Lect 13

Adder
CS221: Digital Design

Ref Chapter 10 of the Book: *Introduction to Digital System*, Ercegovac M, Lang T and Moreno J H, Wiley India, 2013

Dr. A. Sahu
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

1

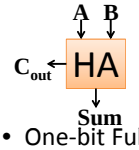
Outline

- Combinational Block
- Adder, Substractor, BCD Adder
- Efficient : Adder Design
  - RCA, CS_kA, CS_lA, CLA
- Binary Multiplier
  - Array, Sequential, Booth
- Floating Point

2

Adding Two One-bit Operands

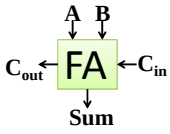
- One-bit Half Adder:



Sum = A ⊕ B
Cout = A.B

A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- One-bit Full Adder:



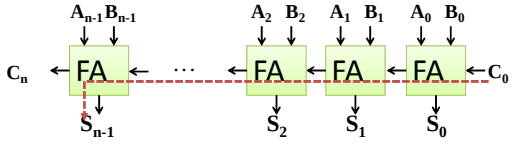
Sum = A ⊕ B ⊕ Cin
Cout = A.B + B.Cin + A.Cin

Cin	A	B	Sum	Cout
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	1	1

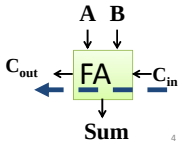
3

N-Bit Ripple-Carry Adder: Series of FA Cells

- To add two n-bit numbers

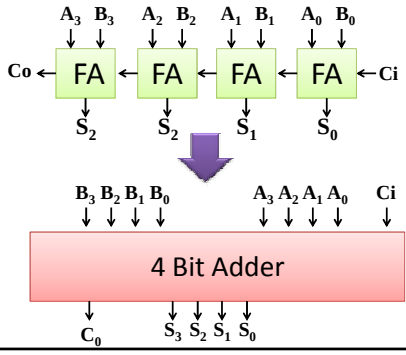


- **Adder delay = Tc * n**
- Tc = (C_{in} to C_{out} delay) of a FA
- **Adder Area: N*A_{FA}**



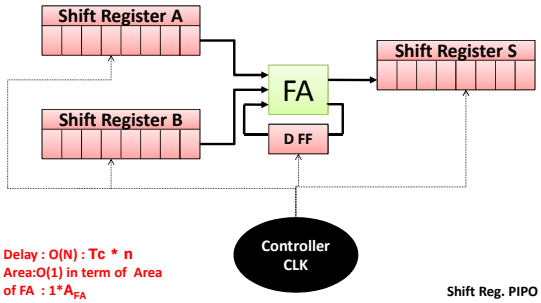
4

4 bit Binary Adder



5

4 bit Binary Adder: Serial

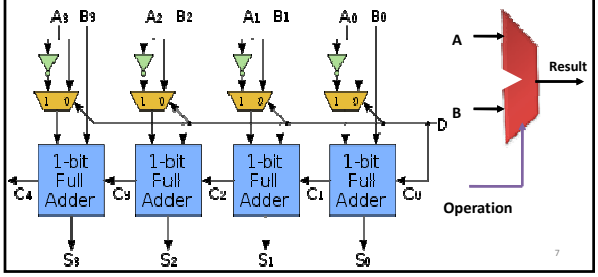


Delay : O(N) : Tc * n
Area:O(1) in term of Area of FA : 1*A_{FA}

Shift Reg. PIPO
Right Shift

Adder/Substractor

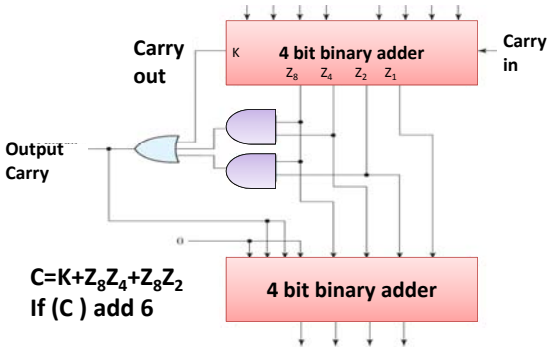
- C= A-B=A+(-B)=A+ (B^b+1), B^b is complement of B
- D is control bit: D=0/1 operation is add/sub



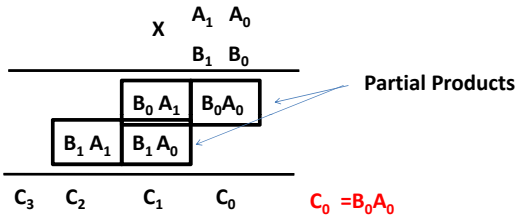
Decimal Adder

- Decimal numbers are represented with BCD code.
- When two BCD digits A and B are added
  - if A+B<10 result is a valid BCD digit
  - if A+B>9 result will not be valid BCD digit. It must be corrected by adding 6 to the result
- If A+B >9 add 6 to solve this issue

Decimal Adder



Binary Multiplier: 2 Bits



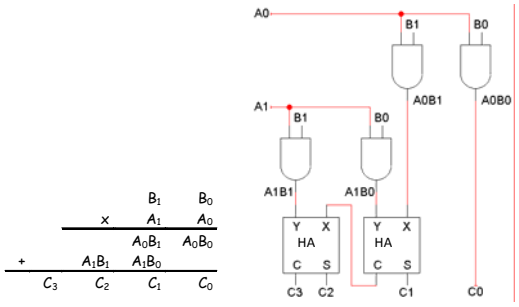
$C_0 = B_0A_0$

$C_1 = B_0A_1 + B_1A_0$

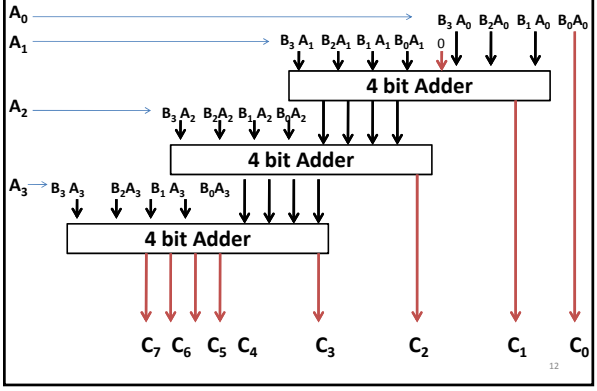
$C_2 = B_1A_1$ +Carry of C_1

C_3 = Carry of C_2

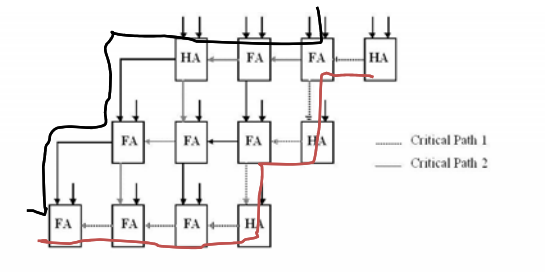
Binary Multiplier: 2 Bits



Binary Multiplier: 4 Bits

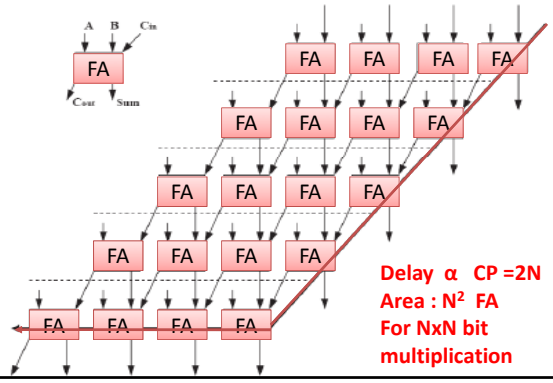


Area & Delay (Critical Path): Multiplier



Delay α CP =2N
Area : N^2 FA
For NxN bit multiplication

Area & Delay (Critical Path): Multiplier



Delay α CP =2N
Area : N^2 FA
For NxN bit multiplication

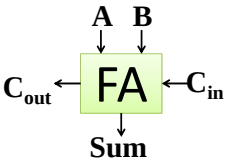
Efficient Adder Design

Adder Universal Use

- Adder : A = B + C
- Substractor: A = B + (-C), 2's complement
- Compare : C = A> B ? 1 : 0 , (A-B > 0) ? 1 : 0
  - Special case of compare with 0
- Multiply
- Divide
- Mod
- Floating point: Add/sub/mul…

Adding Two One-bit Operands

- One-bit Full adder



$Sum = A \oplus B \oplus Cin$
$Cout = A.B + B.Cin + A.Cin$

Cin	A	B	Sum	Cout
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	1	1

Addition of Two N-Bit numbers

$$x + y + c_{in} = 2^n c_{out} + s$$

The solution:

$$s = (x + y + c_{in}) \bmod 2^n$$

$$c_{out} = 1 \text{ if } (x + y + c_{in}) \geq 2^n \text{ else } 0$$

Example

- 011110 + 101101 = 1 (x 2^6)+ 001011

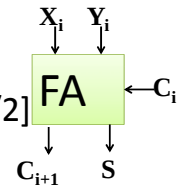
- X=30, Y=45
- 30 + 45 = 75= 2^6 x 1 + 11
- Solution
  - S= (30+45+0) % 2^6=11
  - Cout= 1 if (30+45+0 >= 2^6) else 0 = 1

Primitive module FA

$x_i + y_i + c_i = 2\ c_{i+1} + s_i$

with solution

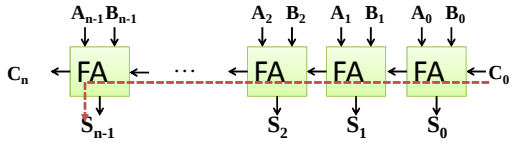
- $s_i = (x_i + y_i + c_i) \bmod 2$
- c_{i+1} =floor [$(x_i + y_i + c_i)/2$]



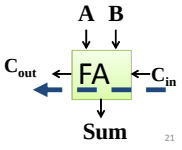
20

N-Bit Ripple-Carry Adder: Series of FA Cells

- To add two n-bit numbers



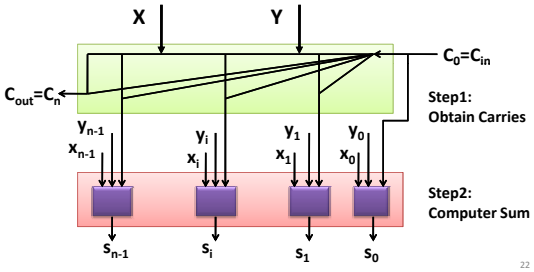
- Adder delay = Tc * n
- Tc = (C_{in} to C_{out} delay) of a FA



21

Adder Schemes

- Step1: Obtain carries
  - (Carry at i depends on j < i), Non-trivial to do fast
- Step2: Compute sum bits (local function)



22

Mathematically: C_i & S_i

- C_i =FuncC (x_{i-1}, ..,x_0, y_{i-1}, ..,y_0, c_{in})

- S_i =FuncS (x_i, y_i, c_i)
  = ($x_i + y_i + c_i$) mod 2

23

Ripple Carry Adder Analysis

i	9	8	7	6	5	4	3	2	1	0	
x_i	1	0	1	0	1	1	1	1	0	0	C_{in}=0
y_i	0	0	0	1	0	1	0	0	1	0	
	P	K	P	P	P	G	P	P	P	K	
C_{i+1}	0	0	1	1	1	1	0	0	0	0	

Case	X_i	Y_i	X_i+Y_i	C_{i+1}	Comment
Kill (K_i=1)	0	0	0	0	Kill/Stop C_{in}/C_{out}=0
Propagate P_i=1	0	1	1	C_i	Propagate C_{in} C_{out}=C_{in}
	1	0	1	C_i	
Generate	1	1	2	1	(G_i=1), Generate C_{out}, C_{out}=1

24

Propagate, Generate & Kill

- Case 1 (Kill): $k_i = x_i' y_i' = (x_i + y_i)'$
- Case 2 (Propagate): $p_i = x_i \text{ XOR } y_i$
- Case 3 (Generate): $g_i = x_i + y_i$

Then

$$c_{i+1} = g_i + p_i c_i = x_i y_i + (x_i \text{ XOR } y_i)\, c_i$$

Alternative (simpler) expression:

$$c_{i+1} = g_i + a_i c_i$$

Since $a_i = k_i'$, we call it "alive"

25

Reducing Adder Delay

- Ripple Carry Adder: $N(t_c)+\max(t_c,t_s)$
- Reducing Carry Delay t_c: Manchester Switch
- Changing linear factor to smaller
  - N/k or logN
  - Carry Look Ahead, Carry Skip, Carry Select, Conditional Sum Adder
- Including a competition signal: addition always may not be the worst case.
- Changing number representation: Carry Saved Adder

26

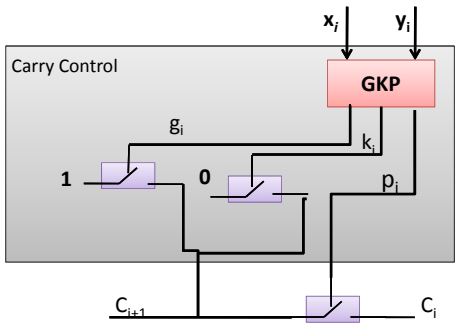
Switched Carry Ripple (Manchester) Adder

- Idea: Fast circuit to propagating carry chain

x_i+y_i	G_i	P_i	K_i	C_{i+1}
0	0	0	1	0
1	0	1	0	Ci
2	1	0	0	1

27

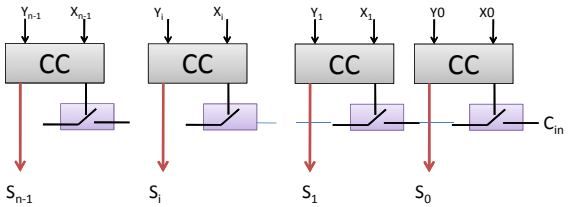
Chain Control Switch Module



28

Manchester Adder (MRCA)

- Delay: $t_{sw}+(n-1)*t_p$
- Here $t_p < t_c$, so total delay is less



29

Carry Skip Adder

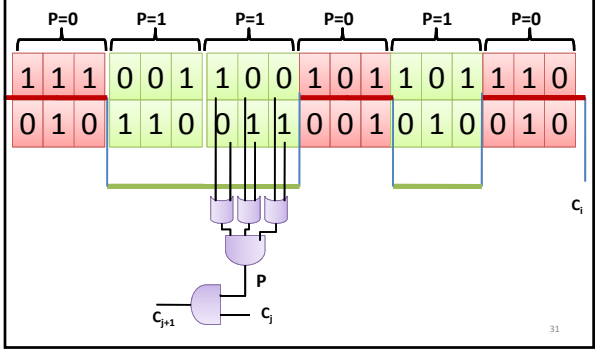
- Smaller modification to Ripple Carry Adder
- Reduce worst case delay by reducing the number of FA cell through carry has to Propagate
- Divide n bits in to (n/m) groups of m bits
- If sum of group is 2^m-1 then carry is propagated
- Carry is propagated when it propagated by all the bits of the groups

$$P_j = p_{j0}\, p_{j1} p_{j2} \ldots p_{jm-1}$$

- $C_{in,j+1} = C_{out,j} \cdot P'_j + C_{in,j} + P_j$

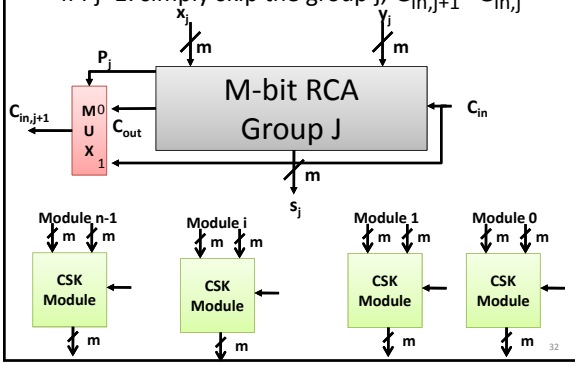
30

Carry Skip Example

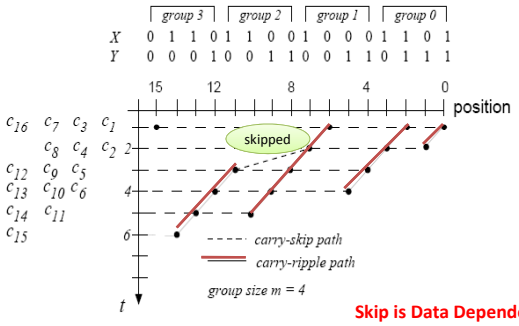


CSK Module

- If Pj=1: simply skip the group j, $C_{in,j+1}=C_{in,j}$



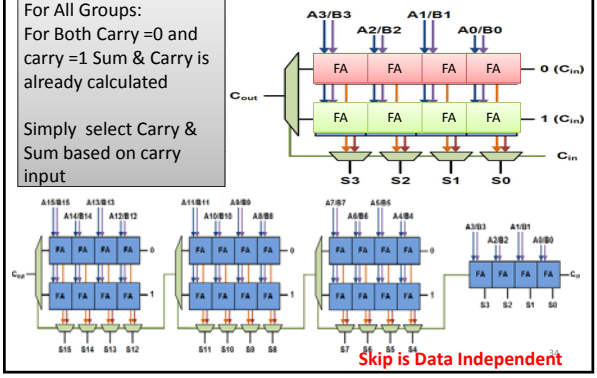
Carry Skip Path



Carry Select Adder

For All Groups:
For Both Carry =0 and carry =1 Sum & Carry is already calculated

Simply select Carry & Sum based on carry input



Delay Analysis of Adder

- Ripple Carry Adder (RCA) = N * T_c
- Machester RCA = N*T_m
- Carry Skip Adder = Depend on Data
  - Let probability of skip is p
  - If skip delay for Group is T_s
  - Else Delay of Group m*T_c

  Total Delay = p (N/m) T_s + (p-1) *(N/m)* m * T_c
- Carry Select Adder = Independent of Data

  Delay of select = T_s

  T = (N/m - 1) T_s + m T_c

Delay of Adder α N : Linear Delay

- Ripple Carry Adder (RCA) = N * T_c = **α N**
- Machester RCA = N * T_m = **α N**
- Carry Skip Adder

  Total Delay = p (N/m) T_s + (p-1) *(N/m)* m * T_c

  T= N * (p/m*T_s+(p-1)T_c) = **α N**
- Carry Select Adder = Independent of Data

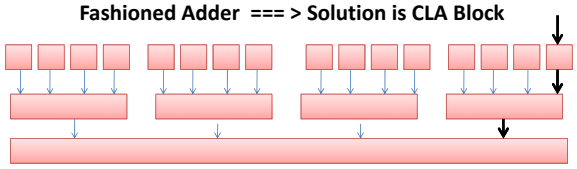
  Delay of select = Ts

  T = (N/m - 1) T_s + m T_c

  T = N*T_s/m + (mT_c-T_s) = **α N +c**

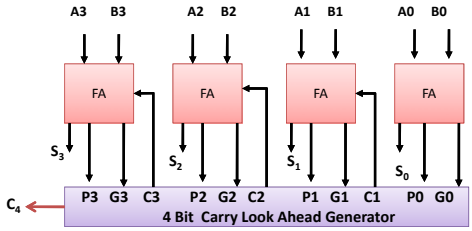
Delay is linear

- Always carry moves from Right to Left
- Some time Skip/Always Skip It have to select
- Linear Arrangement == > O(N) delay
- **How to reduced to Logarithmic delay ?**
  - **Tree Fashion/ Hierarchical Fashion**
  - **Design a block which can be used in Tree Fashioned Adder === > Solution is CLA Block**



Carry Look Ahead Adder (CLA)

- Basic Idea: Compute Several Carries Simultaneously and use it
- Use as ~~Ripple Carry~~ or **MultiLevel CLA**



38

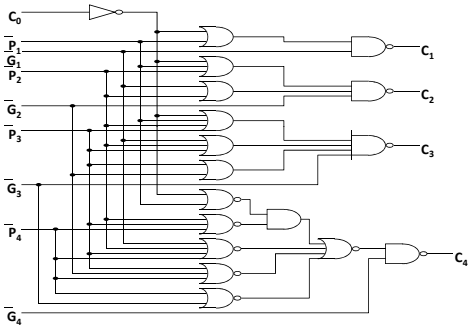
CLG

$$C_i = G_i + P_i \cdot C_{i-1} \text{ where } G_i = A_i \cdot B_i$$
$$= G_i + P_i \cdot G_{i-1} + \ldots + P_i \cdot P_{i-1} \ldots P_2 \cdot P_1 \cdot C_0$$
$$S_i = C_i \oplus P_i$$

Available for (# of inputs ≤ 4)

39

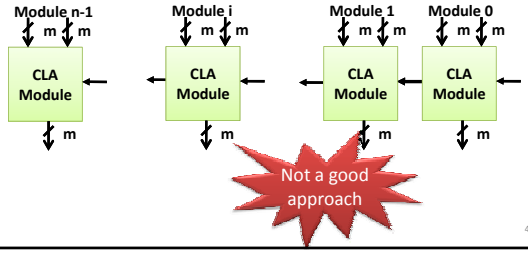
CLG (Carry Lookahead Generator)



6.40

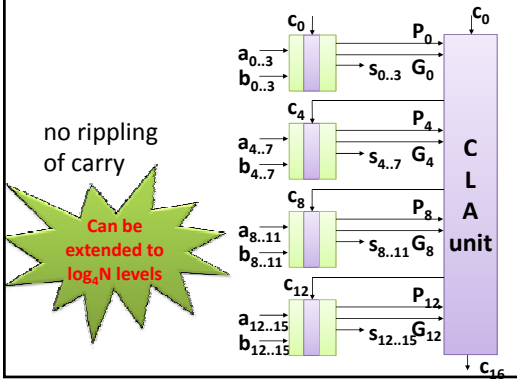
RCA with CLA units

- As Carry are generated with CLG it takes less time
- **Time complexity : O(N) ===> useless approach**



41

2 Levels of look ahead



Group propagate & generate

$c_1 = p_0 c_0 + g_0$

$c_2 = p_1 c_1 + g_1 = p_1p_0c_0+p_1g_0+g_1$

$c_3 = p_2 c_2 + g_2 = p_2p_1p_0c_0+p_2p_1g_0+p_2g_1+g_2$

$c_4 = p_3 c_3 + g_3 =$

$p_3p_2p_1p_0c_0+p_3p_2p_1g_0+p_3p_2g_1+p_3g_2+g_3$

$P_0 = p_3p_2p_1p_0$

$G_0 = p_3p_2p_1g_0+p_3p_2g_1+p_3g_2+g_3$

$c_4 = P_0 c_0 + G_0$

Group propagate & generate

$P_i = p_{i+3} p_{i+2} p_{i+1} p_i$

$G_i = p_{i+3} p_{i+2} p_{i+1} g_i + p_{i+3} p_{i+2} g_{i+1} + p_{i+3} g_{i+2} + g_{i+3}$

$c_4 = P_0 c_0 + G_0$

$c_8 = P_4 P_0 c_0 + P_4 G_0 + G_4$

$c_{12} = P_8 P_4 P_0 c_0 + P_8 P_4 G_0 + P_8 G_4 + G_8$

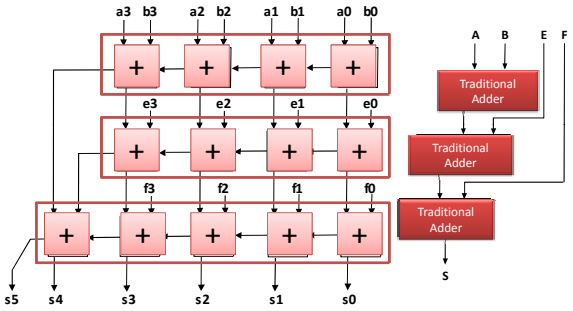
$c_{16} = P_{12} P_8 P_4 P_0 c_0 + P_{12} P_8 P_4 G_0 + P_{12} P_8 G_4 + P_{12} G_8 + G_{12}$

Summery: Two operand Additions

- Speeding up addition
  - –Ripple carry adder (carry propagate): O(n)
  - –Carry look ahead: O(log n)
- What about Multi-Operand Adder
  - $–A=N_0+N_1+N_2+.....+N_n$
- Where we require: possibly in multiply or advance computing places

45

Adding multiple operands



Carry save addition