

DIGITAL LOGIC

LECTURE-20



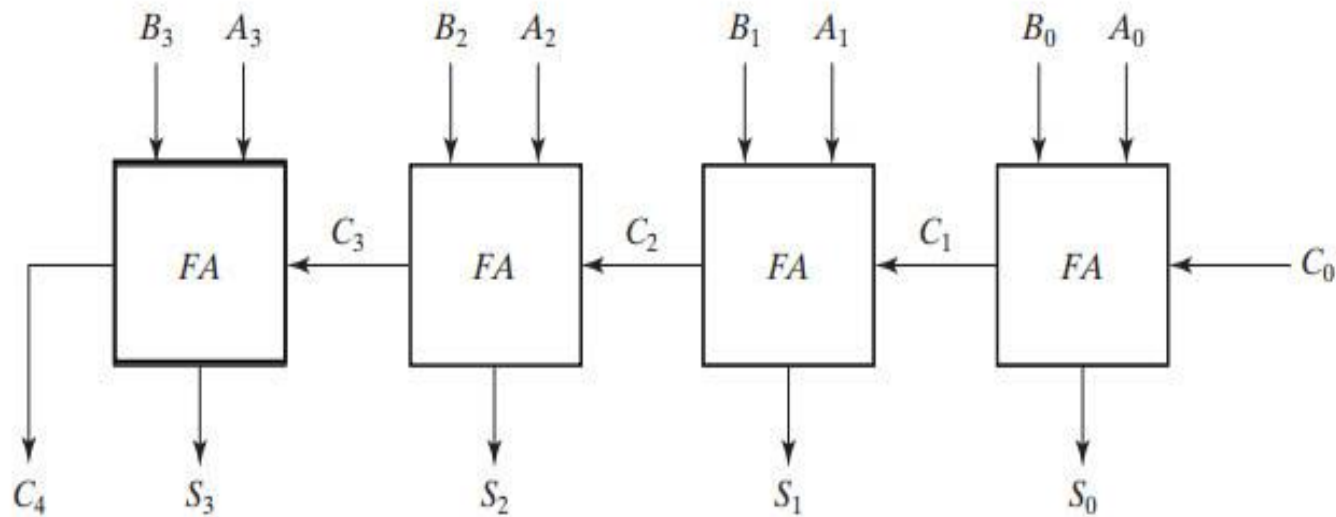
Combinational Logic with MSI & LSI:

Binary Parallel adder


- A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers.
- It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder in the chain.



- Addition of **n-bit numbers** requires a chain of **n full adders** or a chain of **one-half adder** and **n - 1 full adders**.



Four-bit adder

- The above figure shows the interconnection of 4 full-adder (FA) circuits to provide a **4-bit binary ripple carry** adder. The augend bits of A and the addend bits of B are designated by subscript numbers from right to left, with subscript 0 denoting the least significant bit.
 - The carries are connected in a chain through the full adders.
 - The input carry to the adder is C_0 , and it ripples through the full adders to the output carry C_4 .
 - The S outputs generate the required sum bits.
 - **An n -bit adder requires n full adders**, with each output carry connected to the input carry of the next higher order full adder.
- 

- To demonstrate with a specific example, consider the two binary numbers $A = 1011$ and $B = 0011$. Their sum $S = 1110$ is formed with the four-bit adder as follows:

Subscript i:	3	2	1	0	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output carry	0	0	1	1	C_{i+1}



HDL for Binary Adder

// Gate-level description of four-bit ripple carry adder

// Description of half adder

```
// module half_adder (S, C, x, y);
```

```
// output S, C;
```

```
// input x, y;
```

```
module half_adder ( output S, C, in
```

```
// Instantiate primitive gates
```

```
xor (S, x, y);
```

```
and (C, x, y);
```

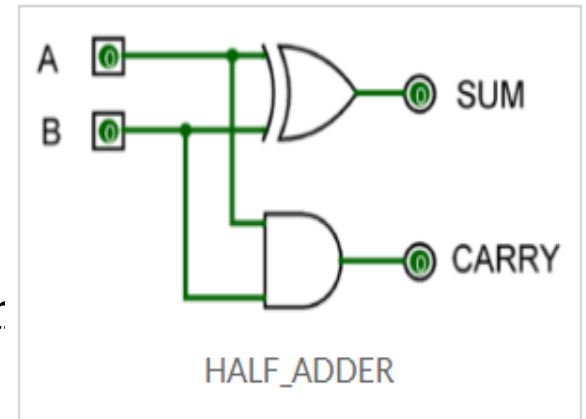
```
endmodule
```

// Description of full adder

```
// module full_adder (S, C, x, y, z);
```

```
// output S, C;
```

```
// input x, y, z;
```



```
module full_adder ( output S, C, input x, y, z);
```

```
wire S1, C1, C2;
```

```
// Instantiate half adders
```

```
half_adder HA1 (S1, C1, x, y
```

```
half_adder HA2 (S, C2, S1, z
```

```
or G1 (C, C2, C1);
```

```
endmodule
```

```
// Description of four-bit adder
```

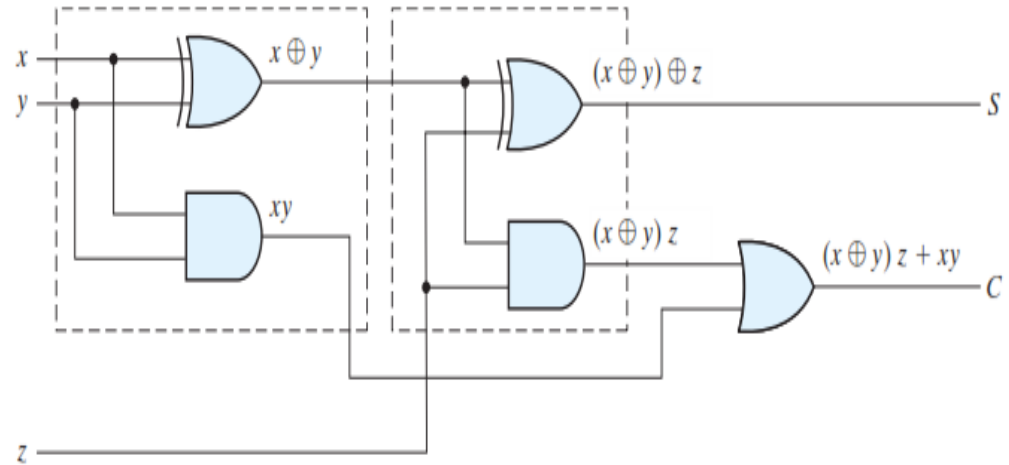
```
// module ripple_carry_4_bit_adder (Sum, C4, A, B, C0);
```

```
// output [3: 0] Sum;
```

```
// output C4;
```

```
// input [3: 0] A, B;
```

```
// input C0;
```



// Alternative Verilog 2001, 2005 syntax:

```
module ripple_carry_4_bit_adder ( output [3: 0] Sum,  
    output C4,  
    input [3: 0] A, B, input C0);  
wire C1, C2, C3; // Intermediate carries  
// Instantiate chain of full adders  
full_adder FA0 (Sum[0], C1, A[0], B[0], C0),  
    FA1 (Sum[1], C2, A[1], B[1], C1),  
    FA2 (Sum[2], C3, A[2], B[2], C2),  
    FA3 (Sum[3], C4, A[3], B[3], C3);  
endmodule
```



Advantage

The four-bit adder is a typical example of a standard component. It can be used in many applications involving arithmetic operations. Observe that the design of this circuit by the classical method would require a truth table with $2^9 = 512$ entries, since there are nine inputs to the circuit. By using an iterative method of cascading a standard function, it is possible to obtain a simple and straightforward implementation.

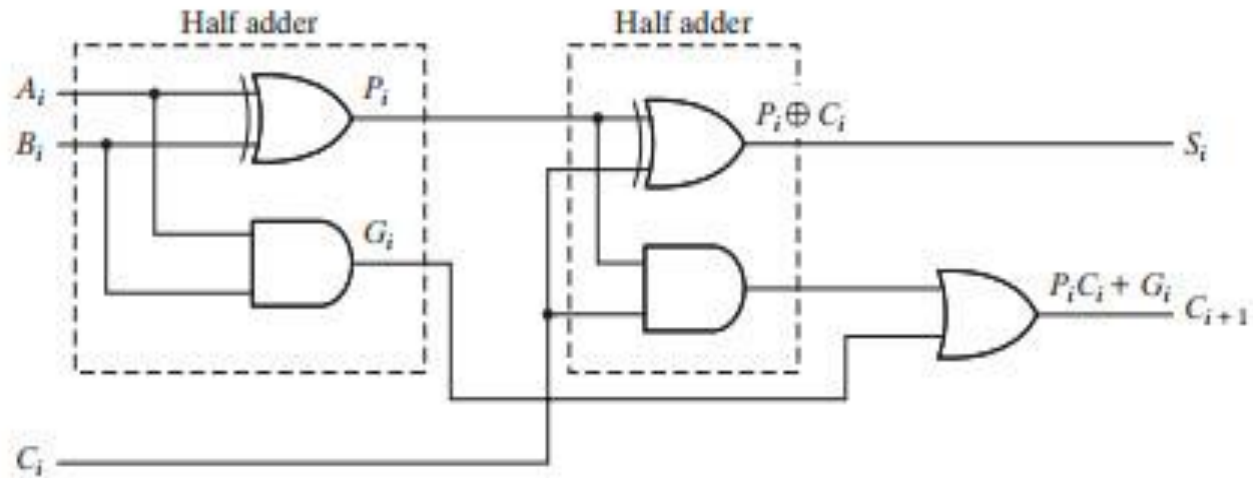


Disadvantage

- The addition of two binary numbers in parallel implies that all the bits of the augend and addend are available for computation at the same time. The longest propagation delaytime in an adder is the time it takes the carry to propagate through the full adders.
- Since each bit of the sum output depends on the value of the input carry, the value of S_i at any given stage in the adder will be in its steady-state final value only after the input carry to that stage has been propagated.



Carry Propagation



Full adder with P and G shown



- The input and output variables use the subscript i to denote a typical stage of the adder. The signals at P_i and G_i settle to their steady-state values after they propagate through their respective gates.
- The signal from the input carry C_i to the output carry C_{i+1} propagates through an AND gate and an OR gate, which constitute two gate levels.
- If there are four full adders in the adder, the output carry C_{i+1} would have $2 * 4 = 8$ gate levels from C_0 to C_4 .
- For an n -bit adder, there are $2n$ gate levels for the carry to propagate from input to output.



- The carry propagation time is an important attribute of the adder because it limits the speed with which two numbers are added.
- Since all other arithmetic operations are implemented by successive additions, the time consumed during the addition process is critical.
- An obvious solution for reducing the carry propagation delay time is to employ faster gates with reduced delays.
- Solution is to increase the complexity of the equipment in such a way that the carry delay time is reduced.
- The most widely used technique employs the principle of *carry lookahead logic*.



If we define two new binary

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

the output sum and carry can respectively be expressed as

$$S_i = P_i \oplus C_i$$

$$C_{i+1} = G_i + P_i C_i$$

G_i is called a *carry generate*, and it produces a carry of 1 when both A_i and B_i are 1, regardless of the input carry C_i . P_i is called a *carry propagate*, because it determines whether a carry into stage i will propagate into stage $i + 1$.



- We now write the Boolean functions for the carry outputs of each stage and substitute the value of each C_i from the previous equations:

$$C_0 = \text{input carry}$$

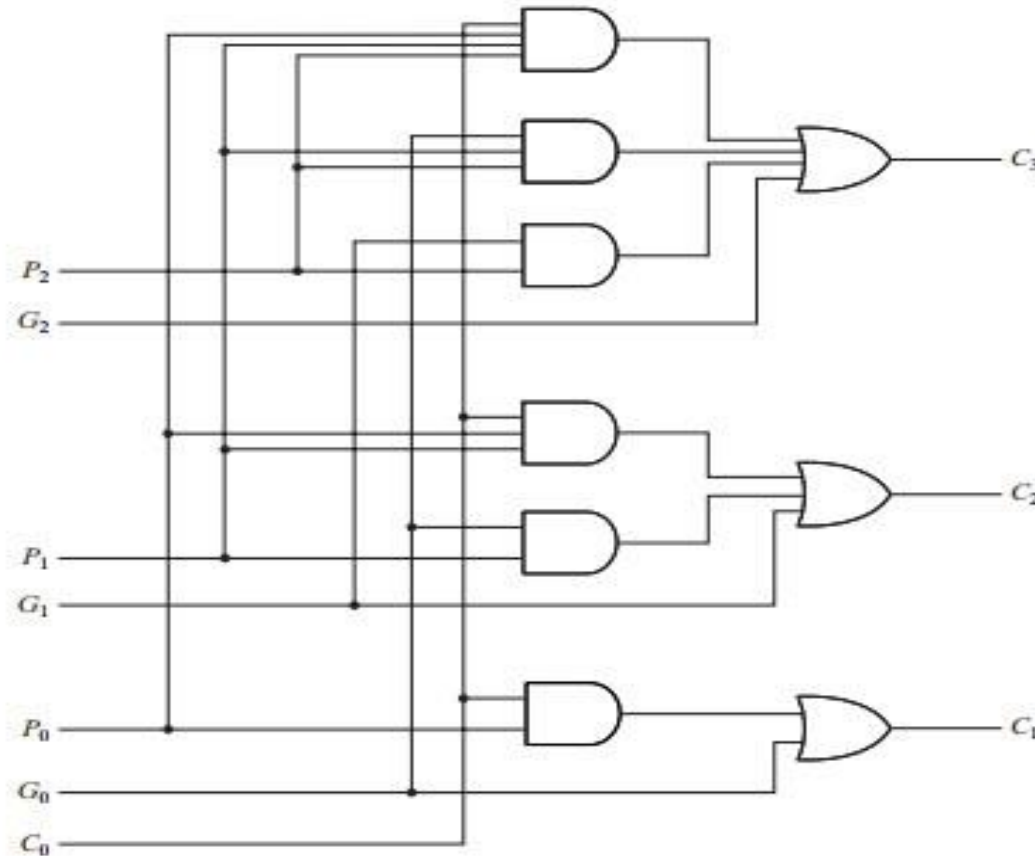
$$C_1 = G_0 + P_0C_0$$

$$C_2 = G_1 + P_1C_1 = G_1 + P_1(G_0 + P_0C_0) = G_1 + P_1G_0 + P_1P_0C_0$$

$$C_3 = G_2 + P_2C_2 = G_2 + P_2G_1 + P_2P_1G_0 = P_2P_1P_0C_0$$

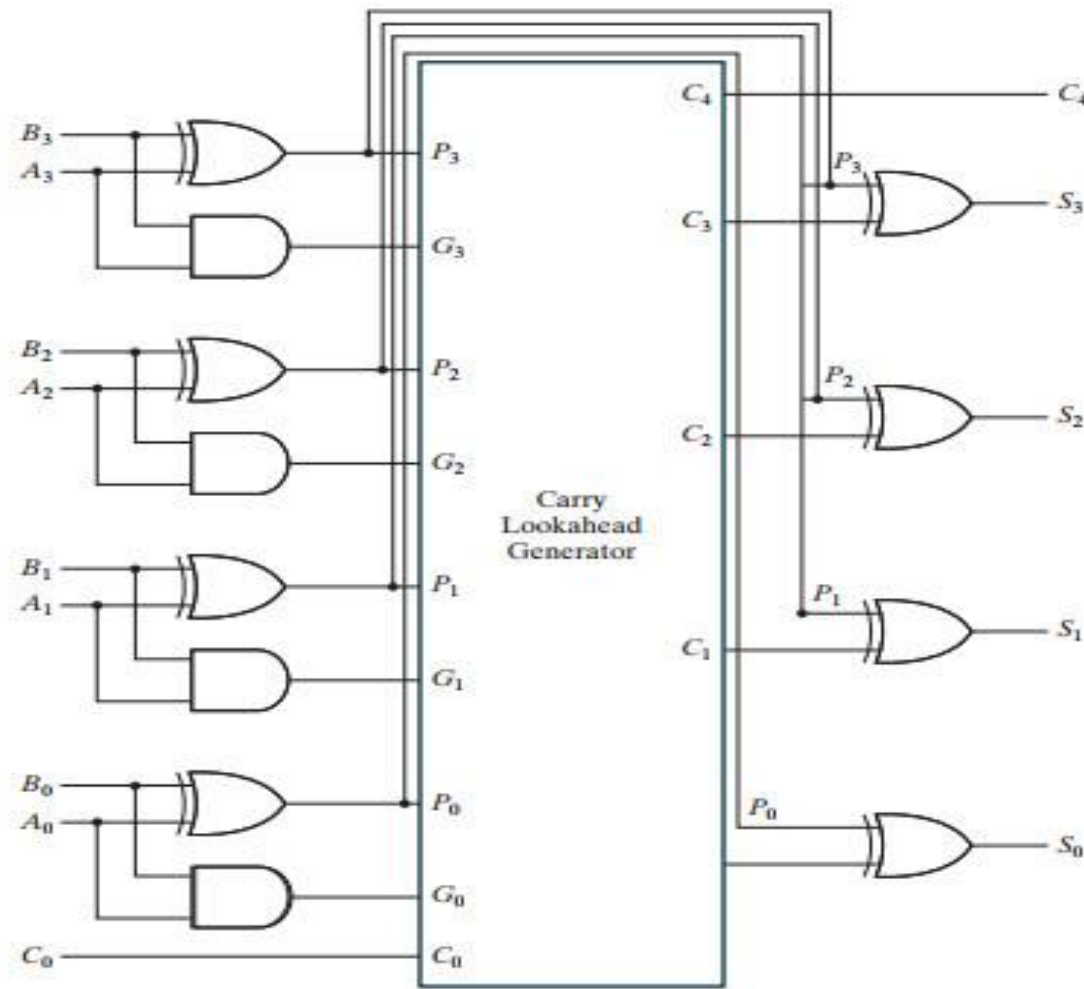


- The three Boolean functions for C_1 , C_2 , and C_3 are implemented in the carry lookahead generator shown in Fig. below



Logic diagram of carry lookahead generator

The construction of a four-bit adder with a carry lookahead scheme is shown below:



Four-bit adder with carry lookahead

THANK YOU

