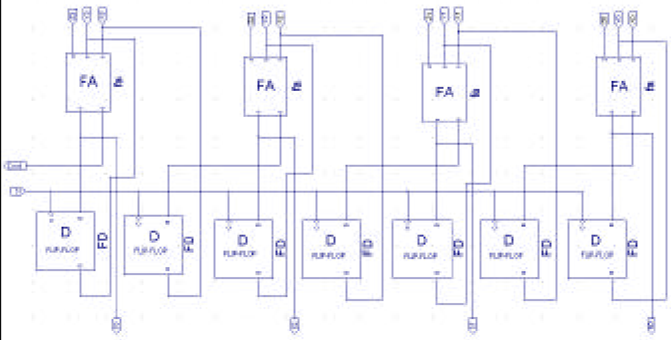
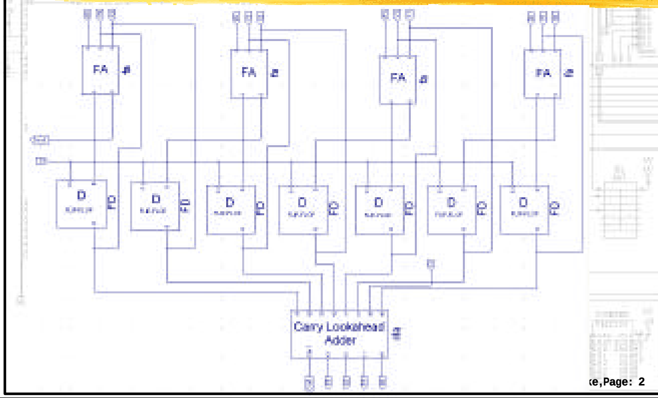
4-bit Carry Save Adder

$$S = \sum_{i=1}^{k} w_i$$



4-bit CSA-CLA

$$S = \sum_{i=1}^{k} w_i$$



Multiplication

- Multiplication of an m-bit number X by and n-bit number Y
- M> N
- To yield $Z=X \bullet Y$ with m+n bits
- This may be accomplished by forming n partial products P_i where:

$$P_i = Y_i \bullet X$$

- This requires m AND gates

4-Bit Partial Products

$Z = P_0 + 2 \bullet P_1 + 2^2 \bullet P_2 + 2^3 \bullet P_3$

				X_3	X_2	X_1	X_0	
			*	Y_3	Y_2	Y_1	Y_0	
0	0	0	0	X_3Y_0	X_2Y_0	X_1Y_0	X_0Y_0	$\rightarrow P_0$
0	0	0	X_3Y_1	X_2Y_1	X_1Y_1	X_0Y_1	0	$\rightarrow P_1$
0	0	X_3Y_2	X_2Y_2	X_1Y_2	X_0Y_2	0	0	$\rightarrow P_2$
0	X_3Y_3	X_2Y_3	X_1Y_3	X_0Y_3	0	0	0	$\rightarrow P_3$

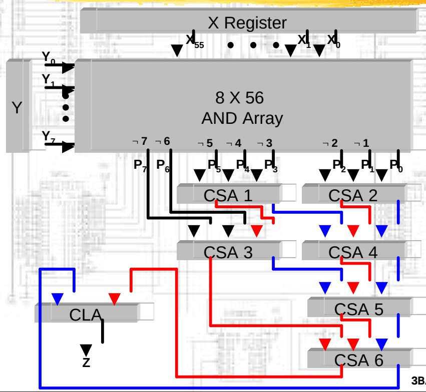
Partial Products

- We then weight each P_i with 2^i and sum them:

$$Z = P_0 + 2 \bullet P_1 + 2^2 \bullet P_2 + \ldots + 2^{n-1} \bullet P_{n-1}$$

- For example if m=56 and n=8 we use a Wallace tree of six CSAs to reduce 8 partial products to 2 which are complited by a single Carry Lookahead Adder.

Wallace Tree Multiplier



- The CSA range from 58-63 bits wide.
- The CLA is 64 bits wide

Shift and Add Multiplication

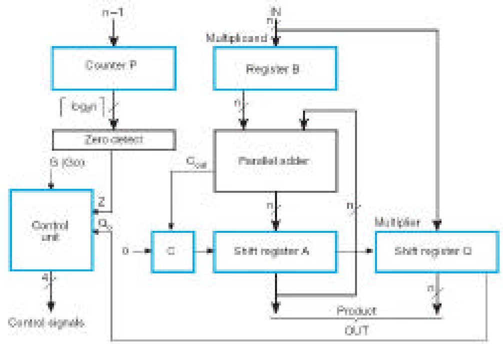
23	10111	Multiplicand	
19	10011	Multiplier	
	10111		← P_0
	10111		← P_1 ´ 2^1
	00000		← P_2 ´ 2^2
	00000		← P_3 ´ 2^3
	10111		¬ P_4 ´ 2^4
437	110110101	Product	

Hardware Multiplication

23	10111	Multiplicand
19	10011	Multiplier
	00000	Initial partial product
	10111	Add multiplicand, since multiplier bit is 1
	10111	Partial product after add and before shift
	010111	Partial product after shift
	10111	Add multiplicand, since multiplier bit is 1
	1000101	Partial product after add and before shift[a]
	1000101	Partial product after shift
	01000101	Partial product after shift
	001000101	Partial product after shift
	10111	Add multiplicand, since multiplier bit is 1
	110110101	Partial product after add and before shift
437	0110110101	Product after final shift

- a. Overflow temporarily occurred

Binary Multiplier Diagram



Booth's Algorithm

- This effectively halves the number of partial products in a multiplication.
- Principle:

$$Z = X * Y$$

$$X = 00\overset{k}{1}11\ldots\overset{j}{1}00$$

$$X = 2^{k+1} - 2^{j}$$

Booth's Algorithm

k-j+1

- We may replace the k-j+1 non-zero partial products with just two:

$$P_1 = 2^{k+1} * Y$$
$$P_2 = -2^{j} * Y$$
$$Z = P_1 + P_2$$

Booth's Algorithm

Worst case

- Of course in general a multiplier will have more than one such sequence of consecutive ONES.
- The most extrem case is that of alternating zero and one.
  - Resulting in n/2

Worst case = 01010101010101

Booth's Algorithm

Sequences of three bits

- Thus to deted and generate all the appropriate partial products we examine averlapping sequences of three bits:

X_{i+1}, X_i,	X_{i-1}
Bits examined	Previously Examined Bit