EECS 452 – Lecture 4

Today:	Gate level arithmetic Sequential logic
Announcements:	Lab 2 starts next week. Pre-project ideas are due today by 11:55PM. PPI Comment period: Sat 6PM to Wed. 11:55PM Hwk 2 due at beginning of class next week Project teaming meeting: Next thurs 7-9PM Dow 3150

Last one out should close the lab door!!!!

Please keep the lab clean and organized.

Computers are good at following instructions, but not at reading your mind.
– D. Knuth

Arithmetic at the signal/bit level

We have

- introduced the two's complement number representation.
- discussed sign extension.
- discussed overflow.
- commented on 2's complement overflow robustness.

Next we look in more detail at implementing addition, subtraction and multiplication using basic gates. This sets the stage for doing arithmetic in an FPGA like the DE2-70.

Binary addition

Consider adding two 8-bit values a and b

$$\begin{array}{cccccccccc} & & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ + & & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 . \\ \hline \end{array}$$

For each bit position we form the sum of the two bits in that position and add in any carry from the previous (lower index) bit position. The carry into position 1 is 0.

$$\begin{array}{cccccccccc} & c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0 & 0 \\ & & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\ + & & b_7 & b_6 & b_5 & b_4 & b_3 & b_2 & b_1 & b_0 \\ \hline & c_7 & s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0 \end{array}$$

Single bit binary addition

Matrix form of input/output map for 1-bit adder

$a \times b$	0	1
0	0	1
1	1	0

XOR accomplishes single bit addition

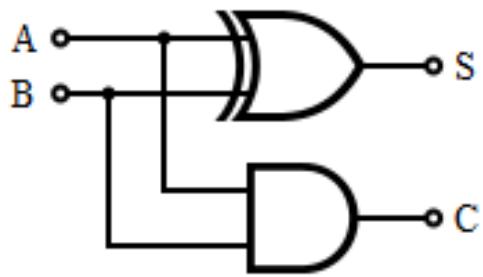
Tabular form of input/output map for 1-bit adder

output	inputs	
S	B	A
0	0	0
1	0	1
1	1	0
0	1	1

- Logic operation is $S = A \cdot \overline{B} + \overline{A} \cdot B$
- XOR adder overflows the 1 bit addition.

A single bit half adder

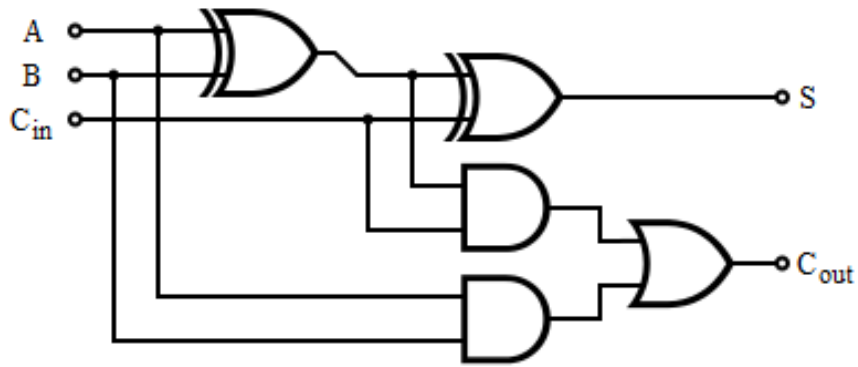
outputs		inputs	
C_{out}	S	B	A
0	0	0	0
0	1	0	1
0	1	1	0
1	0	1	1



Two types of gates are used: AND and XOR.

A single bit full adder unit (1/2)

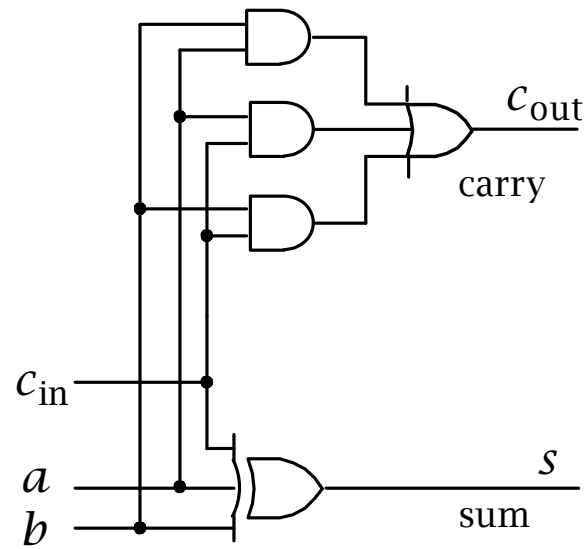
outputs		inputs		
C_{out}	S	B	A	C_{in}
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
1	0	1	0	1
1	0	1	1	0
1	1	1	1	1



Three types of gates are used: AND, OR, and XOR.

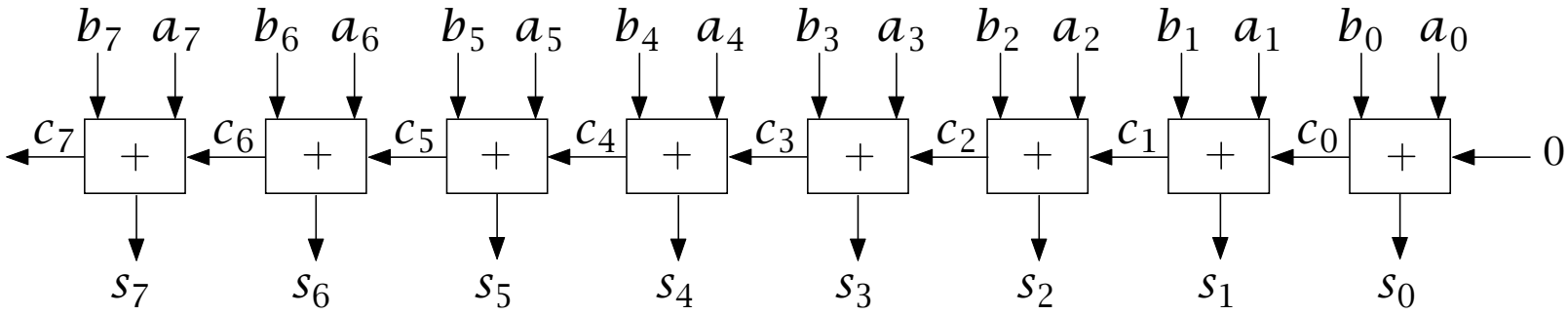
Equivalent 3-input XOR implementation (2/2)

outputs		inputs		
c_{out}	s	b	a	c_{in}
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	1	1
0	1	1	0	0
1	0	1	0	1
1	0	1	1	0
1	1	1	1	1



Three types of gates are used: AND, OR, and 3-input XOR.

Ripple carry adder – multiple-bit adder



Intrinsically serial: cannot clock all of the FA's at the same time due to ripple delays

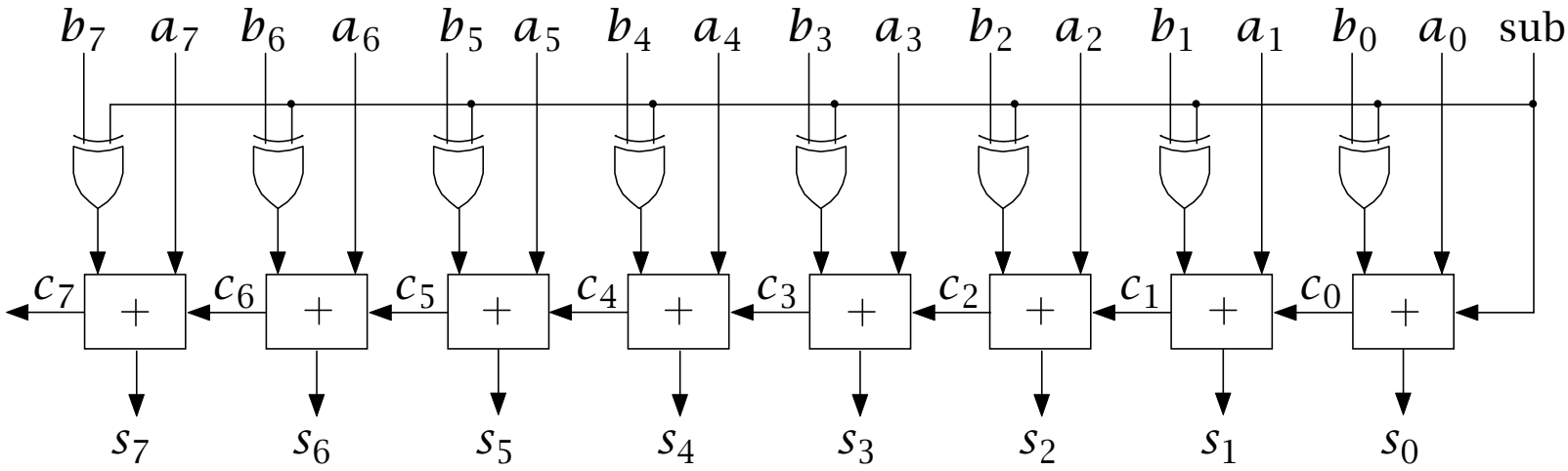
Execution time limited by the time required for carries to propagate from least significant bit to most significant bit..

Subtraction

To subtract b from a simply negate b and add. For two's complement numbers negation consists of complementing the individual bits and adding one. The addition of one can be accomplished by using a carry of one into bit position zero.

$$
\begin{array}{cccccccccc}
 & c_7 & c_6 & c_5 & c_4 & c_3 & c_2 & c_1 & c_0 & 1 \\
 & & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \\
+ & & \overline{b_7} & \overline{b_6} & \overline{b_5} & \overline{b_4} & \overline{b_3} & \overline{b_2} & \overline{b_1} & \overline{b_0} \\
\hline
 & c_7 & s_7 & s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0
\end{array}
$$

Ripple carry add/subtract



Sub is logical 0 for addition, logical 1 for subtraction.

sub	b	sub exor b
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-or gate is used as controlled buffer/inverter.

Sequential operation—bit-serial arithmetic

Using a *single* one-bit full adder to perform multiple-bit addition/subtraction using feedback.

- Used many years ago when logic was dearly expensive, bulky and power hungry.
- Often used in hand held calculators.
- Generally requires less FPGA fabric area than parallel.
- Generally slower than parallel, but not necessarily.
- Dr. Metzger designed and implemented a PN sequence correlator that adds 63 16-bit numbers in 25 clock cycles.
- Generally one trades longer execution time for smaller FPGA footprint.

Bit serial adder (1/2)

The basic element is a one-bit full adder with the carry bit fed back through a register. Values are shifted through the adder starting least significant bit first (see next slide)

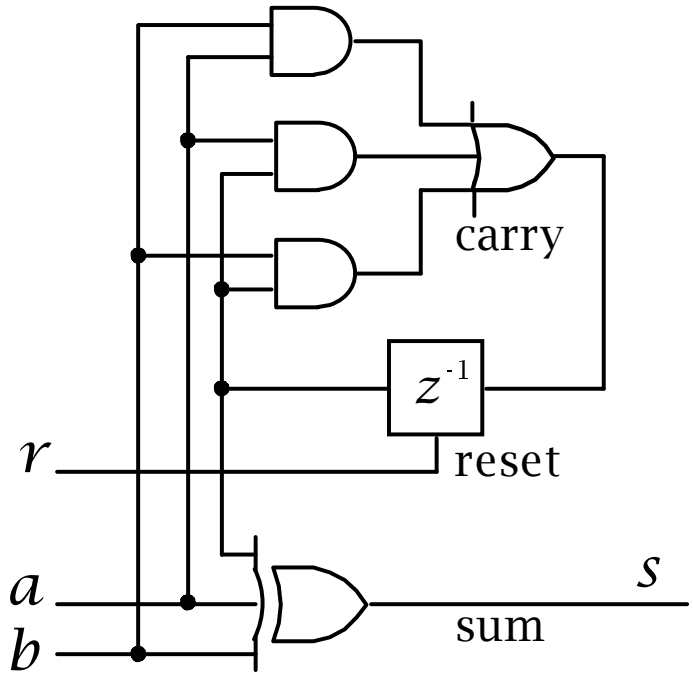
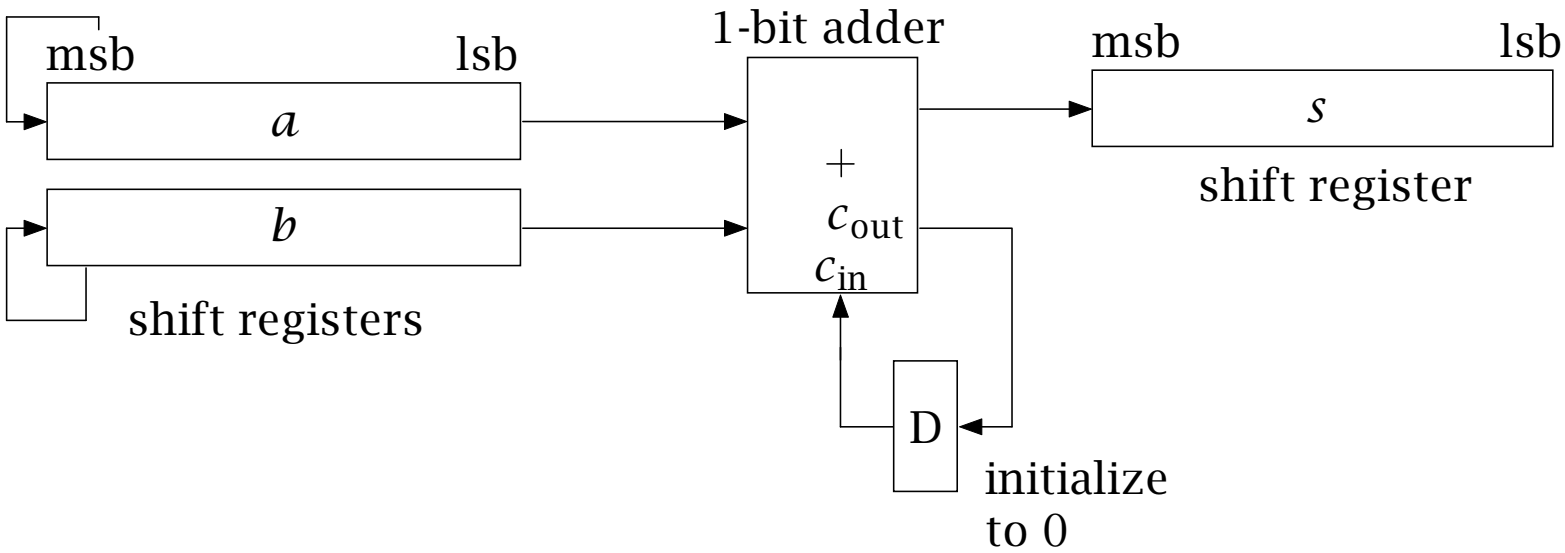


Figure: Bit serial adder block diagram.

To subtract invert the bits of the value being subtracted and initialize the carry bit to one.

Bit serial adder (2/2)

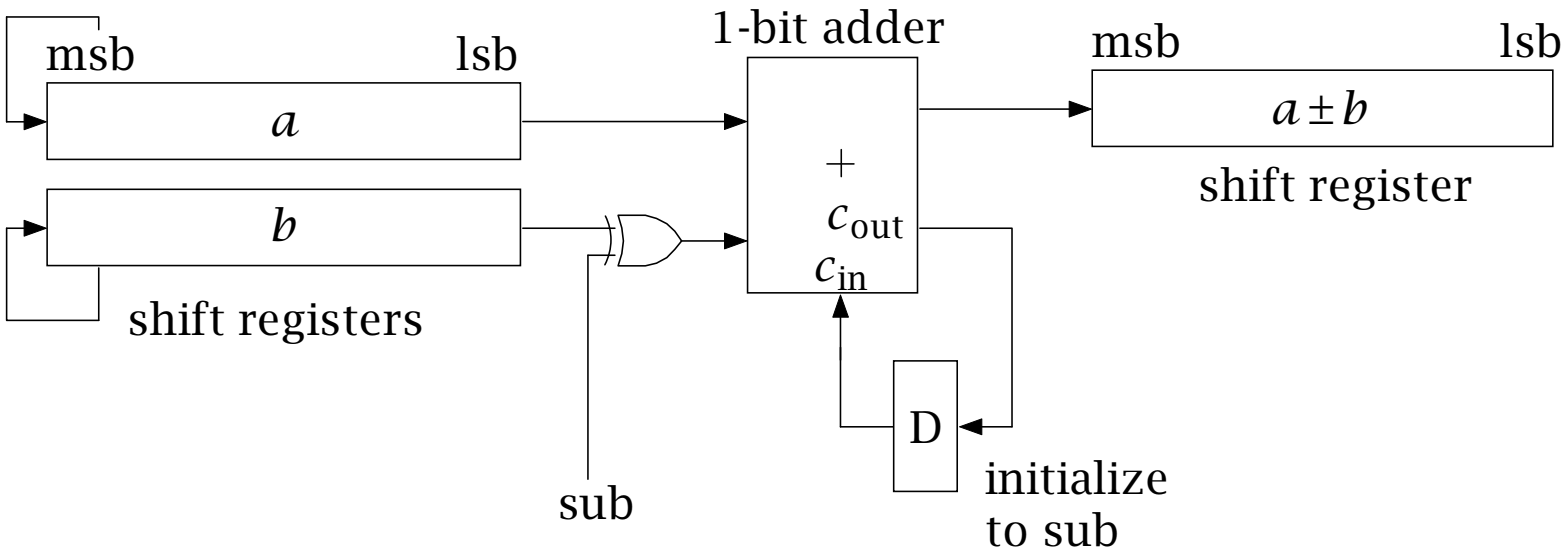


Minimal logic. Can be clocked at high rates.

Execution time strongly influenced by word size.

Not shown is the control logic needed to step the operation. This might be as simple as a counter.

Bit serial add/subtract



Minimal logic. Can be clocked at high rates.

Execution time strongly influenced by word size.

Adder 1-bit carry memory initialized depending on whether adding or subtracting.

Binary multiplication

Matrix form of truth table for 1-bit multiplier

$a \times b$	0	1
0	0	0
1	0	1

Tabular form of truth table for 1-bit multiplier

output	inputs	
S	B	A
0	0	0
0	0	1
0	1	0
1	1	1

Note that this is equivalent to the logical AND operation.

Recall unsigned and two's complement values

A B-bit binary number is an ordered sequence of zero and one values:

$$(b_{B-1}, b_{B-2}, \cdots, b_1, b_0).$$

The unsigned value represented by this sequence is

$$v = b_{B-1}2^{B-1} + b_{B-2}2^{B-2} + \cdots + b_1 2^1 + b_0 2^0 = \sum_{i=0}^{B-1} b_i 2^i \, .$$

v can take on the values from 0 through $2^B - 1$ in steps of one.

The two's complement value represented by this sequence is

$$v = -b_{B-1}2^{B-1} + b_{B-2}2^{B-2} + \cdots + b_1 2^1 + b_0 2^0.$$

v can take on the values from -2^{B-1} through $2^{B-1} - 1$ in steps of one.

Unsigned and two's complement addition and subtraction use the same logic. This is not necessarily so for multiplication.

Unsigned binary multiplication

Consider the multiplication of the two 4-bit unsigned binary numbers $a = a_3a_2a_1a_0$ and $b = b_3b_2b_1b_0$ giving the product $p = a \times b$. Using the standard paper and pencil method we would write:

$$
\begin{array}{cccccccccc}
 & b_0 & \times & & & & a_3 & a_2 & a_1 & a_0 \\
+ & b_1 & \times & & & a_3 & a_2 & a_1 & a_0 & \\
+ & b_2 & \times & & a_3 & a_2 & a_1 & a_0 & & \\
+ & b_3 & \times & a_3 & a_2 & a_1 & a_0 & & & \\
\hline
 & & & p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

The b_i multiplications are by 0 or 1. Rows are added using unsigned addition.

Two's complement multiplication (1/4)

This is the same solution shown in the previous lecture:

- sign extension followed by signed multiplication.

$$
\begin{array}{cccccccccccc}
 & b_0 & \times & & & & & a_3 & a_3 & a_2 & a_1 & a_0 \\
+ & b_1 & \times & & & & a_3 & a_3 & a_2 & a_1 & a_0 & \\
+ & b_2 & \times & & & a_3 & a_3 & a_2 & a_1 & a_0 & & \\
+ & b_3 & \times & & a_3 & a_3 & a_2 & a_1 & a_0 & & & \\
- & b_3 & \times & a_3 & a_3 & a_2 & a_1 & a_0 & & & & \\
\hline
 & & & p_8 & p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

Can we convince ourselves that p in two's complement format is the correct solution?

Two's complement multiplication (2/4)

First, what is the correct answer?

$$\begin{aligned} v_a &= -a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0 \\ v_b &= -b_3 2^3 + b_2 2^2 + b_1 2^2 + b_0 2^0 \end{aligned}$$

The coefficients of different order terms can be given in the table below:

$v_a v_b$	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
			a_3b_3	$-a_3b_2$	$-a_3b_1$	$-a_3b_0$	a_2b_0	a_1b_0	a_0b_0
				$-a_2b_3$	a_2b_2	a_2b_1	a_1b_1	a_0b_1	
					$-a_1b_3$	a_1b_2	a_0b_2		
						$-a_0b_3$			

Two's complement multiplication (3/4)

Let's now verify that the two's complement multiplication does give the correct answer

$$
\begin{array}{cccccccccccc}
 & b_0 & \times & & & & & a_3 & a_3 & a_2 & a_1 & a_0 \\
+ & b_1 & \times & & & & a_3 & a_3 & a_2 & a_1 & a_0 & \\
+ & b_2 & \times & & & a_3 & a_3 & a_2 & a_1 & a_0 & & \\
+ & b_3 & \times & & a_3 & a_3 & a_2 & a_1 & a_0 & & & \\
- & b_3 & \times & a_3 & a_3 & a_2 & a_1 & a_0 & & & & \\
\hline
 & & & p_8 & p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

p	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	a_3b_3	$-a_3b_3$	$-a_3b_2$	$-a_3b_1$	$-a_3b_0$	a_3b_0	a_2b_0	a_1b_0	a_0b_0
		$-a_3b_3$	a_3b_3	a_3b_2	a_3b_1	a_2b_1	a_1b_1	a_0b_1	
			$-a_2b_3$	a_2b_3	a_2b_2	a_1b_2	a_0b_2		
				$-a_1b_3$	a_1b_3	a_0b_3			
					$-a_0b_3$				

Two's complement multiplication (4/4)

Continue. . .

p	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	a_3b_3	$-a_3b_3$	$-a_3b_2$	$-a_3b_1$	$-a_3b_0$	a_3b_0	a_2b_0	a_1b_0	a_0b_0
		$-a_3b_3$	a_3b_3	a_3b_2	a_3b_1	a_2b_1	a_1b_1	a_0b_1	
			$-a_2b_3$	a_2b_3	a_2b_2	a_1b_2	a_0b_2		
				$-a_1b_3$	a_1b_3	a_0b_3			
					$-a_0b_3$				

Simple arithmetic $\Longrightarrow$:

p	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
			a_3b_3	$-a_3b_2$	$-a_3b_1$	$-a_3b_0$	a_2b_0	a_1b_0	a_0b_0
				$-a_2b_3$	a_2b_2	a_2b_1	a_1b_1	a_0b_1	
					$-a_1b_3$	a_1b_2	a_0b_2		
						$-a_0b_3$			

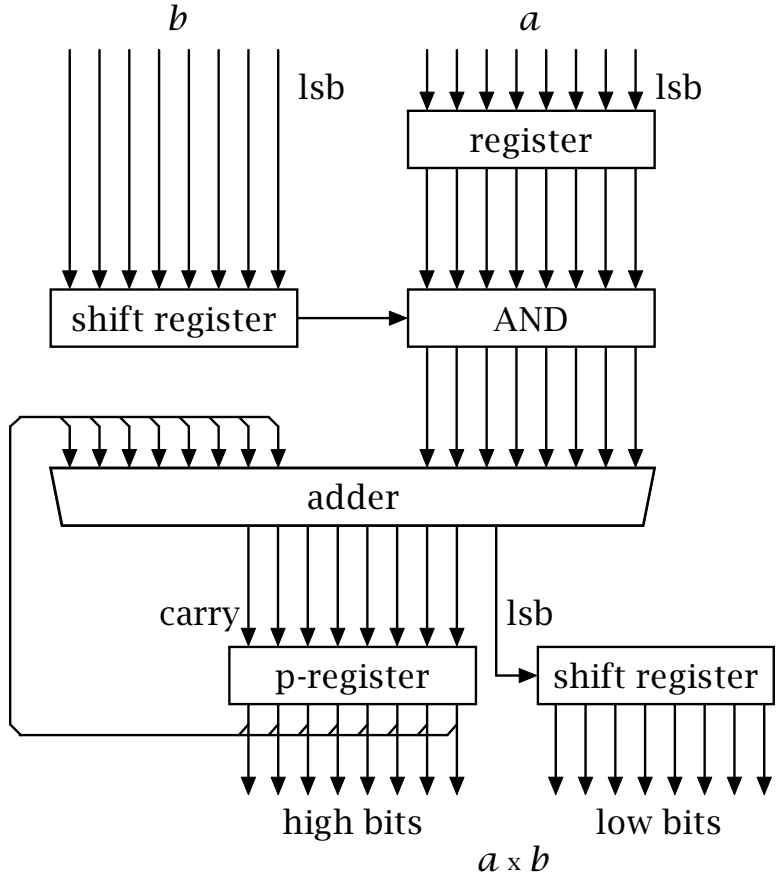
The same with $v_a v_b$!

An equivalent two's complement multiplication operation

$$
\begin{array}{cccccccccc}
 & b_0 & \times & a_3 & a_3 & a_3 & a_3 & a_3 & a_2 & a_1 & a_0 \\
+ & b_1 & \times & a_3 & a_3 & a_3 & a_3 & a_2 & a_1 & a_0 & \\
+ & b_2 & \times & a_3 & a_3 & a_3 & a_2 & a_1 & a_0 & & \\
- & b_3 & \times & a_3 & a_3 & a_2 & a_1 & a_0 & & & \\
\hline
 & & & p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

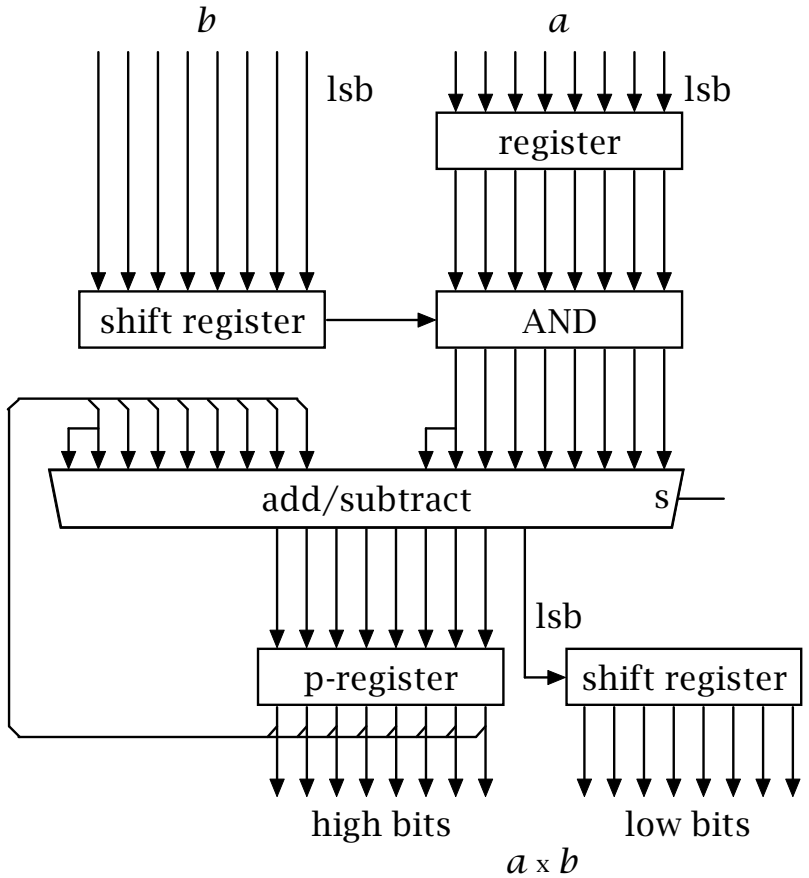
- Again, first sign extension
- Multiply by b_i, either 0 or 1, then two's complement addition.
- The last row subtraction (done using addition: complement all the bits and add 1).
- Can use exactly the same method to show correctness.
- This alternative architecture directly leads to the implementation discussed next.

Unsigned shift and add multiplier



$$
\begin{array}{cccccccc}
 & & & & b_0a_3 & b_0a_2 & b_0a_1 & b_0a_0 \\
 & & & b_1a_3 & b_1a_2 & b_1a_1 & b_1a_0 & \\
 & & b_2a_3 & b_2a_2 & b_2a_1 & b_2a_0 & & \\
 & b_3a_3 & b_3a_2 & b_3a_1 & b_3a_0 & & & \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

Signed shift-and-add multiplier



$$
\begin{array}{cccccccccc}
 & b_0 & \times & a_3 & a_3 & a_3 & a_3 & a_3 & a_2 & a_1 & a_0 \\
+ & b_1 & \times & a_3 & a_3 & a_3 & a_3 & a_2 & a_1 & a_0 & \\
+ & b_2 & \times & a_3 & a_3 & a_3 & a_2 & a_1 & a_0 & & \\
- & b_3 & \times & a_3 & a_3 & a_2 & a_1 & a_0 & & & \\
\hline
 & & & p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

How do these work

These two circuits typically are included in modern texts on computer arithmetic as the starting point for much more interesting logic.

Basically zeros the accumulator. Next adds successive rows shifting the result right 1 each time to simplify the logic.

Once all of the rows have been added you get the product in the accumulator and the register that the accumulator had been shifted into.

For the signed multiply the sign only needs to be extended one position (at a time). The last row is subtracted to form the final sum.

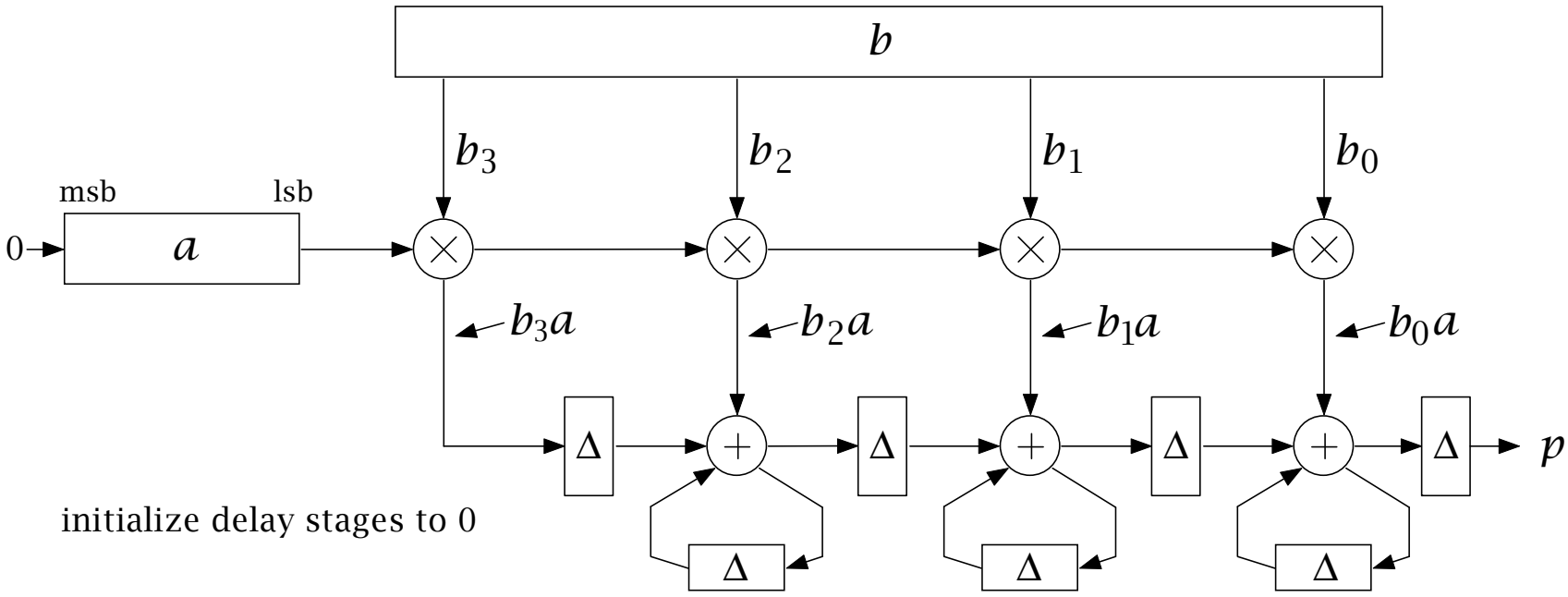
Pipelined bit-serial multiplier

Going from least to most significant bit, generates the sums of the columns bit at a time. (Lyon:76)

$$
\begin{array}{cccccccc}
 & & & & b_0a_3 & b_0a_2 & b_0a_1 & b_0a_0 \\
 & & & b_1a_3 & b_1a_2 & b_1a_1 & b_1a_0 & \\
 & & b_2a_3 & b_2a_2 & b_2a_1 & b_2a_0 & & \\
 & b_3a_3 & b_3a_2 & b_3a_1 & b_3a_0 & & & \\
\hline
p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

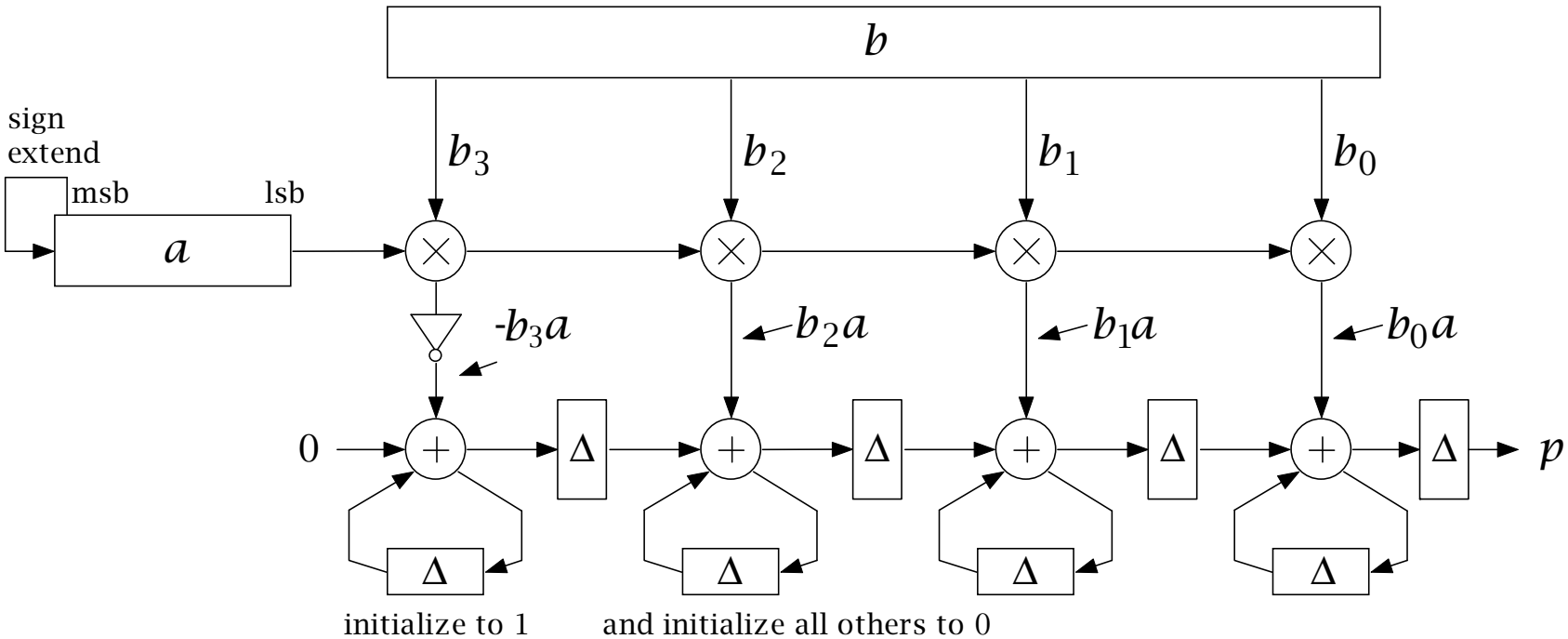
A thirty year old classic. Instead of adding up the rows the columns are generated and summed. Needs about twice as many clock tics per product as the previous method.

Unsigned bit serial multiplier

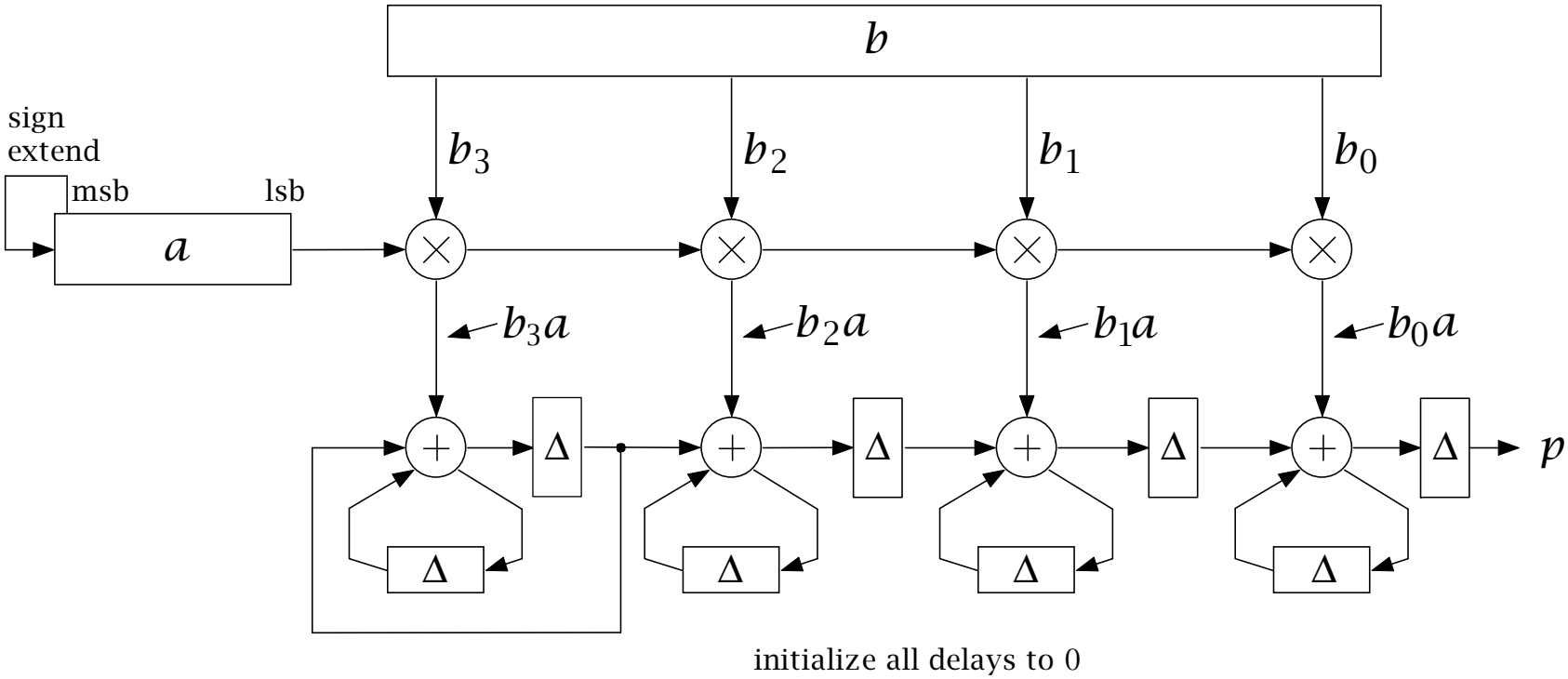


$$
\begin{array}{cccccccccc}
 & b_0 & \times & & & & a_3 & a_2 & a_1 & a_0 \\
+ & b_1 & \times & & & a_3 & a_2 & a_1 & a_0 & \\
+ & b_2 & \times & & a_3 & a_2 & a_1 & a_0 & & \\
+ & b_3 & \times & a_3 & a_2 & a_1 & a_0 & & & \\
\hline
 & & p_7 & p_6 & p_5 & p_4 & p_3 & p_2 & p_1 & p_0
\end{array}
$$

Signed bit serial multiplier

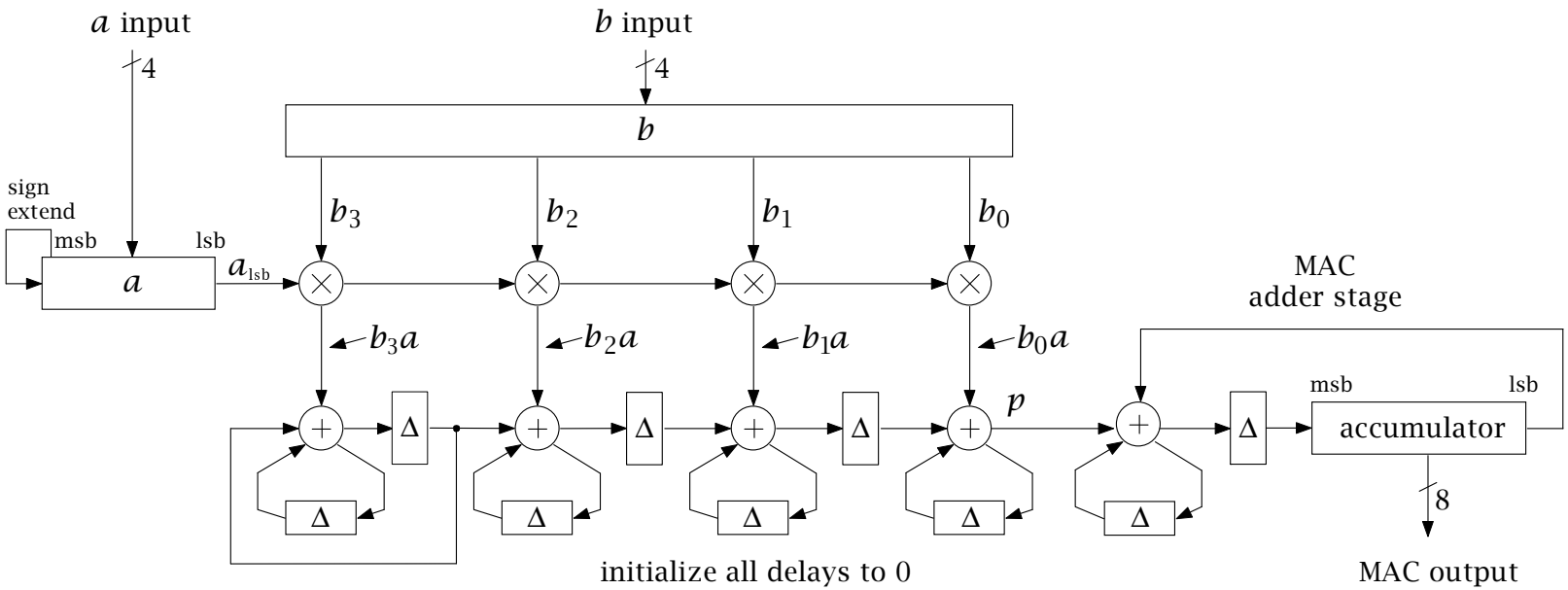


Modified signed bit serial multiplier



I was tracing a particular author's papers in the literature. One paper he was drawing his logic like in the preceding slide and the next he had this very clever version.

Bit-serial multiply-accumulate (MAC)

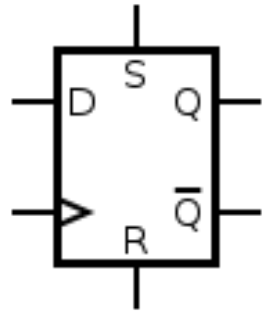


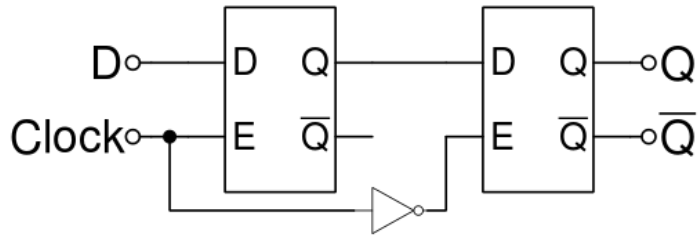
Combining the bit-serial multiplier with the bit-serial adder gives a bit-serial multiply-and-add implementation. (Some control logic needed).

Sequentially adds results of several multiplies: specialized FIR filter computation $y_k = \sum_{i=0}^{M} b_i x[k-i]$

Sequential logic: D flip-flop

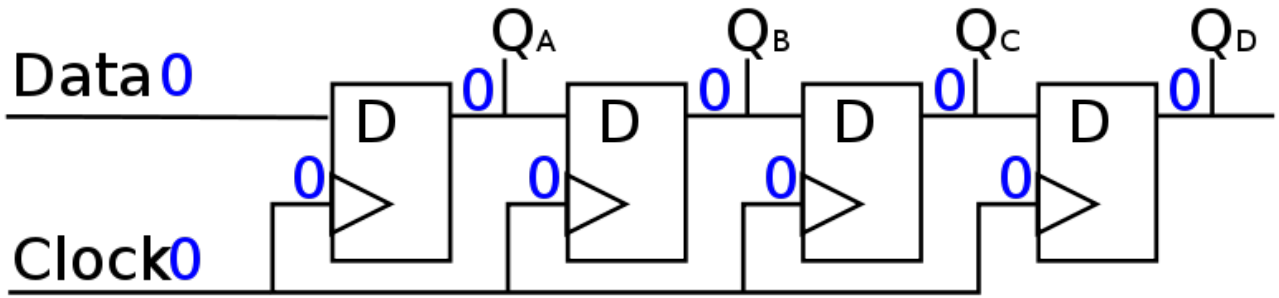
current state		next state
Q	D	Q^+
X	0	0
X	0	0
X	1	1
X	1	1





- A basic storage element and fundamental building block
- Basic D flip-flop: state Q^+ locks onto input logic level D on rising clock edge
- A master-slave D flip-flop: locks onto D on falling edge of the clock

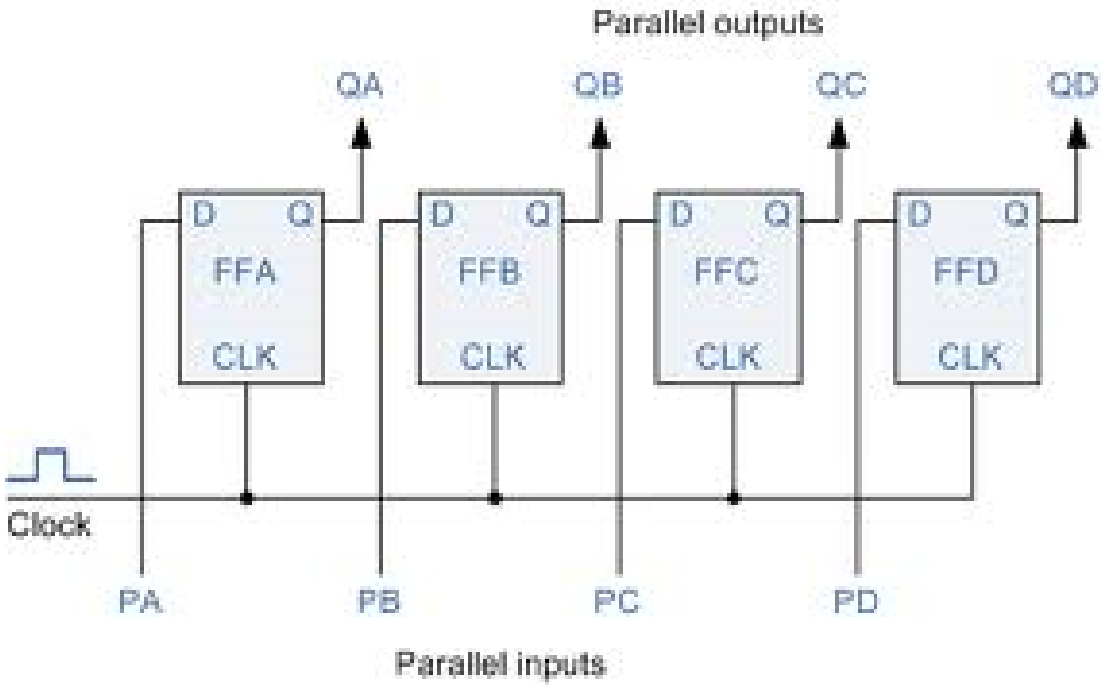
Serial-input parallel output (SIPO) register



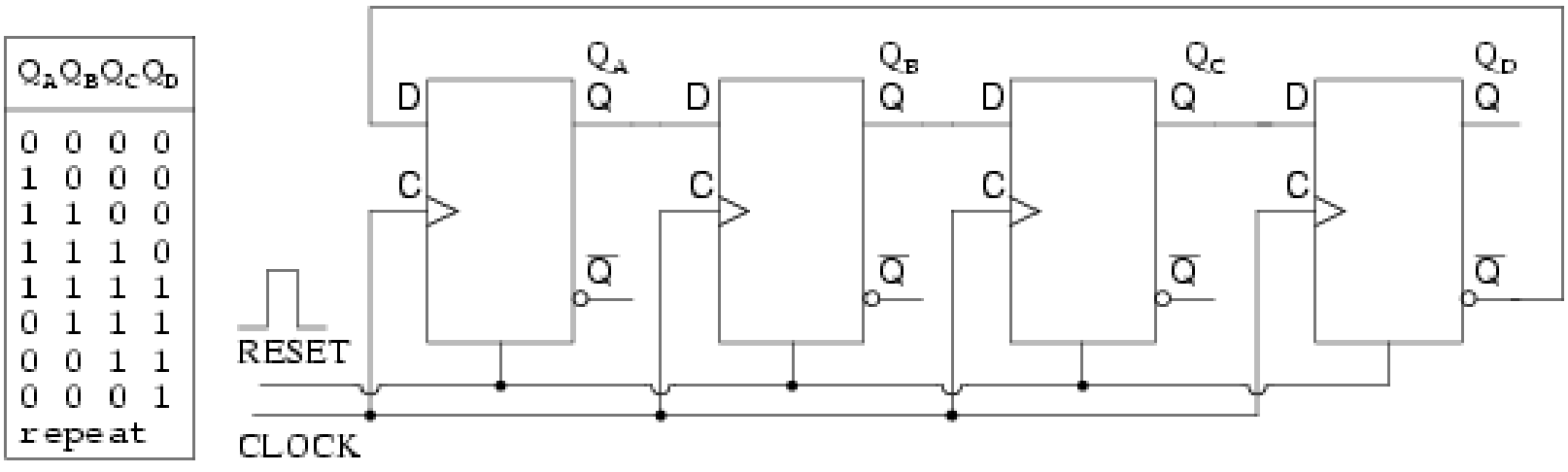
- Input Data is shifted into the leftmost bit position.
- Shifts the contents of the register to the right, one bit position on each active transition of the clock.

Parallel-input parallel output (PIPO) register

4-bit PIPO



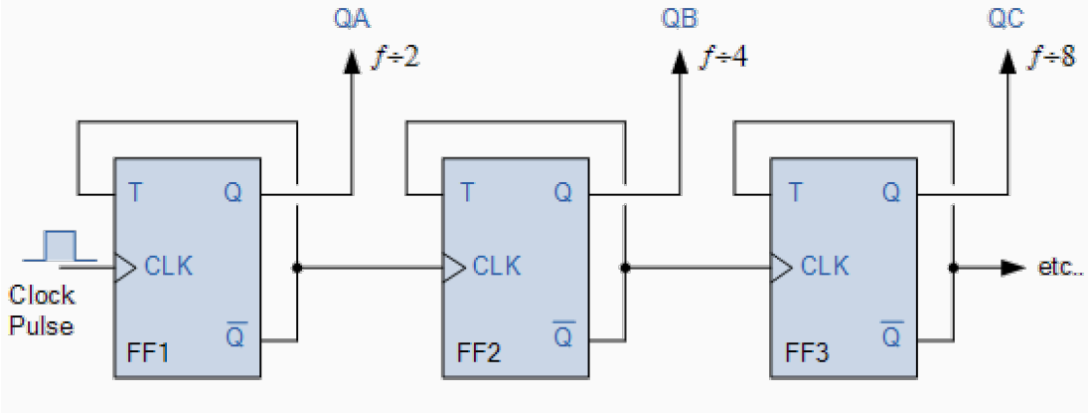
4-bit switch-tail ring counter



Johnson counter (note the $\overline{Q}_D$ to D_A feedback connection)

4-bit switch-tail ring counter (Johnson counter)

3 bit asynchronous counter with Toggle FFs



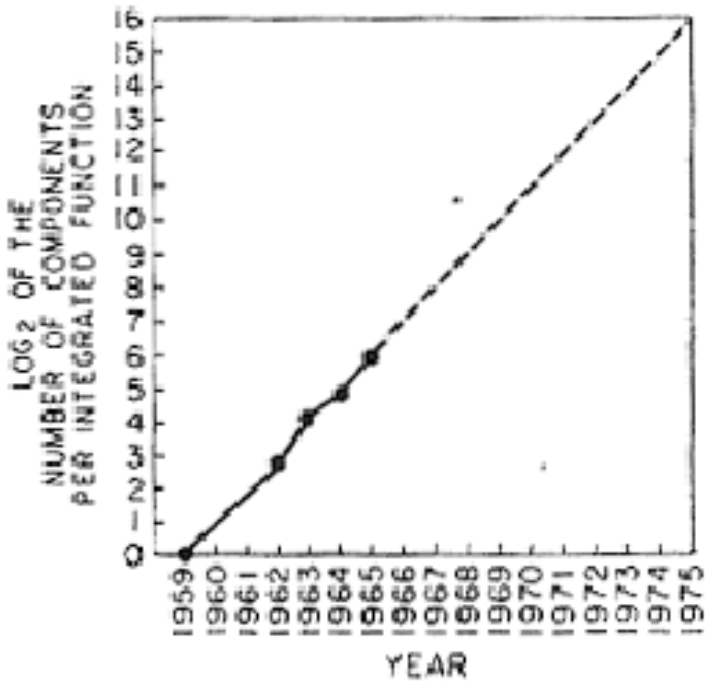
Clock Cycle	Output bit Pattern		
	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

http://www.electronics-tutorials.ws/counter/count_1.html

Performance considerations (1/5)

- Speed and throughput (MOPS/sec)
- Energy and power consumption (Joules, Watts)
- Area (cm/design, μm/gate)
- Reliability and accuracy

• Moore's Law (1965): component density doubles every year



Performance considerations (2/5)

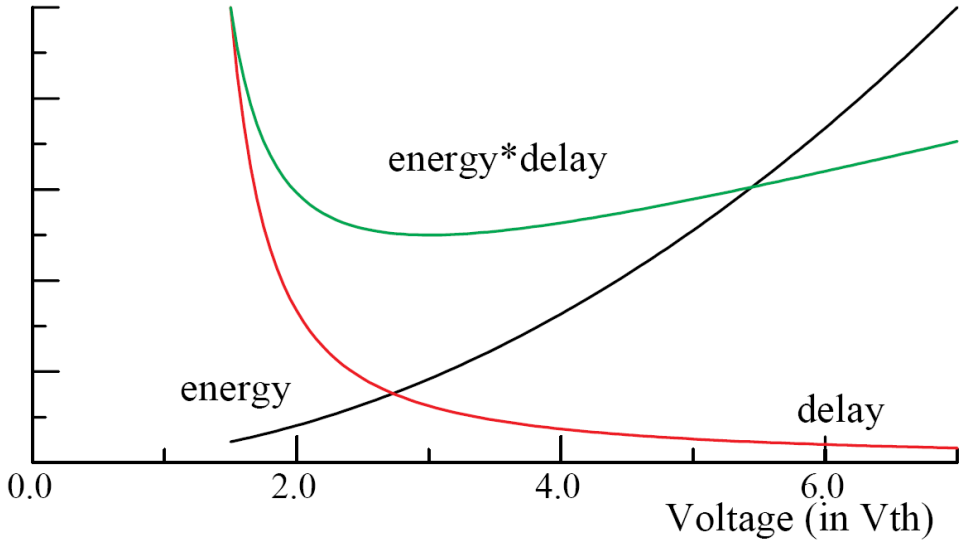
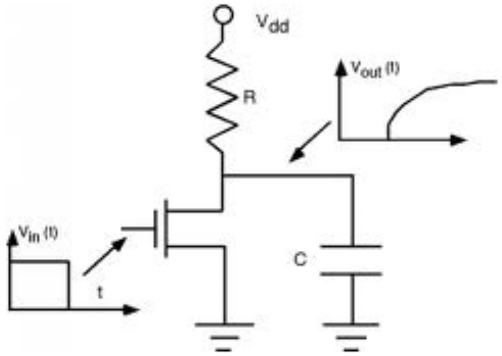


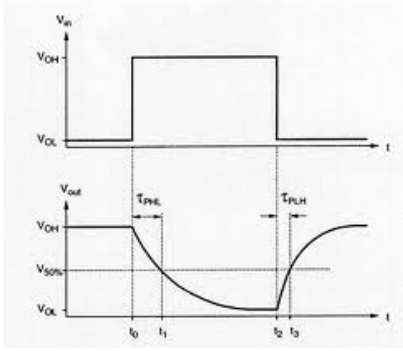
Figure 1. Energy and Delay vs. Voltage

- Energy*delay: measure for comparing efficiency of circuits.
- Power determined by capacitance (C) and logic (supply) voltage
- Delay determined by RC time constants

Ref: Horowitz, M. and Indermaur, T. and Gonzalez, R., IEEE Symp on Low Power Electronics, 1994.

Power consumption and delay (3/5)





- Intrinsic delay of a transition: $T = \frac{kCV_{dd}}{(V_{dd}-V_{th})^2}$ (secs)
- Energy per transition: $E = C\,V_{dd}^2$ $(Joules)$
- Power dissipation if a signal is applied to input:

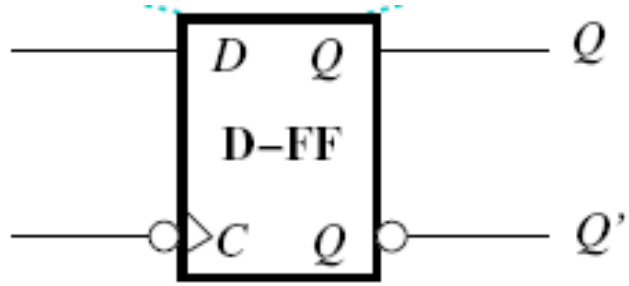
$$P = C\,V_{dd}^2\,f_{clk}\alpha \;\; (Watts)$$

- α is "activity ratio" = average number of input signal transitions per clock cycle

Ref: Chandrasekeran and Brodersen, Proc IEEE, 1995

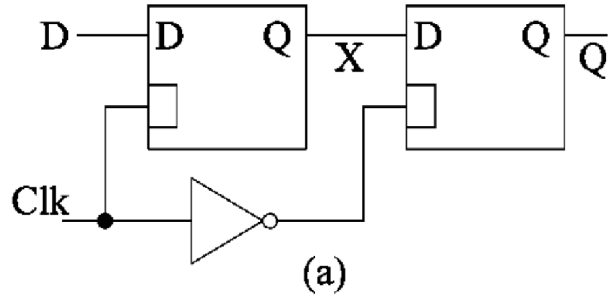
Performance considerations: D flip flop (4/5)

next state	current state	
Q^+	Q	D
0	0	0
0	1	0
1	0	1
1	1	1



- State Q^+ locks onto input logic level D on negative clock edge
- What is power dissipated by the D flip flop?

Performance considerations: D flip flop (4/5)



- Signal nodes X,D (Q) transition on clk leading (falling) edge:

$$P_k = \frac{1}{2} C_k \, V_{dd}^2 \, f_{clk} \alpha_k \;\; (Watts)$$

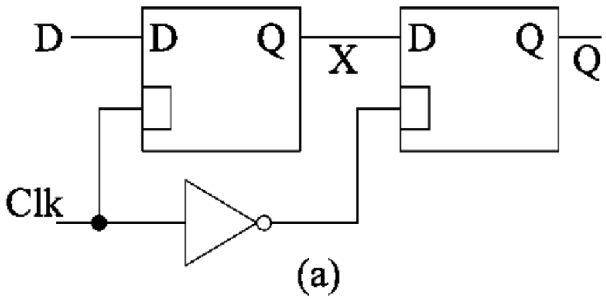
- Clock node (inverter) transitions on both edges:

$$P_{invtr} = C \, V_{dd}^2 \, f_{clk} \;\; (Watts)$$

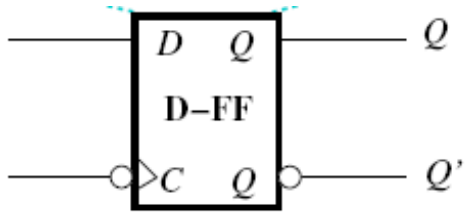
- Total power $P = P_{invtr} + \sum_{k=1}^{3} P_k$. Lumped model $\alpha_k = \alpha, \; C_k = C$

$$P_{DFF}(\alpha) = (1 + \alpha 3/2) C V_{dd}^2 f_{clk}$$

Performance considerations: D flip-flop (5/5)



equivalent to



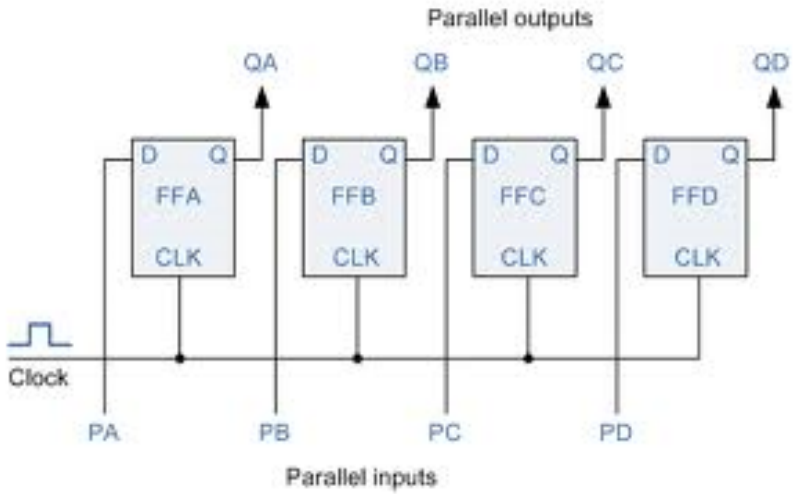
Illustration

- Representative values: $C_k = C = 0.1\text{pF}$, $V_{dd} = 1.8\text{V}$, $f_{clk} = 200\text{MHz}$, $\alpha_k = 1$
- Total power ($\alpha = 1$):

$$P_{DFF}(\alpha) = (0.1\times10^{-12})(1.8)^2(200\times10^6)(1+3/2) = 2\times10^{-4} = 200\mu W$$

Parallel-input parallel output (PIPO) register

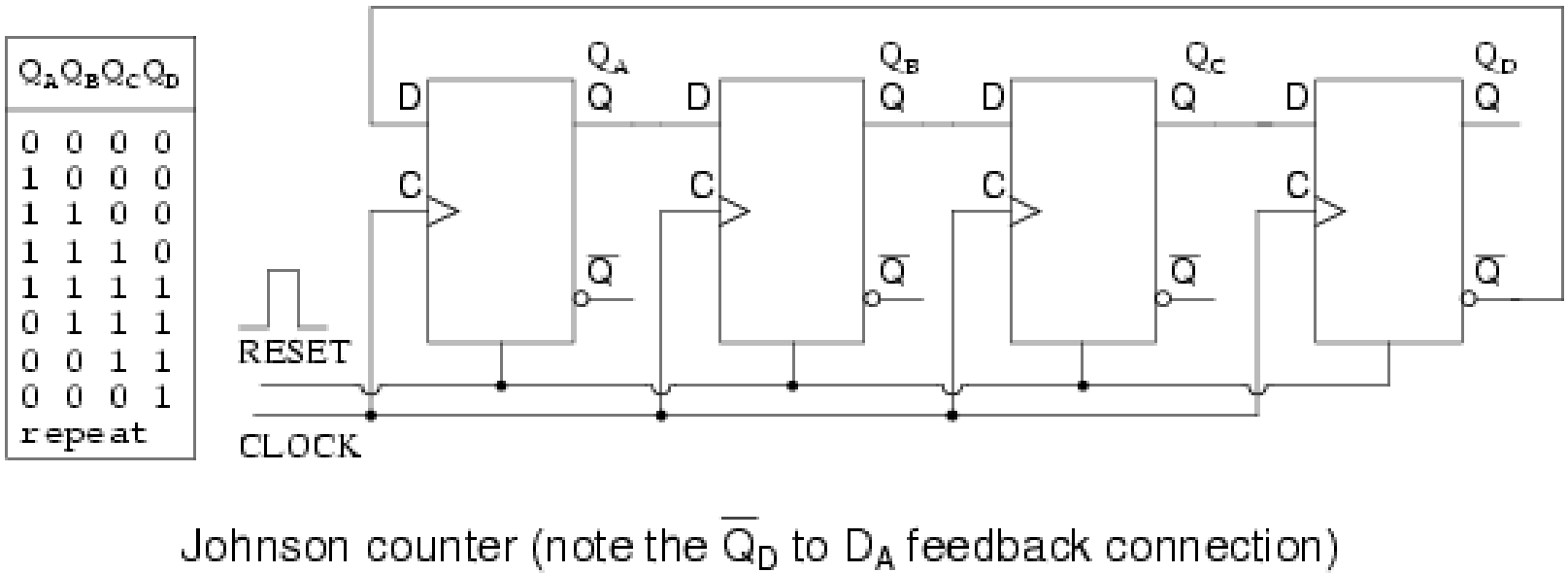
4-bit PIPO



- Assume
  - DFF's have identical output capacitance
  - Average transitions/clock-cyle is identically $\alpha = 1$ for all bit-streams PA, PB, PC, PD.
- Total power dissipation:

$$P_{PIPO} = 4P_{DFF}(\alpha), \quad \alpha = 1$$

4-bit switch-tail ring counter



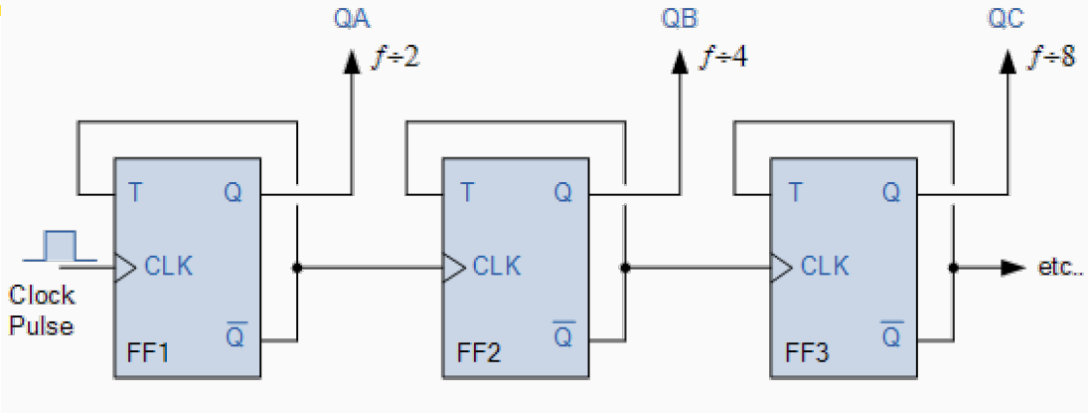
Johnson counter (note the $\overline{Q}_D$ to D_A feedback connection)

4-bit switch-tail ring counter (Johnson counter)

- Assume
  - DFF's have identical input/output capacitance
- Total power dissipation is:

$$P_{JC} = 4P_{DFF}(\alpha), \quad \alpha = 1/4$$

3 bit asynchronous counter with Toggle FFs



Clock Cycle	Output bit Pattern		
	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

- Assume
  - Toggle FF's dissipate same amount of power as DFF's
- Total power dissipation is:

$$P_{Counter} = P_{DFF}(1/4) + P_{DFF}(1/2) + P_{DFF}(1)$$

http://www.electronics-tutorials.ws/counter/count_1.html

Performance considerations

- Power reduction strategies
  - Lowering voltage V_{dd}
  - Reducing resolution (bit-width B)
  - Lowering clock rate f_{clk}
  - Scaling and shortening gate interconnects
  - Reducing idle activity level (uneccessary bit transitions).
- Speed improvement usually comes at cost of increased power
- FPGAs are reconfigurable logic arrays that can be used to explore performance tradeoffs for different designs.
- FPGA translates high level logic and arithmetic descriptions into silicon implementation.
- Design optimizers (Quartus, Xilinx) can be configured to minimize power or runtime, subject to clock, path, or port constraints.

Summary of what we covered today

- Gate-level arithmetic
- Power dissipation and delay
- Next: Altera DE2 and Verilog programming refresher