

CSE140: Adders

Xinyuan Wang

05/24/2019

Full Adder/Subtractor

- How we implement a 1-bit adder/subtractor?

✓ Given 2 **binary** inputs A, B and a carry in/borrow in bit, calculate $A + B / A - B$

$$\begin{array}{r}
 \text{carry in } 1 \\
 0 \quad A \\
 + 1 \quad B \\
 \hline
 \text{carry out } 1 \quad 0
 \end{array}
 \qquad
 \begin{array}{r}
 \text{borrow in } 0 \\
 0 \quad A \\
 - 1 \quad B \\
 \hline
 \text{borrow out } (1) \quad 1
 \end{array}$$

- ✓ A full adder with input $(A, B, C_{in}) \Rightarrow$ output (C_{out}, S)
- ✓ A full subtractor with input $(A, B, B_{in}) \Rightarrow$ output (B_{out}, D)
- ✓ Write the truth table and derive Boolean expressions

Truth Tables of Full Adder/Subtractor

- Derive the Boolean expressions
 - ✓ What can we observe by comparing the truth tables? 

Full Adder

C_{in}	A	B	S	C_{out}
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

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

Full Subtractor

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D = A \oplus B \oplus C_{in}$$

$$B_{out} = A'B + A'B_{in} + BB_{in}$$

Output D is same as output S
 A is complemented in subtractor

Q1: one's complement

- If bit $t = 0$, the function operates as a full adder on (a, b, c_{in}) ; if bit $t = 1$, operates on the complement (a, b', c_{in})
 - ✓ fill in the truth table with full adder

				(a, b, c_{in})		(a, b', c_{in})	
c_{in}	a	b	b'	$t = 0$		$t = 1$	
				c_{out}	s	c_{out}	s
0	0	0	1	0	0	Fill in the outputs	
0	0	1	0	0	1		
0	1	0	1	0	1		
0	1	1	0	1	0		
1	0	0	1	0	1		
1	0	1	0	1	0		
1	1	0	1	1	0		
1	1	1	0	1	1		

Full Adder/Subtractor

- Carry propagate adders (CPAs)

- ✓ Ripple-carry adders

- ✓ Carry-Lookahead adder

- ❖ When a column will generate a carry out? $\Rightarrow G_i = A_i B_i$

- ❖ When a column will propagate a carry in to carry out? $\Rightarrow P_i = A_i + B_i$

- ❖ How to compute G and P for a k-bit block?



Carry out of column i , as
carry in of column $i + 1$

$$\begin{aligned} C_{in,i+1} &= C_{out,i} \\ &= A_i B_i + (A_i + B_i) C_{in,i} \\ &= G_i + P_i C_{in,i} \end{aligned}$$

- ✓ Prefix Adder

- ❖ a hierarchy structure with segmentations into small blocks

- ❖ parallel prefix operations

Prefix Adder

- Compute the carry in for each column as fast as possible
- Has $\log_2(N)$ stages
- Column -1 holds C_{in} , so $G_{-1} = C_{in}$, $P_{-1} = 0$
- Compute the **prefixes**: $G_{0:-1}, G_{1:-1}, G_{2:-1}, \dots, G_{N-1:-1}$
 - ✓ the generate and propagate for a block spanning bits $i:j$

$$\begin{aligned} G_{i:j} &= G_{i:k} + P_{i:k} G_{k-1:j} \\ P_{i:j} &= P_{i:k} P_{k-1:j} \end{aligned}$$

Note: $G_{i:i} = A_i B_i$, $P_{i:i} = A_i + B_i$ are the generate and propagate for column i

- The sum of each column

$$S_i = (A_i \oplus B_i) \oplus \mathbf{G}_{i-1:-1}$$

Prefix Adder

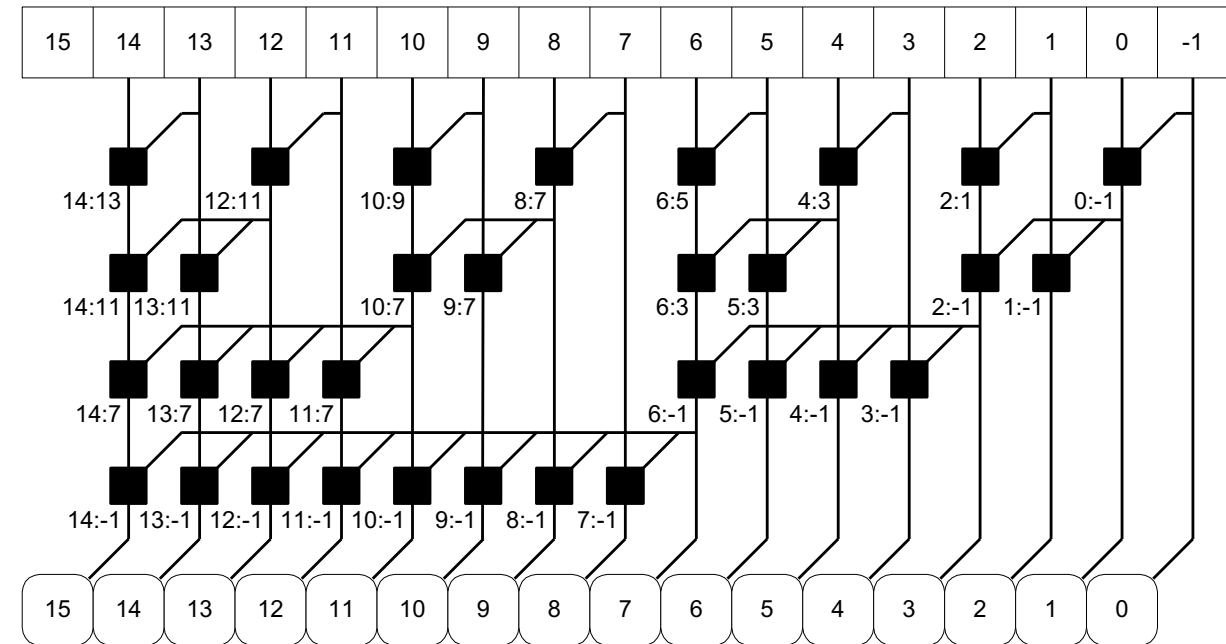
- A 16-bit Prefix Adder from slide 27 of Lecture 14

- ✓ Has 4 stages
- ✓ step 1: column generate and propagate $\Rightarrow (G_{i:i}, P_{i:i})$
- ✓ step2: block generate and propagate

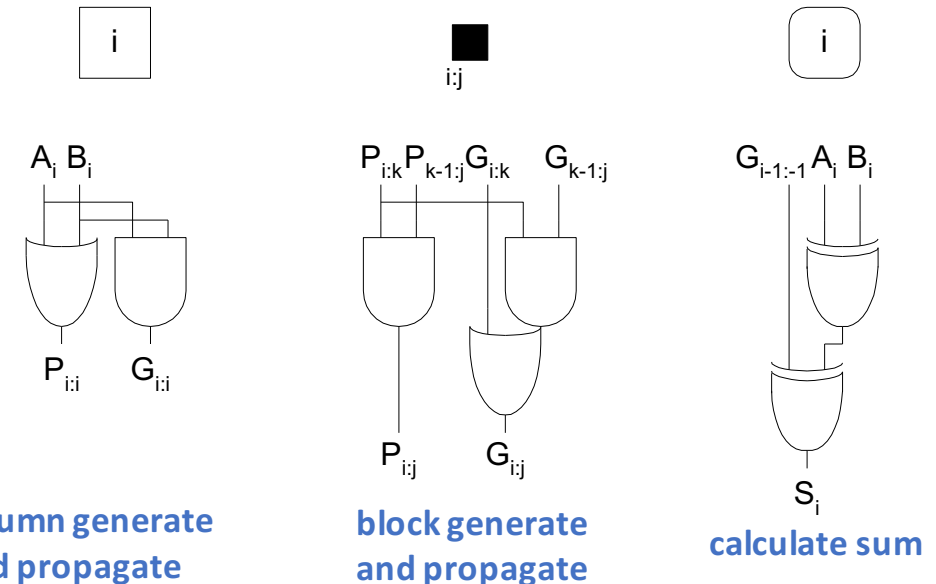
$$\left. \begin{matrix} (G_{i:k}, P_{i:k}) \\ (G_{k-1:j}, P_{k-1:j}) \end{matrix} \right\} \Rightarrow (G_{i:j}, P_{i:j})$$

- ✓ step 3: calculate sum S_i with

$$A_i, B_i, G_{i-1:-1}$$



Legend



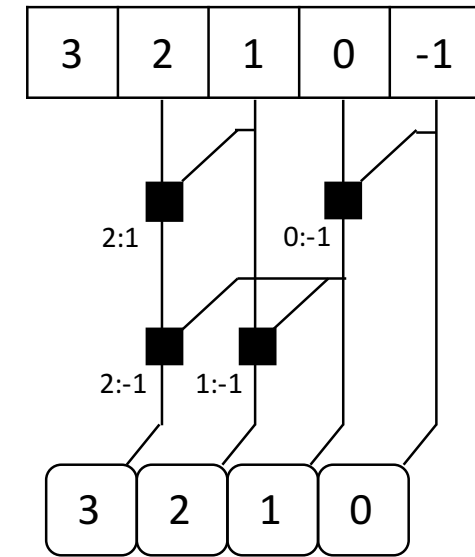
Prefix Adder

- Calculate sum of 13, 6 with a 4-bit Prefix Adder, $C_{in} = 1$
 - ✓ step 1: column generate and propagate
 $\Rightarrow (G_{i:i}, P_{i:i})$
 - ✓ step 2: block generate and propagate

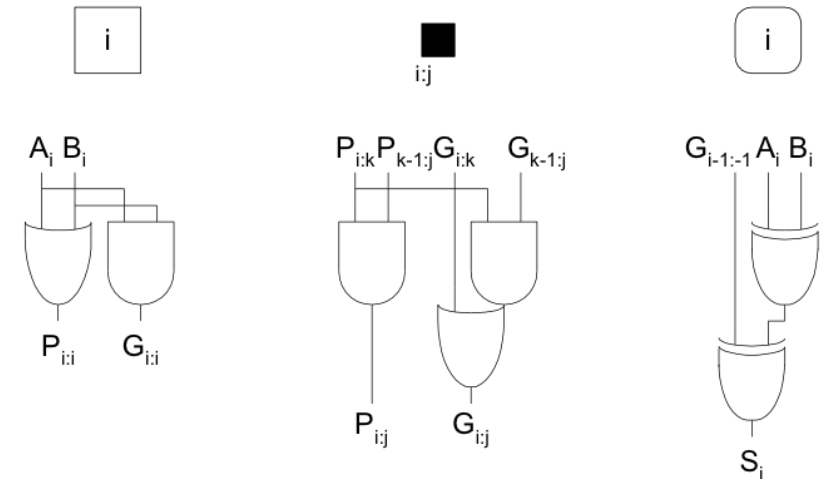
$$\left. \begin{array}{l} (G_{i:k}, P_{i:k}) \\ (G_{k-1:j}, P_{k-1:j}) \end{array} \right\} \Rightarrow (G_{i:j}, P_{i:j})$$

- ✓ step 3: calculate sum S_i with

$$A_i, B_i, G_{i-1:-1}$$



Legend



Prefix Adder

- Calculate sum of 13, 6 with a 4-bit Prefix Adder, $C_{in} = 1$
 - ✓ $1101 + 0110$
 - ✓ step 1: column generate and propagate

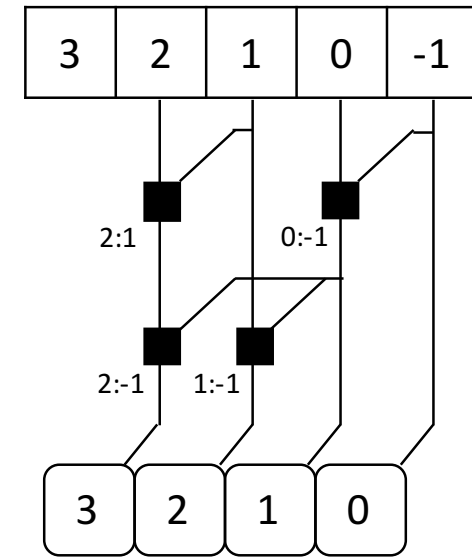
$$(G_{-1:-1}, P_{-1:-1}) = (1, 0)$$

$$(G_{0:0}, P_{0:0}) = (0, 1)$$

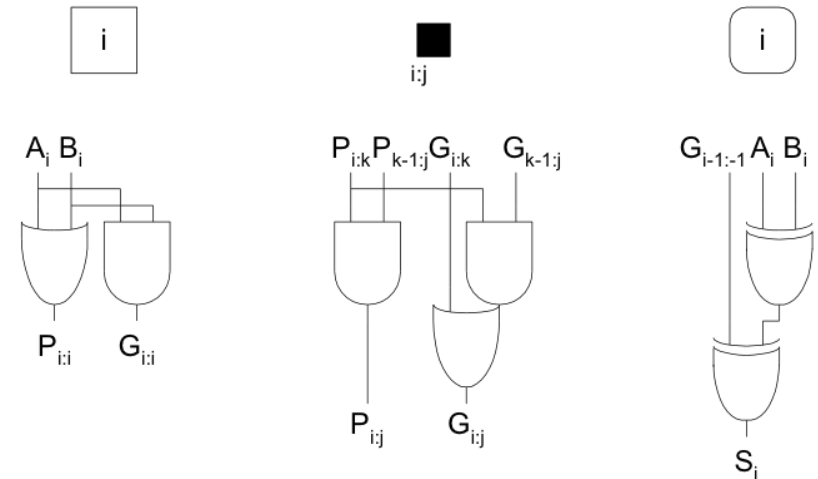
$$(G_{1:1}, P_{1:1}) = (0, 1)$$

$$(G_{2:2}, P_{2:2}) = (1, 1)$$

$$(G_{3:3}, P_{3:3}) = (0, 1)$$



Legend



Prefix Adder

- Calculate sum of 13, 6 with a 4-bit Prefix Adder, $C_{in} = 1$
 - ✓ $1101 + 0110$
 - ✓ step 1: column generate and propagate
 - ✓ step 2: block generate and propagate

$$G_{i:j} = G_{i:k} + P_{i:k}G_{k-1:j}$$

$$P_{i:j} = P_{i:k}P_{k-1:j}$$

Stage 1:

$$(G_{0:-1}, P_{0:-1}) = (1, 0)$$

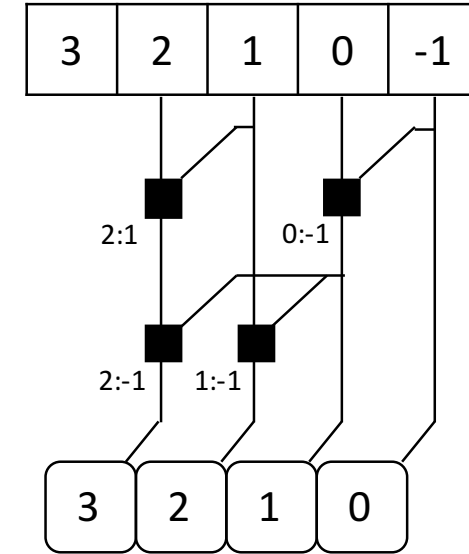
$$(G_{2:1}, P_{2:1}) = (1, 1)$$

Stage 2:

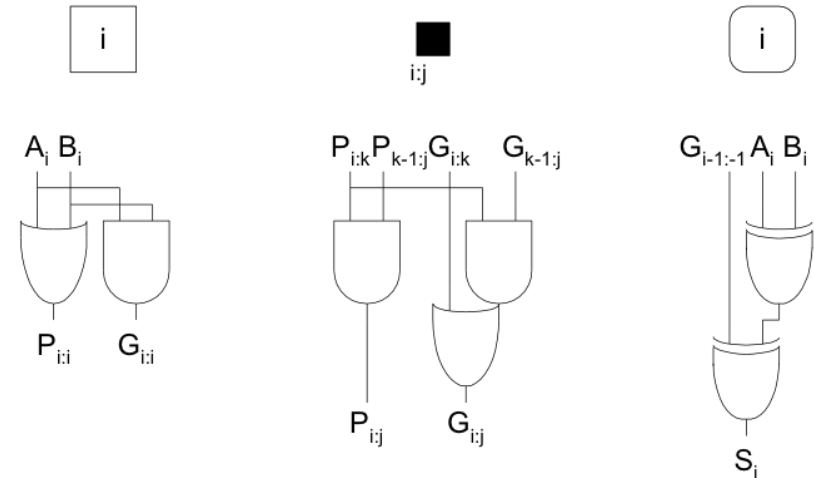
$$(G_{1:-1}, P_{1:-1}) = (1, 0)$$

$$(G_{2:-1}, P_{2:-1}) = (1, 0)$$

$(0,1)(1,1)(0,1)(0,1)(1,0)$



Legend



Prefix Adder

- Calculate sum of 13, 6 with a 4-bit Prefix Adder, $C_{in} = 1$
 - ✓ $1101 + 0110$
 - ✓ step 1: column generate and propagate
 - ✓ step2: block generate and propagate
 - ✓ step 3: calculate sum S_i with $A_i, B_i, G_{i-1:-1}$

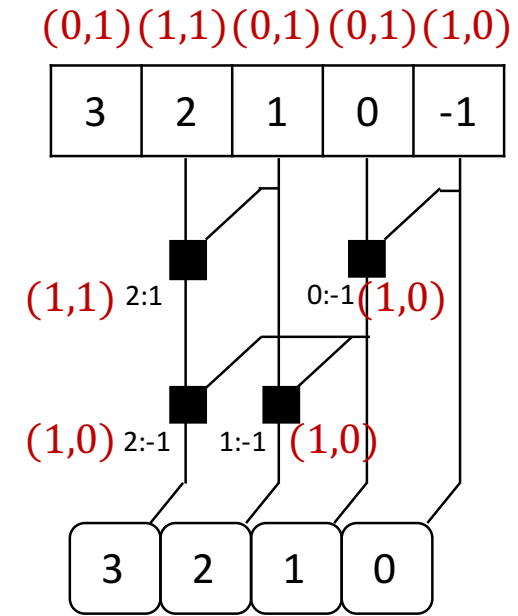
$$S_0 = (A_0 \oplus B_0) \oplus G_{-1:-1} = 0$$

$$S_1 = (A_1 \oplus B_1) \oplus G_{0:-1} = 0$$

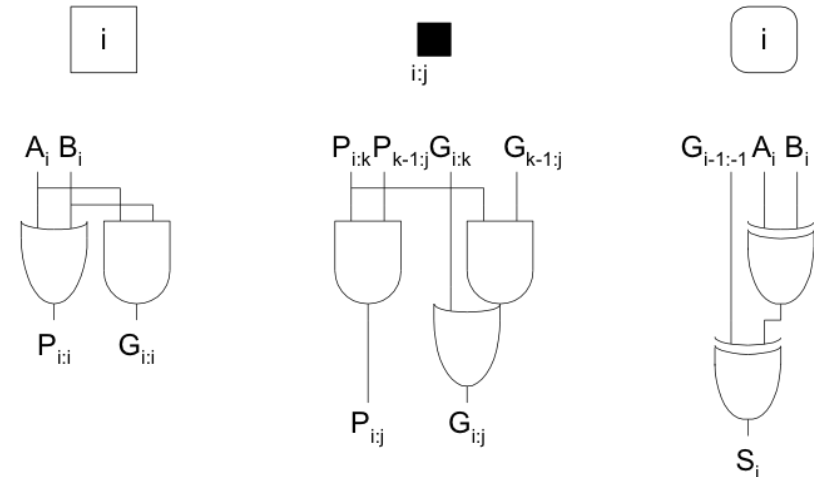
$$S_2 = (A_2 \oplus B_2) \oplus G_{1:-1} = 1$$

$$S_3 = (A_3 \oplus B_3) \oplus G_{2:-1} = 0$$

$$C_{out} = G_{3:-1} = G_{3:3} + P_{3:3}G_{2:-1} = 1$$



Legend



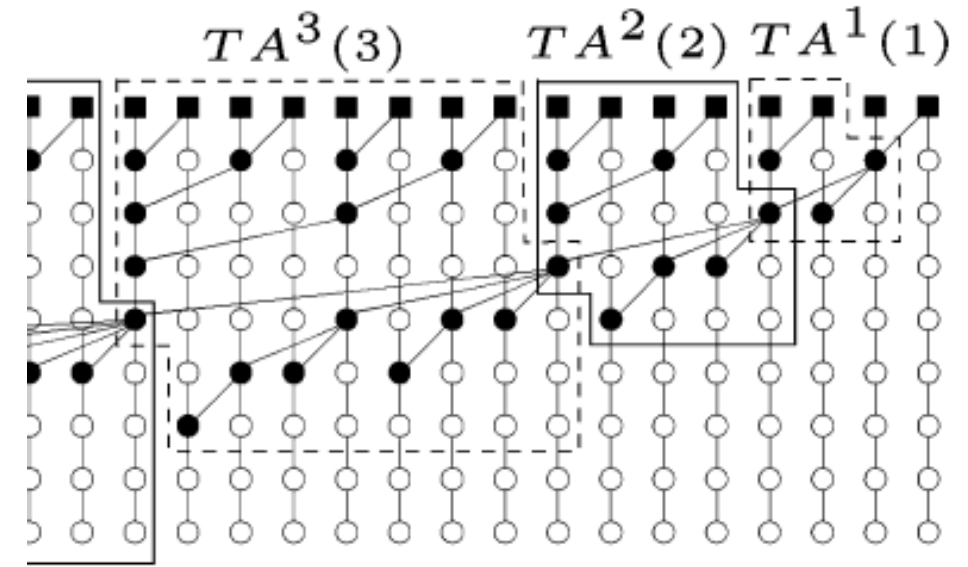
HW: Zero-deficiency prefix circuit

- Denote a prefix circuit with n inputs as $G(n)$
 - ✓ width of $G(n)$ is n
 - ✓ size of $G(n)$ is $s_{G(n)}$ = number of computation nodes
 - ✓ depth of $G(n)$ is $d_{G(n)}$ = number of levels
 - ✓ Deficiency of size of $G(n)$

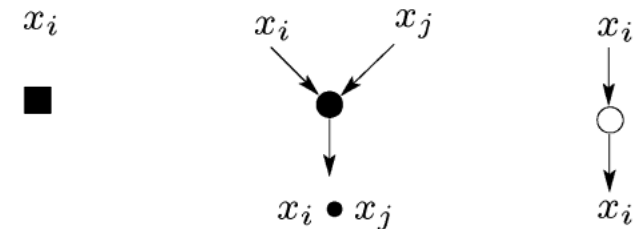
$$def(G(n)) = s_{G(n)} + d_{G(n)} - (2n - 2)$$

zero-deficiency: $def(G(n)) = 0$

- Implement the zero-deficiency prefix adder

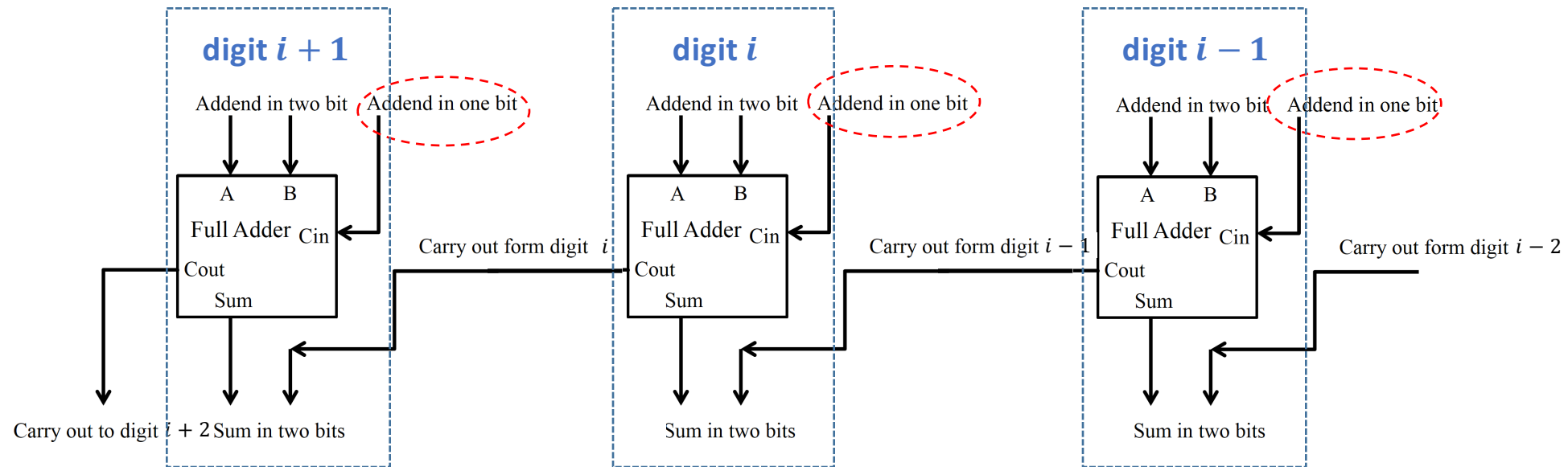


legend



Carry Save Adder

- A Redundant Number System
- Save the Partial Solution for Multiple Additions
- One Bit Propagation for Each Addition



Carry Save Adder

- Example: calculate the sum of 23, 11, 17

Weight	32	16	8	4	2	1
n1=23	0	1	0	1	1	1
n2=11	0	0	1	0	1	1
n3=17	0	1	0	0	0	1

Carry Save Adder

- Example: calculate the sum of 23, 11, 17

Weight	64	32	16	8	4	2	1
n1=23	0	0	1	0	1	1	1
n2=11	0	0	0	1	0	1	1
n3=17	0	0	1	0	0	0	1

	Addend in one bit						
n1	00	00	10	00	10	10	10
n2	0	0	0	1	0	1	1
A=n1+n2							
n3							
B=A+n3							
Sum							

Redundant system

Carry Save Adder

- Example: calculate the sum of 23, 11, 17

Weight	64	32	16	8	4	2	1
n1=23	0	0	1	0	1	1	1
n2=11	0	0	0	1	0	1	1
n3=17	0	0	1	0	0	0	1

n1	00	00	10	00	10	10	10
n2	0	0 carry	0 carry	1 carry	0 carry	1 carry	1
A=n1+n2	00	00	10	10	11	01	00
n3							
B=A+n3							
Sum							

sum

Add three bits with one bit propagation

Carry Save Adder

- Example: calculate the sum of 23, 11, 17

Weight	64	32	16	8	4	2	1
n1=23	0	0	1	0	1	1	1
n2=11	0	0	0	1	0	1	1
n3=17	0	0	1	0	0	0	1

n1	00	00	10	00	10	10	10
n2	0	0	0	1	0	1	1
A=n1+n2	00	00	10	10	11	01	00
n3	0	0	1	0	0	0	1
B=A+n3							
Sum							

Partial Solution

Carry Save Adder

- Example: calculate the sum of 23, 11, 17

Weight	64	32	16	8	4	2	1
n1=23	0	0	1	0	1	1	1
n2=11	0	0	0	1	0	1	1
n3=17	0	0	1	0	0	0	1

n1	00	00	10	00	10	10	10
n2	0	0	0	1	0	1	1
A=n1+n2	00	00	10	10	11	01	00
n3	0	0	1	0	0	0	1
B=A+n3	00	01	00	11	00	10	10
Sum							

Add three bits with one bit propagation

Carry Save Adder

- Example: calculate the sum of 23, 11, 17

Weight	64	32	16	8	4	2	1
n1=23	0	0	1	0	1	1	1
n2=11	0	0	0	1	0	1	1
n3=17	0	0	1	0	0	0	1

n1	00	00	10	00	10	10	10
n2	0	0	0	1	0	1	1
A=n1+n2	00	00	10	10	11	01	00
n3	0	0	1	0	0	0	1
B=A+n3	00	01	00	11	00	10	10
Sum							

Partial Solution
An extra addition is needed!

Carry Save Adder

- Example: calculate the sum of 23, 11, 17

Weight	64	32	16	8	4	2	1
n1=23	0	0	1	0	1	1	1
n2=11	0	0	0	1	0	1	1
n3=17	0	0	1	0	0	0	1

n1	00	00	10	00	10	10	10
n2	0	0	0	1	0	1	1
A=n1+n2	00	00	10	10	11	01	00
n3	0	0	1	0	0	0	1
B=A+n3	00	01	00	11	00	10	10
Two bits of B	0	1	0	1	0	0	0
	0	0	0	1	0	1	1
Sum							

Split the redundant number

Carry Save Adder

- Example: calculate the sum of 23, 11, 17

Weight	64	32	16	8	4	2	1
n1=23	0	0	1	0	1	1	1
n2=11	0	0	0	1	0	1	1
n3=17	0	0	1	0	0	0	1

n1	00	00	10	00	10	10	10
n2	0	0	0	1	0	1	1
A=n1+n2	00	00	10	10	11	01	00
n3	0	0	1	0	0	0	1
B=A+n3	00	01	00	11	00	10	10
Two bits of B	0	1	0	1	0	0	0
	0	0	0	1	0	1	1
Sum	0	1	1	0	0	1	1

HW: Carry Save Adder

- Add five numbers for each addition
 - Weight 0 (0, 0, 0, 0, 0) $\Rightarrow$ 000
 - Weight 1 (1, 0, 0, 0, 0) (0, 1, 0, 0, 0) ... (0, 0, 0, 0, 1) $\Rightarrow$ 001
 - $\vdots$
 - Weight 5 (1, 1, 1, 1, 1) $\Rightarrow$ 100
- How many bits should be propagated?

