Lecture – 09

System Design

- A system is defined as a collection of components, that are connected to form a coherent entity with a specific function or purpose. Example: Computer.

- The function of a system is defined by the functions of its components and how the components are connected.

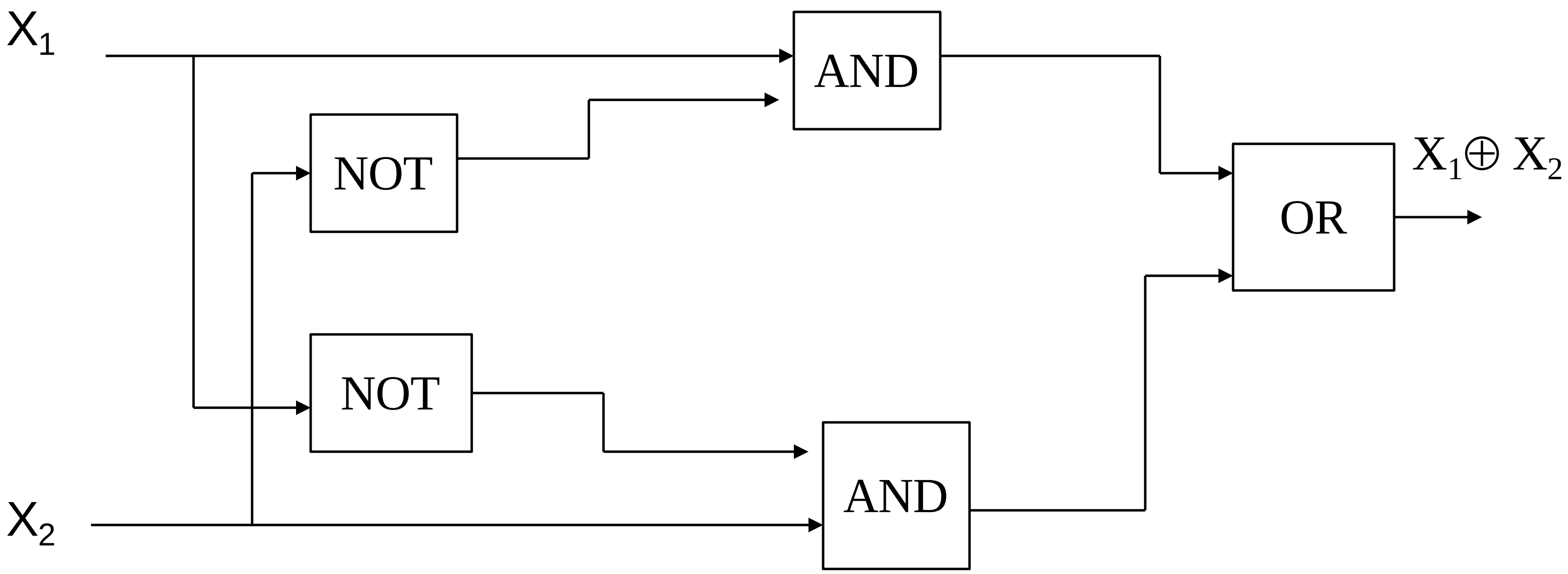
System Representation

- A directed graph is used to represent a system. It consists of a set of objects V = {v_1, v_2,….,v_n} called nodes or vertices and a set of edges E whose members are ordered pairs of nodes. The edge E = (v_i, v_j) joins node v_i to node v_j.

- Our system of interest consists of two classes of components: a set of information processing components (C) and a set of lines that carry information signals between components.

Block Diagram

- In modeling the system by a graph, we associate C with the nodes of G and S with the edges. The resulting graph is known as block diagram.
- Each component is represented by a block or box in which it's name or function can be written.
- Example: Block diagram of EX-OR logic circuit.



Structure

- The structure of a system is an abstract graph consisting of its block diagram with no functional information.
- A structural description names components and defines their interconnection.

Behavior

- The behavioral description determines for any given input signal *a* to the system, the corresponding output *f(a)*. Function *f* is the behavior of the system.

- The behavior *f* may be represented in many different ways:

1. Truth table.
2. Mathematical equation of form $f(a) = f(x_1, x_2)$
3. HDL (Hardware Description Language).

Input a		Output
x_1	x_2	f(a)
0	0	0
0	1	1
1	0	1
1	1	0

f(0,0) = 0
f(0,1) = 1
f(1,0) = 1
f(1,1) = 0

A block diagram conveys structure rather than behavior.

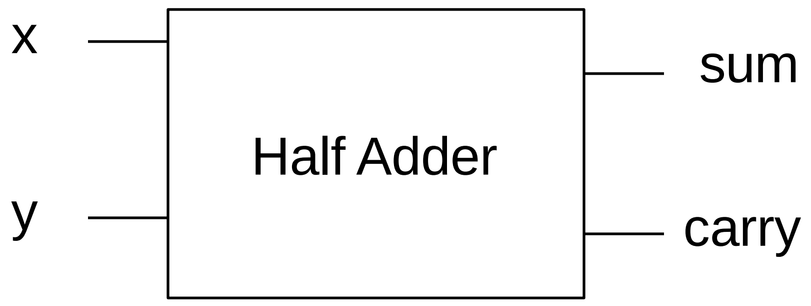
Hardware Description Language (HDL)

HDL resembles a high-level programming language such as ADA and C.

Advantages:

- It provides precise, technology-independent descriptions of digital circuits at various levels of abstraction, primarily the gate and the register levels.
- It is used for documentation.
- It can be processed by computers and is suitable for CAD.

Hardware Description Language (HDL)



x	y	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

```
entity half_adder is
port (x, y: in bit; sum, carry: out bit);
end  half_adder ;


architecture  behavior of half_adder is
begin
sum <= x xor y;
carry <= x and y;
end behavior ;
```

Design Process

Design Problem:

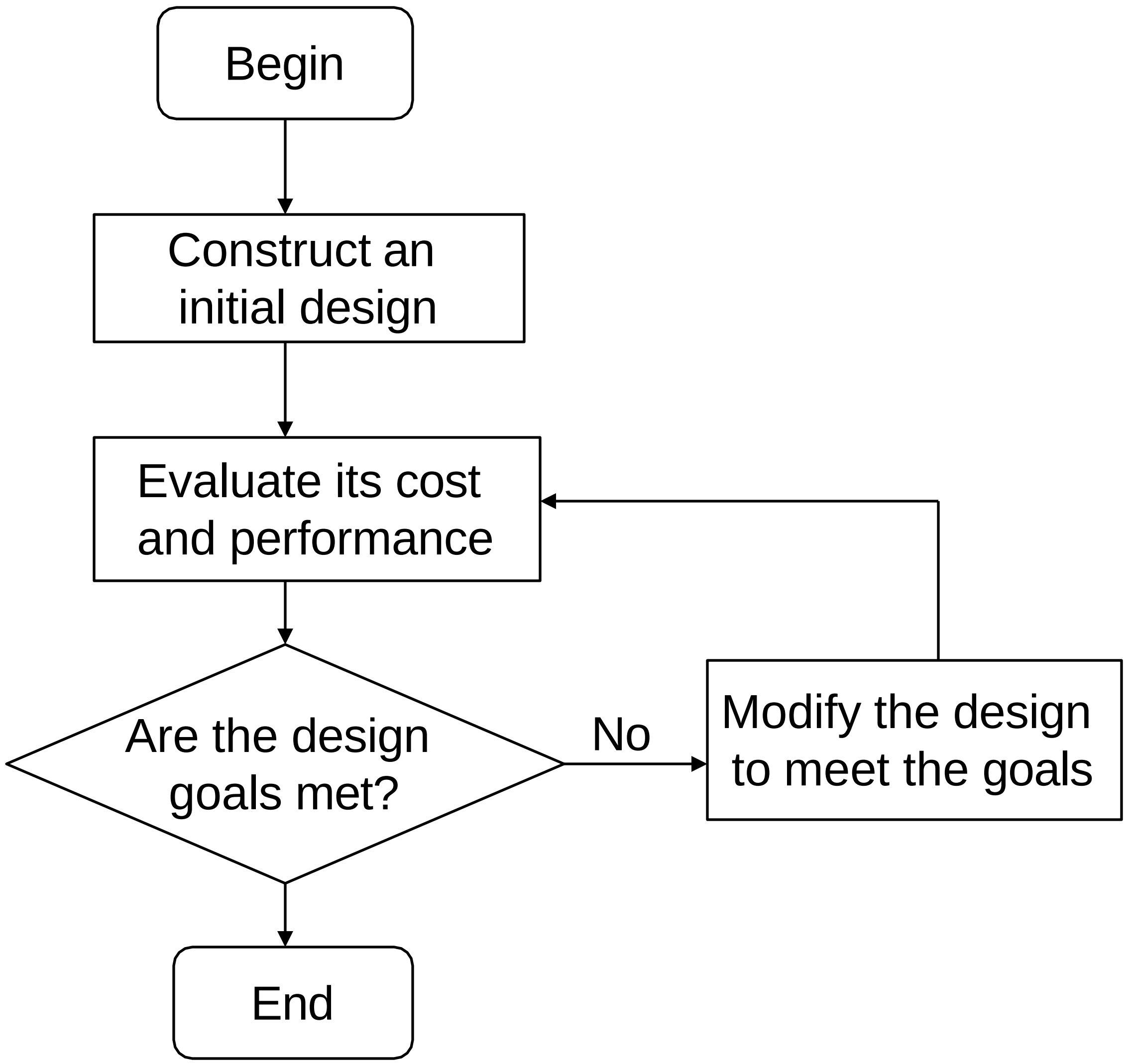
Given a desired range of behaviors and a set of available components, a structure formed these components that achieves the desired behavior with acceptable cost and performance.

- Beside assuring the correct behavior of the new design, it is required to minimize the cost measured by the cost of manufacture and to maximize the performance as measured by the speed of operation.

- Other performance and cost related constraints are high reliability, low power consumption and compatibility with existing system.

- Generally the design problem is broken into smaller, easier tasks. These smaller problems can be solved independently by different designers. Each major steps is implemented by an iterative design process.

Flow chart of an Iterative Design Process


Begin
Construct an initial design
Evaluate its cost and performance
Are the design goals met?
No
Modify the design to meet the goals
End

Computer-aided Design

CAD tools are used to automate, at least in part, the more tedious design and evaluation steps and contribute in three important ways to the overall design process:

- **CAD editors** or **translators** convert design data into forms such as HDL descriptions or schematic diagrams, which humans, computers, or both can efficiently process.
- **Simulators** create computer models of a new design, which can mimic the design's behavior and help designers determine how well the design meets various performance and cost goals.
- **Synthesizers** automate the design process itself by deriving structures that implement all or part of some design step.

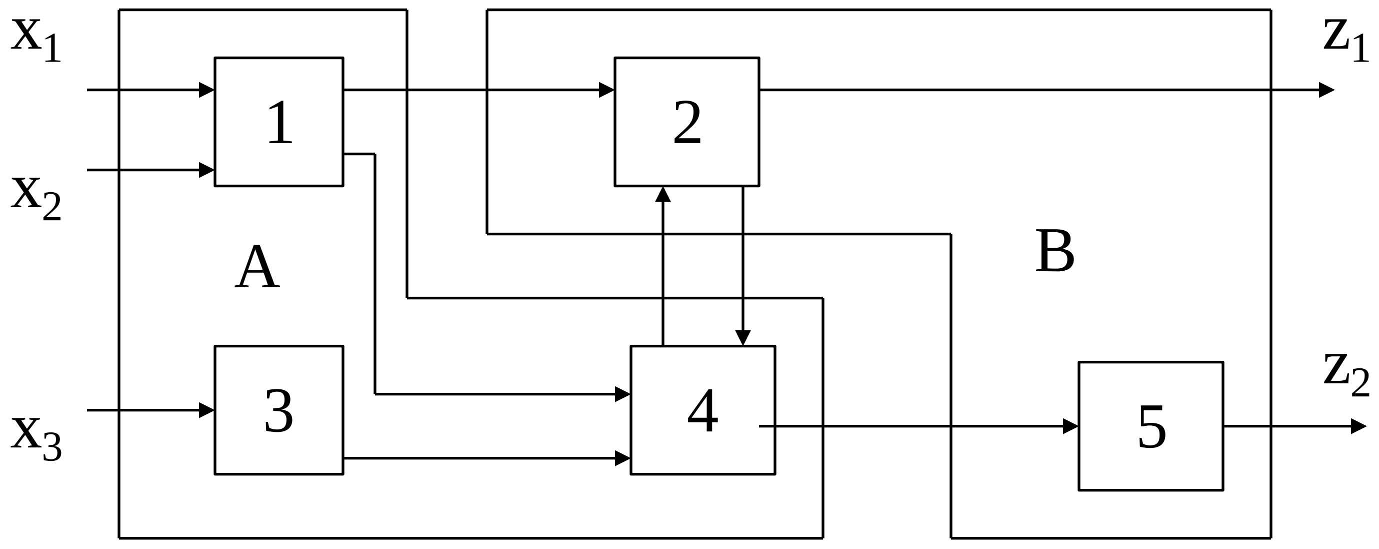
Design Levels

- **Processor** level, also known as **architecture**, **behavior** or **system** level.
- **Register** level, also called the **register-transfer** level.
- **Gate** level, also called the **logic** level.

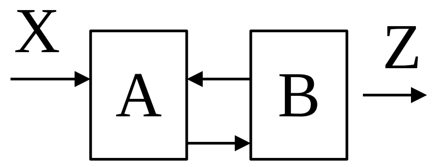
Level	**Components**	**IC Density**	**Information unit**	**Time units**
Gate	Logic gates, filp-flops	SSI	Bits	10^{-12} to 10^{-9}s
Register	Registers, counters, combinational circuits, small sequential circuits	MSI	Words	10^{-9} to 10^{-6} s
Processor	CPUs, memories, IO devices	VLSI	Blocks of Words	10^{-3} to 10^{3} s

System Hierarchy

- Design levels are treated as high or low, depending on the complexity.
- A component in any level L_i is equivalent to a (sub) system of components taken from the level L_{i-1} beneath it.

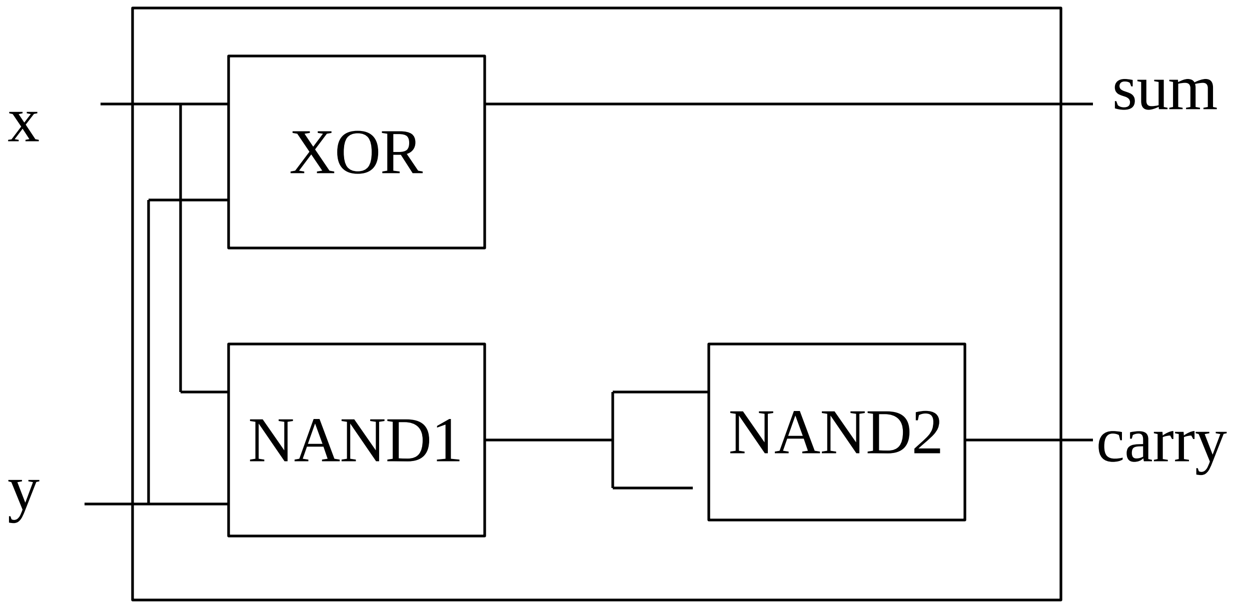


Low level

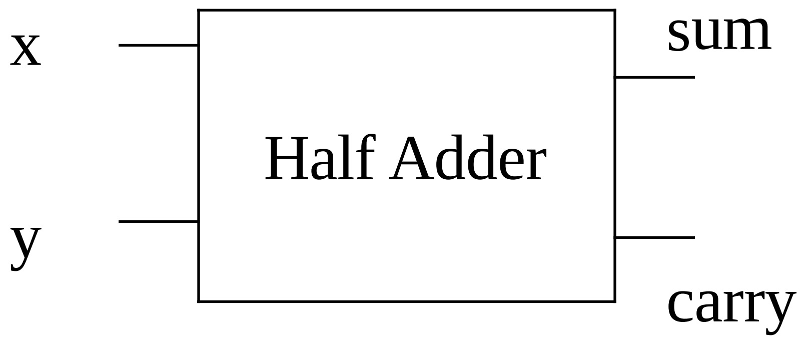


High level

Example of Hierarchical System



Half adder (Low level)



Half adder (High level)

System Hierarchy

- The components of each level of a hierarchical system is self-contained and stable entities. The evolution of a system from simple to complex organization is possible for the existence of stable intermediate structures.

- The design of a complex system is composed of 3 steps:

  1. Specify the processor level structure

  2. Specify the register level structure of each components identified in step 1

  3. Specify the gate level structure of each components identified in step 2

  This design approach is known as top-down.

The Gate Level

- **Combinational Logic:** Read Yourself
- **Flip-flops:** Read Yourself

Sequential Circuits

- A sequential circuit consists of a combinational circuit and a set of flip-flops.
- The combinational logic forms the computational or data-processing part of the circuit.
- The flip-flops store information on the circuit's past behavior. This stored information defines the circuit's **internal state *Y***.
- If the **primary inputs** are ***X*** and the **primary outputs** are ***Z***, then *Z* is a function of both *X* and *Y*, denoted by *Z(X,Y).*
- It is usual to supply a sequential circuit with a precisely controlled clock signal that determines the times at which the flip-flops change state; the resulting circuit is said to be **clocked** or **synchronous**.

Sequential Circuits

- It is usual to supply a sequential circuit with a precisely controlled clock signal that determines the times at which the flip-flops change state; the resulting circuit is said to be **clocked** or **synchronous**.

Serial Adder

- It computes two unsigned binary numbers X_1 and X_2 of arbitrary length $Z = X_1$ *plus* X_2.

- The numbers are supplied serially and the result is also produced serially.

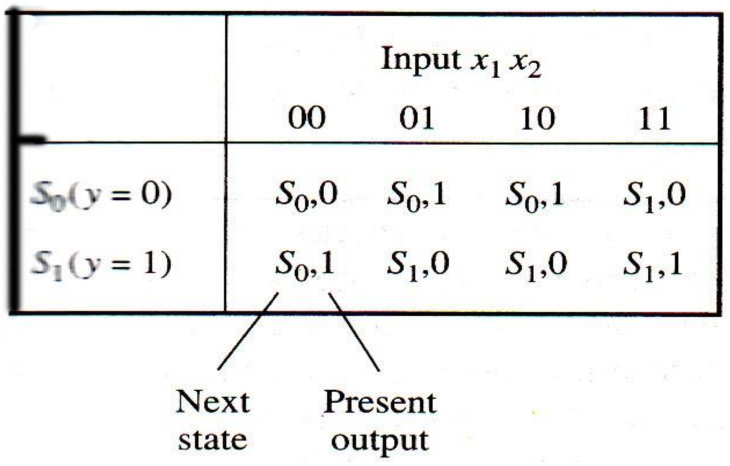
- The output computed in clock cycle i is

  $$c(i)z(i) = x_1(i) \text{ plus } x_2(i) \text{ plus } c(i-1)$$

  where c(i-1) is determined from the adder's present state S(i).

- S_0 means that c(i-1) = 0 and S_1 means that c(i-1) = 1

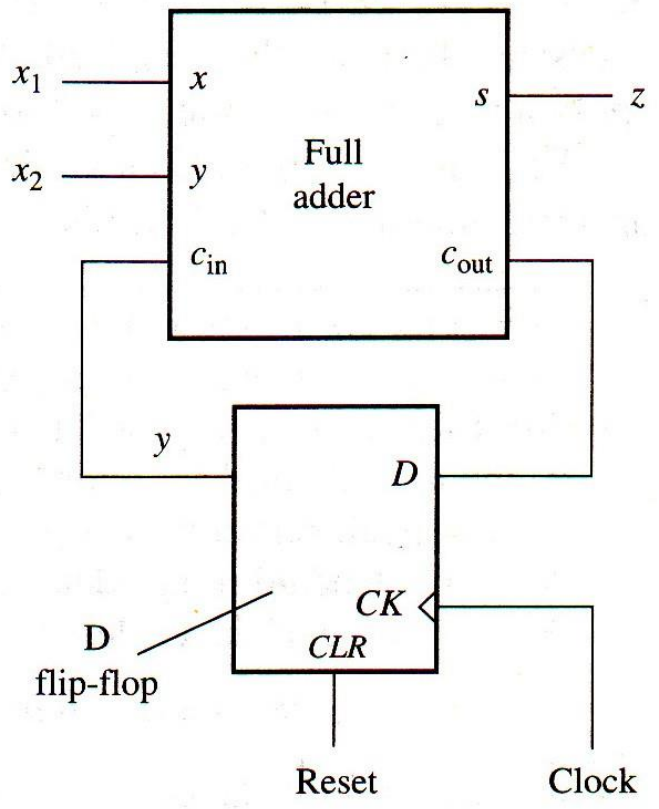
Serial Adder



	Input $x_1 x_2$			
	00	01	10	11
$S_0 (y = 0)$	S_0,0	S_0,1	S_0,1	S_1,0
$S_1 (y = 1)$	S_0,1	S_1,0	S_1,0	S_1,1

Next state
Present output

State Table



Logic Circuit

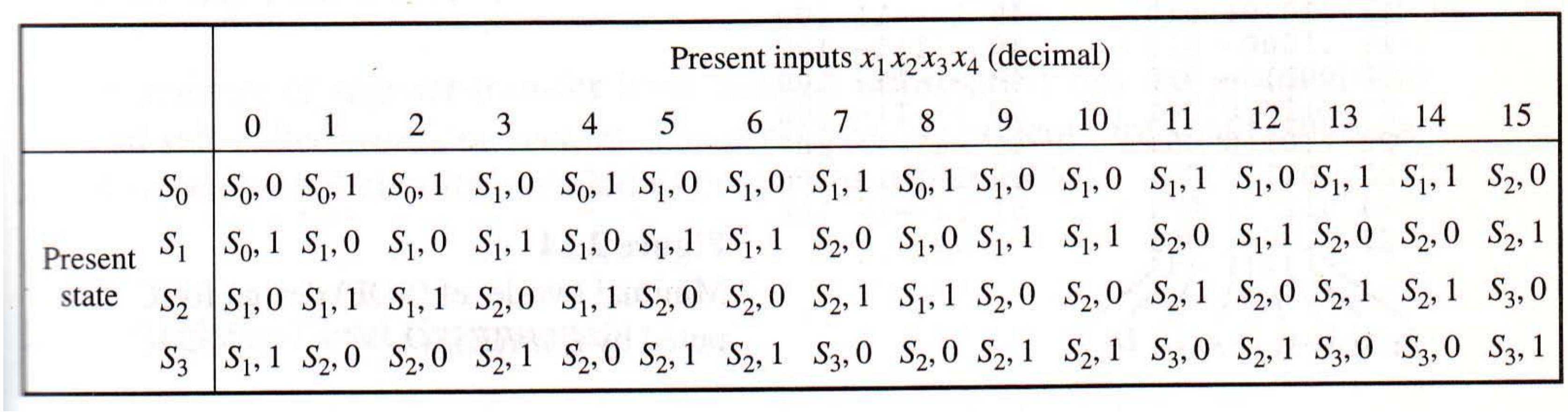
4-bit Stream Serial Adder

- It adds 4 number streams.

- The output computed in clock cycle i is
  sum(i) = c(i)z(i) = x_1(i) plus x_2(i) plus x_3(i) plus x_4(i) plus c(i-1)

- If c(i-1)=0 and each x_j(i)=1, then
  sum(i) = 1 plus 1 plus 1 plus 1 plus 0 = 4 = 100_2, so
  c(i) = 10_2, sum(i) becomes 6 = 110_2, making c(i) = 11_2.
  Finally, c(i-1) = 11_2 makes sum(i) = 111_2, and c(i)= 11_2.
  So, the carry data ranges from 00_2 to 11_2.

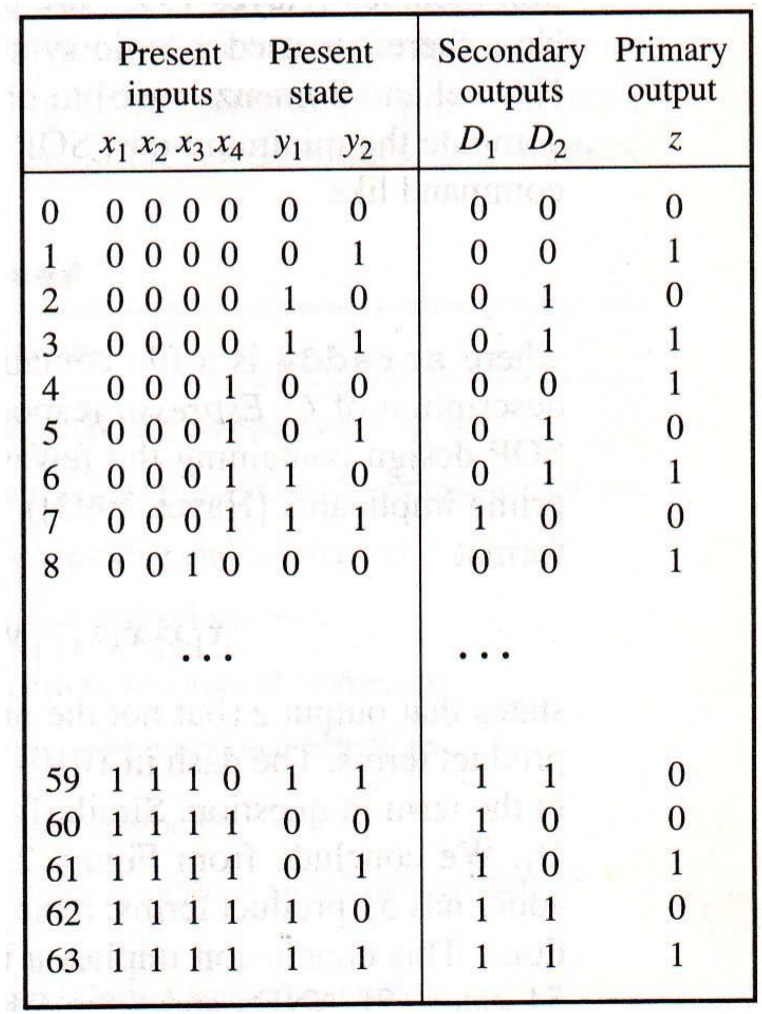
4-bit Stream Serial Adder



		Present inputs $x_1x_2x_3x_4$ (decimal)															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Present state	S_0	S_0, 0	S_0, 1	S_0, 1	S_1, 0	S_0, 1	S_1, 0	S_1, 0	S_1, 1	S_0, 1	S_1, 0	S_1, 0	S_1, 1	S_1, 0	S_1, 1	S_1, 1	S_2, 0
	S_1	S_0, 1	S_1, 0	S_1, 0	S_1, 1	S_1, 0	S_1, 1	S_1, 1	S_2, 0	S_1, 0	S_1, 1	S_1, 1	S_2, 0	S_1, 1	S_2, 0	S_2, 0	S_2, 1
	S_2	S_1, 0	S_1, 1	S_1, 1	S_2, 0	S_1, 1	S_2, 0	S_2, 0	S_2, 1	S_1, 1	S_2, 0	S_2, 0	S_2, 1	S_2, 0	S_2, 1	S_2, 1	S_3, 0
	S_3	S_1, 1	S_2, 0	S_2, 0	S_2, 1	S_2, 0	S_2, 1	S_2, 1	S_3, 0	S_2, 0	S_2, 1	S_2, 1	S_3, 0	S_2, 1	S_3, 0	S_3, 0	S_3, 1

State Table

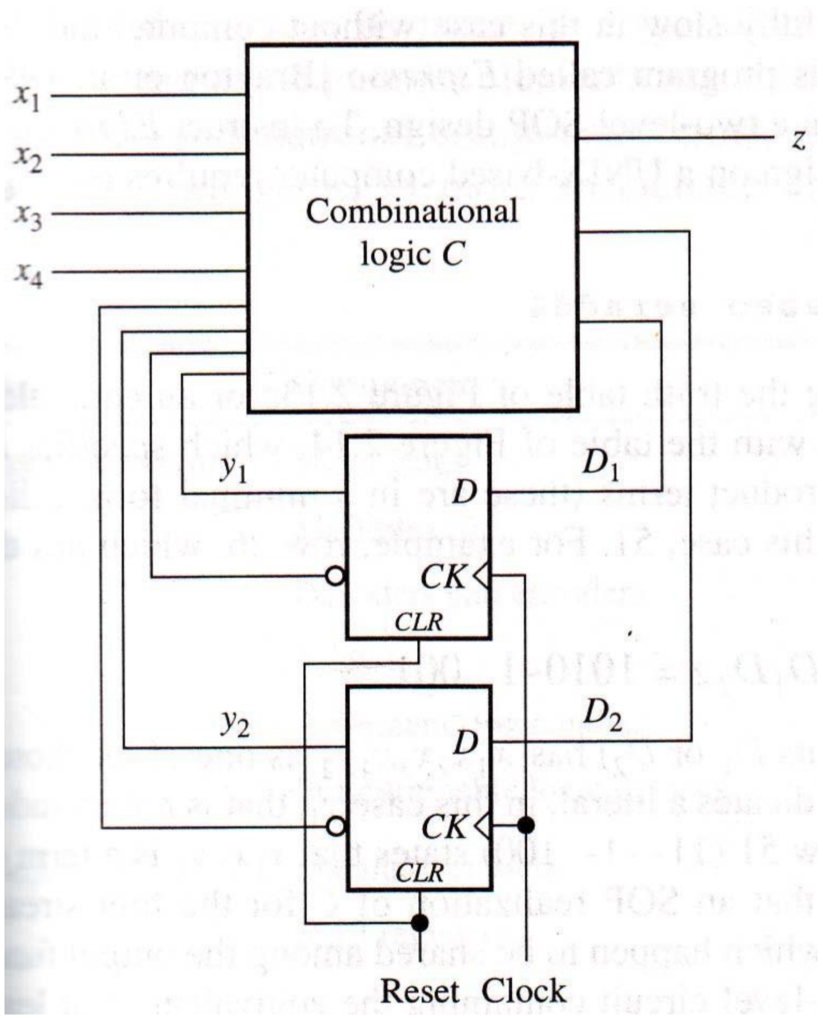
4-bit Stream Serial Adder



	Present inputs				Present state		Secondary outputs		Primary output
	x_1	x_2	x_3	x_4	y_1	y_2	D_1	D_2	z
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	1
2	0	0	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0	1	1
4	0	0	0	1	0	0	0	0	1
5	0	0	0	1	0	1	0	1	0
6	0	0	0	1	1	0	0	1	1
7	0	0	0	1	1	1	1	0	0
8	0	0	1	0	0	0	0	0	1
	…						…		
59	1	1	1	0	1	1	1	1	0
60	1	1	1	1	0	0	1	0	0
61	1	1	1	1	0	1	1	0	1
62	1	1	1	1	1	0	1	1	0
63	1	1	1	1	1	1	1	1	1

Truth Table

4-bit Stream Serial Adder



Overall Structure

The Register Level

- **Word Gates:** Read Yourself
- **Multiplexers:** Read Yourself
- **Decoders:** Read Yourself
- **Encoders:** Read Yourself

4-bit Magnitude Comparator

It has eight input lines, so the truth table has $2^8 = 256$ rows. So a SOP or POS realization is impractical because of the many gates involved.

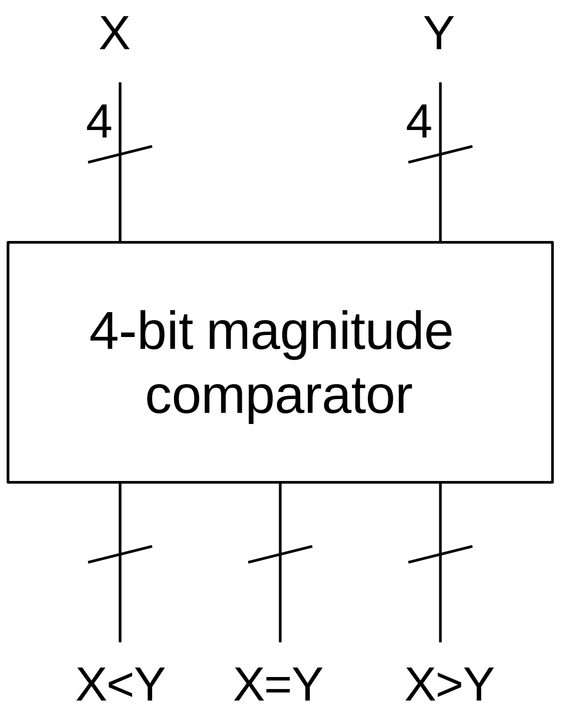
X>Y is equivalent to X-Y>0.

$Y=2^n-1-\overline{Y}$.

So, $X-(2^n-1-\overline{Y})>0$ implies

$$X+\overline{Y}>2^n-1=11\ldots1$$

If the above inequality is satisfied, the C_{out} will be 1.



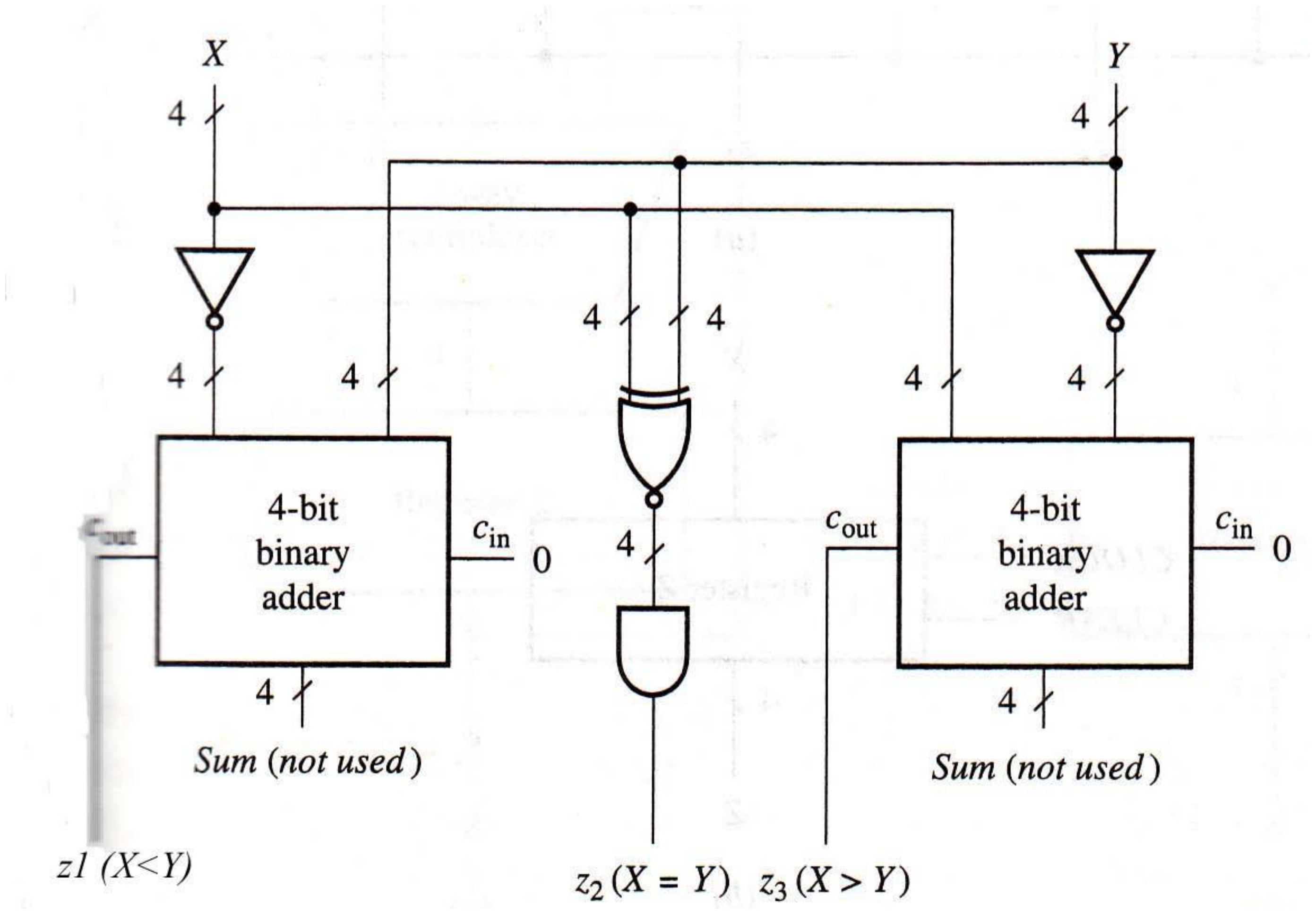
4-bit Magnitude Comparator

The original magnitude test X-Y can be performed as follows:

1. Compute $\overline{Y}$ from Y using an n-bit word inverter.
2. Add X and Y via an n-bit adder and use C_{out} as the primary output. If $C_{out} = 1$, then $X>Y$; if $c_{out} = 0$, then $X \leq Y$

By switching X and Y, $X<Y$ can be generated in exactly the same manner.

4-bit Magnitude Comparator



X
Y
4
4
4
4
4
4
4
4
4
4
4
4-bit binary adder
4-bit binary adder
c_{out}
c_{in}
0
c_{out}
c_{in}
0
Sum (not used)
Sum (not used)
z1 (X<Y)
z_2 (X = Y)
z_3 (X > Y)