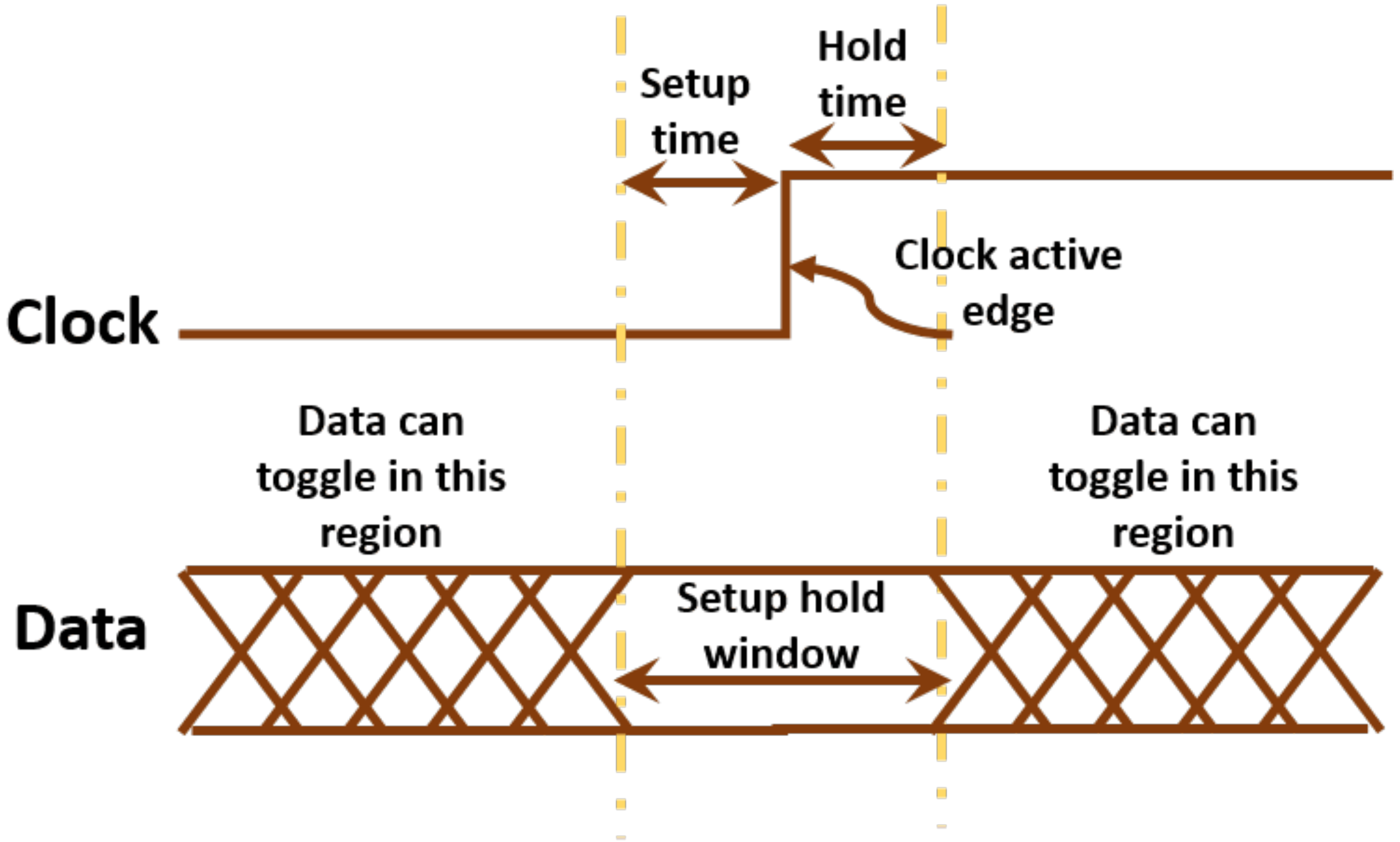
Binary Division

Andrew Lukefahr
Indiana University - Bloomington

Setup and Hold Time



Binary Multiplication

```
  0110
×    1
------
  0110
```

```
  0110
×    0
------
  0000
```

```
       0110
×      0011
-----------
       0110
      01100
     000000
+   0000000
-----------
```

Binary Multiplication

```
 0110
× 1
```

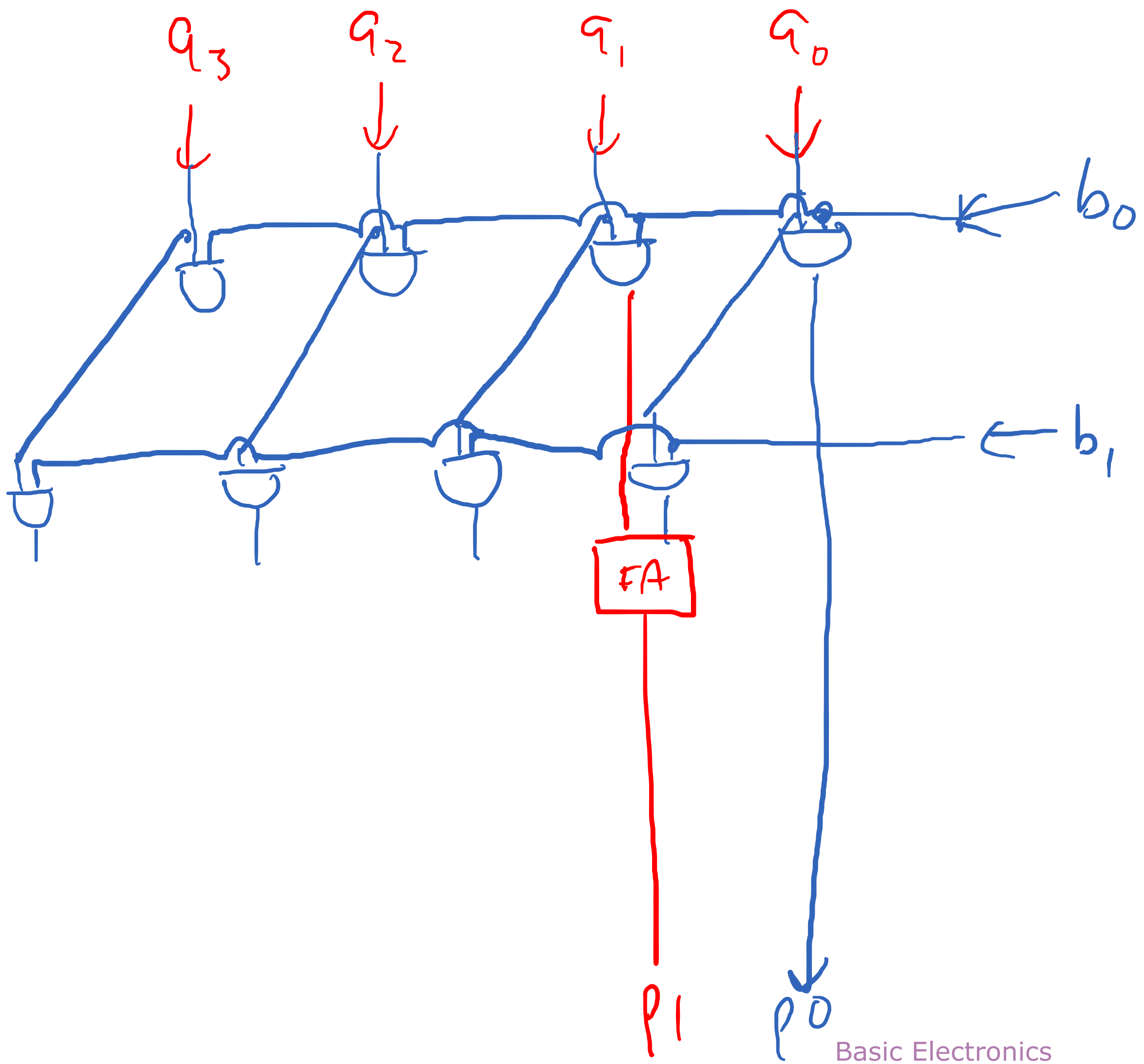
```
 0110
× 0
```

```
    0110
×   0011
    0110
+  01100
+ 000000
+0000000
00010010
```

```
  6
× 3
```

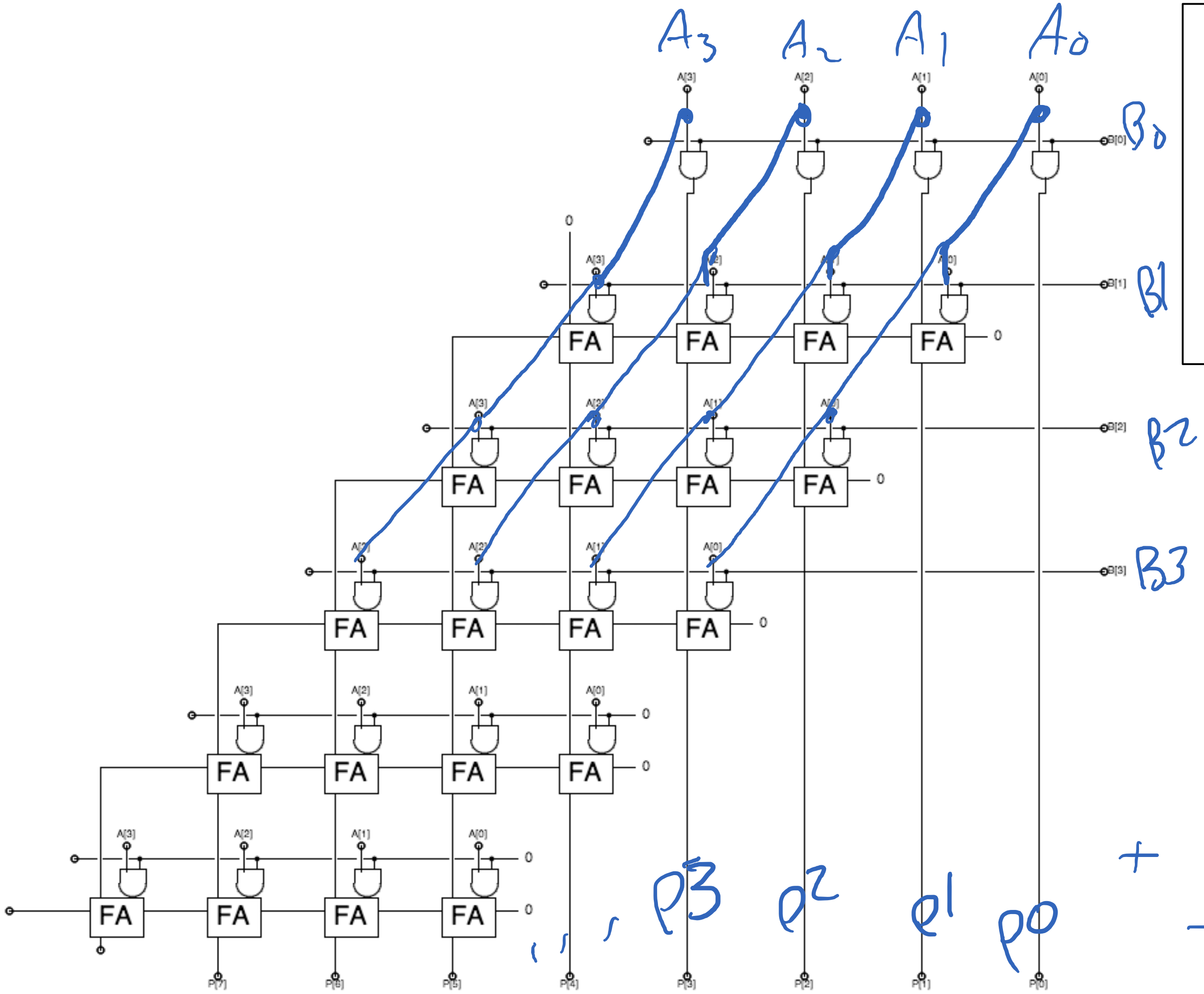
18

Combinational Multiplication



				a3	a2	a1	a0
			×	b3	b2	b1	b0
				b0a3	b0a2	b0a1	b0a0
+			b1a3	b1a2	b1a1	b1a0	0
+		b2a3	b2a2	b2a1	b2a0	0	0
+	b3a3	b3a2	b3a1	b3a0	0	0	0
p7	p6	p5	p4	p3	p2	p1	p0

Combinational Multiplication



				a3	a2	a1	a0
			×	b3	b2	b1	b0
				b0a3	b0a2	b0a1	b0a0
+			b1a3	b1a2	b1a1	b1a0	0
+		b2a3	b2a2	b2a1	b2a0	0	0
+	b3a3	b3a2	b3a1	b3a0	0	0	0
p7	p6	p5	p4	p3	p2	p1	p0

b0a3 b0a2 b0a1 b0a0

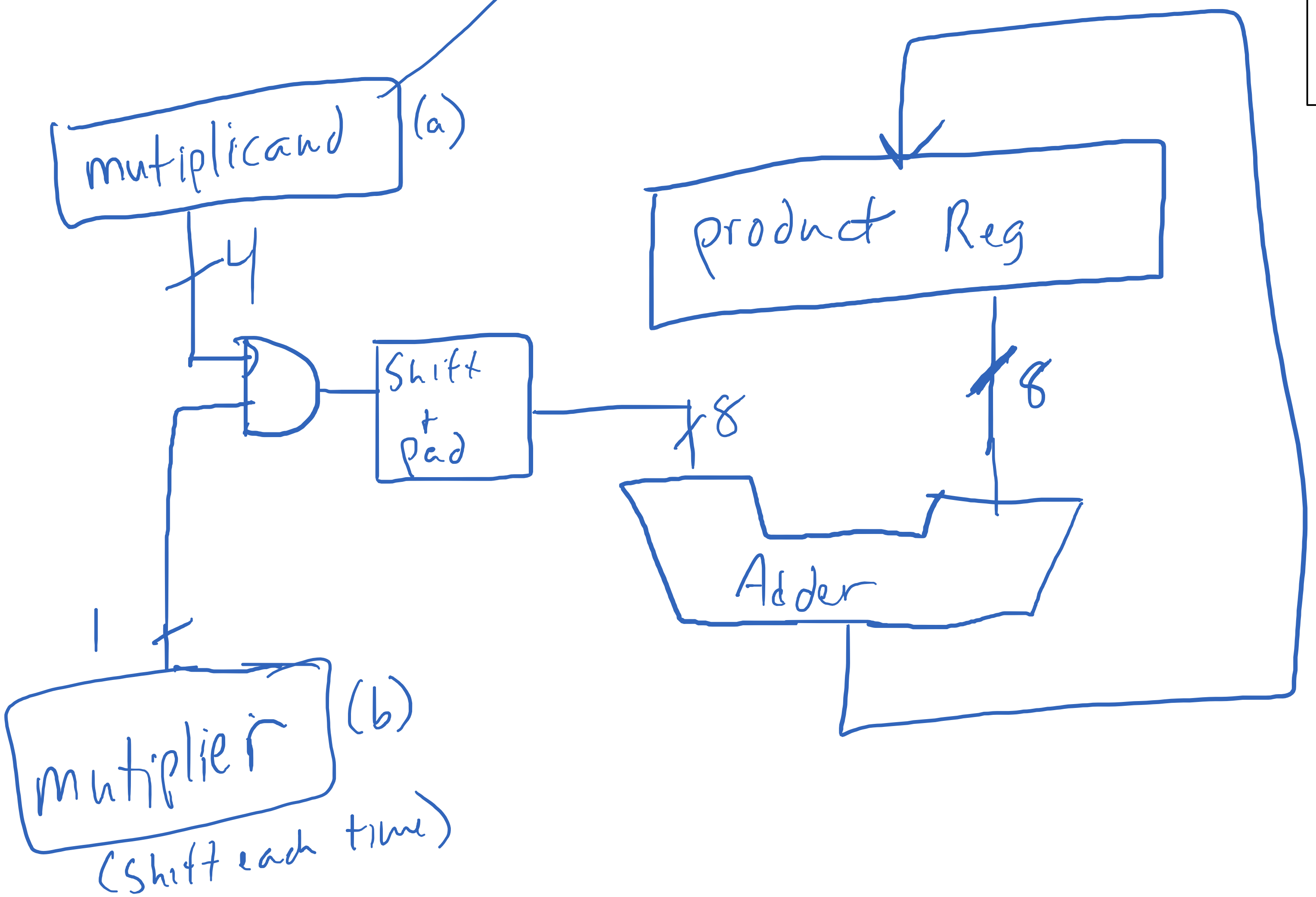
\+ b1a3 b1a2 b1a1 b1a0 0

q4 q3 q2 q1 q0

\+ b2a3 b2a2 b2a1 b2a0

r5 r4 r3 r2 r1 r0

Sequential Multiplication



				a3	a2	a1	a0
			×	b3	b2	b1	b0
0	0	0	0	b0a3	b0a2	b0a1	b0a0
+ 0	0	0	b1a3	b1a2	b1a1	b1a0	0
+ 0	0	b2a3	b2a2	b2a1	b2a0	0	0
+ 0	b3a3	b3a2	b3a1	b3a0	0	0	0
p7	p6	p5	p4	p3	p2	p1	p0

Cycle 0: 0 0 0 0 0 0 0 0

\+ 0 0 0 b0a3 b0a2 ... b0a0

partial product 0

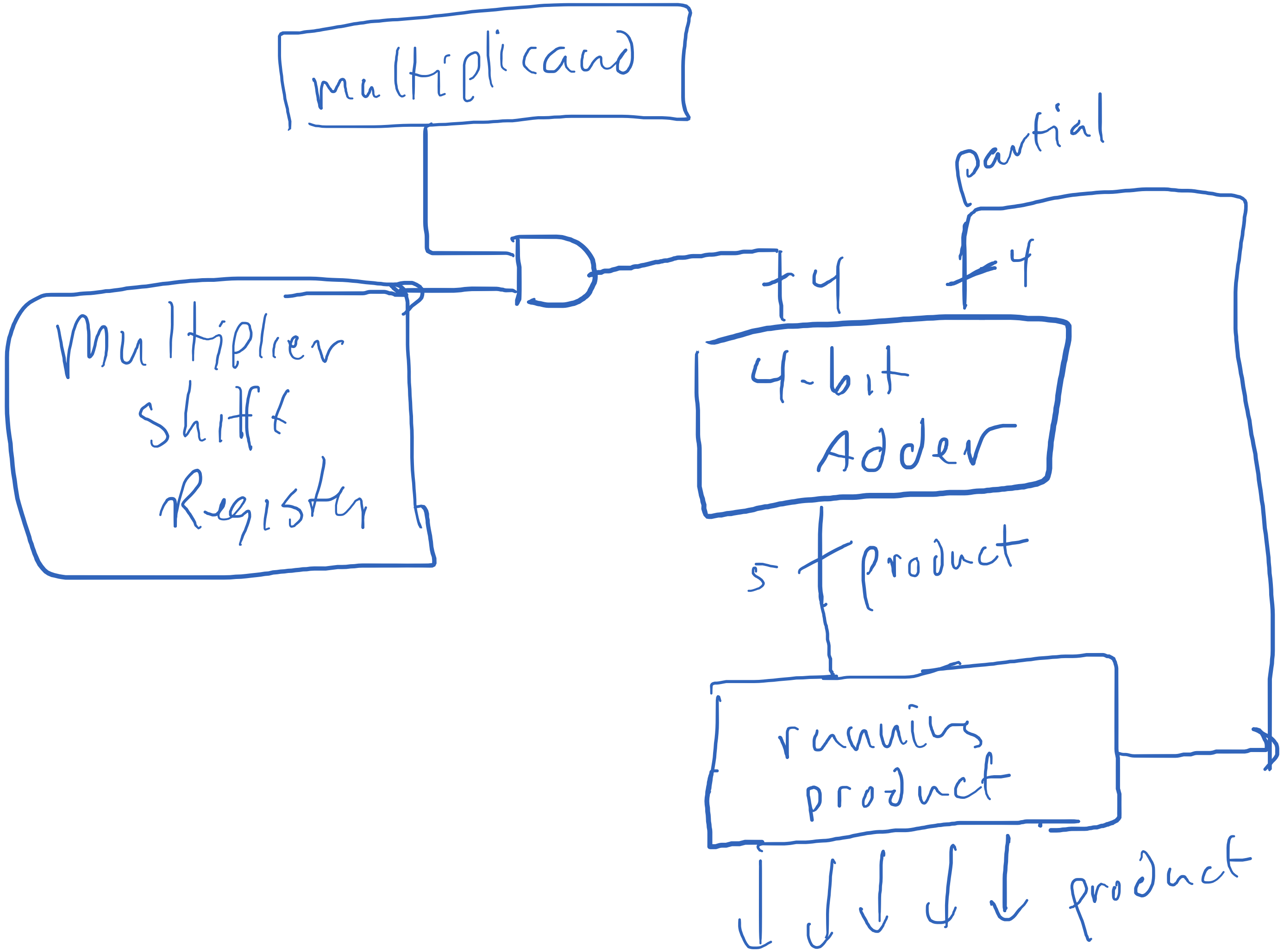
cycle

\+ 0 0 b1a3 b1a2 ... b1a0 0

partial product 1

Sedo Demo

Sequential Multiplication



				a3	a2	a1	a0
			×	b3	b2	b1	b0
				b0a3	b0a2	b0a1	b0a0
+			b1a3	b1a2	b1a1	b1a0	0
+		b2a3	b2a2	b2a1	b2a0	0	0
+	b3a3	b3a2	b3a1	b3a0	0	0	0
p7	p6	p5	p4	p3	p2	p1	p0

clk 1

b0a3 b0a2 b0a1 b0a0
\+ 0 0 0 0

Hybrid Multiplication

				a3	a2	a1	a0
			×	b3	b2	b1	b0
				b0a3	b0a2	b0a1	b0a0
+			b1a3	b1a2	b1a1	b1a0	0
+		b2a3	b2a2	b2a1	b2a0	0	0
+	b3a3	b3a2	b3a1	b3a0	0	0	0
p7	p6	p5	p4	p3	p2	p1	p0

Cycle 1

		b0a3	b0a2	b0a1	b0a0	
+	b1a3	b1a2	b1a1	b1a0	0	
	q5	q4	q3	q2	q1	q0

		b2a3	b2a2	b2a1	b2a0	
+	b3a3	b3a2	b3a1	b3a0	0	
	w7	w6	w5	w4	w3	w2

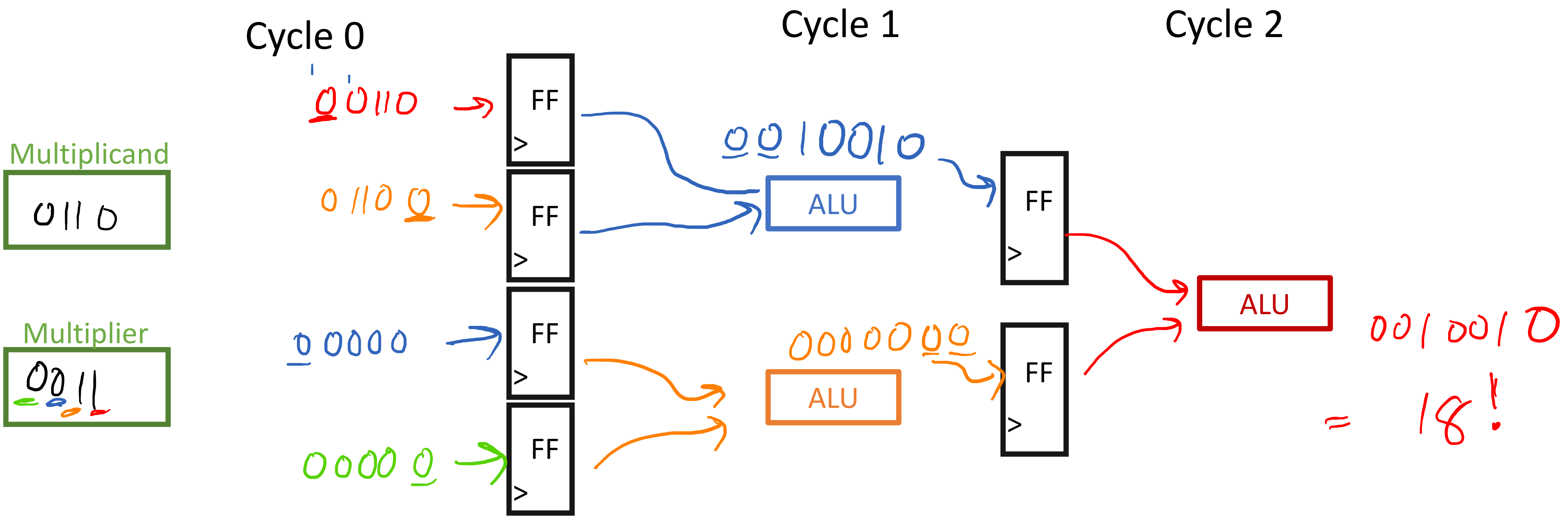
Parallel

Cycle 2

	0	0	q5	q4	q3	q2	q1	q0
+	w7	w6	w5	w4	w3	w2	0	0
	p7	p6	p5	p4	p3	p2	p1	p0

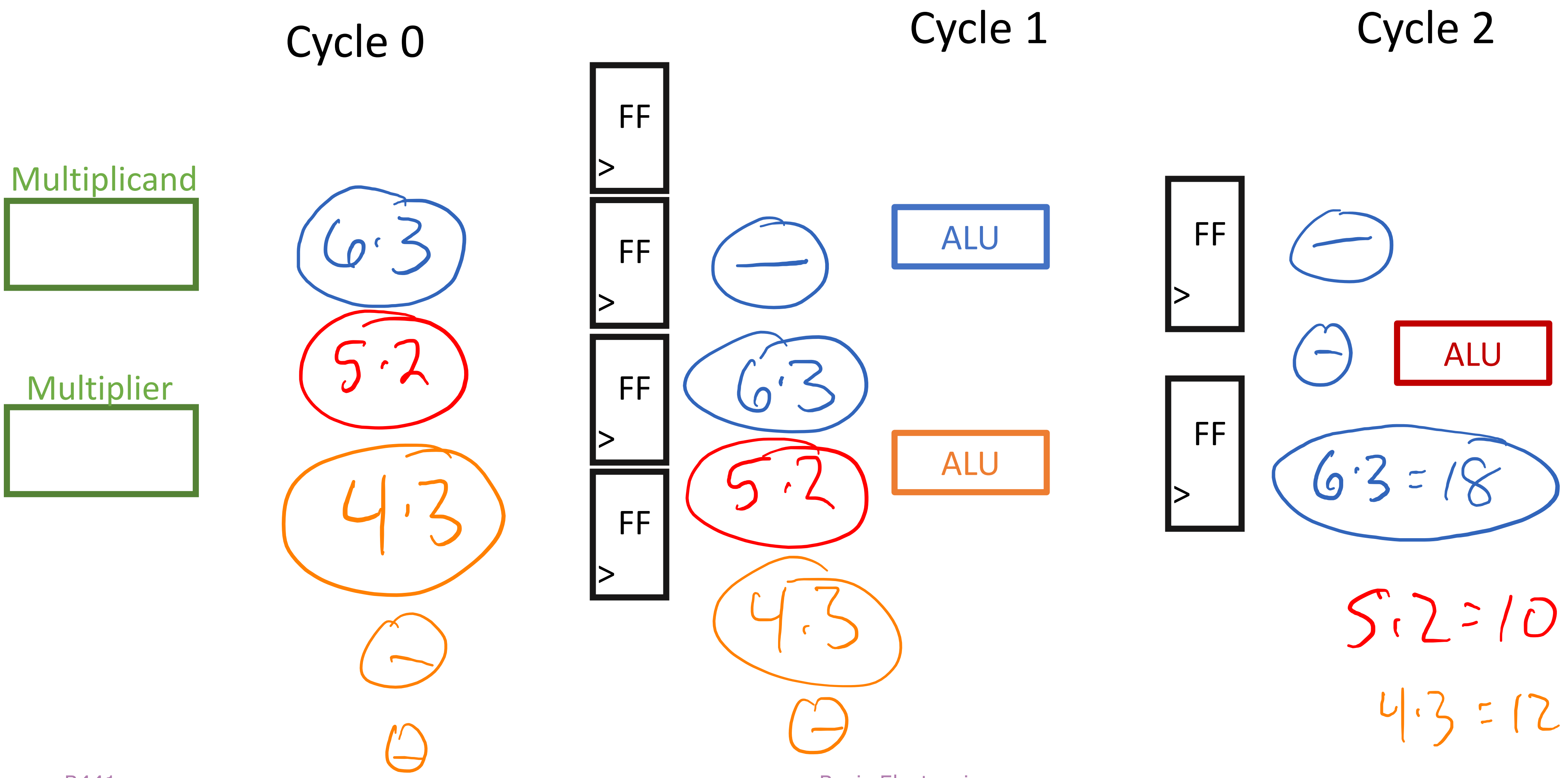
Pipelined Multiplier

6 * 3 = 0110 * 0011 (18)



Pipelined Multiplier

6 * 3
5 * 2
4 * 3



Pipelined Multiplier

	Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4
6·3	X	X	X		
5·4		X	X	X	
4·3			X	X	X

Latency: 3 cycles

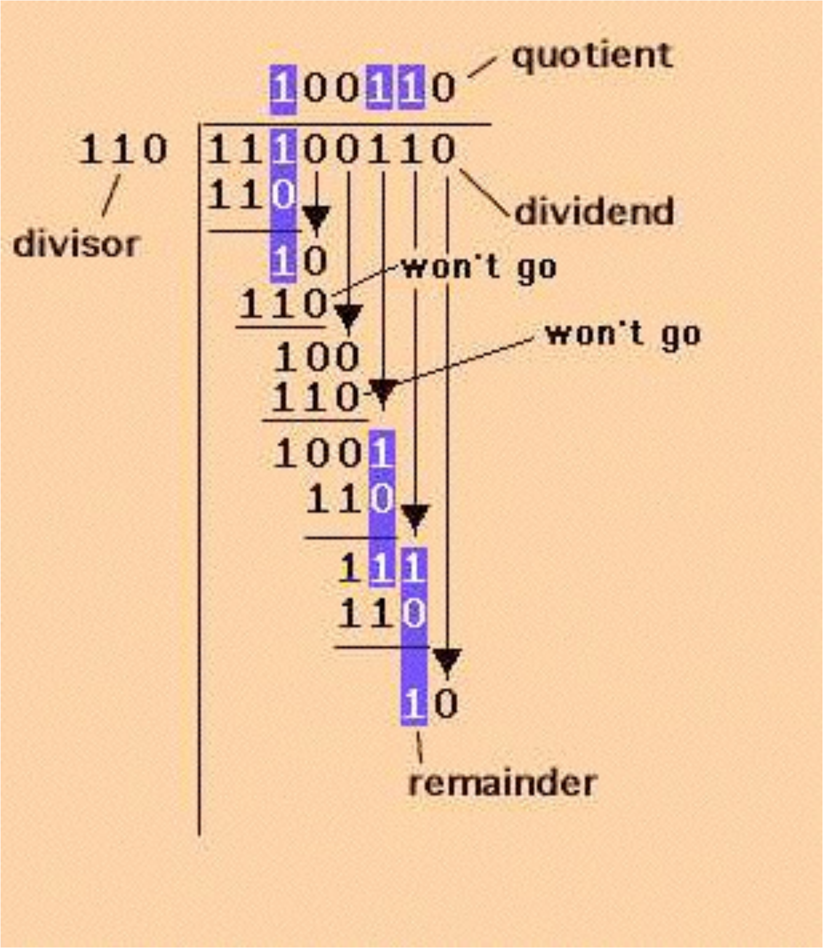
Throughput: 1 multiply/cycle

Latency vs. Throughput

- Latency: How long does it take <u>a single</u> operation to complete
- Throughput: On average, how many operations can you complete <u>per cycle</u>?
- Basis for modern CPUs
- More in E312

Long Division

```
   038  R2
6 | 230
   -18↓
    50
   -48
     2
```



Division Algorithm

- Set quotient to 0
- Align leftmost digits in dividend and divisor
- Repeat
  - If that portion of the dividend above the divisor is greater than or equal to the divisor
    - Then subtract divisor from that portion of the dividend and
    - Concatenate 1 to the right hand end of the quotient
    - Else concatenate 0 to the right hand end of the quotient
  - Shift the divisor one place right
- Until dividend is less than the divisor
- quotient is correct, dividend is remainder
- STOP

Division Example

```
     38  R2
6 | 230
```

```
       000100110
0110 | 011100110
      -0
       01
      -000
       011
      -000
       0111
      -0110
       00010
      -0000
        00100
       -0000
         01001
        -0110
         00111
        -0110
          0001
```

```
  0010
- 0000
    10
```

Remainder

```
  1001
- 0110
  0011
```

Division Example

```
     38 R2
6 | 230
```

```
0110 | 11100110
```

Powers of 2

$$\begin{array}{r} 5 \\ \times \quad 2 \\ \hline 10 \end{array} \qquad \begin{array}{r} 0101 \\ 0010 \\ \hline 01010 \end{array}$$

$$\begin{array}{r} 19 \\ \times \quad 8 \\ \hline 152 \end{array} \qquad \begin{array}{r} 00010011 \\ 00001000 \\ \hline 10011000 \end{array}$$

$\underline{18 \ / \ 2}$ $= 9$

00010010 / 0010

00001001

$$\begin{array}{r} 5 \\ \cdot 10 \\ \hline 50 \end{array}$$

- Multiplication and Division by powers of 2 is just shift.

Take Aways:

1. Multiplication/Division is harder than Addition/Subtraction
    1. Takes longer
    2. Consumes more power
2. Exception: Multiply / Divide by 2^n
    1. This is just shift
3. It is possible to PIPELINE these units
    1. Same <u>Latency</u>, more <u>Throughput</u>

Next Time

- CPU ISAs