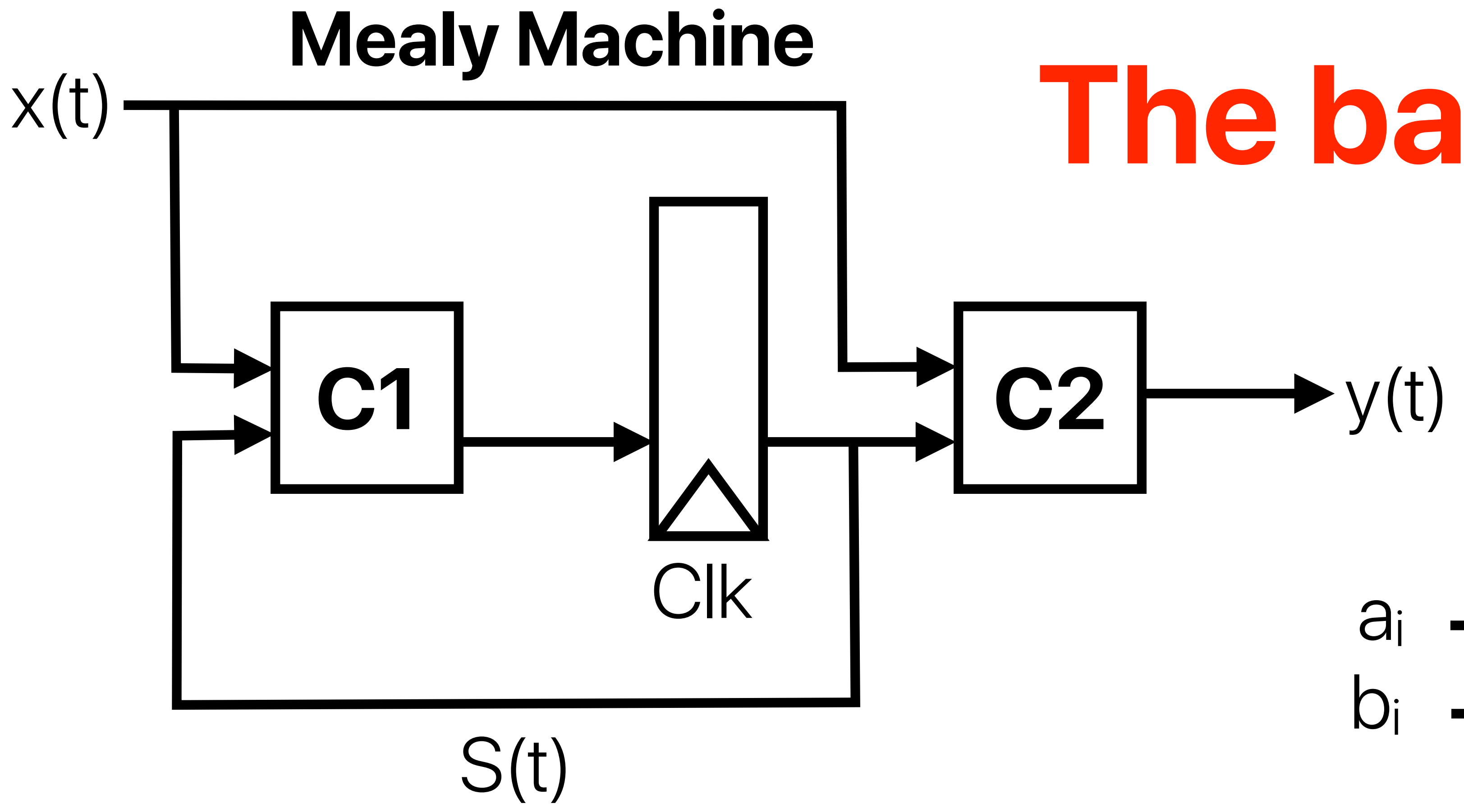
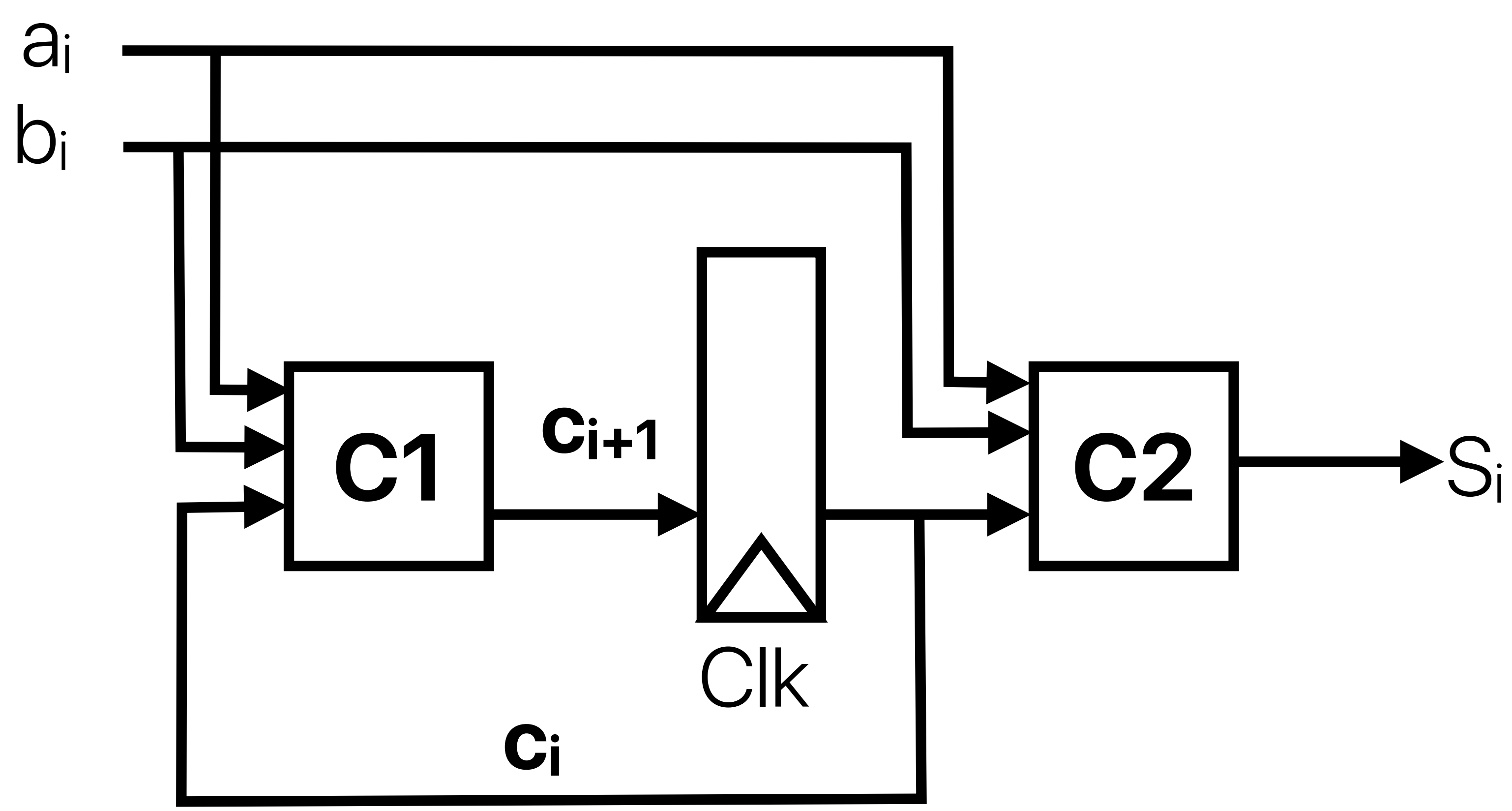
When sequential circuits meet datapath components (3)

Prof. Usagi

Serial Adder

The basic idea





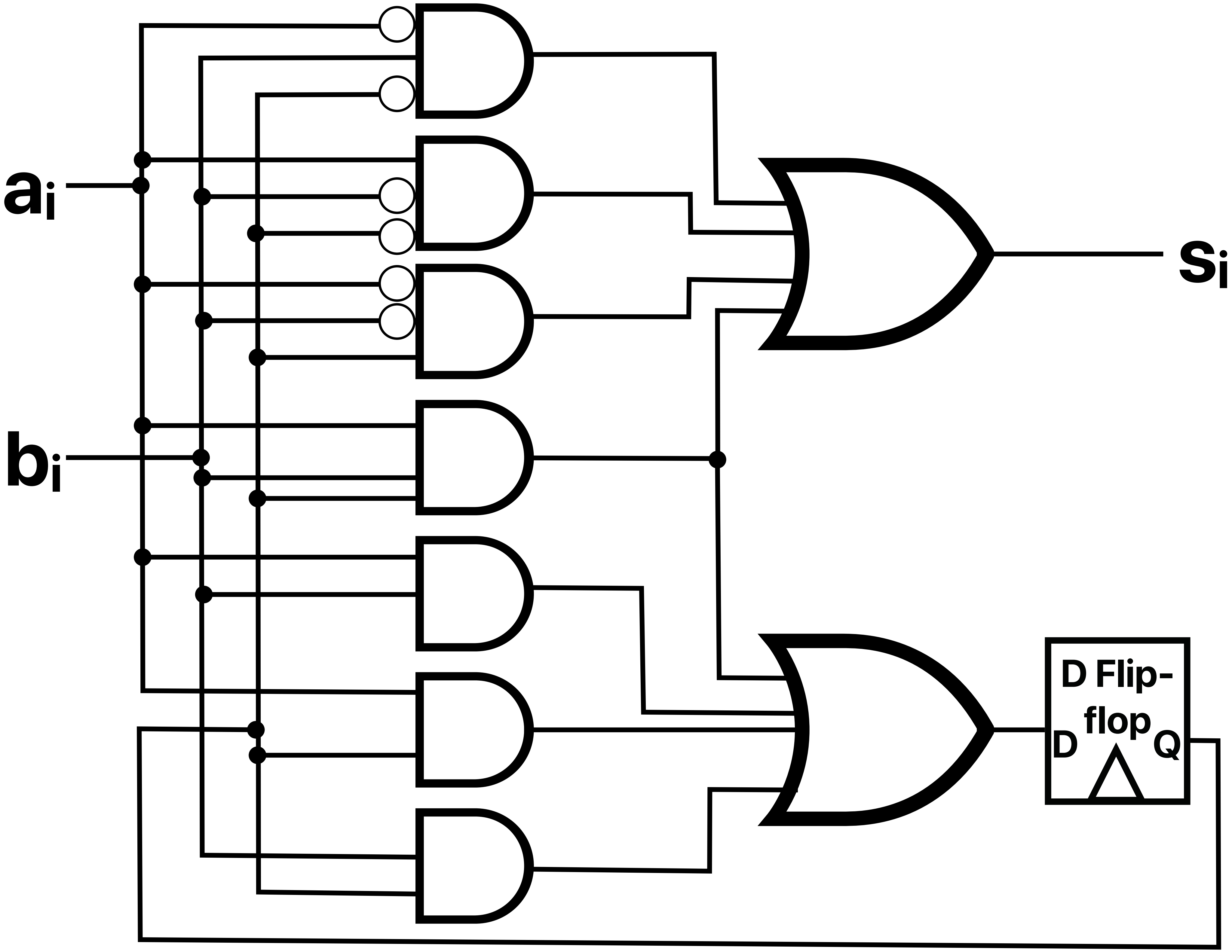
Feed a_i and b_i and generate s_i at time i. Where is c_i and c_{i+1}?

Excitation Table of Serial Adder

a_i	b_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Excitation Table of Serial Adder

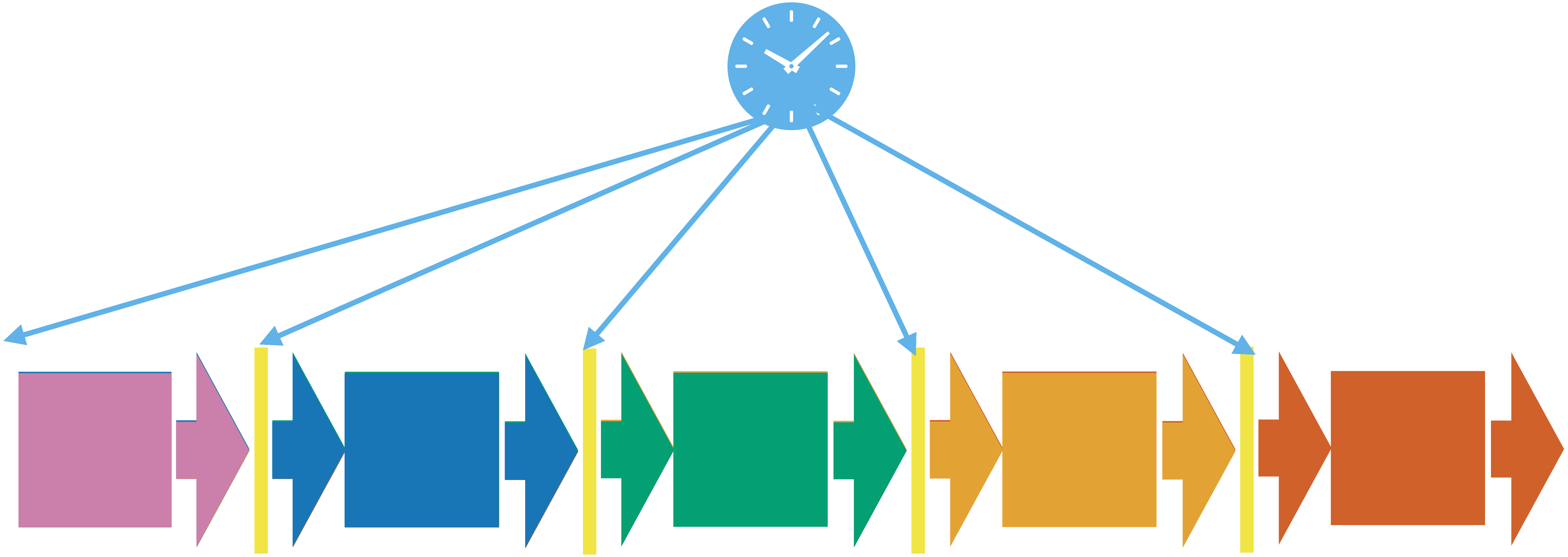
a_i	b_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



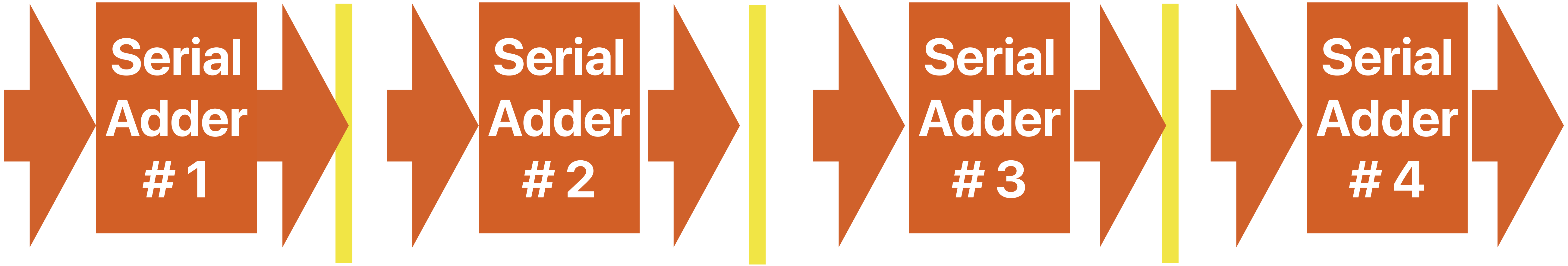
Pipelining

- Different parts of the hardware works on different requests/commands simultaneously
- A clock signal controls and synchronize the beginning and the end of each part/**stage** of the work
- A **pipeline register** between different parts of the hardware to keep intermediate results necessary for the upcoming work
  - Register is basically an array of flip-flops!

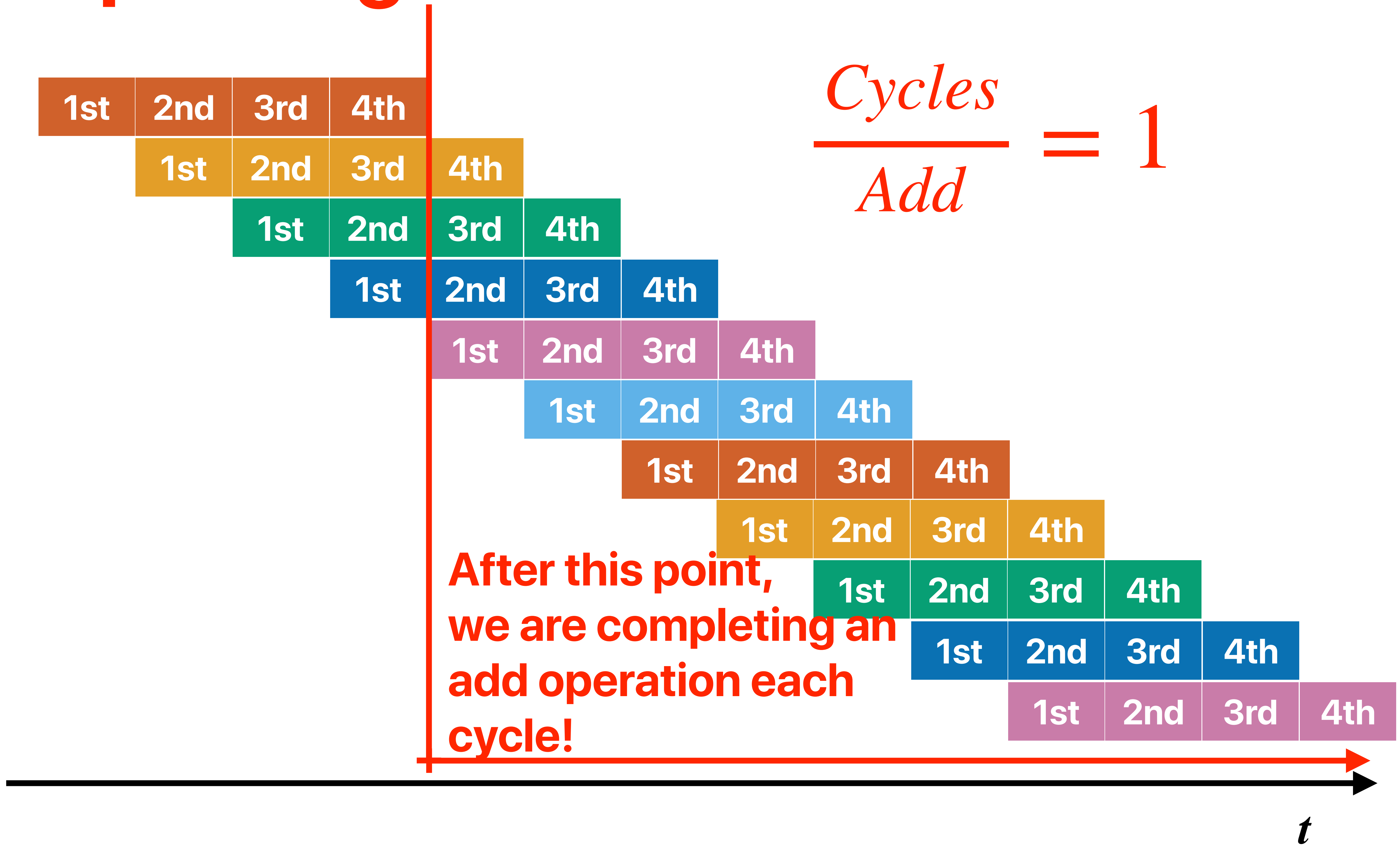
Pipelining



Pipelining a 4-bit serial adder



Pipelining a 4-bit serial adder



`add a, b`	1st	2nd	3rd	4th										
`add c, d`		1st	2nd	3rd	4th									
`add e, f`			1st	2nd	3rd	4th								
`add g, h`				1st	2nd	3rd	4th							
`add i, j`					1st	2nd	3rd	4th						
`add k, l`						1st	2nd	3rd	4th					
`add m, n`							1st	2nd	3rd	4th				
`add o, p`								1st	2nd	3rd	4th			
`add q, r`									1st	2nd	3rd	4th		
`add s, t`										1st	2nd	3rd	4th	
`add u, v`											1st	2nd	3rd	4th

$$\frac{Cycles}{Add} = 1$$

After this point, we are completing an add operation each cycle!

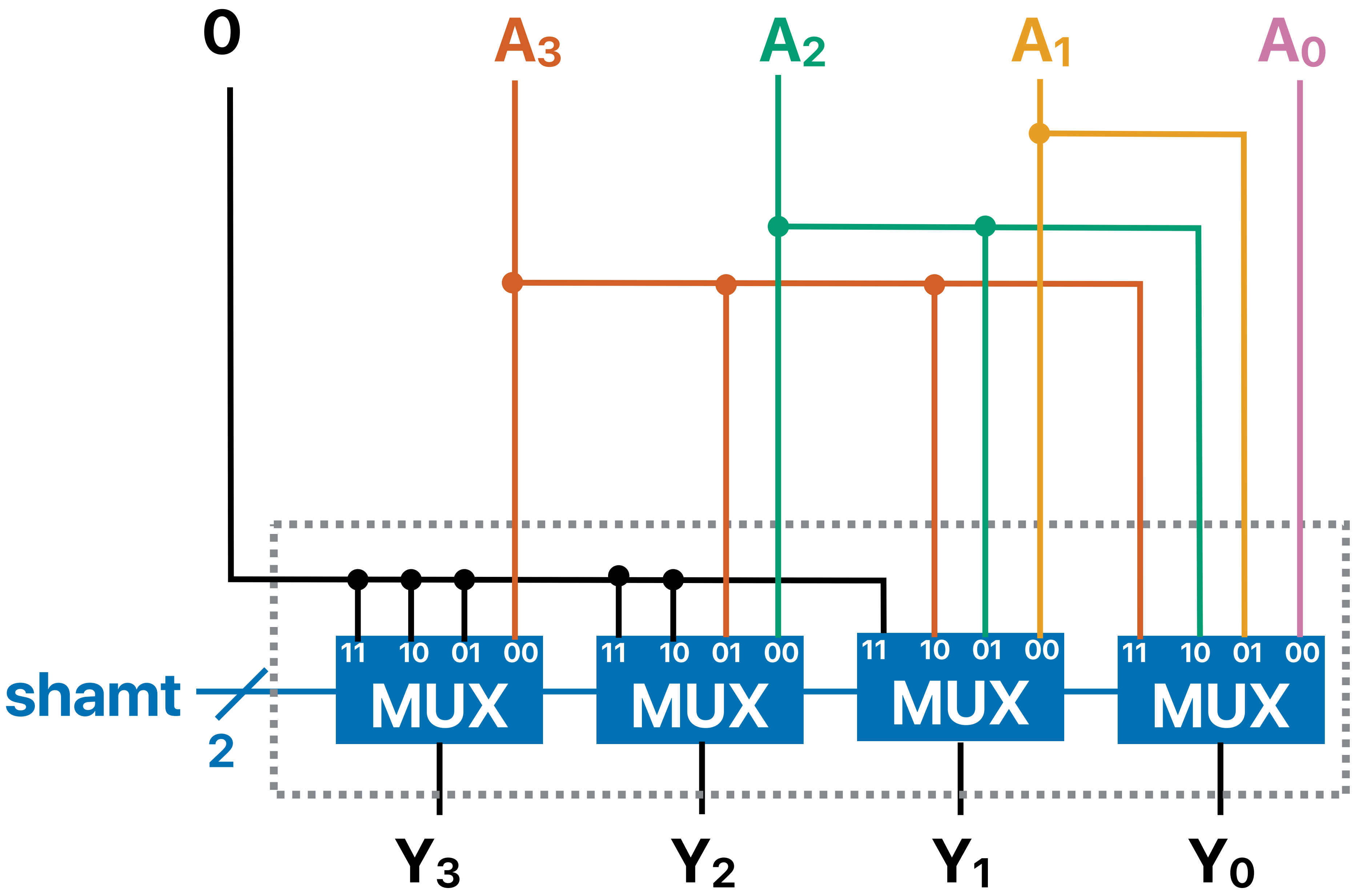
Latency/Delay v.s. Bandwidth/Throughput

- Latency — the amount of time to finish an operation
  - access time
  - response time
- Throughput — the amount of work can be done within a given period of time
  - bandwidth (MB/Sec, GB/Sec, Mbps, Gbps)
  - IOPs
  - MFLOPs

Latency/Delay v.s. Throughput

	Toyota Prius	100 Gb Network
	●100 miles (161 km) from UCSD ●75 MPH on highway! ●Max load: 374 kg = 2,770 hard drives (2TB per drive) 	●100 miles (161 km) from UCSD ●Lightspeed! — $3*10^8$m/sec ●Max load:4 lanes operating at 25GHz
bandwidth	290GB/sec	100 Gb/s or 12.5GB/sec
latency	3.5 hours	2 Peta-byte over 167772 seconds = 1.94 Days
response time	You see nothing in the first 3.5 hours	You can start watching the movie as soon as you get a frame!

Recap: Shift "Right"



Based on the value of the selection input (shamt = shift amount)

Example:
if S = 11
then
Y3 = 0
Y2 = 0
Y1 = 0
Y0 = A3

Example:
if S = 10
then
Y3 = 0
Y2 = 0
Y1 = A3
Y0 = A2

Example:
if S = 01
then
Y3 = 0
Y2 = A3
Y1 = A2
Y0 = A1

The "chain" of multiplexers determines how many bits to shift

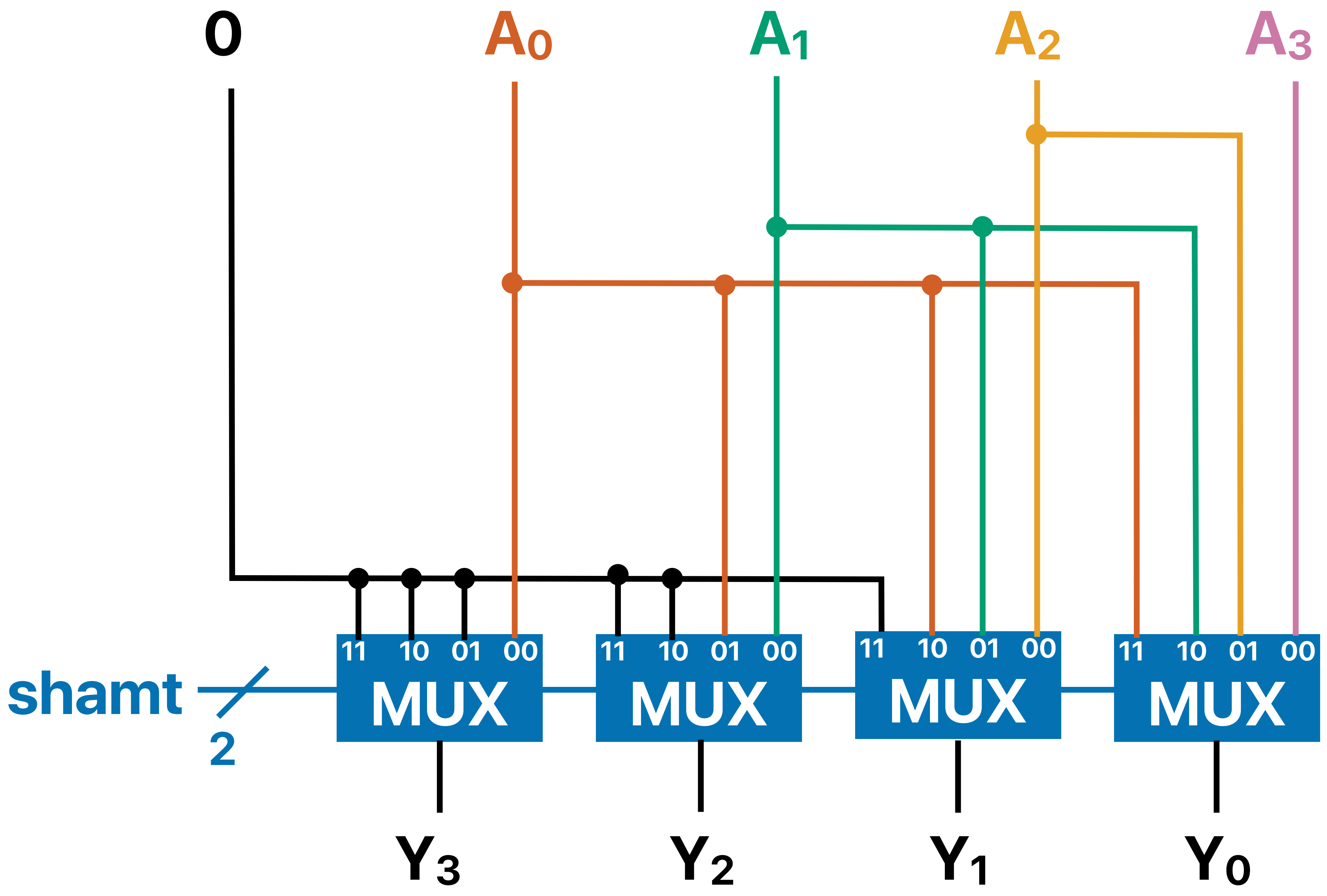
Shifters

Shift "Left"

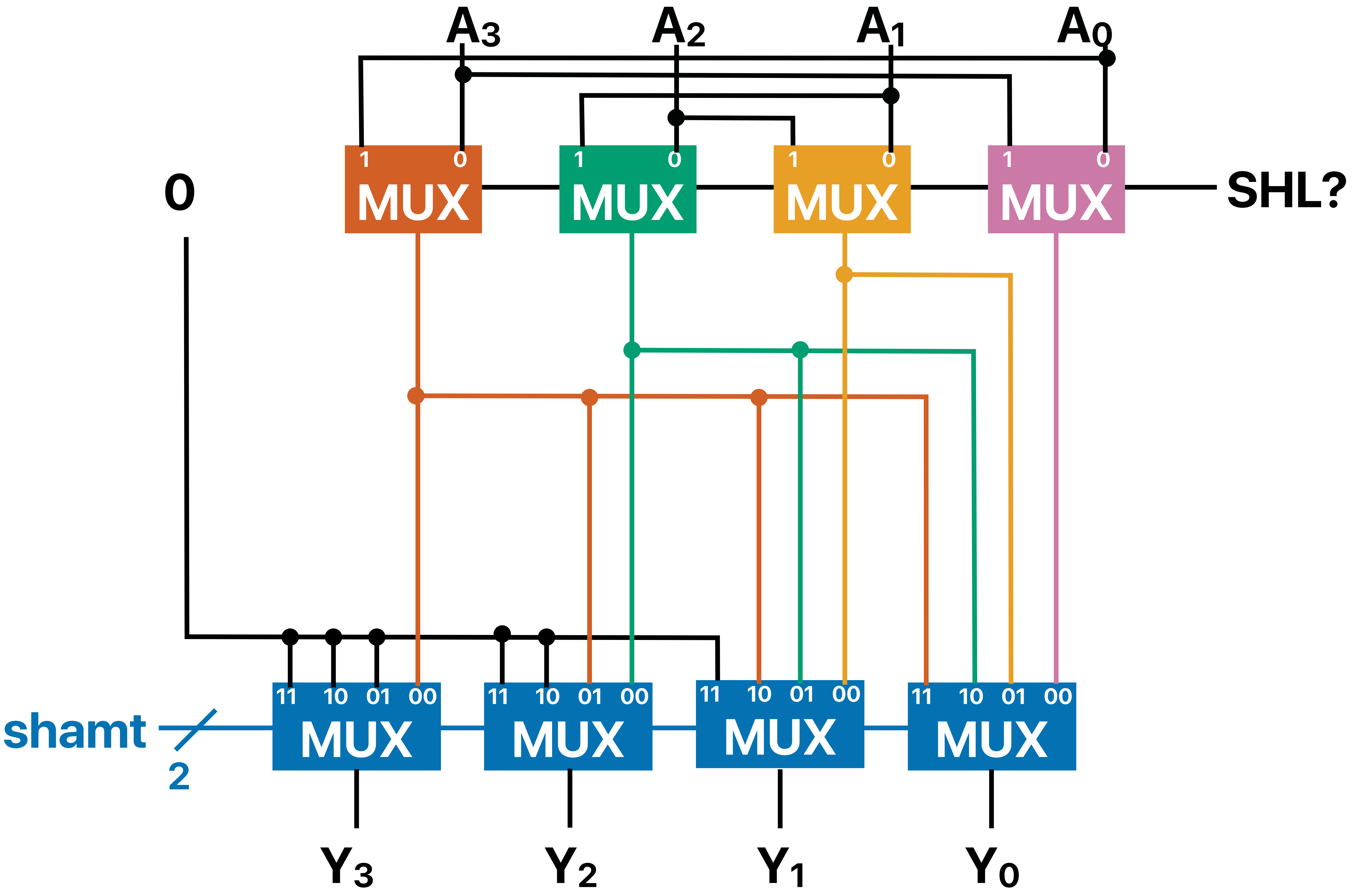
Example:
if S = 01
then
Y3 = A2
Y2 = A1
Y1 = A0
Y0 = 0

Example:
if S = 10
then
Y3 = A1
Y2 = A0
Y1 = 0
Y0 = 0

Example:
if S = 11
then
Y3 = A0
Y2 = 0
Y1 = 0
Y0 = 0



Generic Shifter



Multiplier

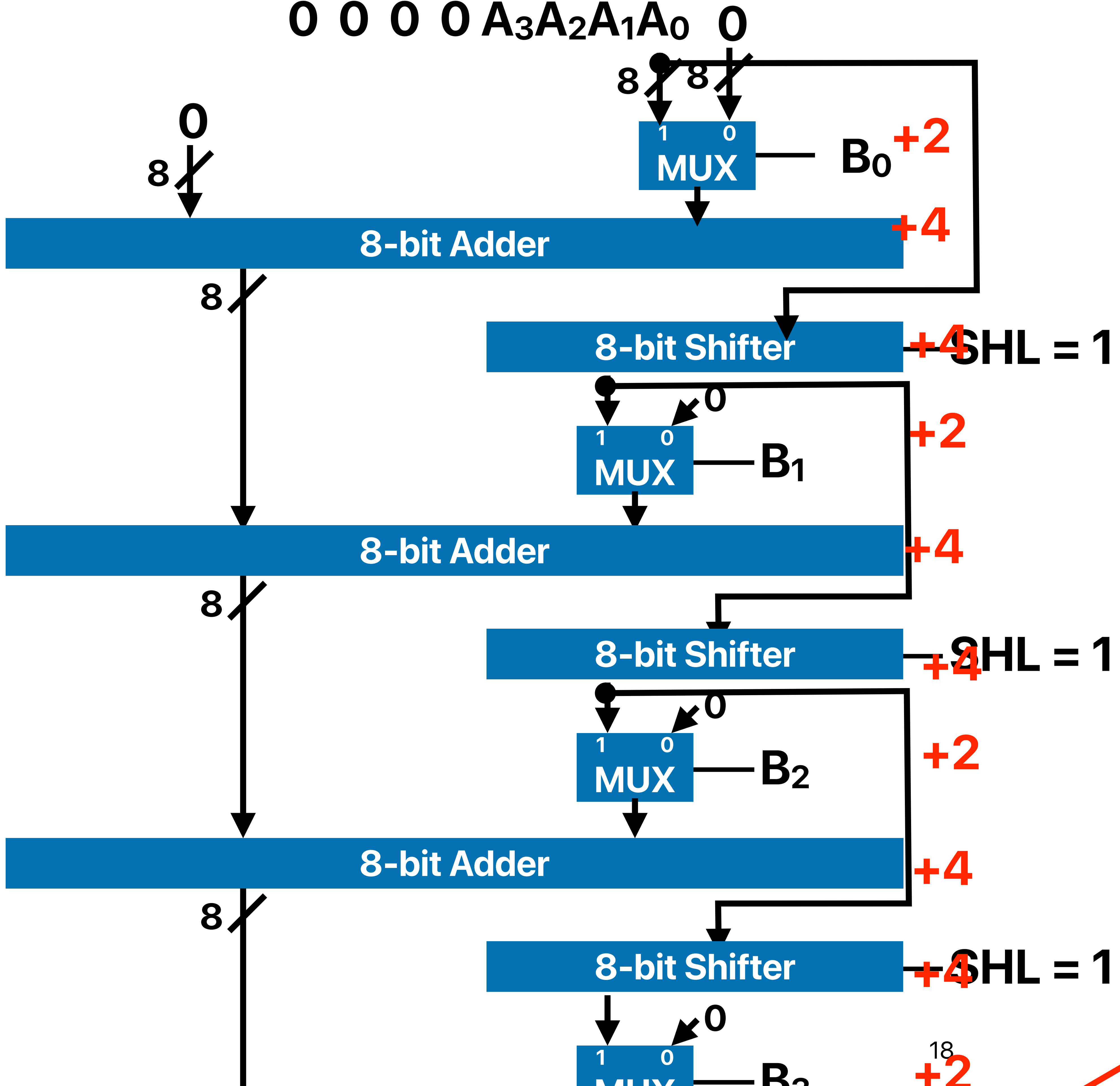
Binary multiplication

- Thinking about how you do this by hand in decimal!

$$\begin{array}{ccccccc}
 & & & 1 & 2 & 3 & 4 \\
 & & \times & 5 & 6 & 7 & 8 \\
\hline
 & & & 9 & 8 & 7 & 2 \\
 & & 8 & 6 & 3 & 8 & \\
 & 7 & 4 & 0 & 4 & & \\
6 & 1 & 7 & 0 & & & \\
\hline
7 & 0 & 0 & 6 & 6 & 5 & 2
\end{array}
\qquad
\begin{array}{ccccccc}
 & & & 0 & 1 & 1 & 1 \\
 & & \times & 1 & 1 & 0 & 0 \\
\hline
 & & & 0 & 0 & 0 & 0 \\
 & & 0 & 0 & 0 & 0 & \\
 & 0 & 1 & 1 & 1 & & \\
0 & 1 & 1 & 1 & & & \\
\hline
1 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}$$

$$\begin{array}{l|cccccccc}
 & & & & & a_3 & a_2 & a_1 & a_0 \\
 & & & & \times & b_3 & b_2 & b_1 & b_0 \\
\hline
\text{pp1} & & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
\text{pp2} & & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & 0 \\
\text{pp3} & & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & 0 & 0 \\
\text{pp4} & & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & 0 & 0 & 0 \\
\hline
 & p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}$$

Shift and add

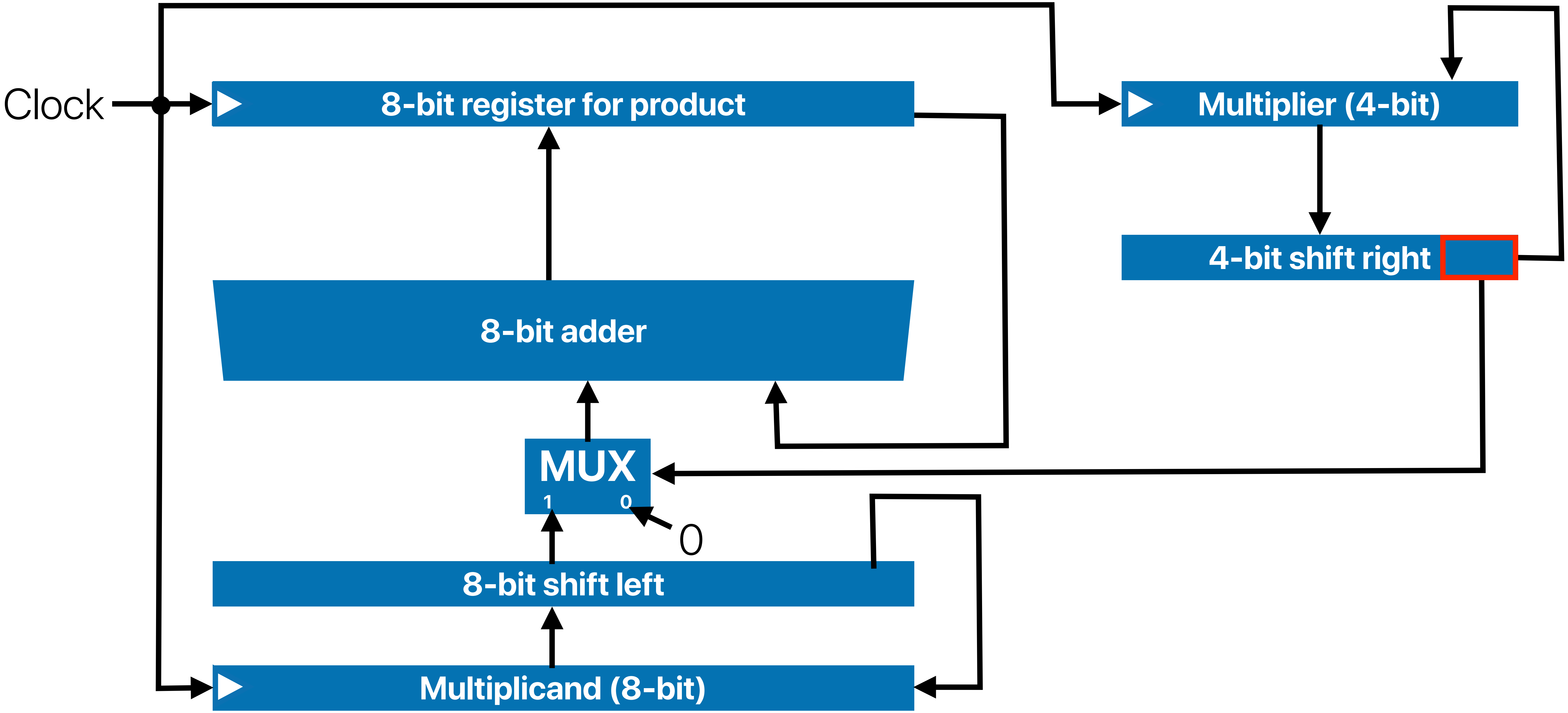


				a_3	a_2	a_1	a_0
				× b_3	b_2	b_1	b_0
				a_3b_0	a_2b_0	a_1b_0	a_0b_0
			a_3b_1	a_2b_1	a_1b_1	a_0b_1	0
		a_3b_2	a_2b_2	a_1b_2	a_0b_2	0	0
	a_3b_3	a_2b_3	a_1b_3	a_0b_3	0	0	0
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

— 36 gate delays

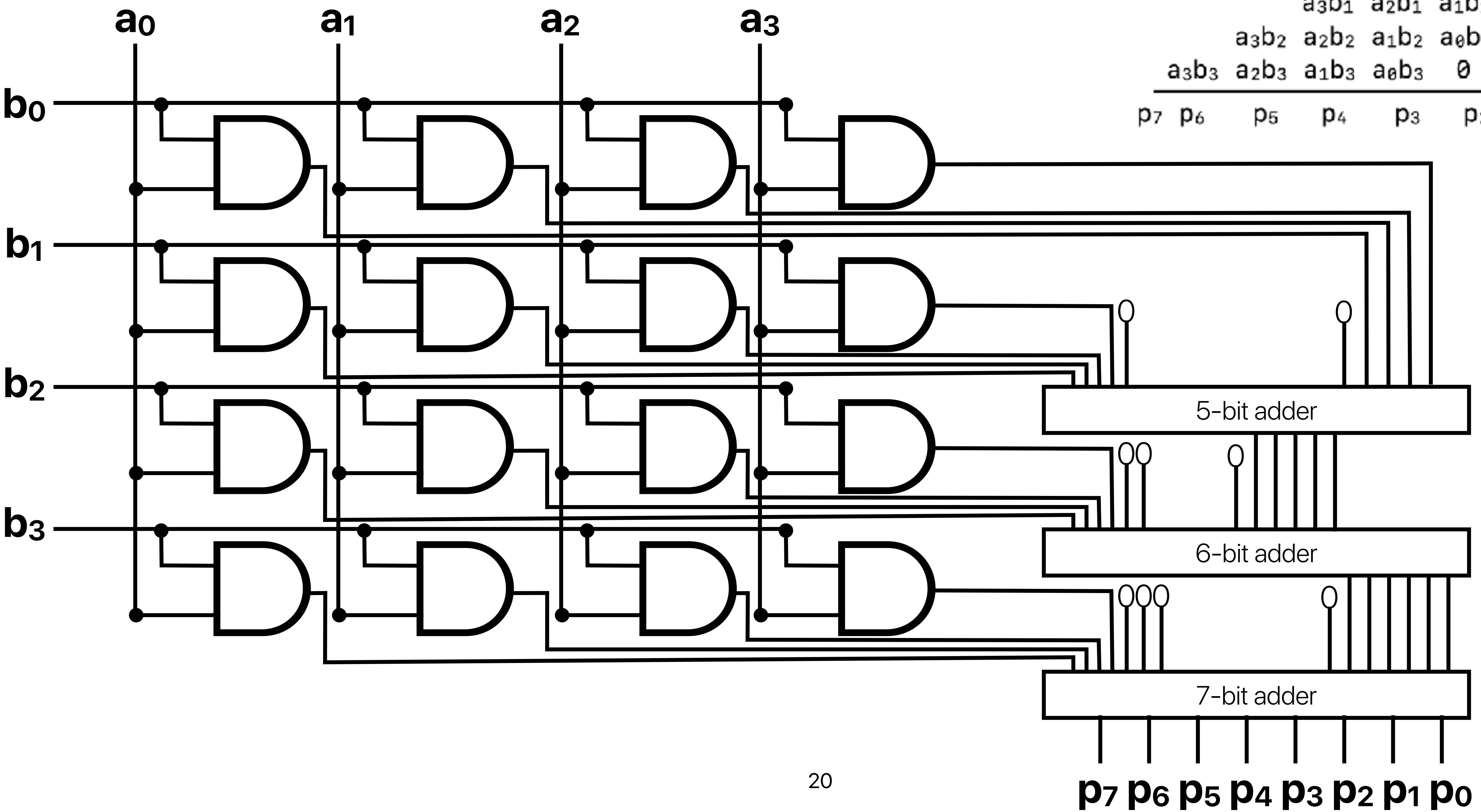
+4

4-bit serial shift-and-add multiplier

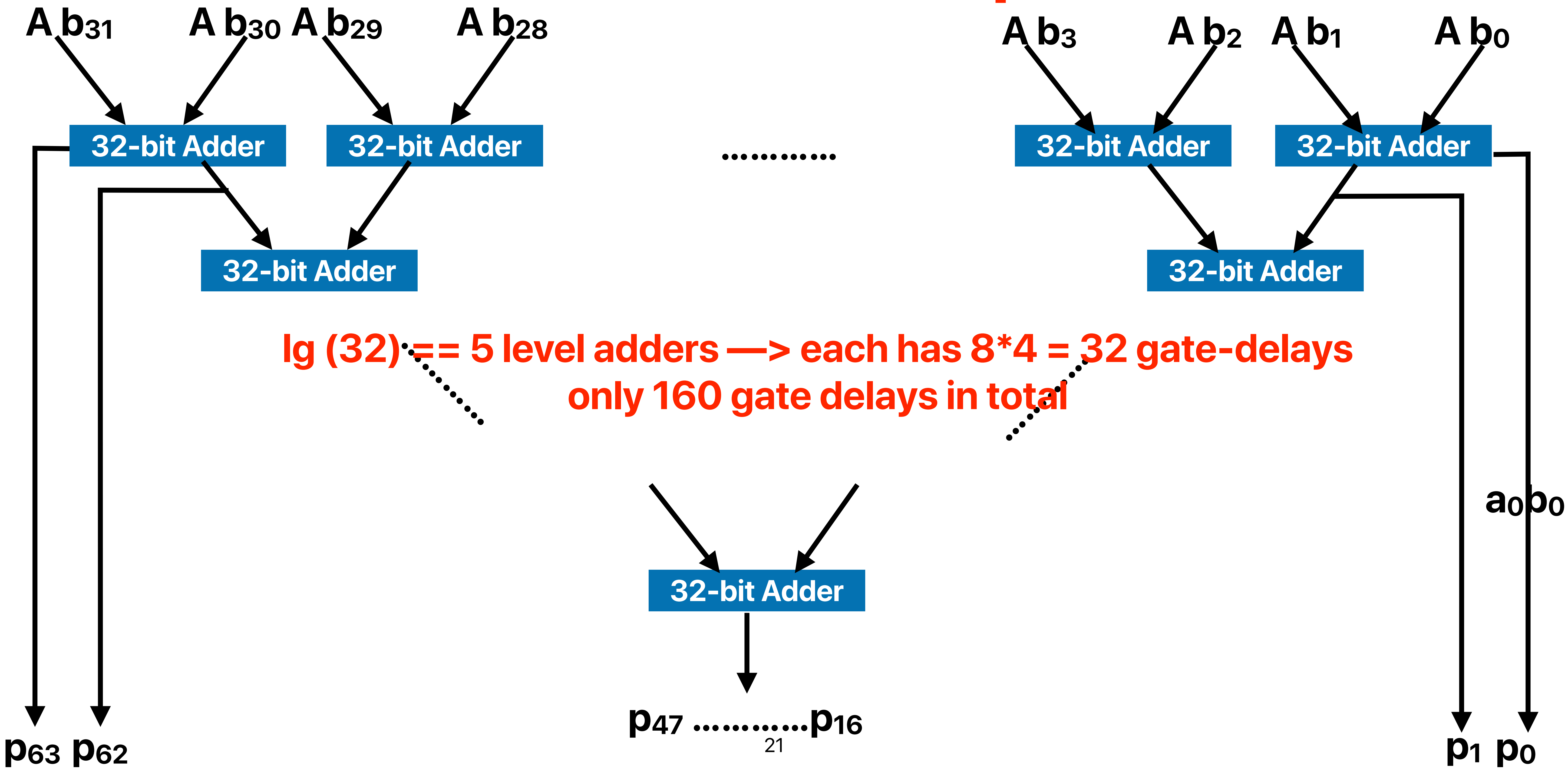


Array style

$$
\begin{array}{cccccccc}
 & & & & a_3 & a_2 & a_1 & a_0 \\
 & & & \times & b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & 0 \\
 & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & 0 & 0 \\
 & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & 0 & 0 & 0 \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$



Parallel-tree Multiplier



lg (32) == 5 level adders —> each has 8*4 = 32 gate-delays

only 160 gate delays in total

Divider

Division of positive binary numbers

- Repeated subtraction
  - Set quotient to 0
  - Repeat while (dividend >= divisor)
    - Subtract divisor from dividend
    - Add 1 to quotient
  - When dividend < divisor:
    - Reminder = dividend
    - Quotient is correct

Put everything all together!
ALU — arithmetic logic unit

Announcement

- Assignment 2 due **TONIGHT**
  - All challenge questions up to **3.5**
- Reading quiz 5 due 4/28 **BEFORE** the lecture
  - Under iLearn > reading quizzes
- Lab 3 due 4/30
  - Watch the video and read the instruction BEFORE your session
  - There are links on both course webpage and iLearn lab section
  - Submit through iLearn > Labs
- Midterm on 5/7 during the lecture time, access through iLearn — no late submission is allowed — make sure you will be able to take that at the time
- Check your grades in iLearn

Electrical
Computer Science
Engineering

120A

つづく