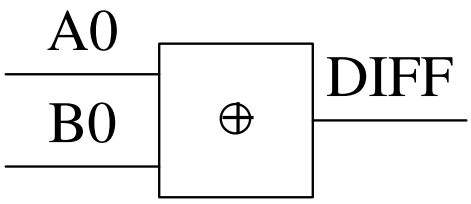
Chapter 7.4. Comparing

A *comparator* compares two binary numbers and indicates whether or not they are equal.

A comparator can interpret its inputs as signed or unsigned numbers.
We will consider unsigned numbers.

For non-equal numbers some comparators produce a relationship (greater or less than).
Such comparators are called *magnitude comparators*.

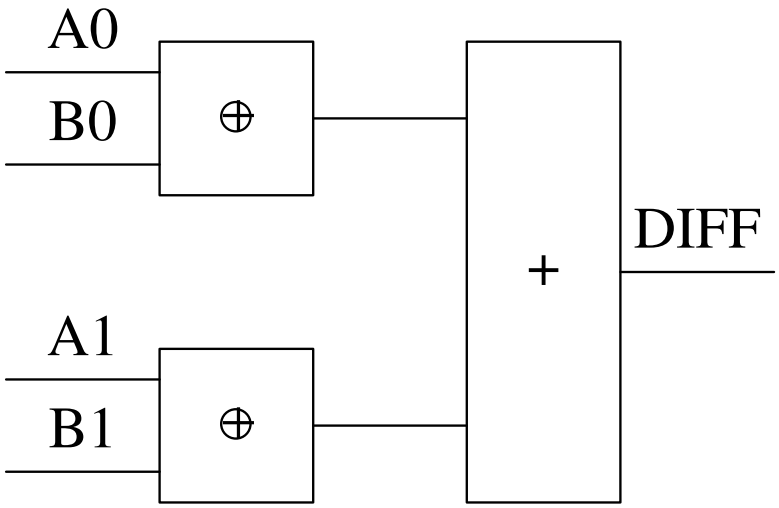
An XOR gate is a 1-bit comparator.



DIFF=0 if A0=B0
DIFF=1 if A0≠B0

A0	B0	DIFF
0	0	0
0	1	1
1	0	1
1	1	0

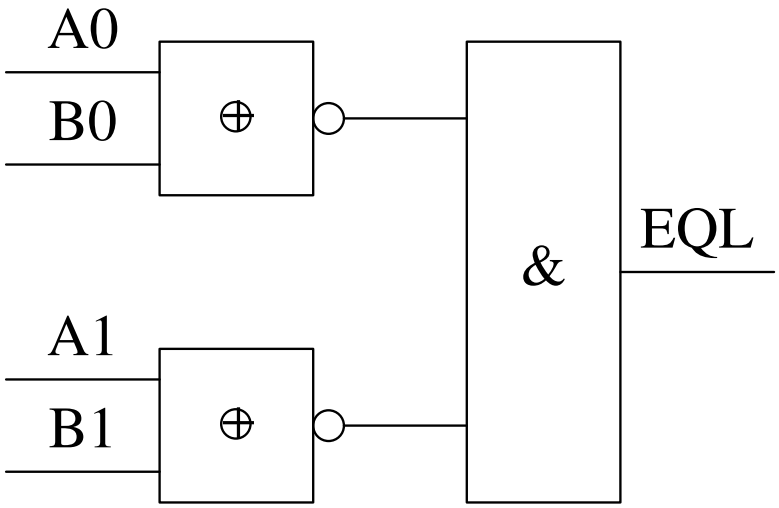
A 2-bit comparator (two 2-bit operands):



DIFF=0 if A0,A1=B0,B1
DIFF=1 if A0,A1≠B0,B1

An *n*-bit comparator requires *n* two-input XOR gates and an *n*-input OR gate.

Another way to design an *n*-bit comparator:



EQL=0 if A0,A1≠B0,B1
EQL=1 if A0,A1=B0,B1

These comparators consider all the input bits in parallel.
This requires *n*-bit OR or AND gates.

Iterative Circuits

An iterative comparator produces a result for the ith bit position before it can produce a result for the $(i + 1)$st bit position.

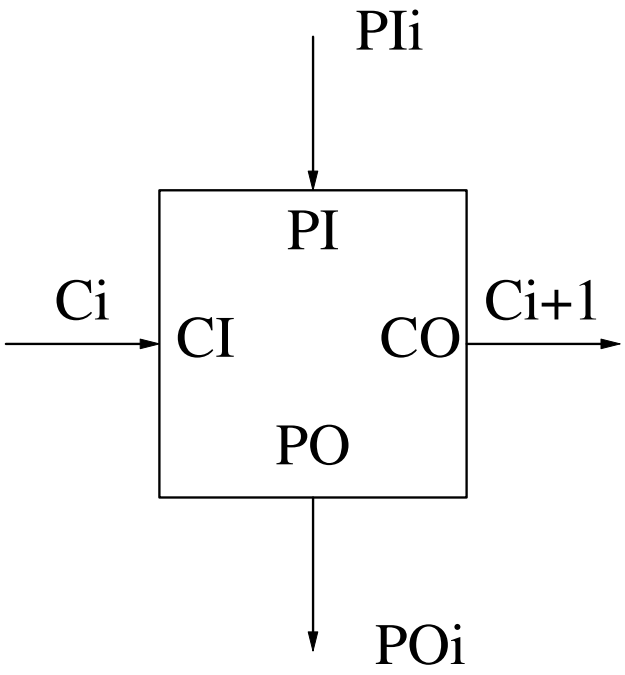
It does not require gates whose numbers of inputs grow with n.

However, its propagation delay is proportionate to n.

An iterative comparator can be obtained by breaking the n-input OR (AND) gate into $n - 1$ two-input OR (AND) gates.

We will see a more general method that applies to other types of units.

A general iterative circuit has the following basic building block.

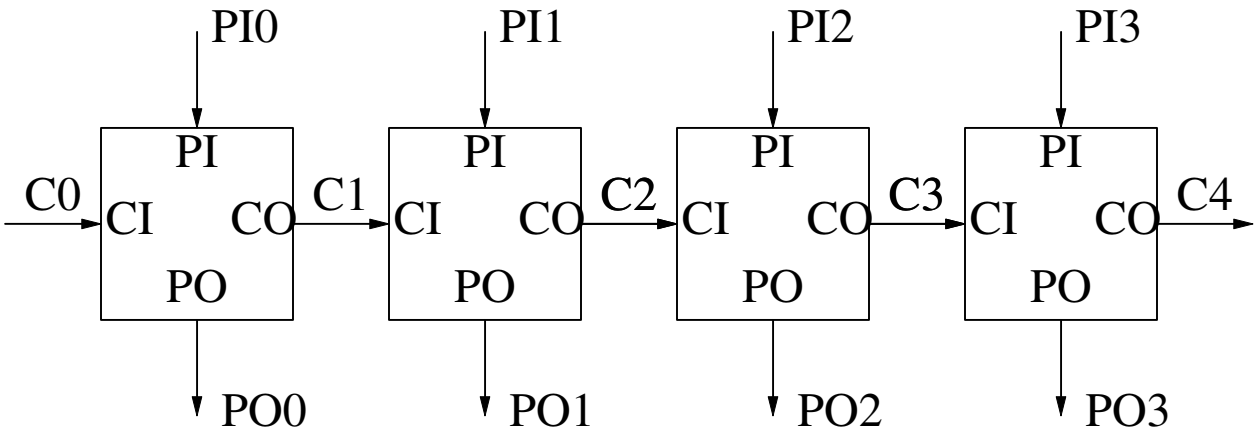


PI and PO are *primary inputs* and *primary outputs*.
The inputs and outputs are connected to them.

CI and CO are *cascading inputs* and *cascading outputs*.
They are used for transferring information from one bit position to the next.

The direction is not unique.
In some cases it is convenient for information to flow from the MSB to the LSB or left to right.
In other cases it is convenient for information to flow from the LSB to the MSB or right to left.

The building block is repeated n times to handle n-bit inputs.

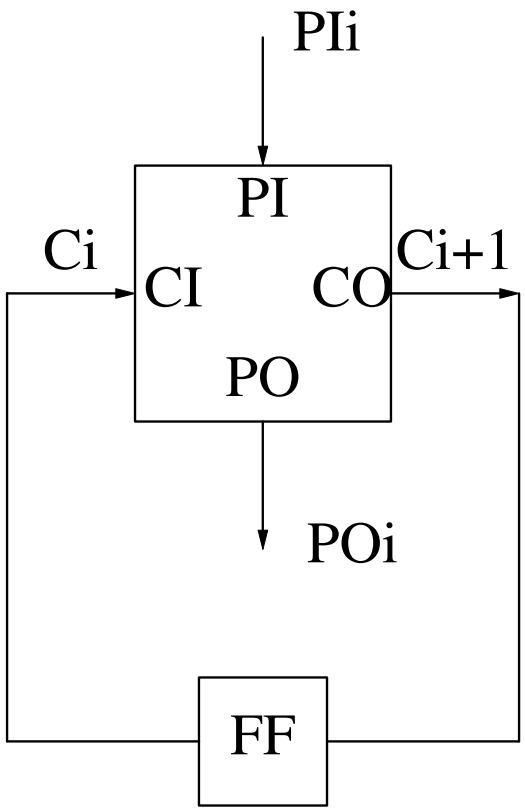


An iterative circuit performs the following iterative algorithm:

1. Assign to C_0 its initial value.
   Assign $i = 0$.
2. Use C_i and PI_i to compute PO_i and C_{i+1}.
3. Assign $i = i + 1$.
4. If $i < n$, go to Step 2.

The index i of the algorithm corresponds to traversing the iterative circuit from left to right.

The index i can also correspond to increasing time in a synchronous sequential circuit with the same building block as the iterative circuit.



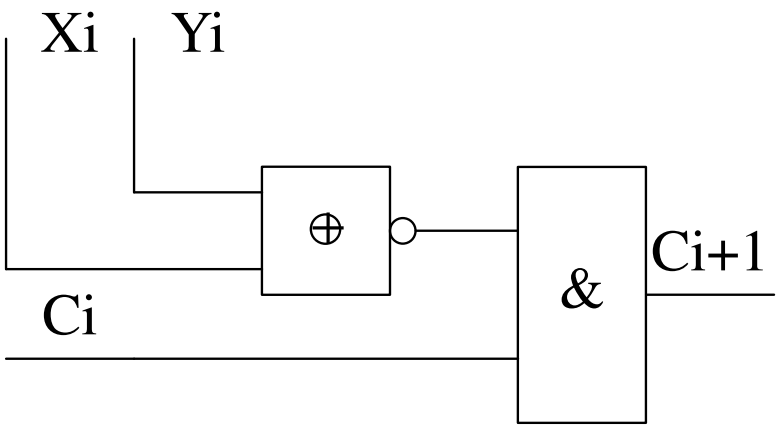
An iterative algorithm for a comparator that produces $C_n = 1$ when two n-bit inputs X and Y are equal:

1. Assign $C_0 = 1$.
   Assign $i = 0$.
2. If $C_i = 1$ and $X_i = Y_i$, assign $C_{i+1} = 1$.
   Else assign $C_{i+1} = 0$.
3. Assign $i = i + 1$.
4. If $i < n$, go to Step 2.

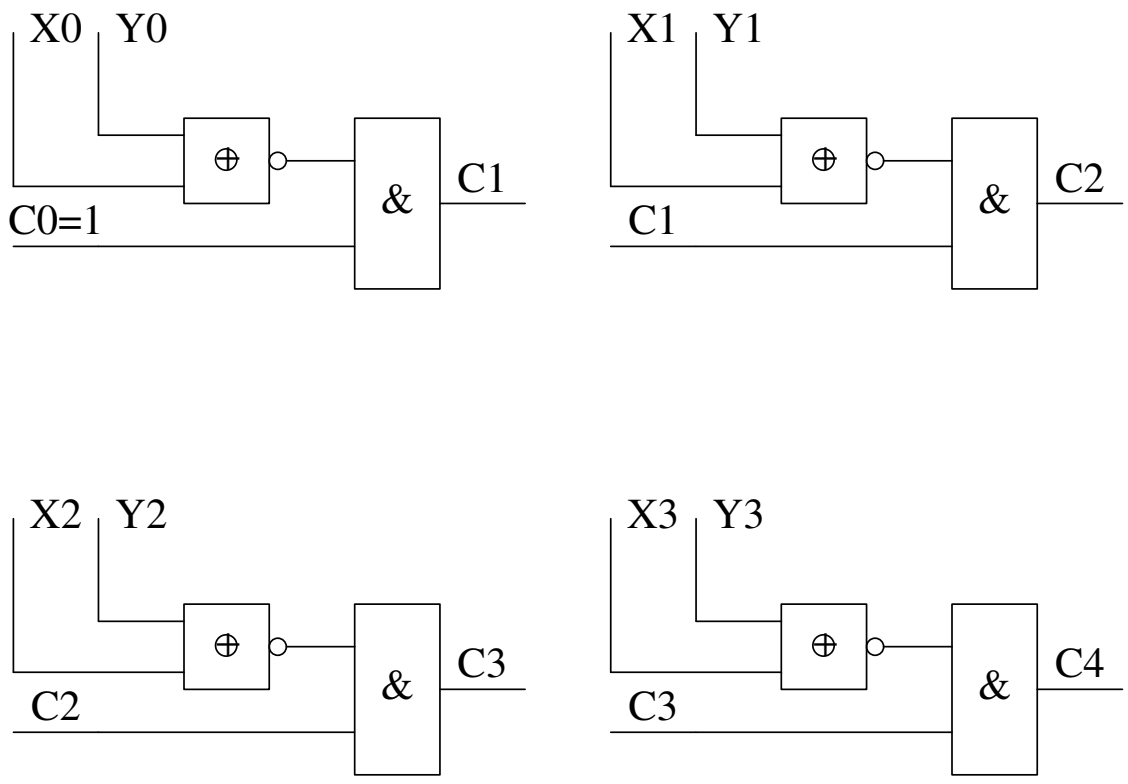
Note that there is no need for primary outputs in this case.
The output is obtained from C_n.

The iterative circuit requires a basic logic block that implements:
If $C_i = 1$ and $X_i = Y_i$, assign $C_{i+1} = 1$.
Else assign $C_{i+1} = 0$.

$$C_{i+1} = C_i \cdot (X_i \oplus Y_i)'$$



Note that an *n*-input AND gate in the parallel comparator is replaced with *n* two-input AND gates in the iterative comparator.



In the previous comparator, the basic building block was a 1-bit comparator (for two 1-bit operands).

In general, it is possible to use a basic building block that implements an m-bit parallel comparator for $m \geq 1$.
The m-bit building blocks can be cascaded to form an n-bit iterative comparator.

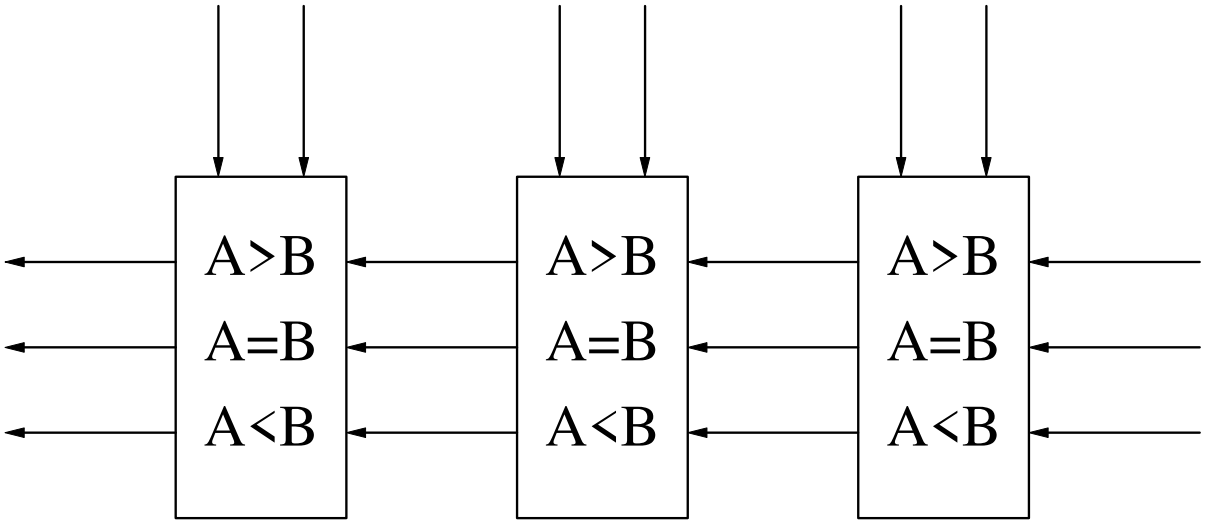
We will see how this is done in the case of a magnitude comparator.

Magnitude Comparator

A four-bit parallel comparator has the following inputs and outputs.
Primary inputs
A_0, A_1, A_2, A_3,
B_0, B_1, B_2, B_3.
Cascading inputs
$CI_{A>B}$, $CI_{A<B}$, $CI_{A=B}$.
Cascading outputs
$CO_{A>B}$, $CO_{A<B}$, $CO_{A=B}$.

The comparator is used to form an iterative circuit by connecting corresponding cascading inputs and outputs.
The connection is such that a less significant bit position produces values for a more significant bit position.

	11	10	9	8	7	6	5	4	3	2	1	0
A	0	1	1	0	1	1	0	1	0	0	1	0
B	1	0	0	1	1	0	1	0	0	0	1	0



Internally the comparator needs to implement the following functions.

Intermediate functions:

$PI_{A>B}$: $A_3A_2A_1A_0 > B_3B_2B_1B_0$

$PI_{A=B}$: $A_3A_2A_1A_0 = B_3B_2B_1B_0$

$PI_{A<B}$: $A_3A_2A_1A_0 < B_3B_2B_1B_0$

Cascading outputs:

$$CO_{A>B} = PI_{A>B} + PI_{A=B} \cdot CI_{A>B}$$

$$CO_{A=B} = PI_{A=B} \cdot CI_{A=B}$$

$$CO_{A<B} = PI_{A<B} + PI_{A=B} \cdot CI_{A<B}$$

The intermediate functions
$PI_{A>B}$, $PI_{A=B}$ and $PI_{A<B}$:

A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	$PI_{A>B}$
1	x	x	x	0	x	x	x	1
v3	1	x	x	v3	0	x	x	1
v3	v2	1	x	v3	v2	0	x	1
.	.	.	.	.	.	.	.	.

$$PI_{A>B} = A_3 \cdot B_3' +$$
$$+(A_3 \oplus B_3)' \cdot A_2 \cdot B_2' +$$
$$+(A_3 \oplus B_3)' \cdot (A_2 \oplus B_2)' \cdot A_1 \cdot B_1' +$$
$$+(A_3 \oplus B_3)' \cdot (A_2 \oplus B_2)' \cdot (A_1 \oplus B_1) \cdot$$
$$\cdot A_0 \cdot B_0'$$

A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	$PI_{A=B}$
v3	v2	v1	v0	v3	v2	v1	v0	1

$$PI_{A=B} = (A_3 \oplus B_3)' \cdot (A_2 \oplus B_2)' \cdot (A_1 \oplus B_1) \cdot \cdot (A_0 \oplus B_0)'$$

A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	$PI_{A<B}$
0	x	x	x	1	x	x	x	1
v3	0	x	x	v3	1	x	x	1
v3	v2	0	x	v3	v2	1	x	1
.	.	.	.	.	.	.	.	.

$$\begin{aligned} PI_{A<B} = {} & A_3' \cdot B_3 + \\ & + (A_3 \oplus B_3)' \cdot A_2' \cdot B_2 + \\ & + (A_3 \oplus B_3)' \cdot (A_2 \oplus B_2)' \cdot A_1' \cdot B_1 + \\ & + (A_3 \oplus B_3)' \cdot (A_2 \oplus B_2)' \cdot (A_1 \oplus B_1) \cdot \\ & \quad \cdot A_0' \cdot B_0 \end{aligned}$$

The XNOR functions need to be implemented only once.

Products of XNOR functions need to be implemented only once.

The resulting circuit is a multi-level circuit.