

Design of Arithmetic circuits

5/8/14

①

Adder

Fast addition of 2-n bit unsigned number

→ use 2-level combination net expressed in

SOP & POS of n-2lp variables

$C(n)$ — # gates

$f(n)$ — fan-in
multilevel combination

Practical Circuits ↙ sequential net

Half adder

$$S_i = x_i \oplus y_i$$

$$C_i = x_i y_i$$

Full adder

$$S_i = x_i \oplus y_i \oplus C_{i-1}$$

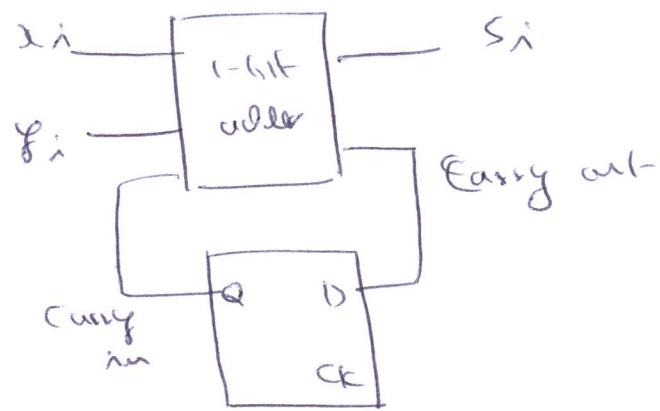
$$C_i = x_i y_i + x_i C_{i-1} + y_i C_{i-1}$$

x_i	y_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	10	1

Serial adder

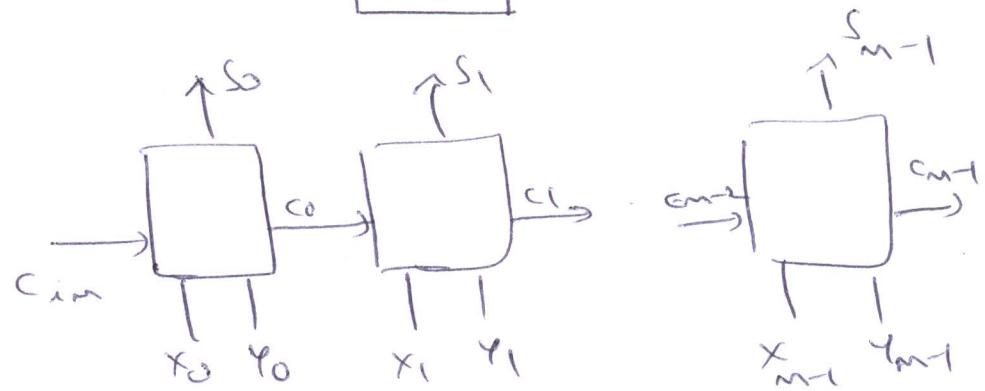
For n -bit addition

n - clock cycles



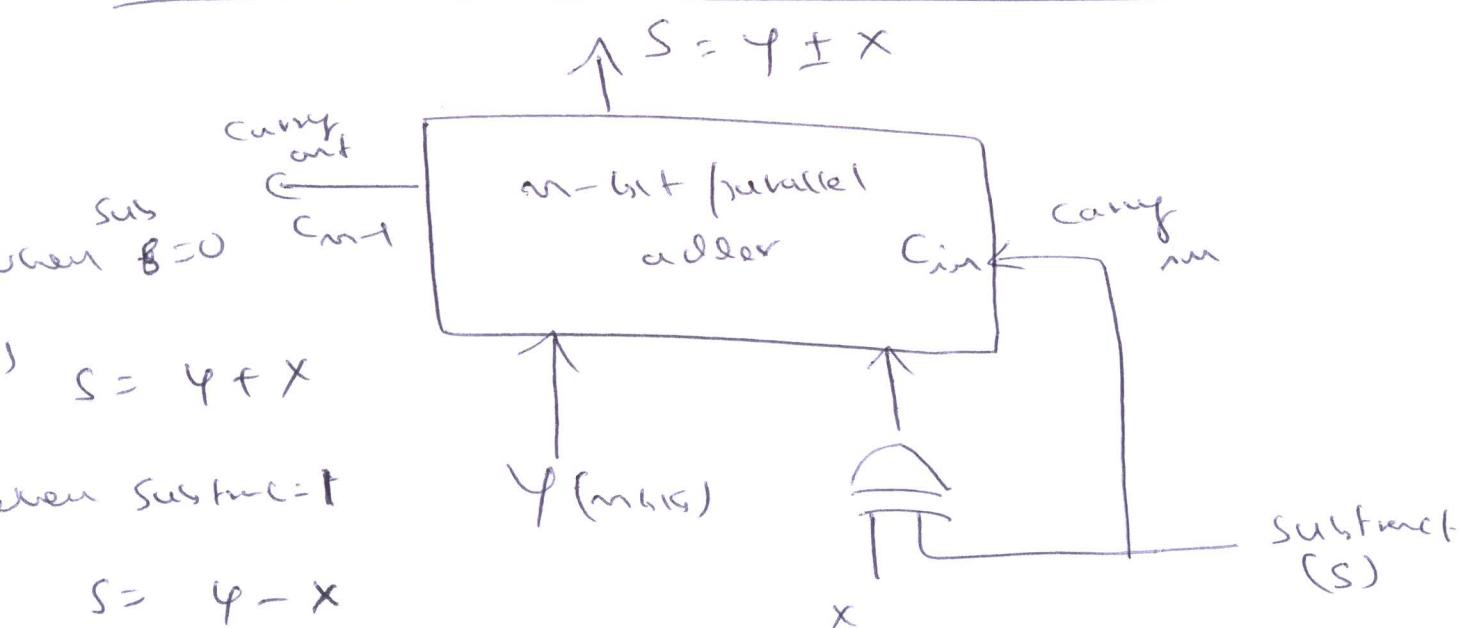
Parallel adder

n -bit ripple carry adder



$$\text{Time delay} = \frac{nd}{\text{delay in FA stage}}$$

n -bit 2's complement adder - subtractor



Binary subtractor

S_x = difference bit

b_{x-1} = borrow-in

b_x = borrow out

$$S_x = x_i \oplus y_i \oplus b_{x-1}$$

$$b_x = x_i \bar{y}_i + x_i b_{x-1} + \bar{y}_i b_{x-1}$$

overflow

unsigned number $\Rightarrow c_{m-1} = 1$ (2)

signed numbers $\begin{cases} \text{both +ve} \Rightarrow c_{m-2} = 1 \\ \text{both -ve} \Rightarrow c_{m-2} = 0 \end{cases}$

$$V = x_{m-1} y_{m-1} \bar{c}_{m-2} + \bar{x}_{m-1} \bar{y}_{m-1} c_{m-2}$$

$$c_{m-1} = x_{m-1} y_{m-1} + x_{m-1} c_{m-2} + y_{m-1} c_{m-2}$$

$$\Rightarrow V = c_{m-1} \oplus c_{m-2}$$

High Speed Adder

Delay = Sum of logic gate delays along the
longest signal propagation path through N(n)
n-bit n/pipe adder

$c_{m-1} \rightarrow 2(m-1)$ gate delay

$s_{m-1} \rightarrow 2(m-1) + 1$ xor delay

$c_m \rightarrow 2m$ gate delay

overflow $\rightarrow 2m+2$ gate delay

Reducing delay 1) Fastest electronic technology

2) modify the logic gate structure
(Fast propagation of carry)

Carry - look-ahead - addition

$$S_i = x_i \oplus y_i \oplus c_{i-1}$$

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$= x_i y_i + c_i (x_i + y_i)$$

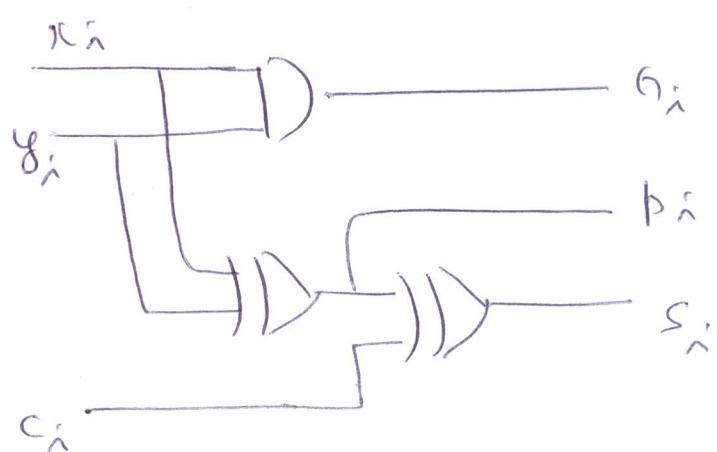
$$= g_i + p_i c_i$$

$$g_i = x_i y_i$$

$$p_i = x_i + y_i$$

\hookrightarrow generate f_m .

\hookrightarrow propagate f_m



$$c_{i+1} = g_i + p_i (g_{i-1} + c_{i-1} p_{i-1})$$

$$= g_i + p_i g_{i-1} + p_i p_{i-1} c_{i-1}$$

$$= g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \dots + p_i p_{i-1} \dots p_1 g_0$$

1 gate delay $\Rightarrow g_i \in P_i + p_i p_{i-1} \dots p_0 c_0$

2 gate delays $\Rightarrow c_{i+1} \text{ (AND-OR)}$
 1 gate delay $\Rightarrow s_i \text{ (X-OR gate)}$

$\Rightarrow 4\text{-gate delays}$

overflow

unsigned addition

7	7	0	1	2	3	4	5	6
6	6	7	0	1	2	3	4	5
5	5	6	7	0	1	2	3	4
4	4	5	6	7	0	1	2	3
3	3	4	5	6	7	0	1	2
2	2	3	4	5	6	7	0	1
1	1	2	3	4	5	6	7	0
0	0	1	2	3	4	5	6	7

$c_m = 1$

$s < u$

$s < b$

unsigned subtract

7	1	2	3	4	5	6	7	0
6	2	3	4	5	6	7	0	1
5	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
3	5	6	7	0	1	2	3	4
2	6	7	0	1	2	3	4	5
1	7	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6	7

$c_m = 0$

$a < s$

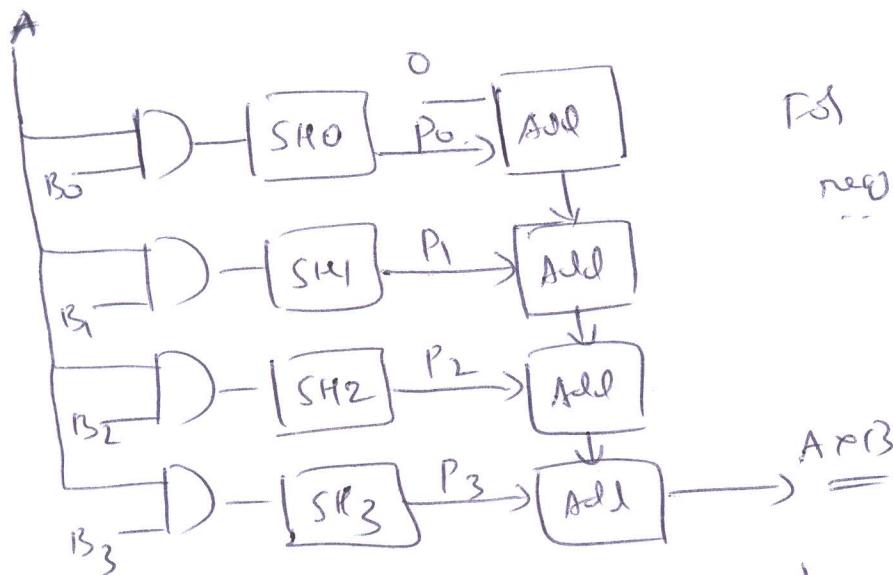
$a < b$

Multiplication

Simple Method

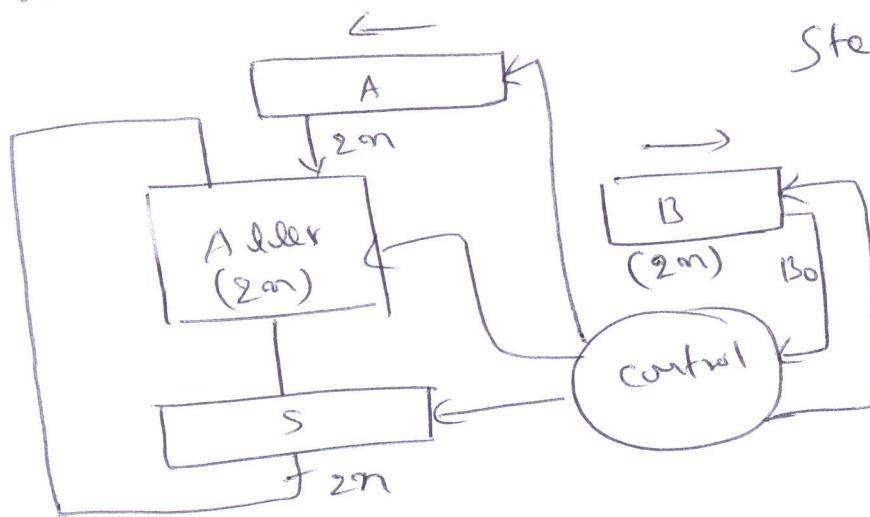
$$\begin{array}{l}
 P_0 \rightarrow 0 \ 0 \ 0 \ 0 \ 0 \\
 P_1 \rightarrow 0 \ 0 \ 0 \ 1 \ 1 \\
 P_2 \rightarrow 0 \ 0 \ 0 \ 1 \ 1 \\
 P_3 \rightarrow 0 \ 1 \ 1 \ 1 \ 1
 \end{array}$$

$$\begin{array}{r}
 A \quad 0 \ 1 \ 1 \\
 B \quad 1 \ 0 \ 1 \\
 \hline
 & 0 \ 1 \ 1 \\
 & 0 \ 0 \ 0 \times \\
 & 0 \ 1 \ 1 \times \times \\
 \hline
 & 0 \ 1 \ 1 \ 1 \ 1 \Rightarrow \sum_{i=0}^{n-1} A \cdot B_i \cdot 2^i
 \end{array}$$



For $N \times N$ multiplication
req N additions.

Shift and Multiply (sequential)



Step 1: $S = 0; S = 0$

do {

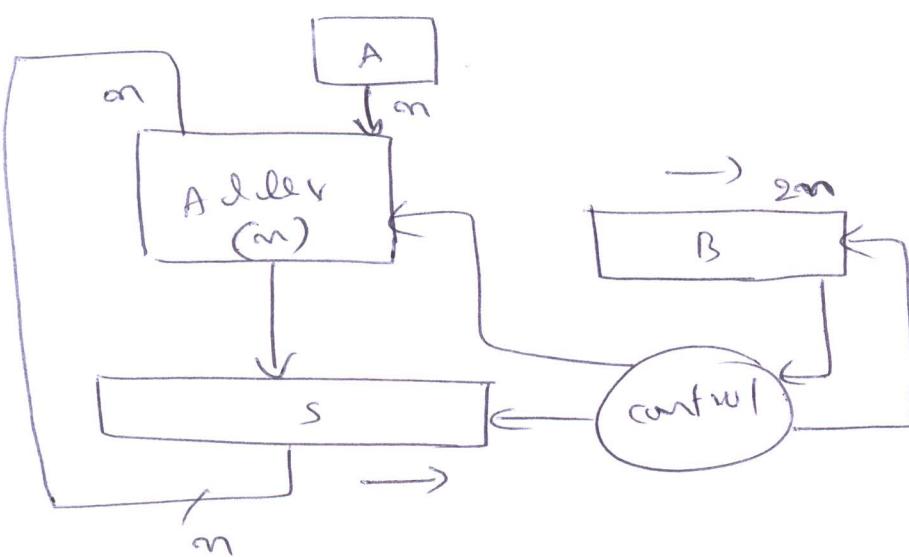
Step 2:

$A = 2 \times A$

$B = B/2; S + B$

} while ($S < n$)

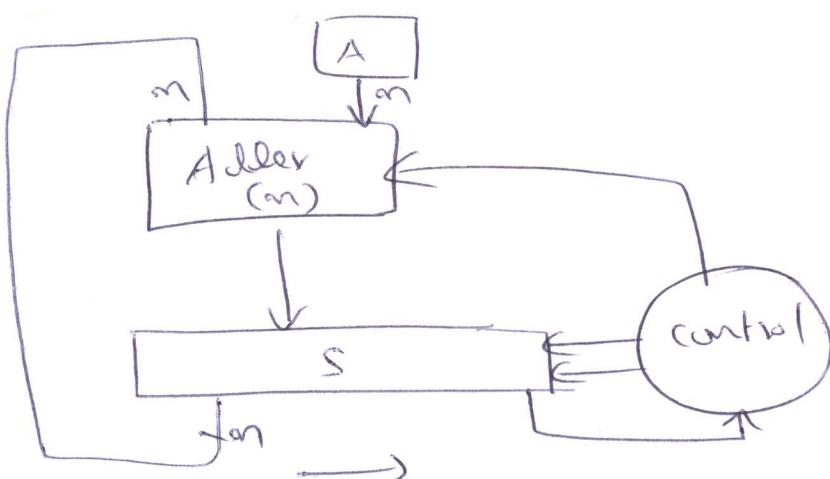
Sequential Shift add multiplier 2



Step 1: $i = 0, S = 0$
 do
 Step 2:
 if (B_0) $S_H \leftarrow A$
 $S = S_L$
 $B = B_{L2}$
 $i = i + 1$
 } while $i < m$

- ✓ Reducing size of A from $2m$ to m
- ✓ Reducing the size of adder from $2m$ to m

Sequential Shift add multiplier 3



Step 1: $i = 0; S = [01B]$
 do {
 Step 2:
 if (S_0) $S_H \leftarrow A$
 $S = S_L$; set
 } while ($i < m$)

- ✓ Avoid the use of separate reg for B

H, L Reg pair for holding 64 bits

Signed multiplication

Common expression to get both positive as well as negative values.

$$B = -B_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} B_i 2^i$$

$$\text{for } B > 0 \quad B = \sum_{i=0}^{m-1} B_i 2^i = -B_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} B_i 2^i \quad (\because B_{m-1} = 0)$$

$$\text{for } B < 0, \quad B = -|B|$$

$$|B| = 2^m - \sum_{i=0}^{m-1} B_i 2^i = 2^m - B_{m-1} 2^{m-1} - \sum_{i=0}^{m-2} B_i 2^i$$

$$= B_{m-1} 2^{m-1} - \sum_{i=0}^{m-2} B_i 2^i$$

$$B = -|B| = -B_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} B_i 2^i$$

~~$$= -B_{m-1} 2^{m-1} + B_{m-2} \left(2^{m-1} - 2^{m-2} \right) + \dots + B_0$$~~

$$= -B_{m-1} 2^{m-1} + B_{m-2} 2^{m-2} + \dots + B_1 2 + B_0$$

$$= -B_{m-1} 2^{m-1} + B_{m-2} 2^{m-2} - B_{m-2} 2^{m-2} + \dots + B_1 2^2 - B_1 2 + B_0 2 - B_0 2^0$$

$$= \sum_{i=0}^{m-1} (B_{i-1} - B_i) 2^i \quad \text{where } B_{-1} = 0$$

$$\therefore A \times B = \sum_{i=0}^{m-1} A \cdot (B_{i-1} - B_i) 2^i$$

Comparison bet. Signed and unsigned multiply

<u>unsigned</u>	<u>Signed</u>	<u>operator</u>
B_i	$B_i \ B_{i-1}$	
0	no addition	0 0
1	add A	0 1
		1 0
		1 1

Motivation of Booth's Algorithm

0 0 0 1 1 1 1 1 0 0 0 0
 [add] [sub]

additions will come down

Range of Multiplier

unsigned multiplier $0 \leq \text{result} \leq (2^m - 1)^2$
 $\leq 2^{2m} - 2 \cdot 2^m + 1$

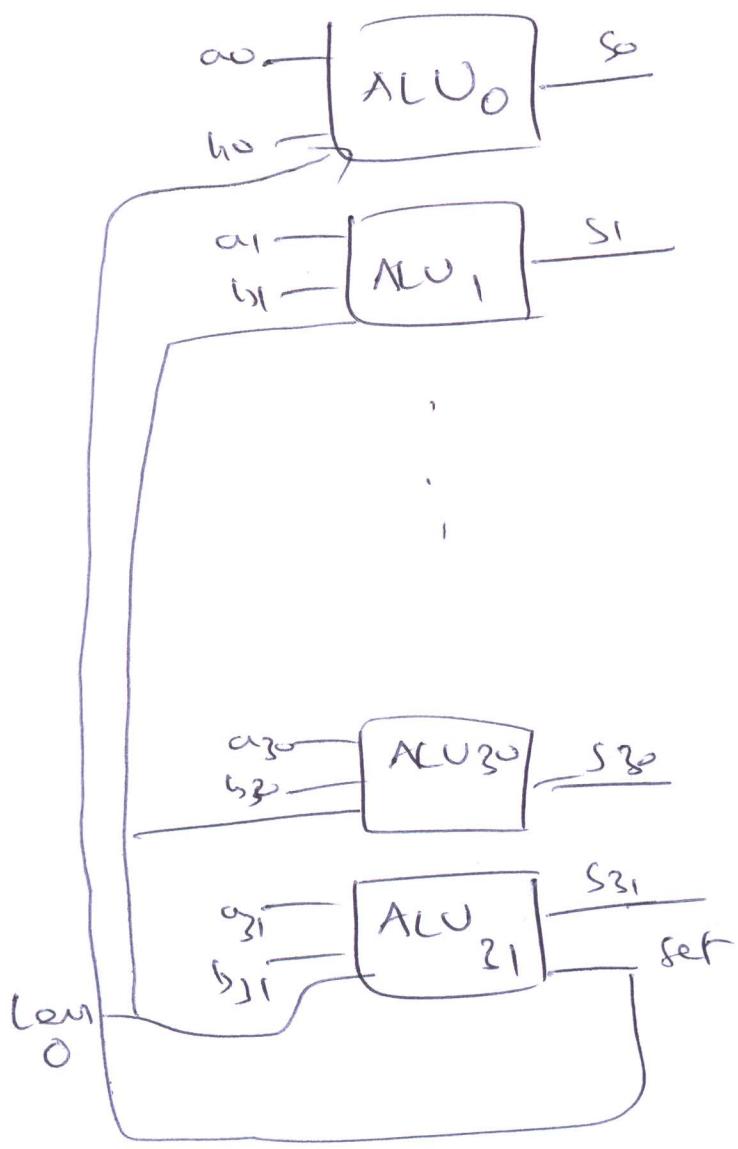
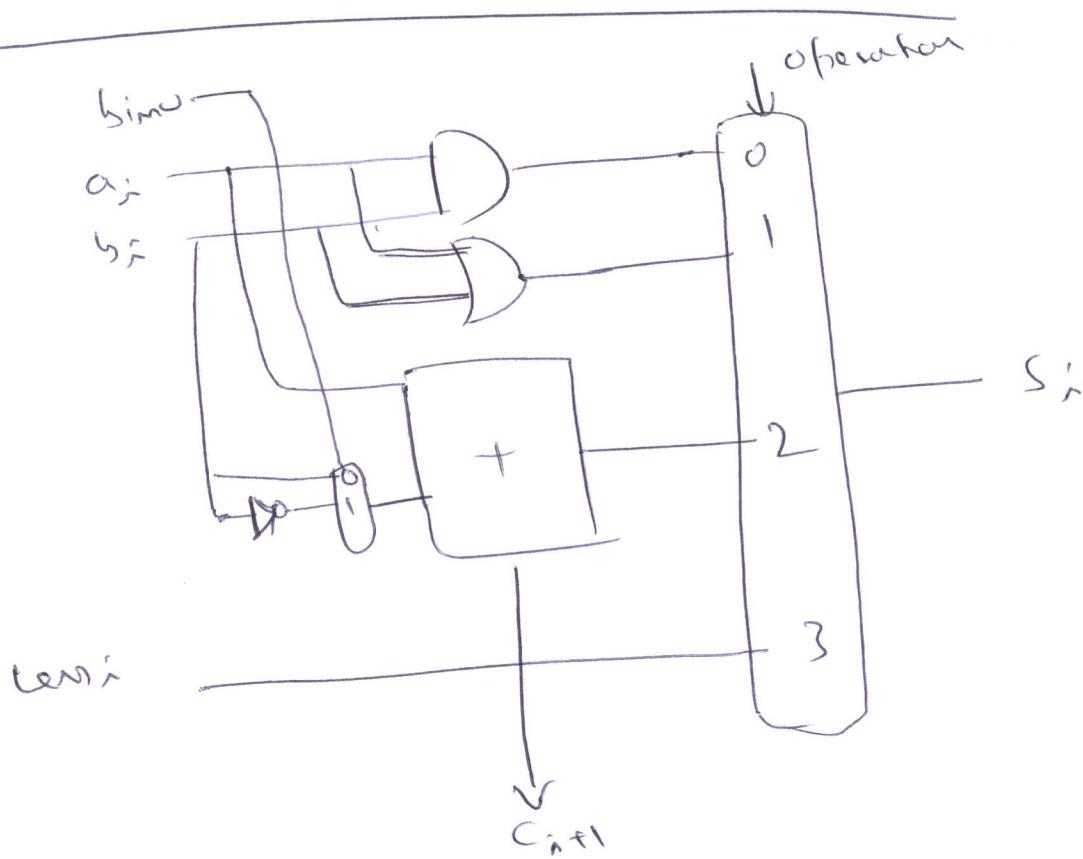
$$\begin{array}{r} 1 1 \dots 1 0 0 \dots 0 1 \\ \hline m-2 \quad m-1 \end{array}$$

Sign multiplier

$$-2^{m-1} - 2 \leq \text{result} \leq 2^{m-1}$$

No overflow

ALU for Add, Sub, AND, OR, SLT



Comparison



② Direct comparison

greater than

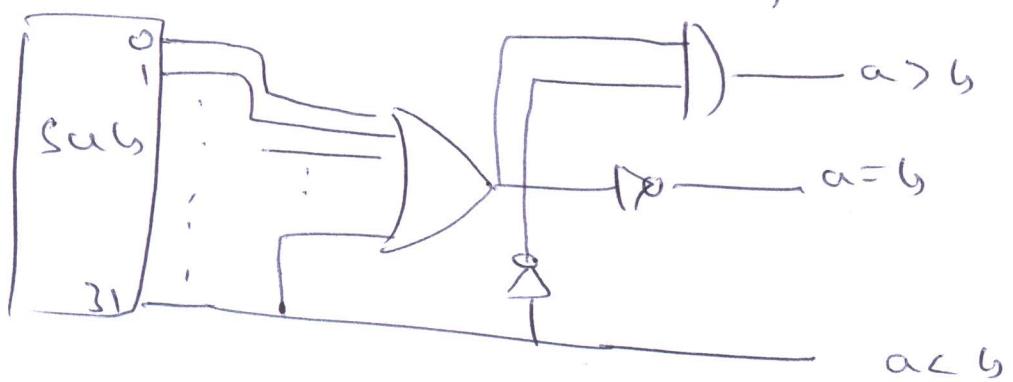
$$a_{0 \dots n-1} > b_{0 \dots n-1} = (a_{0 \dots i} > b_{0 \dots i}) (a_{i+1 \dots n-1} \overline{+} b_{i+1 \dots n-1}) + (a_{0 \dots i} \neq b_{0 \dots i}) a_i \overline{b}_{i+1 \dots n-1}$$

From MSB

$$a_{n-1 \dots 0} > b_{n-1 \dots 0} = (a_{n-1} > b_{n-1}) + (a_{n-1} = b_{n-1}) a_{0 \dots n-1} \overline{b}_{0 \dots n-1}$$

Equal

$$a = b \Rightarrow \prod_{i=0}^{31} (a_i \oplus b_i)$$



Division

$$A \div B = Q \times B + R / B$$

$$\underline{A = Q \times B + R}$$

$$0100) 00001101 (00011$$

$$\underline{01000000}$$

$$00001101$$

$$\underline{0100 \dots}$$

$$00001101$$

$$\underline{0100 \dots}$$

$$00001101$$

$$\underline{0100 \dots}$$

$$0101$$

$$\underline{0100}$$

(U)

$$2^4 \times B$$

Step 1: $i=0, R=A, Q=0$
 $D=B$

do {

Step 2:

if ($0 \times 2^{n-i-1} \leq R$) then

$$R = R - D \times 2^{n-i-1}$$

$$Q_{n-i-1} = 1$$

else

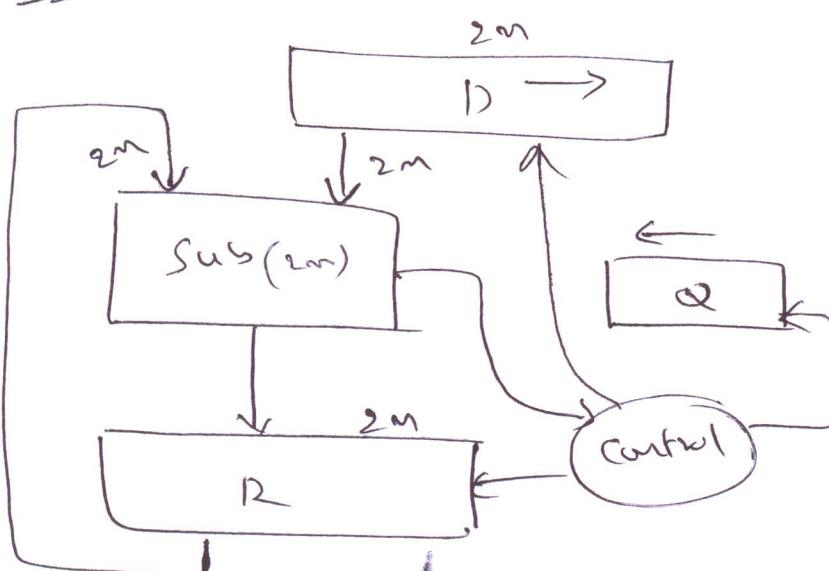
$$Q_{n-i-1} = 0$$

} while $i < n$

$$A = Q \times D + R \quad \text{and} \quad 0 \leq R \leq D \times 2^{n-i}$$

Loop invariant

Design



Step 1: $i=0, R=A, Q=0, D=B \times 2^{n-i}$

do {

Step 2:

if ($D \leq R$)

$$R = R - D; Q = 2Q + 1$$

else

$$Q = 2Q$$

$$D = D/2; \text{ if } P$$

} while ($i < n$)

Reducing Subtractor Size

$$A = Q \times B + R$$

Step 1: $i=0; R=2 \times A; Q=0; D=B$

do {

 Step 2:

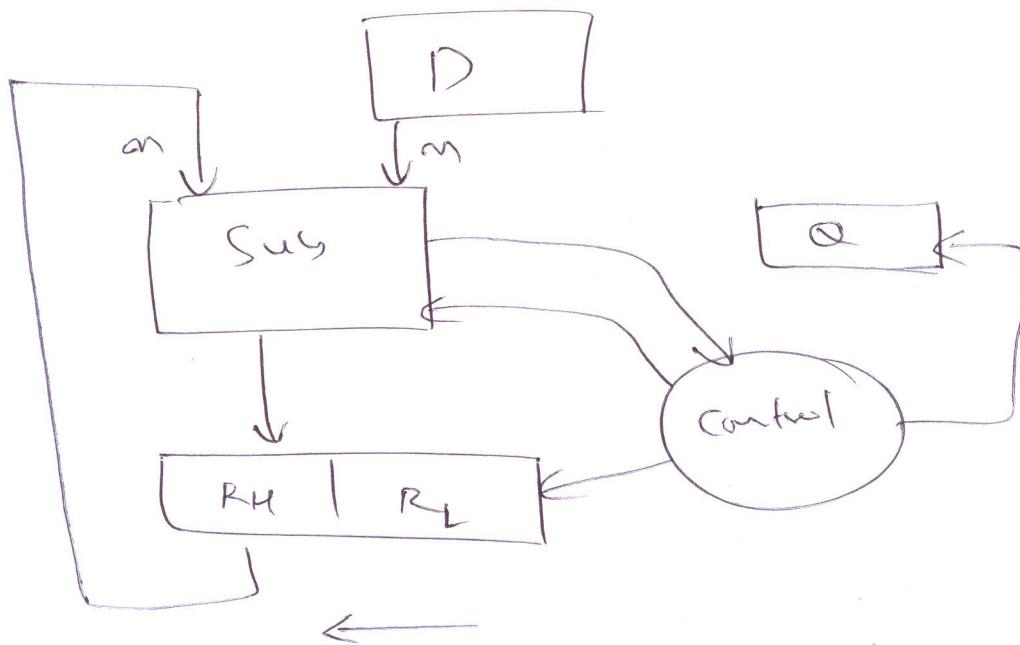
 if ($D \leq R_H$) $R_H = R_H - D; Q = 2Q + 1$

 else

$$Q = 2Q$$

 } $R = 2R; i++$
 } while ($i < m$)

Design-2



Reducing Registers

$$A = Q \times B + R$$

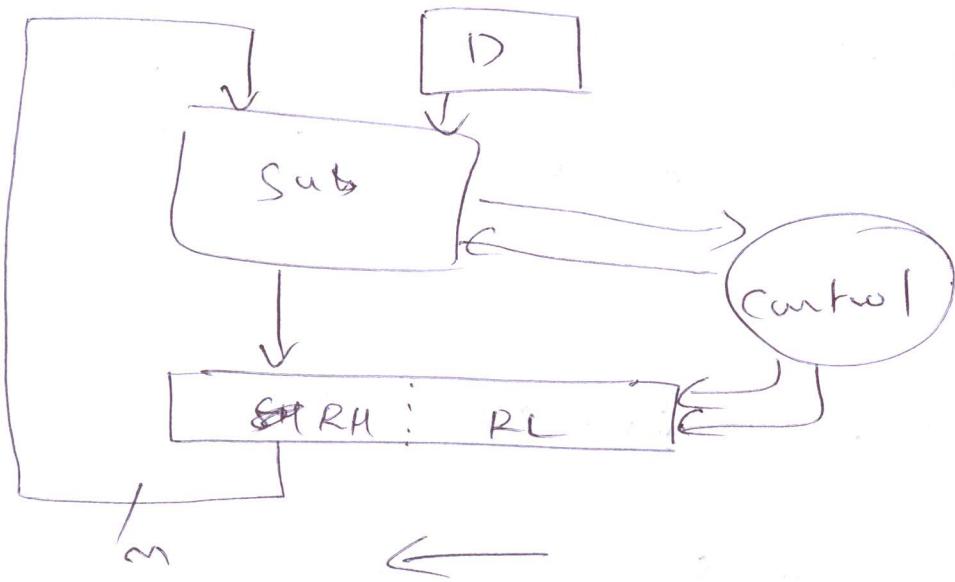
Step 1: $i=0; R=2 \times A; D=B$

do Step 2: if ($D \leq R_H$) $R_H = R_H - D; R=2R+1$

else

$$R = 2R$$

} $i++$
 while ($i < m$)



Design-3

Restoring Division

Non-restoring algorithm

$\begin{cases} \text{if } (D \leq R) \quad R = R - D; \quad D = D/2; \quad Q = 2Q + 1 \\ \text{Else } \quad D = D/2 \quad Q = 2Q \\ \quad \quad \quad \text{if } \end{cases}$

restored algorithm

$$R = R - D$$

$$\text{if } (R < 0) \quad R = R + D; \quad Q = 2Q$$

$$\text{Else } \quad Q = 2Q + 1$$

$$D = D/2; \quad \text{if } \quad \quad \quad$$

Delayed Restoration

$$R = R + D \quad \& \quad R = R - D/2 \quad \Rightarrow \quad R \leftarrow R + D/2$$

Improved restoring division

Step 1: $i \leq 0$; $R = A$; $Q = 0$; $D = B \times 2^{m-1}$

do {

Step 2A:

if ($R < 0$) $R = R + D$ else $R = R - D$

Step 2B:

if ($R < 0$) $Q = 2Q$ else $Q = 2Q + 1$

} $D = D_2$; i++

} while ($i < m$)

if ($R < 0$) $R = R + B$

Further improvement

Step 1 $i \leq 0$; $R = A$; $Q = 0$; $D = B \times 2^{m-1}$

do {

Step 2 if ($R < 0$)

$R = R + D$; $Q = 2 \times Q - 1$

else

$R = R - D$; $Q = 2 \times Q + 1$

$D = D_2$; i++

} while ($i < m$)

if ($R < 0$) $R = R + B$; $Q = Q - 1$

Extending to Signed division

Step 1: $i \leq 0 \quad R \leftarrow A, Q \leftarrow 0; D = B \times 2^{m-1}$

do {

Step 2

if ($R_{m-1} \neq D_{m-1}$)

$R = R + 0; Q = 2^Q - 1$

ELSE

$R = R - 0; Q = 2^Q + 1$

$D = DR_2; i \leftarrow i + 1$

} while ($i < m$)

if ($R_{m-1} \neq D_{m-1}$) $R = R + B; Q = Q - 1$

Signed in Signed division

Dividend	Divisor	Quotient	Remainder
+	+	+	+
+	-	-	+
-	+	-	-
-	-	+	-

Carry-lookahead adder

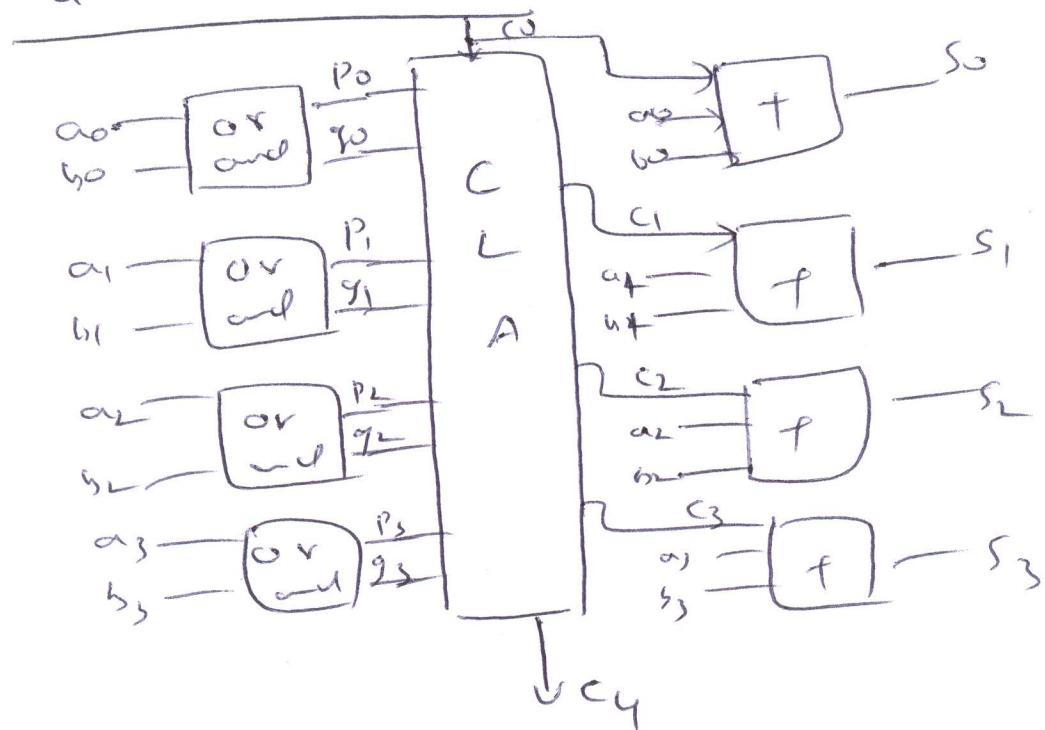
$$C_1 = P_0 C_0 + g_0$$

$$C_2 = P_1 C_1 + g_1 = P_1 P_0 C_0 + P_1 g_0 + g_1$$

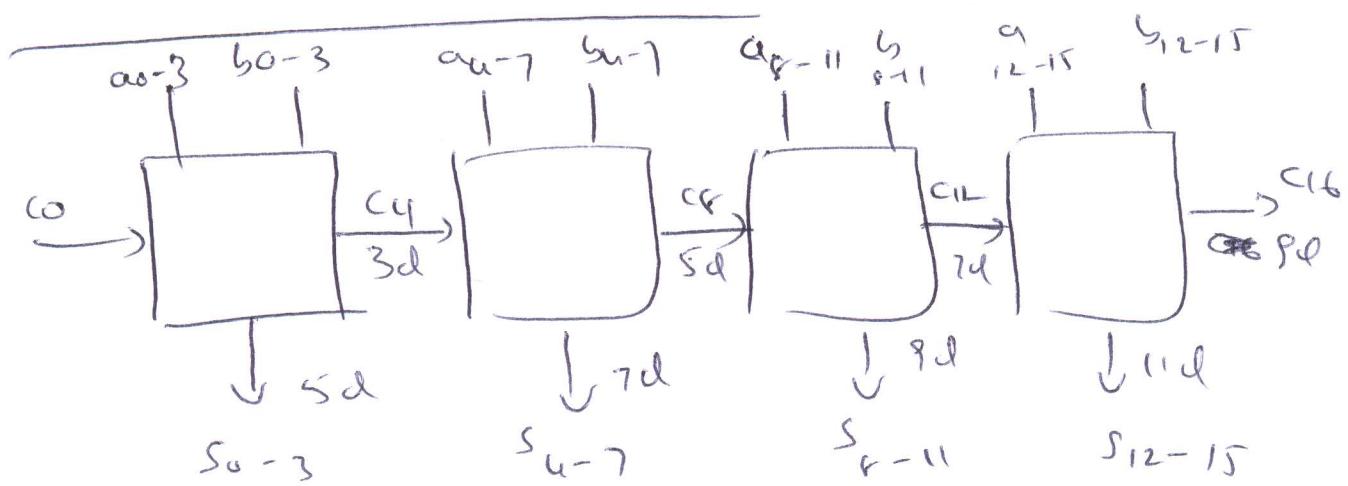
$$C_3 = P_2 C_2 + g_2 = P_2 P_1 P_0 C_0 + P_2 P_1 g_0 + P_2 g_1 + g_2$$

$$C_4 = P_3 C_3 + g_3 = P_3 P_2 P_1 P_0 C_0 + P_3 P_2 P_1 g_0 + P_3 P_2 g_1 + P_3 g_2 + g_3$$

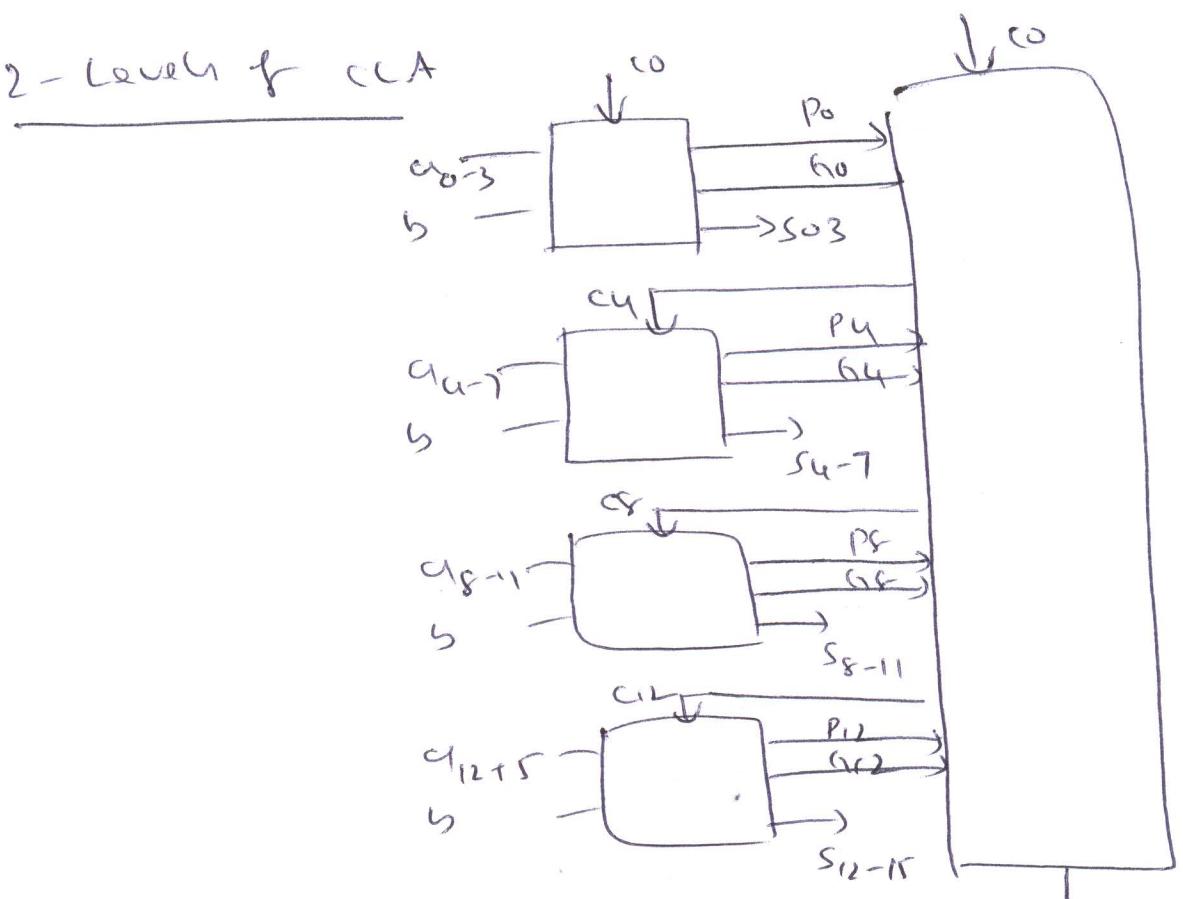
4-bit CLA adder



16-bit addition with 4-bit CLA



2-Level f CCA



$$P_0 = P_3 P_2 P_1 P_0$$

$$h_0 = P_3 P_2 P_1 g_0 + P_3 P_2 g_1 + P_3 g_2 + g_3$$

$$c_0 = P_0 c_0 + h_0$$

$$c_8 = P_4 P_0 c_0 + P_4 h_0 + h_4$$

$$c_{12} = P_8 P_4 P_0 c_0 + P_8 P_4 h_0 + P_8 h_4 + h_{12}$$

$$c_{16} = P_{12} P_8 P_4 P_0 c_0 + P_{12} P_8 P_4 h_0 + P_{12} P_8 h_4 + P_{12} h_{12}$$

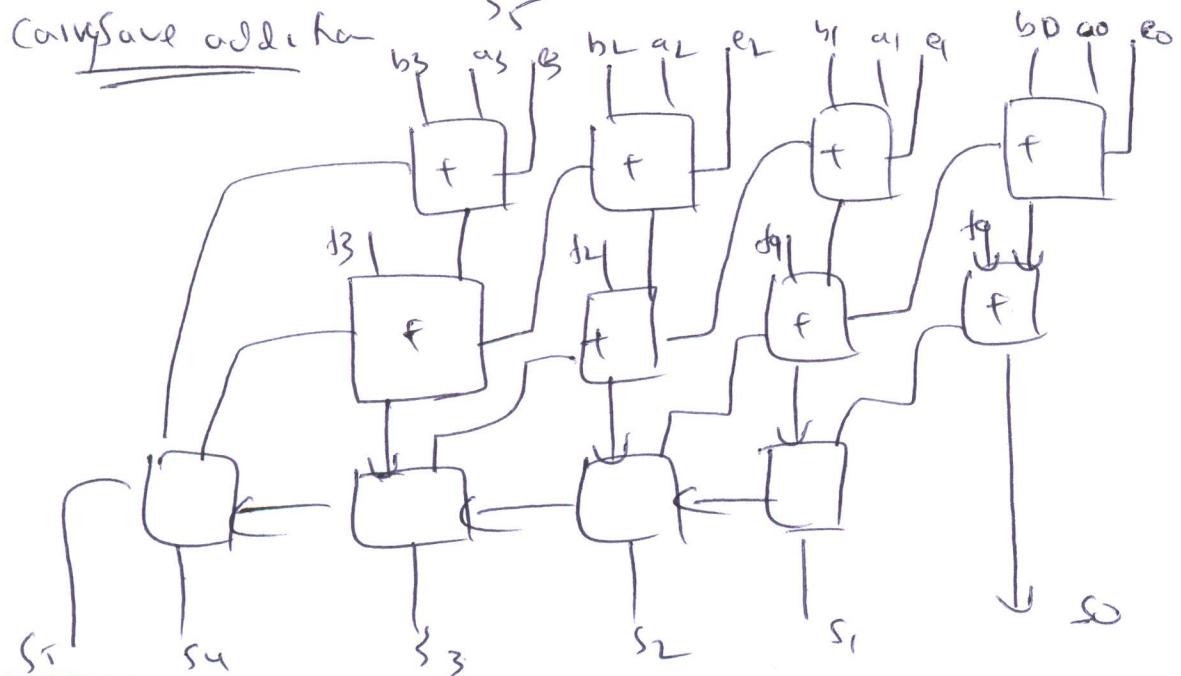
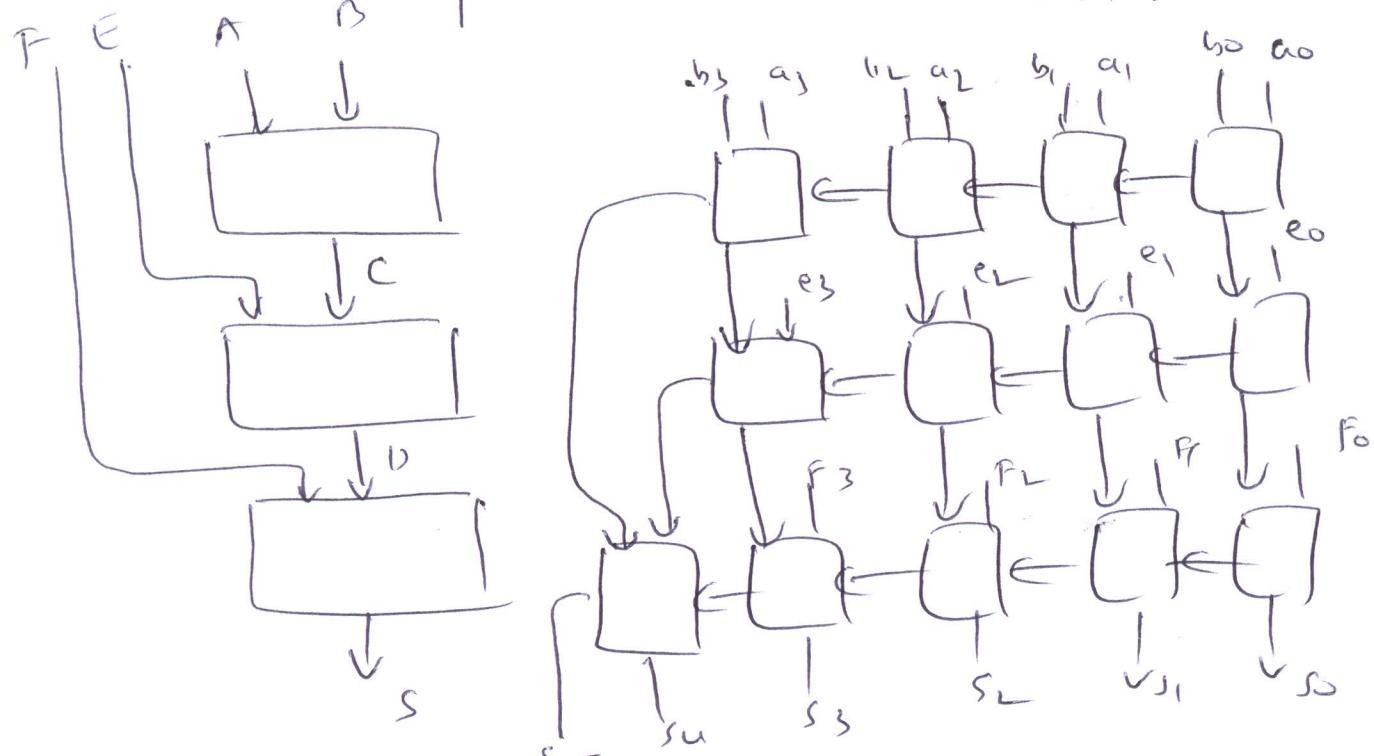
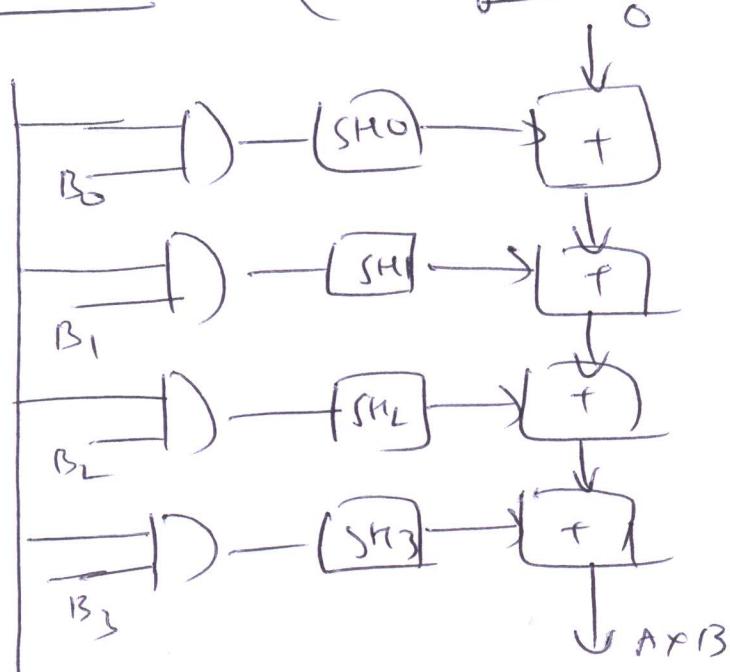
Array Multiplier (Carry save adder)

$$A \times B = i$$

and

$$\sum A \cdot B_i \cdot 2^i$$

$i \leq 0$



Floating Point Numbers 8

12/8/14

System Design

Data types: Integer, rational, real & complex

Scientific Application → Astronomy (distance bet Sun & Pluto) 5.9×10^{12} m
weight of an electron 9.1×10^{-31} kg

Representing Fractions

Integer form (for rational numbers)

$$\boxed{5} \quad \boxed{8} \Rightarrow 5/8$$

Strings with explicit decimal point

$$\boxed{1} \boxed{2} \boxed{4} \boxed{5} \boxed{.} \boxed{9} \Rightarrow -245.9$$

fixed point rep (significant fit in a fixed field)

0.00110, 01001

Floating point

Fracton × base^{power}

Precision & accuracy → mantissa
Base & power → range

$$0.6 = 0.100110011001\ldots$$

$$\begin{aligned} 0.6 \times 2 &= 1 + 0.2 \\ 0.2 \times 2 &= 0 + 0.4 \\ 0.4 \times 2 &= 0 + 0.8 \\ 0.8 \times 2 &= 1 + 0.6 \\ 0.6 \times 2 &= 1 + 0.2 \\ \vdots & \vdots \end{aligned}$$

FP Number with base 2

$$(-1)^S \times F \times 2^E$$

S = Sign

F = Fracton / mantissa / significant

E = Exponent (\pm integer)

Fixed point

IEEE - 754

Single precision

1	8	23
S	E	F

Double precision

1	11	52
S	E	F

Rep of F in IEEE 754

(Representation of mantissa)

11	23
----	----

111	52
-----	----

$$1 \leq F \leq 2 - 2^{-23} \approx 2$$

$$1 \leq F \leq 2 - 2^{-52} \approx 2$$

Rep of Exponent

< Single precision : bias 127
(8)

Double precision : bias 1023
(11)

$$0 - 255 \Rightarrow 1 - 254 \Rightarrow \cancel{126}$$

$$\underline{1-127 \text{ to } 254-127}$$

$$\cancel{-126 \text{ to } 127}$$

Double precision

$$0 - 2047 \Rightarrow 1 - 2046 \Rightarrow \cancel{1-1023 \text{ to } 2046-1023}$$

$$\cancel{-1022 \text{ to } 1023}$$

Rep of zero

All 0's (S, E, F) are to us.

Testing & Comparing

zero, +ve, -ve is same as integers
magnitude comparison is same as integers

overflows underflows

$$\text{Largest +ve Number (SP)} = \pm (2 - 2^{-23}) \times 2^{127} \approx \pm 2 \times 10^{38}$$

$$\text{" (DP)} = \pm (2 - 2^{-52}) \times 2^{1023} \approx \pm 2 \times 10^{308}$$

$$\text{smallest +ve number (SP)} = \pm 1 \times 2^{-126} \approx \pm 2 \times 10^{-38}$$

$$\text{" (DP)} = \pm 1 \times 10^{-1022} \approx \pm 2 \times 10^{-308}$$

$$\text{Precision} \quad SP \rightarrow \frac{23}{\log_2 10} \approx 8 \quad DP \rightarrow \frac{52}{\log_2 10} \approx 17$$

Floating Point operations

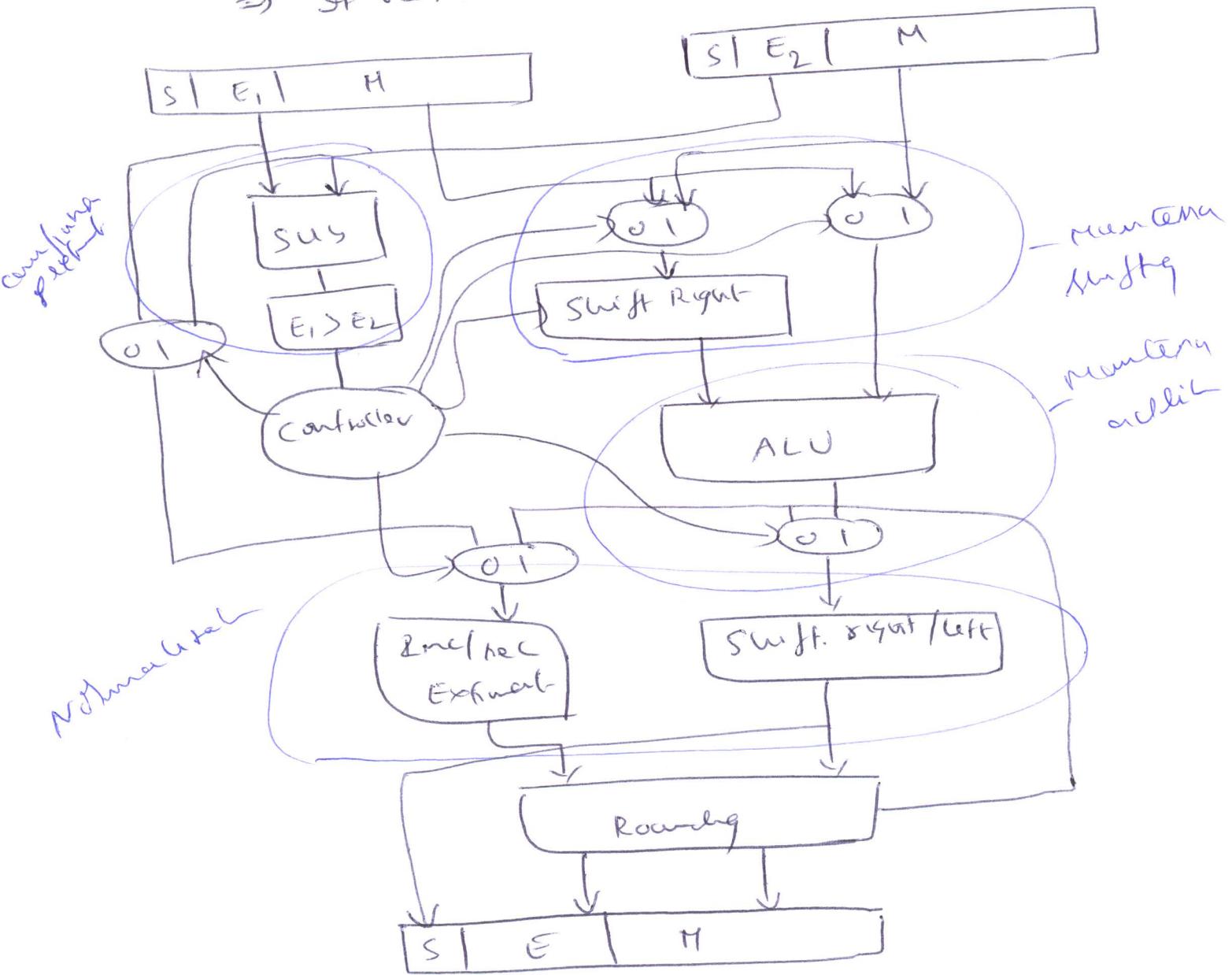
$$\text{Add / Subtract} \quad [(-1)^{S_1} \times F_1 \times 2^{E_1}] \pm [(-1)^{S_2} \times F_2 \times 2^{E_2}]$$

if $E_1 > E_2$

$$[(-1)^{S_1} \times F_1 \times 2^{E_1}] \pm [(-1)^{S_2} \times F_2' \times 2^{E_1}] ; \quad F_2' = \frac{F_2}{2^{E_1 - E_2}}$$

$$\Rightarrow (-1)^{S_1} \times (F_1 \pm F_2) 2^{E_1}$$

\Rightarrow if want to be normalized



$$\frac{\text{Multiply}}{\left[(-1)^{S_1} \times F_1 \times 2^{E_1} \right] \times \left[(-1)^{S_2} \times F_2 \times 2^{E_2} \right]}$$

$$\Rightarrow (E_1 \times (F_1 \times F_2) \times 2)^{E_1 + E_2}$$

$\therefore 1 \leq (F_1 \times F_L) \leq 4 \Rightarrow$ the result has to be uniform and

$$\text{Divide} \quad \left[(-1)^{S_1} \times F_1 \times 2^{E_1} \right] \div \left[(-1)^{S_2} \times F_2 \times 2^{E_2} \right]; \quad F_2 \neq 0$$

$$\therefore (-1)^{S_1 + S_2} \times (E_1 - E_2) 2^{E_1 - E_2}$$

$$\therefore 0.5 \leq (F_1 + F_2) < 2 \Rightarrow \text{the result has to be}\text{and must be}\text{real}$$

Accuracy & Rounding off

- Precision is lost when bits are shifted to the right of rightmost bit
 - 3 extra bits are used internally
g (guard) r (round) & s (sticky)

G&R and 24.5 after LSB

$s=1$ iff any bit to the right of R is non-zero

1.1101 - - - - - 11011001

Rounding using R, R&S if $R=1$ & $S=1$; add 1 to LSB

If $\theta = 0^\circ$ & $R = 0.31$; no angle

if $R=1$ & $R=0$ (look at 'S')

If $s=1 \Rightarrow$ add 1 to LSB

If $s=0 \Rightarrow$ you'll never "exit" (i.e., add 1 to LSB,)

SP	M	E	D P	O	destf	if LSO = 1
0	0	0	0	0	new	
Nz	any	0	Nz	any	devaluer	
0	0	0-2046	0	0	other	
2046	2047	2047	2047	2047	infinity	
2047	2047	2047	2047	2047	None	
numInv						