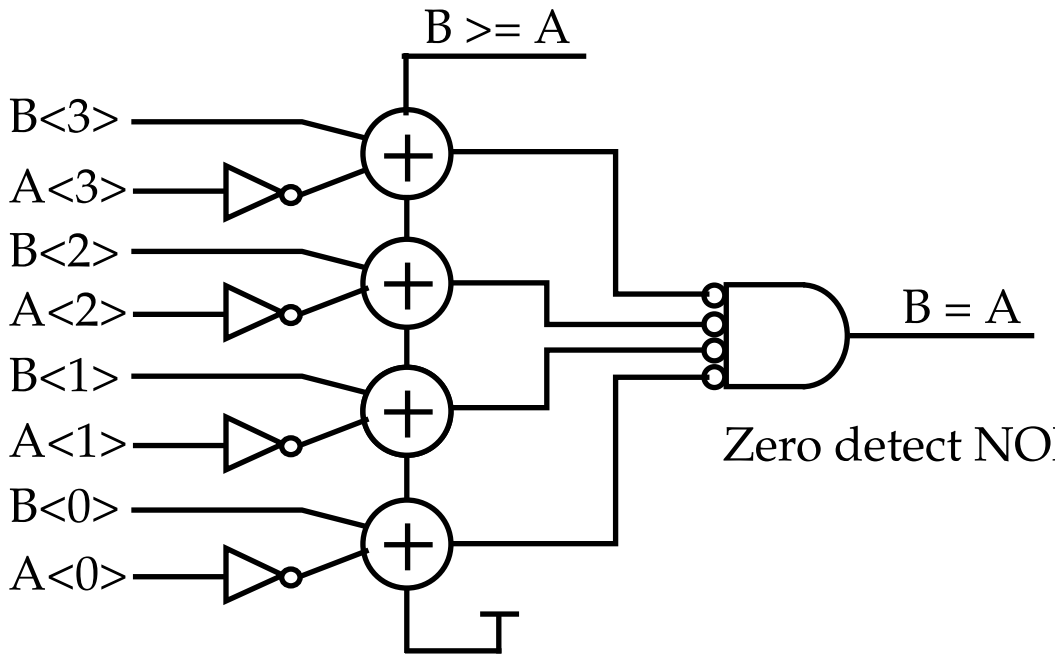
Datapath Operators: Comparison

Magnitude Comparators:

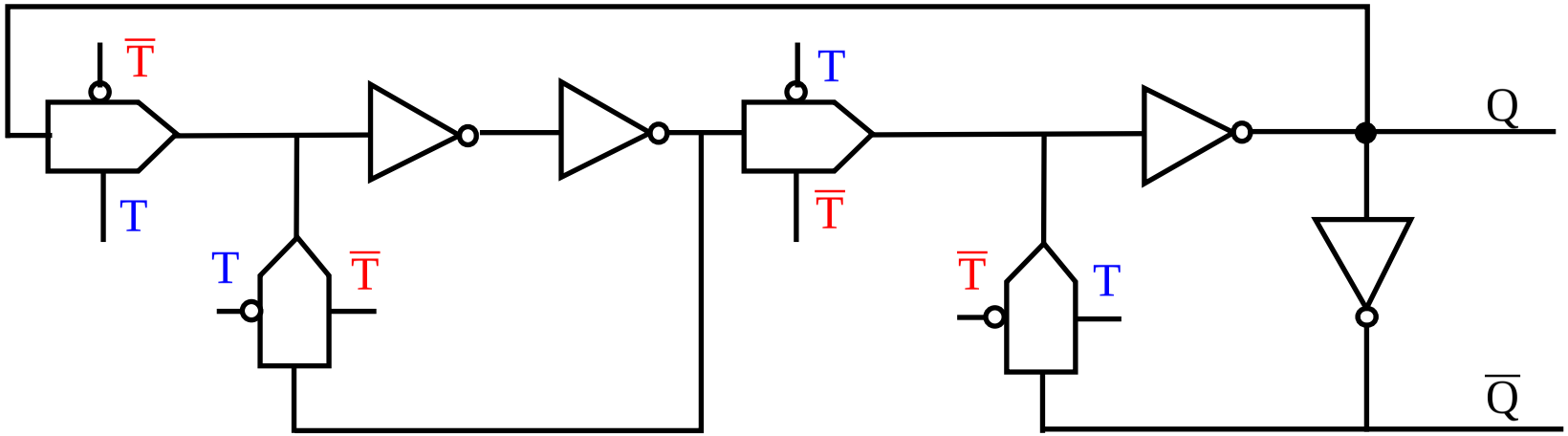
May be built from an adder, complementer (XOR gates) and a zero detect unit.

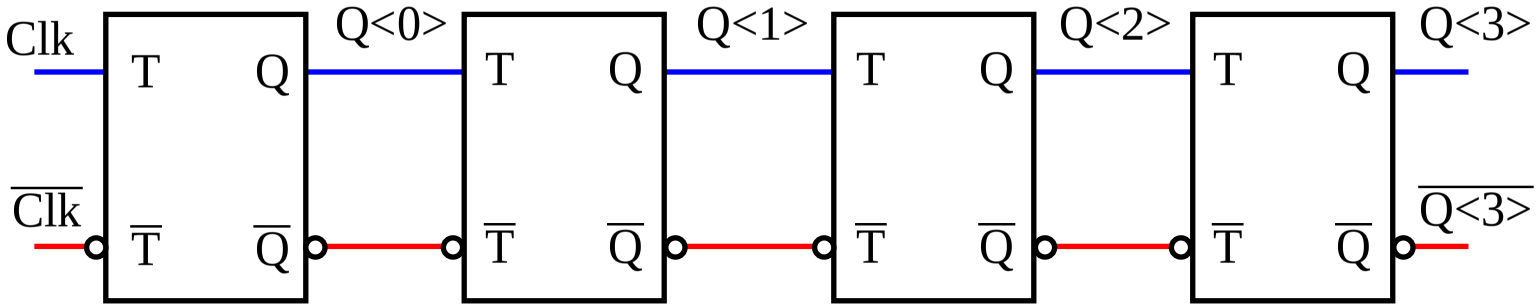


Think about the modifications necessary to make it a signed comparator (Hint: A couple of XOR gates).

Datapath Operators: Binary Counters

Asynchronous: Based on the Toggle register.



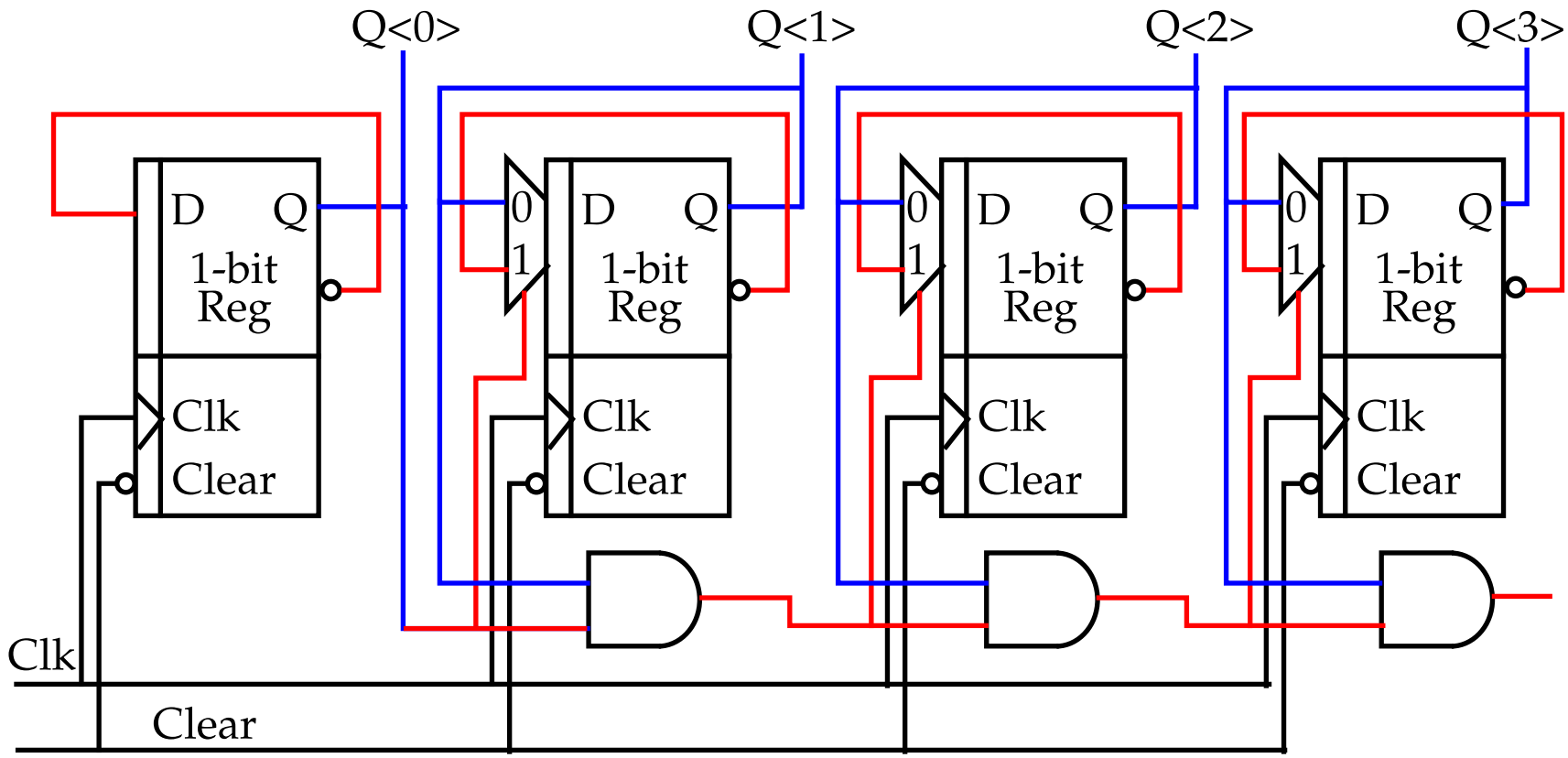


"Ripple Carry" Binary counter →

Not a good choice for performance and testability (with no reset).

Datapath Operators: Binary Counters

Synchronous counter.



Replace AND gate with an adder for up/down counting capability.

Weste and Eshraghian also show a version that can be initialized.

Datapath Operators: Multiplication

Multiplication can be broken down into two steps:

- Computation of partial products.
- Accumulation of the shifted partial products.

```
    1100
X   0101
------
    1100   <-- Binary multiplication equivalent to
   0000        AND operation
  1100
0000
-------
0111100
```

Multipliers may be classified by the format in which data words are accessed:

- Serial
- Serial/parallel
- Parallel

The parallel form computes the partial products in parallel.

Datapath Operators: Multiplication

Parallel Unsigned Multiplication:

$$X = \sum_{i=0}^{m-1} X_i 2^i$$

$$Y = \sum_{j=0}^{n-1} Y_j 2^j$$

Multiplying 2 unsigned binary integers results in:

$$P = X \times Y = \sum_{i=0}^{m-1} X_i 2^i \sum_{j=0}^{n-1} Y_j 2^j = \sum_{k=0}^{m+n-1} P_k 2^k$$

			X_3	X_2	X_1	X_0		Multiplicand
			Y_3	Y_2	Y_1	Y_0		Multiplier
			X_3Y_0	X_2Y_0	X_1Y_0	X_0Y_0		
		X_3Y_1	X_2Y_1	X_1Y_1	X_0Y_1			
	X_3Y_2	X_2Y_2	X_1Y_2	X_0Y_2				
X_3Y_3	X_2Y_3	X_1Y_3	X_0Y_3					
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	

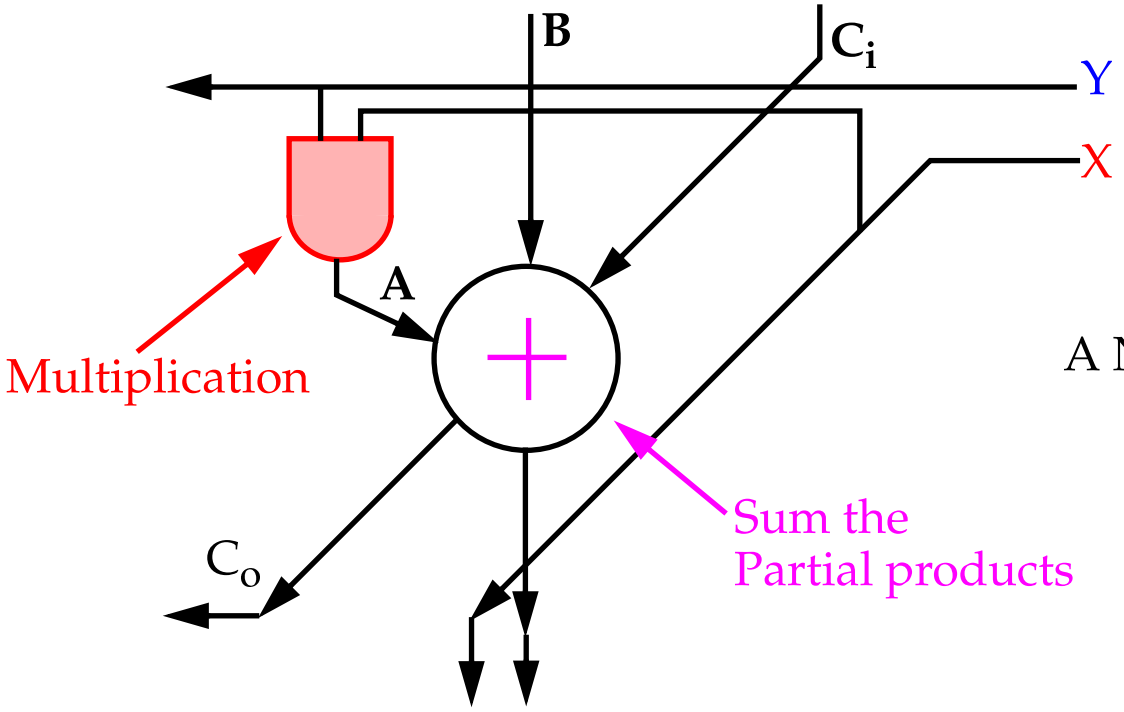
There are m*n summands produced by a set of m*n AND gates in parallel.

Datapath Operators: Multiplication

Parallel Multiplication:

Multiplication is carried out using a bitwise AND of the operands, X_i and Y_i.

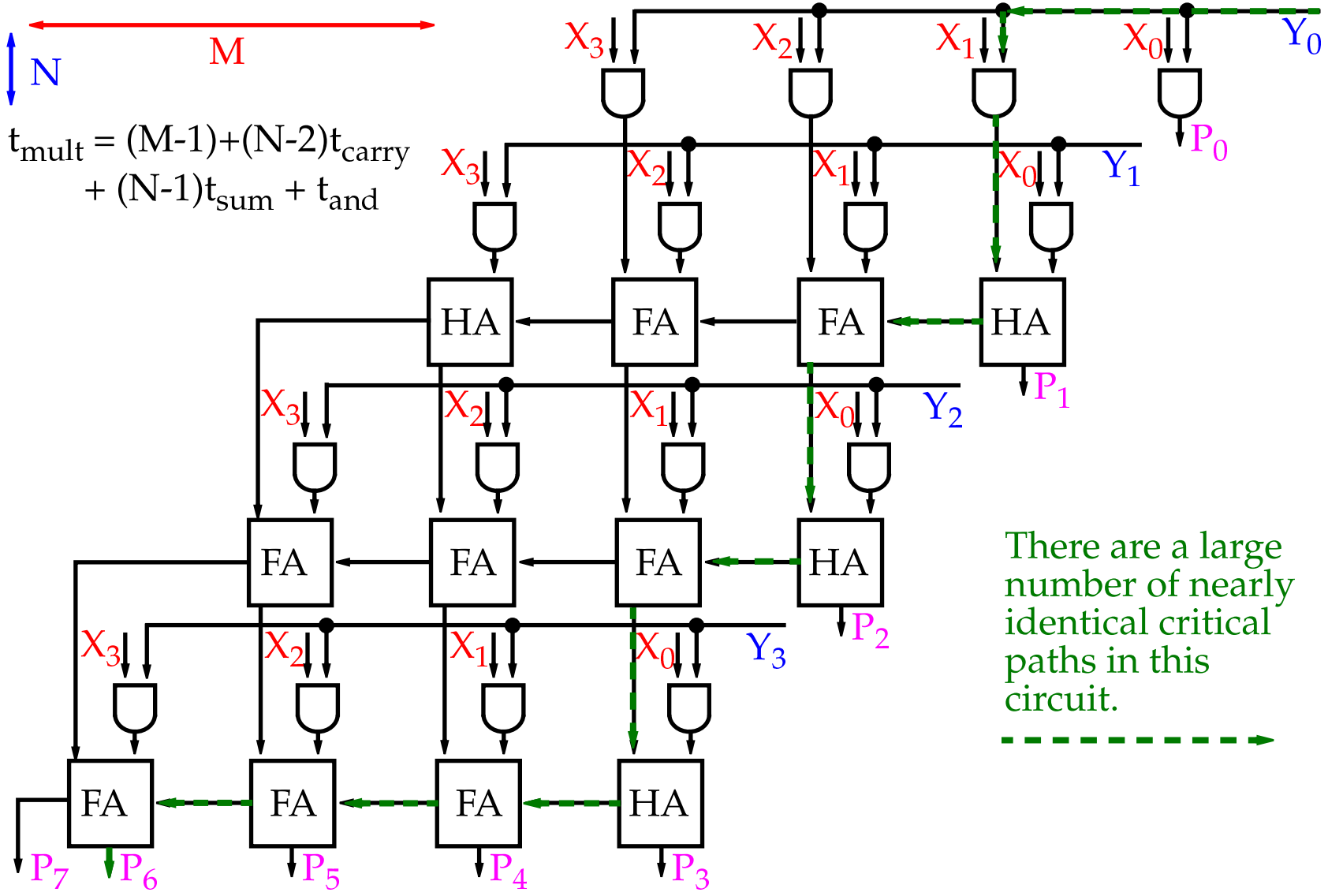
Most of the work (and delay) is in summing the partial products.



A NxN multiplier requires:
- N(N-2) full adders
- N half adders
- N^2 AND gates

Datapath Operators: Multiplication

Array multiplier:



Datapath Operators: Multiplication

From the delay expression and the fact that all critical paths have the same length, minimizing t_{mult} requires minimizing both t_{carry} and t_{sum}.

This is in contrast with the adder where minimizing t_{carry} was key.

The transmission gate adder is a good choice here.

Parallel Signed Multiplication:

Baugh-Wooley algorithm: Only 3 additional adders required over the unsigned version.

$$A = -a_{m-1}2^{m-1} + \sum_{i=0}^{m-2} a_i 2^i$$

$$B = -b_{m-1}2^{m-1} + \sum_{i=0}^{m-2} b_i 2^i$$

Let A and B represent signed integers.

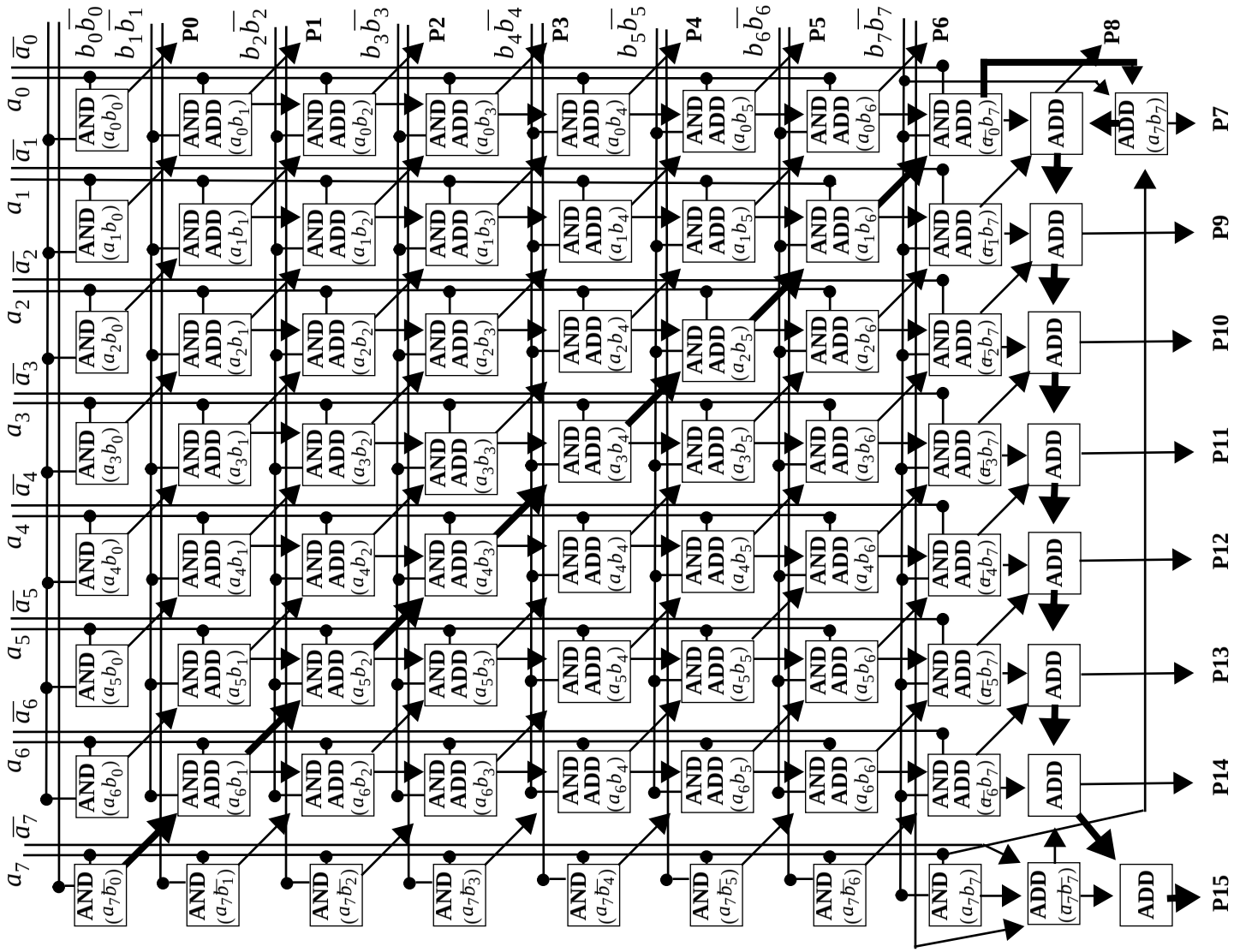
Expanding shows that the last two rows of summands are all negative so the algorithm simply adds in their negations.

$$P = \left(-a_{m-1}2^{m-1} + \sum_{i=0}^{m-2} a_i 2^i\right)\left(-b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i\right)$$

$$= a_{m-1}b_{n-1}2^{m+n-2} + \sum_{i=0}^{m-2}\sum_{j=0}^{n-2} a_i b_j 2^{i+j} - \sum_{i=0}^{m-2} a_i b_{n-1} 2^{n-1+i} - \sum_{i=0}^{n-2} a_{m-1} b_i 2^{m-1+i}$$

Datapath Operators: Multiplication

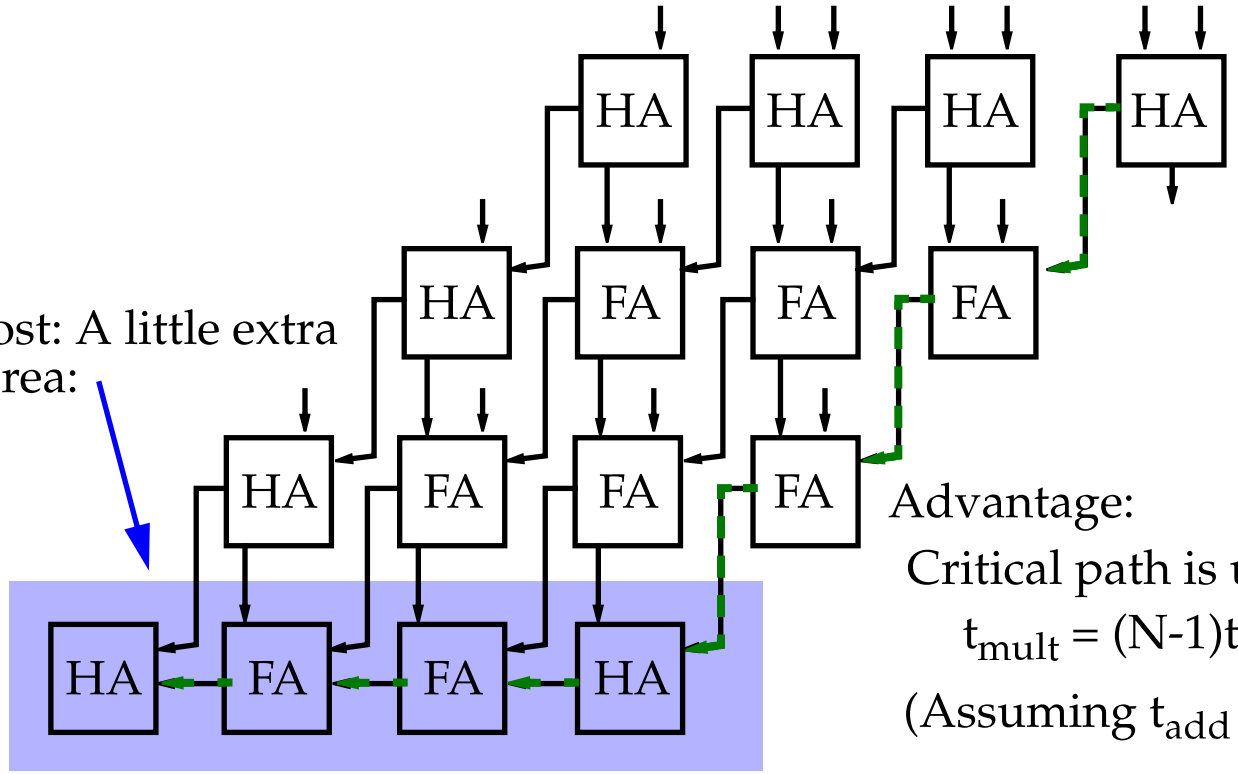
Parallel Signed Multiplication:



Datapath Operators: Multiplication

Carry-Save Multiplier:

Carry bits can be passed diagonally downwards instead of to the left.



4x4 version

Advantage:

Critical path is uniquely defined:

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

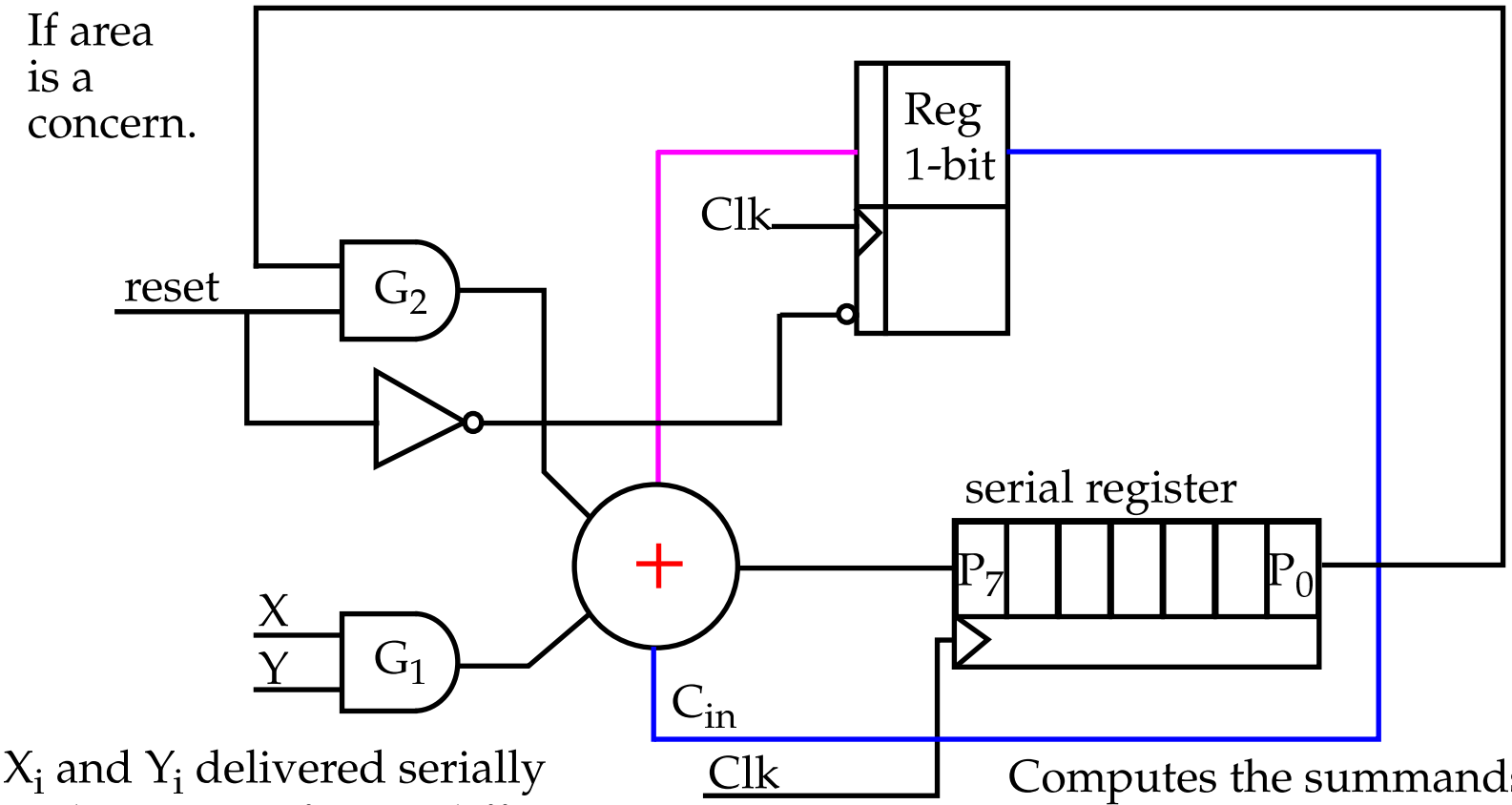
(Assuming $t_{add} = t_{carry}$).

Minimizing t_{merge} is useful, e.g. use carry-select or lookahead.

Here the carry bits are not immediately added but rather "saved" for the next adder stage.

Datapath Operators: Multiplication

Serial Unsigned Multiplication:



X_i and Y_i delivered serially to the inputs of G_1 at different rates.

Computes the summands row-wise from right to left.

Disadv: Quadratic delay: $t_{mult} = M \times N \times t_{carry}$

Serial/Parallel Unsigned Multiplier shown in Weste and Eshraghian.

Datapath Operators: Multiplication

Booth Encoding:

A special encoding of the multiplier word reduces the number of required addition stages and speeds up multiplication substantially.

Radix-4 scheme:

$$Y = \sum_{j=0}^{(N-1)/2} Y_j 4^j \text{ with } (Y_j \in \{-2, -1, 0, 1, 2\})$$

The number of partial products (and additions) is halved, resulting in area and speed advantage.

The disadvantage is a somewhat more involved multiplier cell.

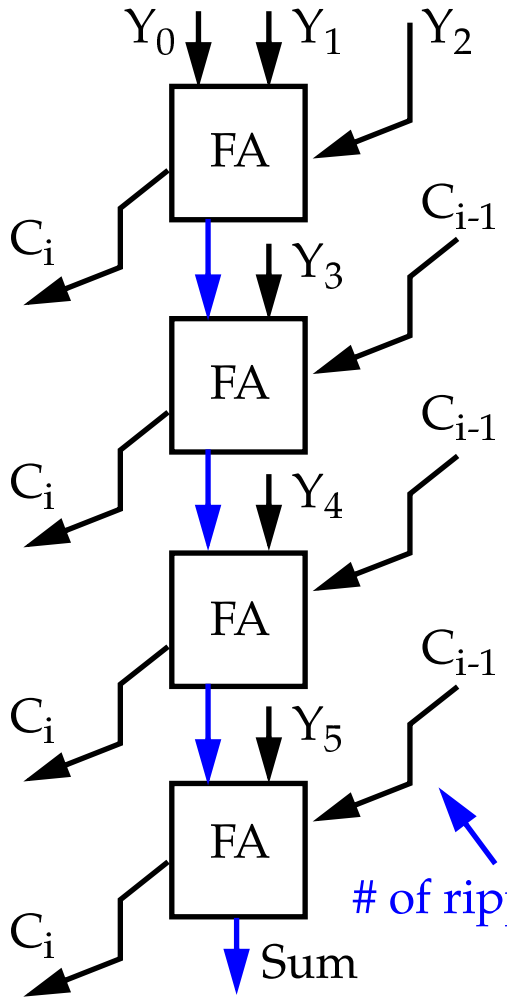
AND operation replaced with inversion and shift logic.

Virtually every multiplier in use employs the Booth scheme.

Datapath Operators: Multiplication

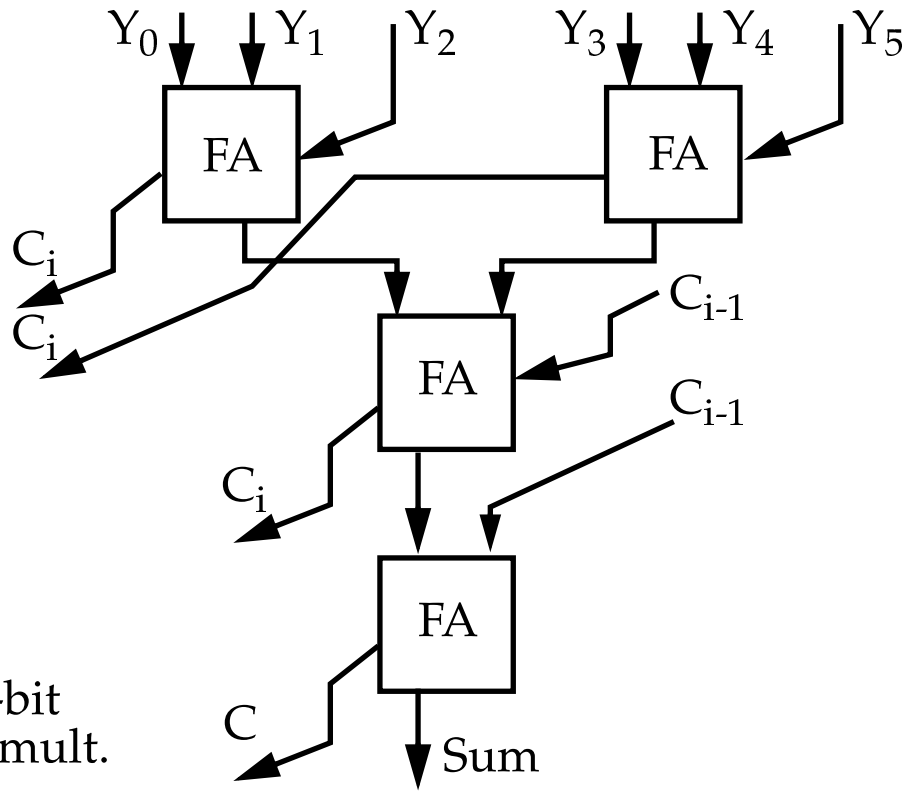
Wallace Multiplier:

Trees can be used to replace the linear partial-sum adders:



Slice of a 6-bit carry-save mult.

# of ripple stages is N-2

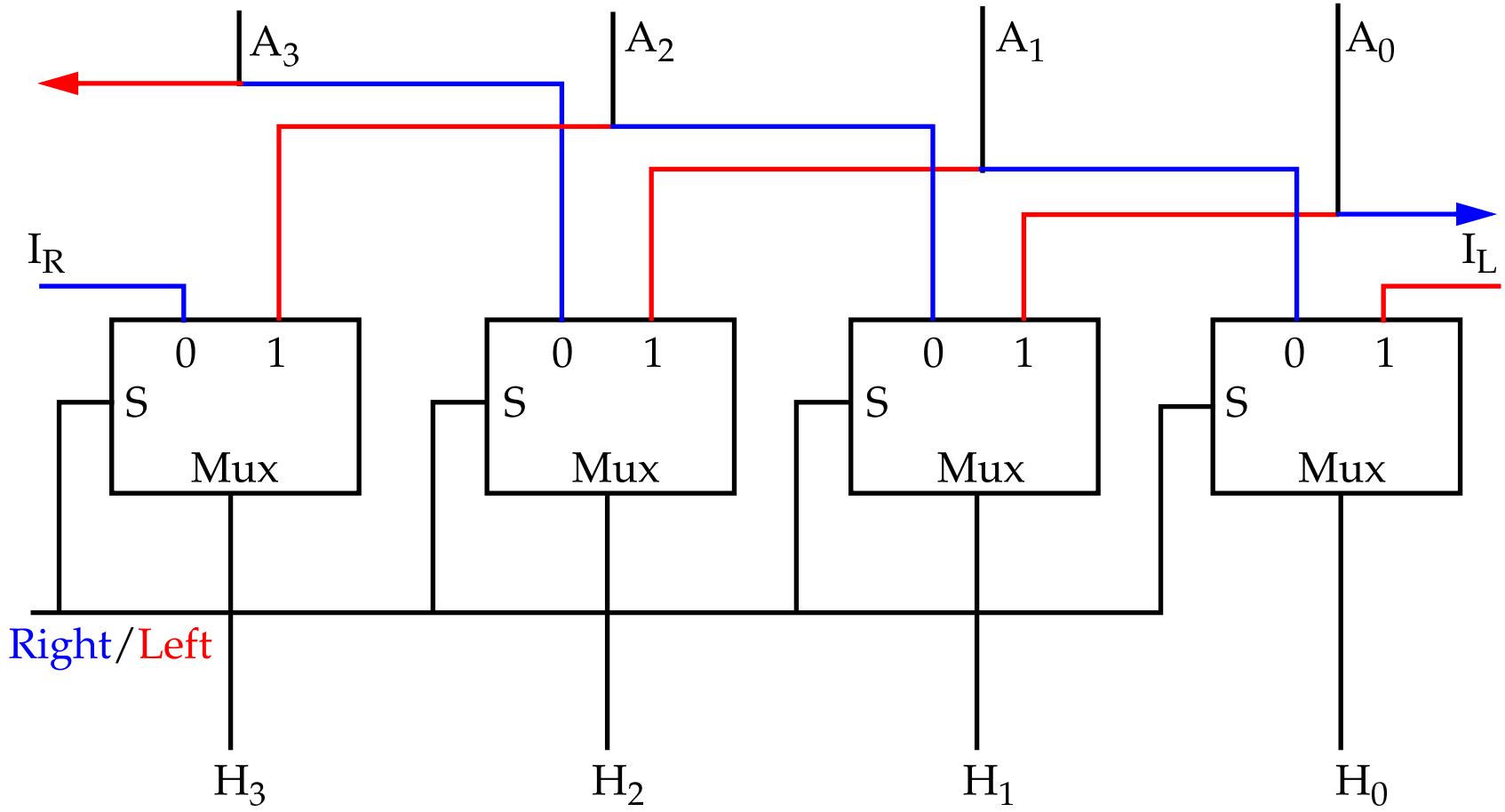


Adv: $O(\log_2 N)$ mult time.

Disadv: Very irregular -- difficult to layout.

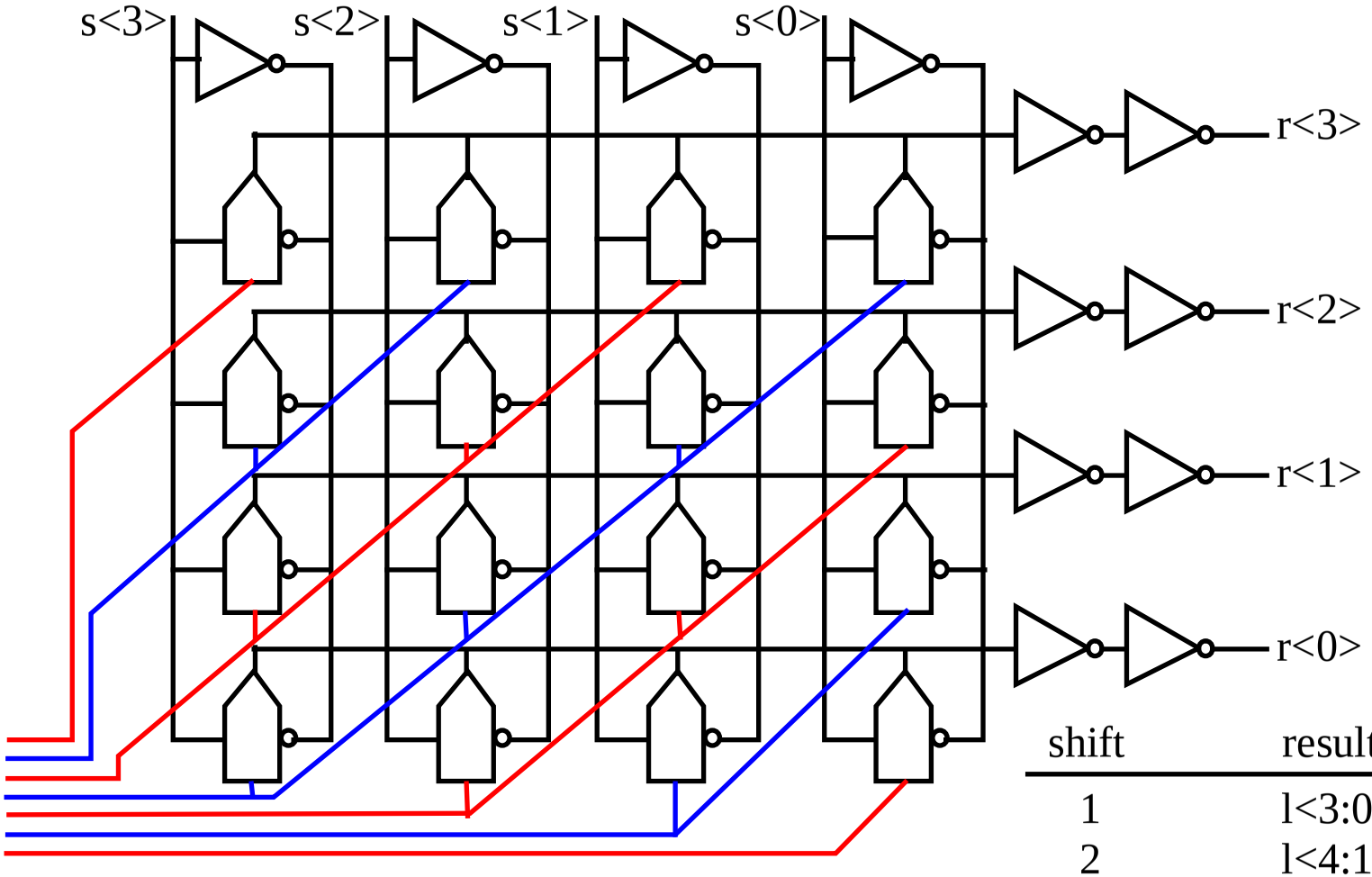
Datapath Operators: Shifters

Right/Left 1-bit shifter:



Datapath Operators: Shifters

Barrel shifter:



shift	result
1	l<3:0>
2	l<4:1>
4	l<5:2>
8	l<6:3>

Arithmetic and logical shifts and rotates possible by muxing l<6:0> to the appropriate values.