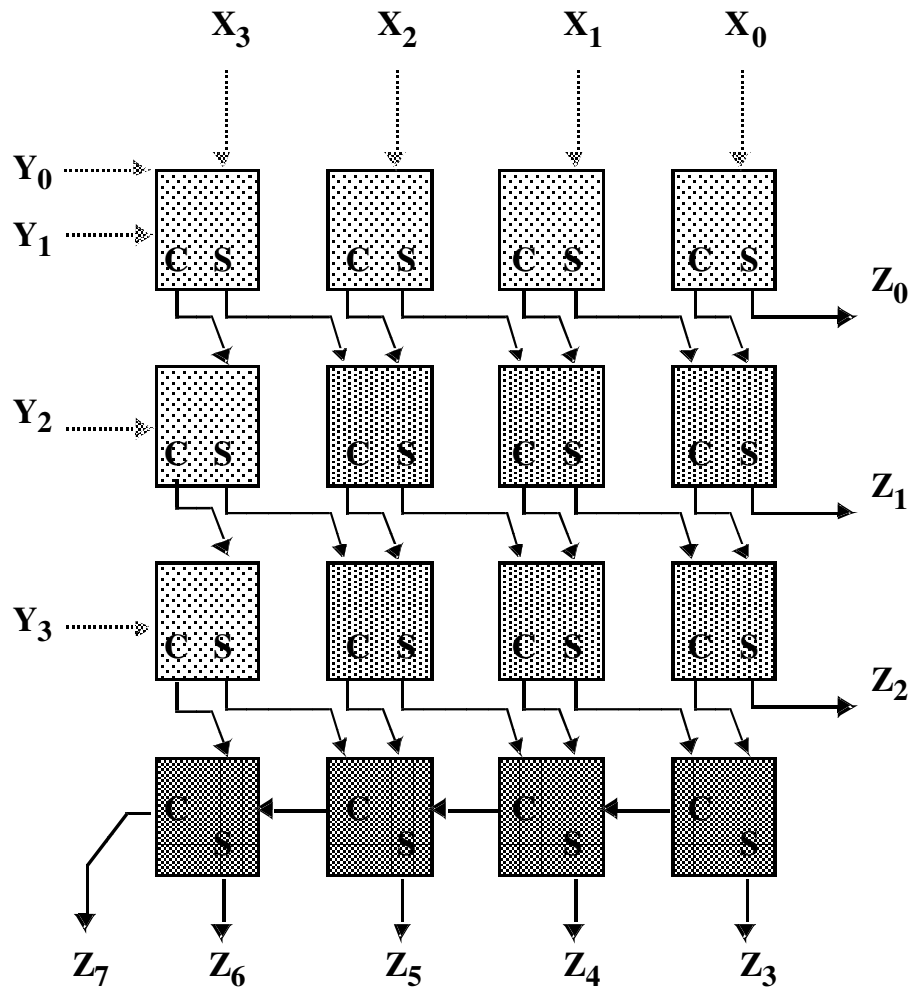






Identical Delays for Carry and Sum

Multiplier Floorplan



HA Multiplier Cell

FA Multiplier Cell

Vector Merging Cell

X and Y signals are broadcasted through the complete array.

Multipliers —Summary

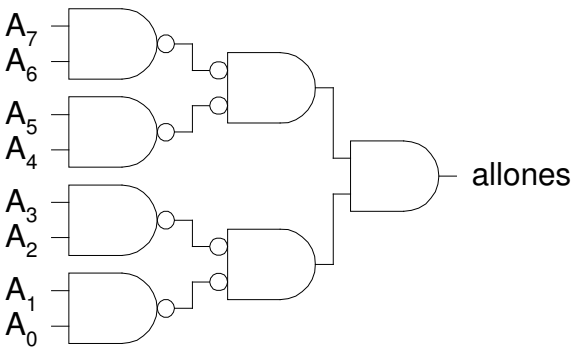
- **Optimization Goals Different Vs Binary Adder**
- **Once Again: Identify Critical Path**
- **Other possible techniques**
  - **Logarithmic versus Linear** (Wallace tree multiplier)
  - **Data encoding (Booth)**
  - **Pipelining**

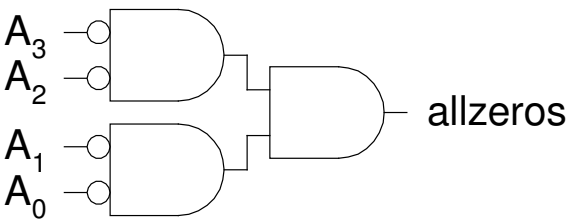
Comparators

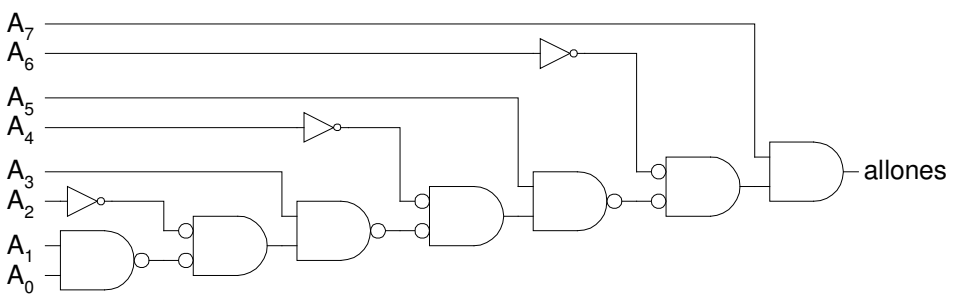
- 0's detector: $A = 00\ldots000$
- 1's detector: $A = 11\ldots111$
- Equality comparator: $A = B$
- Magnitude comparator: $A < B$

1's & 0's Detectors

- 1's detector: N-input AND gate
- 0's detector: NOTs + 1's detector (N-input NOR)

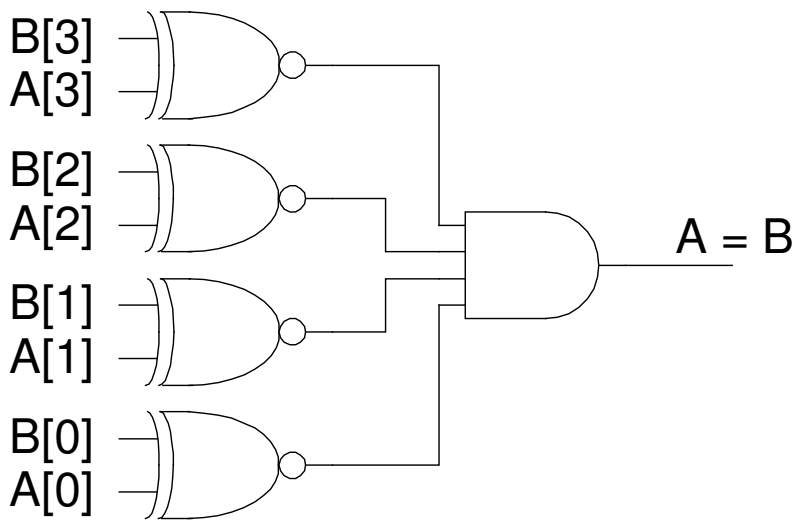






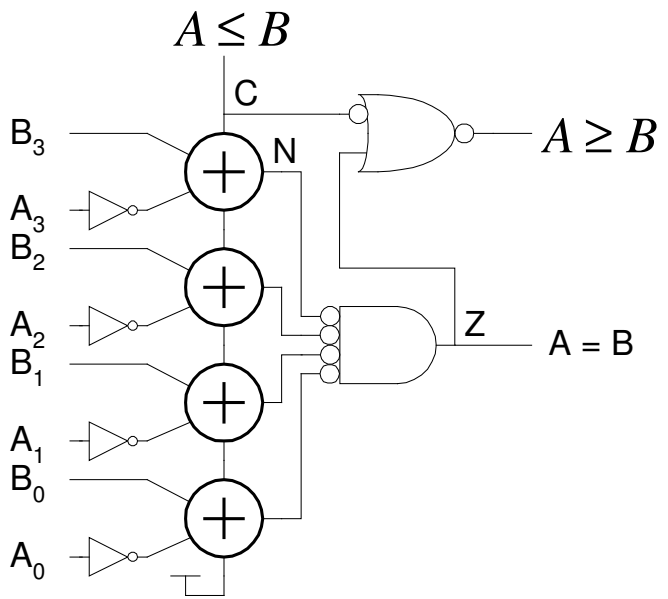
Equality Comparator

- Check if each bit is equal (XNOR, aka equality gate)
- 1's detect on bitwise equality



Magnitude Comparator

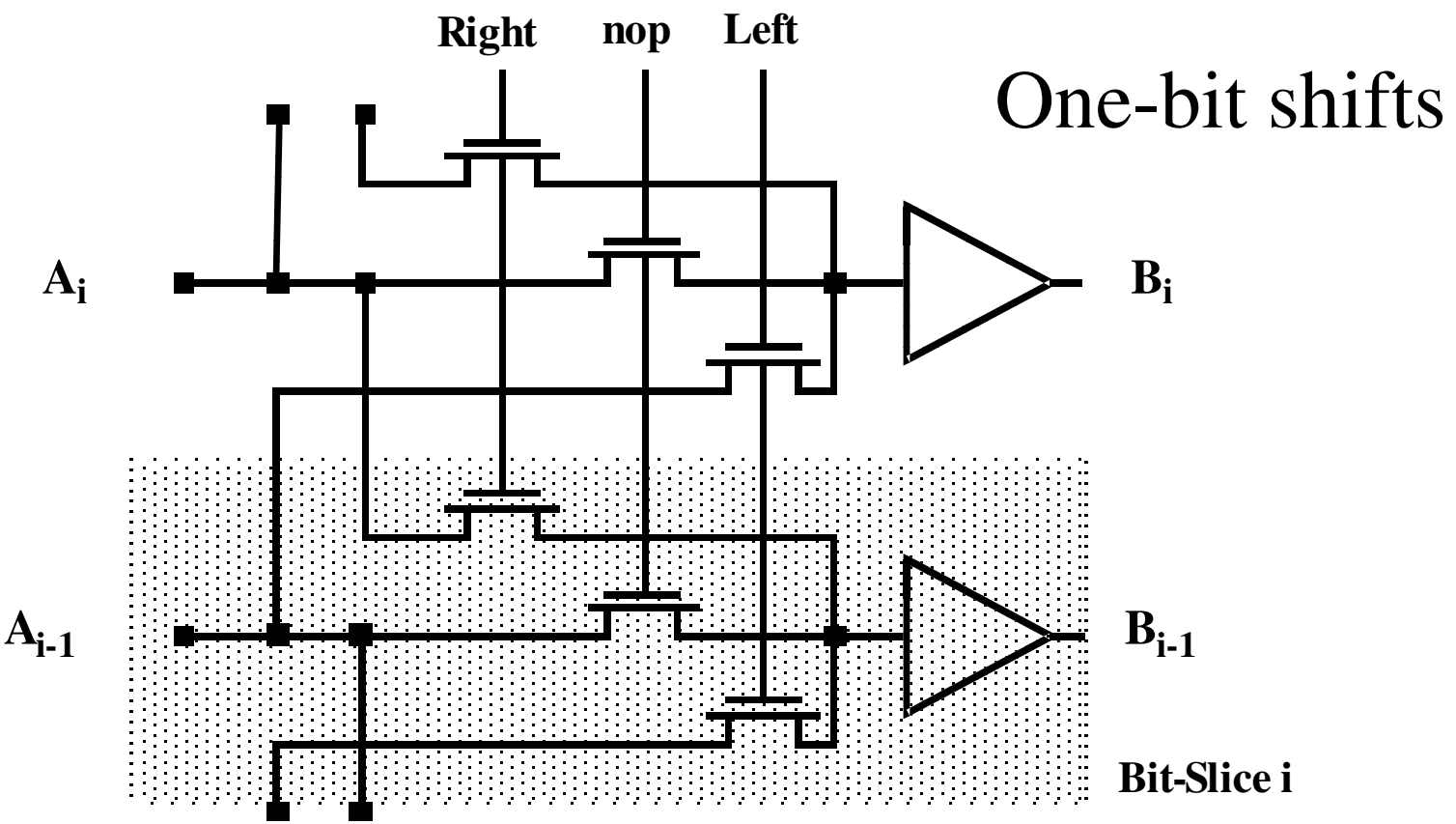
- Compute B-A and look at sign
- B-A = B + ~A + 1
- For unsigned numbers, carry out is sign bit



Shifters

- Logical Shift:
  - Shifts number left or right and fills with 0's
    - 1011 LSR 1 = 0101 1011 LSL1 = 0110
- Arithmetic Shift:
  - Shifts number left or right. Rt shift sign extends
    - 1011 ASR1 = 1101 1011 ASL1 = 0110
- Rotate:
  - Shifts number left or right and fills with lost bits
    - 1011 ROR1 = 1101 1011 ROL1 = 0111

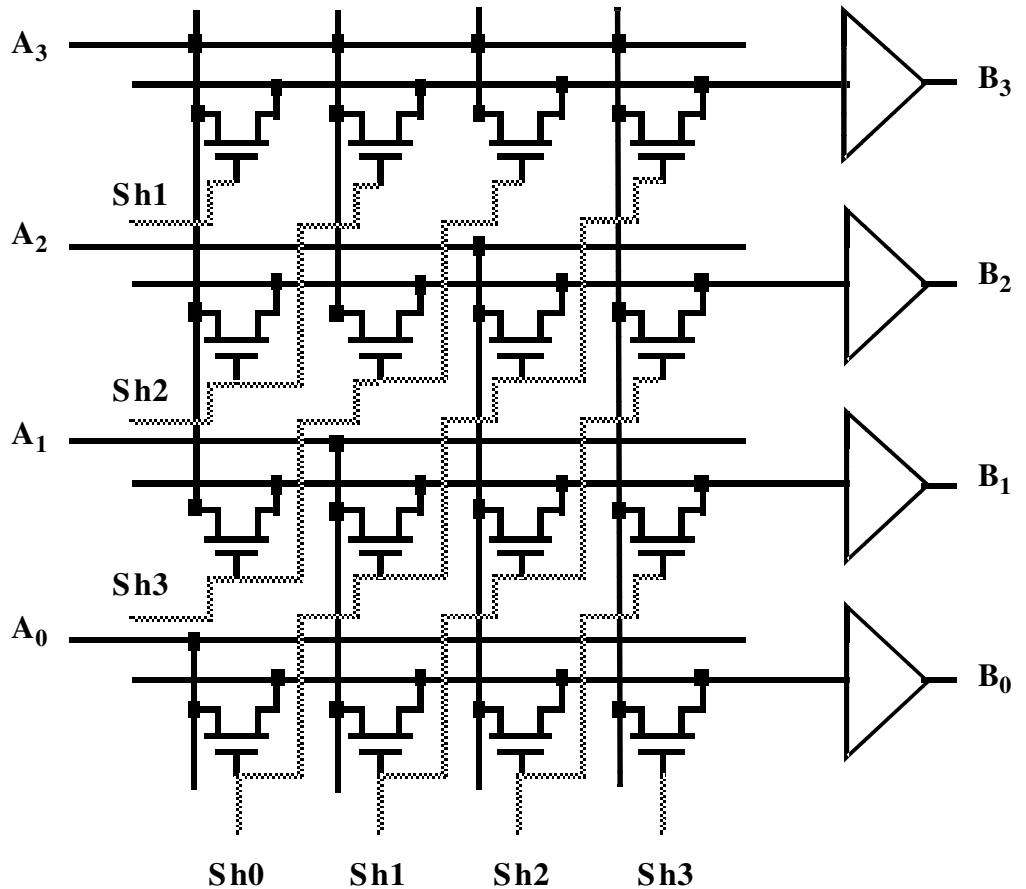
The Binary Shifter



Multi-bit Shifters

- Cascade one-bit shifters?
- Complex, unwieldy, slow for larger number of shifts
- Two other types of shifters
  - Barrel shifter
  - Logarithmic shifter

The Barrel Shifter



Shift by 0 to 3 bits

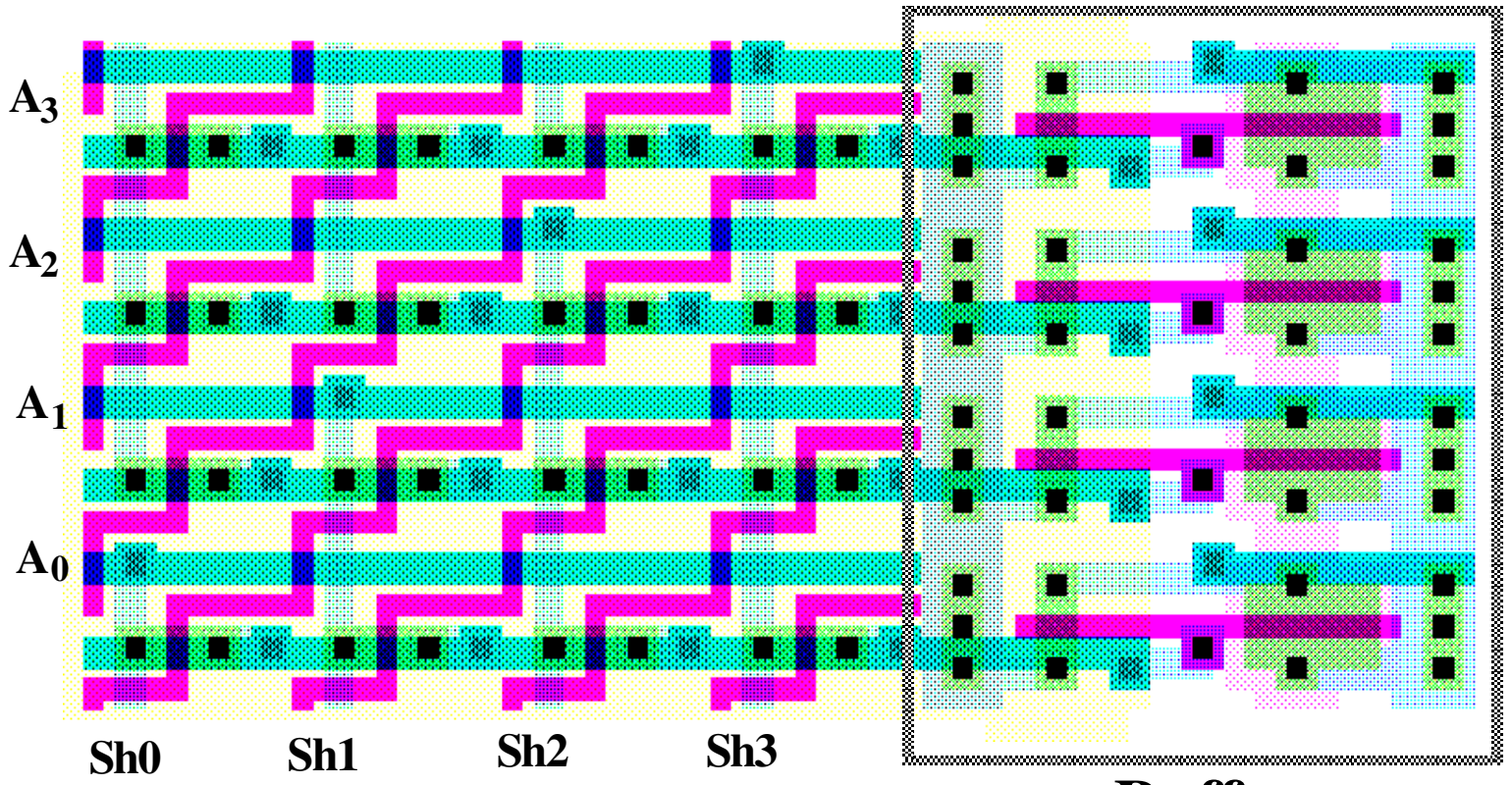
: Data Wire

: Control Wire

The Barrel Shifter

- Area dominated by wiring
- Propagation delay is theoretically constant (at most one transmission gate), independent of shifter size, no. of shifts
- Reality: Capacitance on buffer input α maximum shift width

4x4 barrel shifter

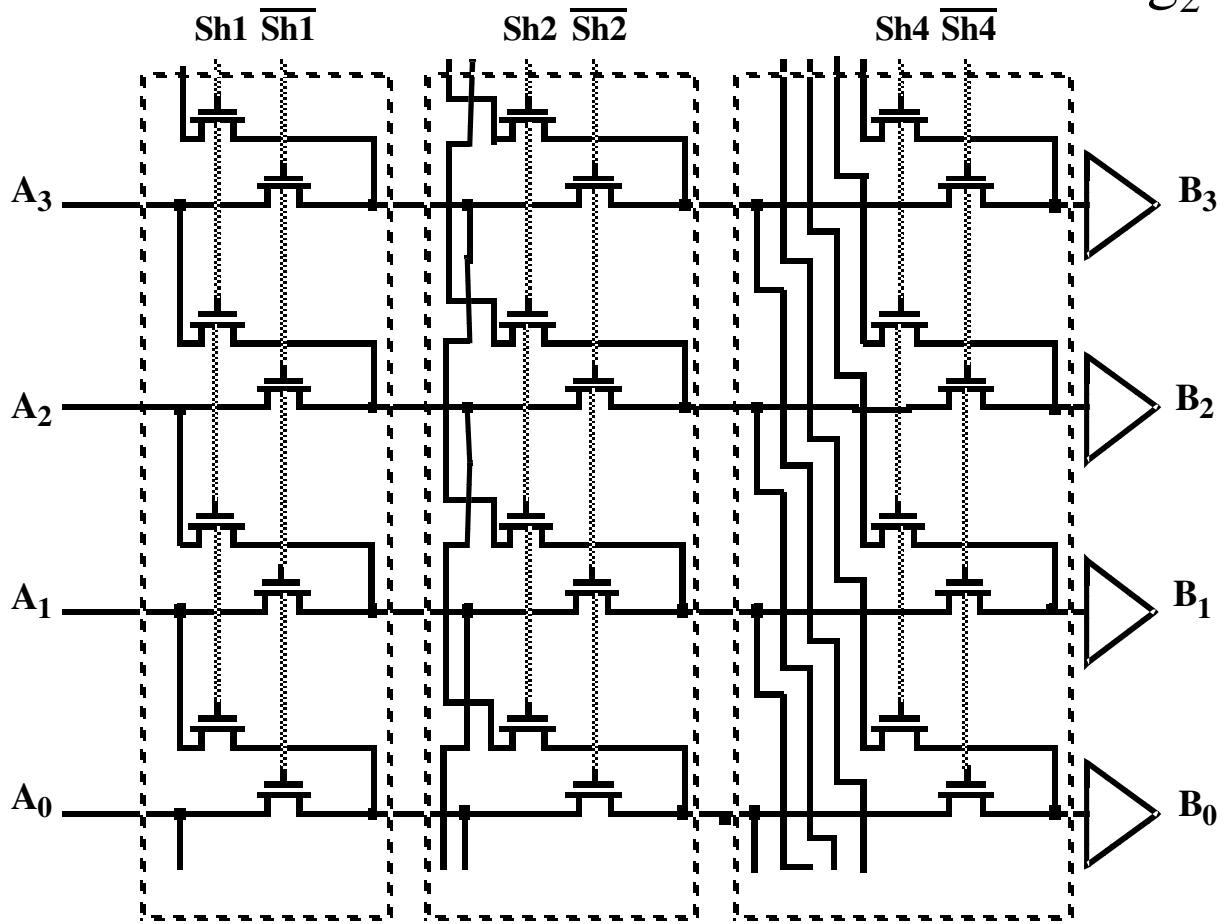


$\mathbf{Width_{barrel} \sim 2\ p_m\ M}$

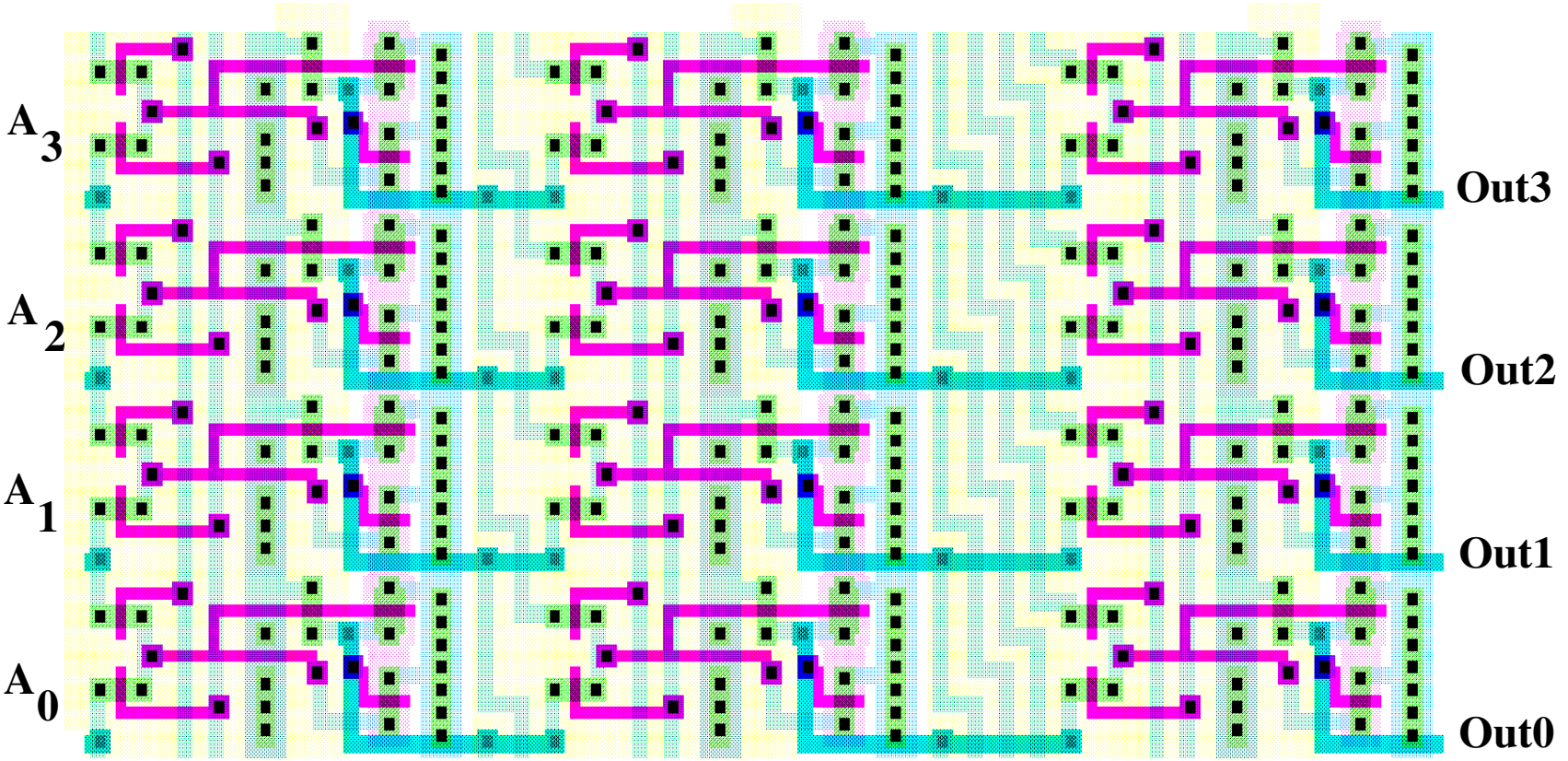
p_m = metal/poly pitch
M = no. of shifts

Logarithmic Shifter

- Staged approach, e.g. 7 = 1 + 2 + 4, 5 = 1 + 0 + 4
- Shifter with maximum shift width M consists of $\log_2 M$ stages



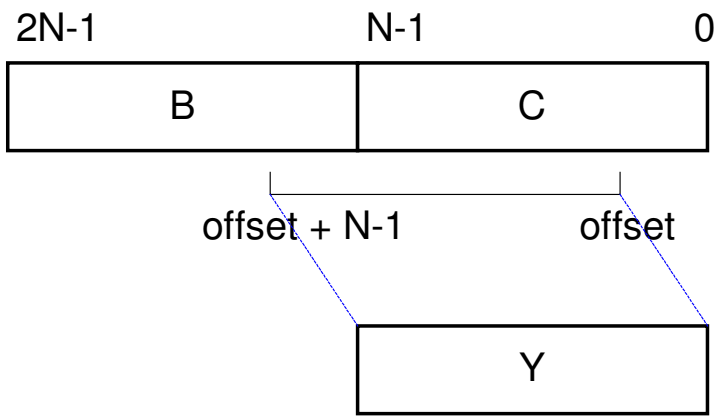
0-7 bit Logarithmic Shifter



$$width_{\log} \approx p_m\left(2K + \left(1 + 2 + \ldots + 2^{K-1}\right)\right) = p_m\left(2^K + 2K - 1\right)$$

Funnel Shifter

- A funnel shifter can do all six types of shifts
- Selects N-bit field Y from 2N-bit input
  - Shift by k bits ($0 \leq k < N$)



Funnel Shifter Operation

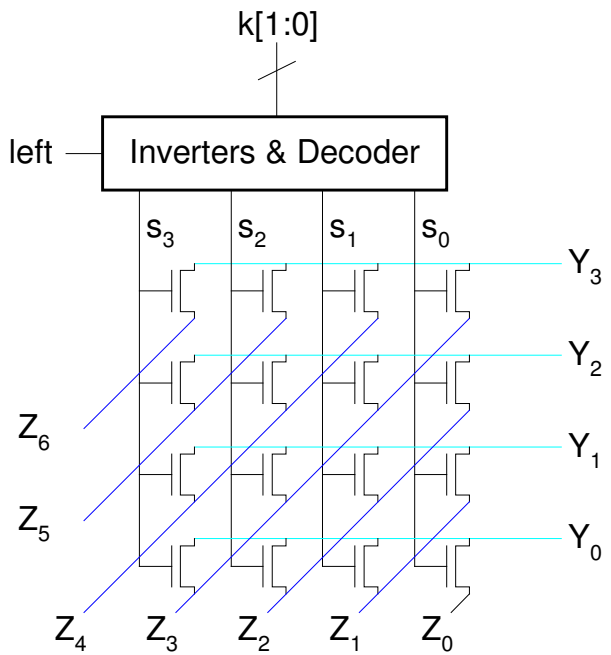
Table 10.10 Funnel shifter operation

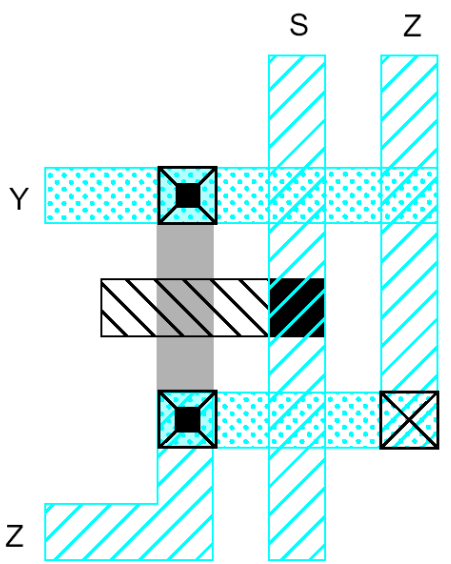
Shift Type	*B*	*C*	Offset
Logical Right	0…0	$A_{N-1}…A_0$	k
Logical Left	$A_{N-1}…A_0$	0…0	$N-k$
Arithmetic Right	$A_{N-1}…A_{N-1}$ (sign extension)	$A_{N-1}…A_0$	k
Arithmetic Left	$A_{N-1}…A_0$	0	$N-k$
Rotate Right	$A_{N-1}…A_0$	$A_{N-1}…A_0$	k
Rotate Left	$A_{N-1}…A_0$	$A_{N-1}…A_0$	$N-k$

- Computing N-k requires an adder

Funnel Shifter Design 1

- N N-input multiplexers
  - Use 1-of-N hot select signals for shift amount
  - nMOS pass transistor design (V_t drops!)





Funnel Shifter Design 2

- Log N stages of 2-input muxes
  - No select decoding needed