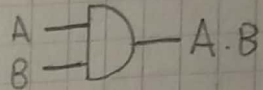
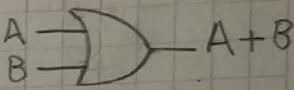
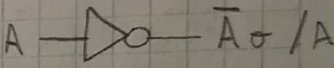
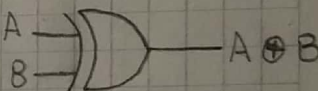


# RESUMEN MODULO 1

## ALGEBRA DE BOOL:

OPERADORES LOGICOS: INTERRELACIONAN VARIABLES LÓGICAS

OPERADOR	SIMBOLO	FUNCIONAMIENTO	COMPUESTA
AND	•	1 si $A=B=1$ 0 OTROS CASOS	
OR	+	1 si $A=1 \vee B=1$ 0 OTROS CASOS	
NOT	/	1 si $A=0$ 0 si $A=1$	
XOR	$\oplus$	1 si $A=1 \oplus B=1$ PERO NO AMBOS	

CONECTIVIDADES: NUMERO FINITO DE FUNCIONES DIFERENTES QUE PUEDEN OBTENERSE CON "N" VARIABLES. SE CALCULAN:

$$\boxed{2^N}$$

1 VARIABLE:  $2^1 = 4$

$$F=1 \vee F=0 \vee F=A \vee F=\bar{A}$$

PROPIEDADES: BASICAS

AND:  $A \cdot 1 = A$

$$A \cdot 0 = 0$$

$$A \cdot A = A$$

$$A \cdot \bar{A} = 0$$

OR:  $A + 1 = 1$

$$A + 0 = A$$

$$A + A = A$$

$$A + \bar{A} = 1$$

NOT:  $\bar{\bar{A}} = A$

$$\bar{\bar{\bar{A}}} = \bar{A}$$

XOR:  $A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$

OTRAS:

$$A + A \cdot B = A$$

$$A \cdot (A + B) = A$$

DISTRIBUTIVA  $\rightarrow$

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

CONMUTATIVA  $\rightarrow$

$$A \cdot B = B \cdot A$$

$$C + D = D + C$$

DE MORGAN  $\rightarrow$

$$A + B = \overline{\bar{A} \cdot \bar{B}}$$

$$A \cdot B = \overline{\bar{A} + \bar{B}}$$

FUNCIONES CANÓNICAS: FUNCIONES QUE CONTIENEN TODAS LAS VARIABLES DE ENTRADA

MINTERMINOS: INTERSECCIÓN ( $\cdot$ ) DE LAS VARIABLES EN JUEGO.

MAXTERMINOS: UNION (+) DE LAS VARIABLES EN JUEGO.

NOTA:

DIAGRAMAS DE KARNAUGH: SIRVEN PARA SIMPLIFICAR FUNCIONES LÓGICAS PARA USAR LA MENOR CANTIDAD DE COMPUERTAS Y VARIABLES POSIBLE.

PASOS:

1. REESCRIBIR LA FUNCIÓN "F" CON TODOS SUS MINTERMINOS COMPLETOS.

Si  $F = A \cdot B$  (VARIABLES A, B, C)  $\rightarrow F = A \cdot B \cdot C + A \cdot B \cdot \bar{C}$

2. ESCRIBIR LA TABLA DE KARNAUGH PONIENDO 1 EN LOS CASILLEROS DE CADA MINTERMINO

		$\bar{B}\bar{C}$	$\bar{B}C$	$BC$	$B\bar{C}$	
$\bar{A}$	$BC$	00	01	11	10	
0		0	1	3	2	$\rightarrow$ NRO DE MINTERMINO
A	$BC$	1	4	5	7	6
1						

$\downarrow$  EN ESTA FILA, A SIEMPRE ES 1.  
 $\downarrow$  ABC  $\downarrow$  A=1, B=1, C=1  
 $\rightarrow AB\bar{C} \rightarrow A=1, B=1, C=0$

LA TABLA ESTA CONSTRUIDA DE FORMA QUE ENTRE FILAS Y COLUMNAS NO SE CAMBIE MAS DE UN DÍGITO (EN BINARIO).

3. AGRUPAR LOS 1 DE LA TABLA FORMANDO RECTÁNGULOS (DE TAMAÑO DE POTENCIAS DE 2 (1, 2, 4, 8, 16) CASILLEROS) O CUADRADOS, Y SIMPLIFICAR LAS VARIABLES QUE NO SE MANTIENEN IGUAL EN LAS FILAS O COLUMNAS



$F = A.B \rightarrow$  LA EXPRESION ORIGINAL EN ESTE CASO ES LA MAS SIMPLIFICADA POSIBLE.

4. SI APARECE UNA X EN UN DIAGRAMA DE KARNUGH, SIGNIFICA QUE NO SE ESPERA QUE SE DE ESE MINTERMINO EN LA FUNCION, POR LO QUE PUEDE TOMAR CUALQUIER VALOR (1 O 0) SEGUN CONVENGA.

$X = 1 \text{ o } 0$  SEGUN SIMPLIFIQUE O NO A F

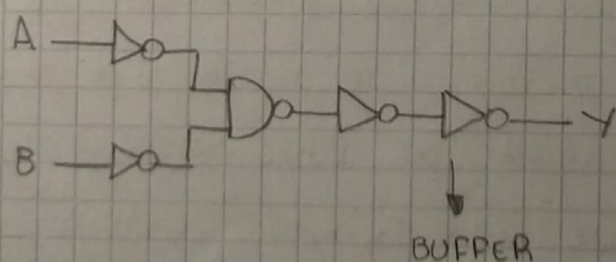
### RIESGOS DE TEMPORIZACION:

**RIESGO ESTÁTICO:** ES AQUEL QUE PUEDE HACER QUE UNA SALIDA VAYA TEMPORALMENTE A UN ESTADO DIFERENTE AL DEFINITIVO.

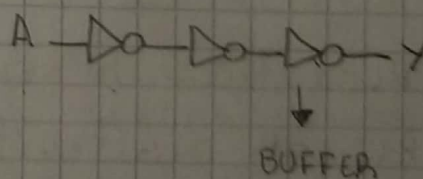
**RIESGO DINAMICO:** RESPUESTA DE UNA SALIDA LA CUAL CAMBIA DE ESTADO REPETIDAS VECES AL GENERARSE UN SIMPLE CAMBIO A SU ENTRADA.

### CIRCUITOS COMBINATORIOS

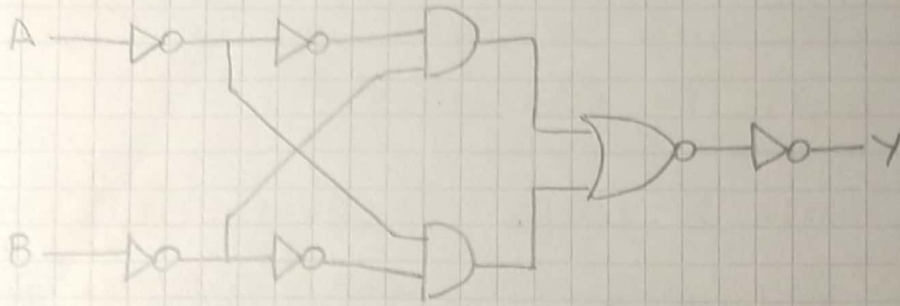
#### COMPUERTA OR:



#### COMPUERTA NOT:

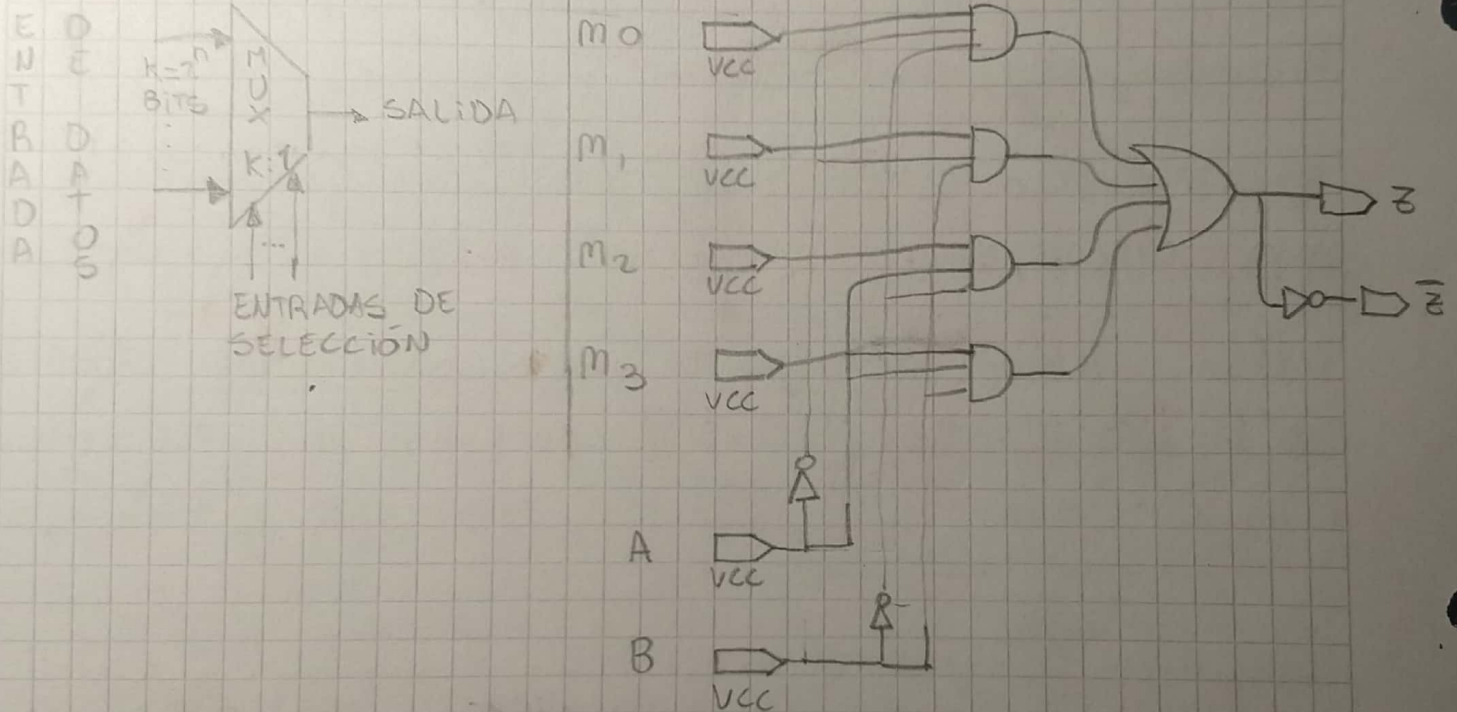


## COMPUERTA XOR:



## MULTIPLEXORES:

PUEDEN SER ANALÓGICOS (LLAVES SELECTORAS) O DIGITALES



FUNCIONAMIENTO: EL MINITERMINO QUE ESTÉ PRENDIDO SE GENERARÁ A LA SALIDA. EJ: Si  $m_0 = 1$  y  $m_1, m_2$  y  $m_3 = 0$ ,  $z = \bar{A}\bar{B}$ .

UN MULTIPLEXOR ES UN GENERADOR DE FUNCIONES LÓGICAS.

UN MULTIPLEXOR DIGITAL TIENE UNA ENTRADA EXTRA (ENABLE) QUE SI ESTA APAGADA, GENERA AUTOMÁTICAMENTE UN 0 A LA SALIDA, Y PERMITE EL FUNCIONAMIENTO NORMAL PRENDIDO

## DEMULTIPLEXOR:



FUNCIONA A LA INVERSA DE UN MULTIPLEXOR, ES DECIR, 1 ENTRADA Y MUCHAS SALIDAS.

**BARREL SHIFTER:** CIRCUITOS BASADOS EN MULTIPLEXORES QUE SIRVEN PARA ROTAR O DESPLAZAR NUMEROS. CON 2 ENTRADAS SE SELECCIONA CUANTO SE ROTA UN NUMERO (EN BINARIO)

ENTRADAS		$S_1$	$S_0$	$Y_3$	$Y_2$	$Y_1$	$Y_0$	→ SALIDAS
		0	0	$D_3$	$D_2$	$D_1$	$D_0$	
		0	1	$D_2$	$D_1$	$D_0$	$D_3$	} DIGITOS
		1	0	$D_1$	$D_0$	$D_3$	$D_2$	
		1	1	$D_0$	$D_3$	$D_2$	$D_1$	

**COMPARADOR DE MAGNITUD:** COMPARA 2 NUMEROS DE 8 BITS C/U. SI SON IGUALES PONE LA SALIDA EN BAJO, CASO CONTRARIO, EN ALTO. ES EL TÍPICO EJEMPLO DEL USO DE COMPUERTAS XNOR.

**DECODIFICADOR DE 7 SEGMENTOS:** GENERA UN NUMERO EN UNA PANTALLA SEGUN 7 ENTRADAS QUE ENCIENDEN UNA BARRA (LUZ)

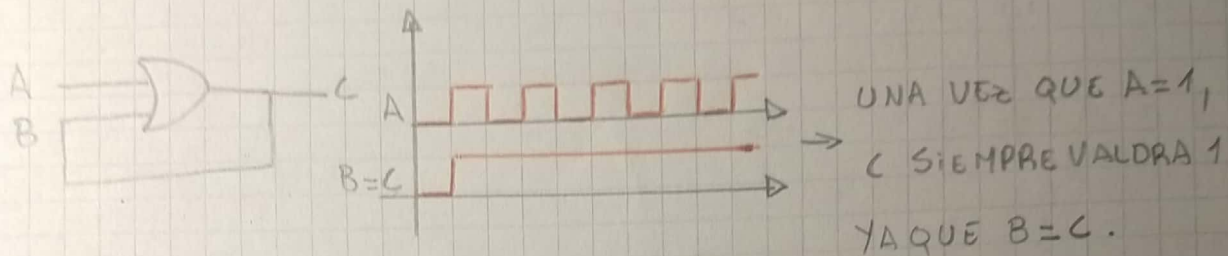
$\begin{matrix} a \\ F/g/b \\ e/_/c \\ d \end{matrix}$

→ SI TODAS LAS ENTRADAS A...G SON 1, SE VE EL NUMERO 8 EN LA PANTALLA



**FLIP FLOPS** COMPONENTES QUE DEPENDEN DE SU SALIDA ACTUAL PARA DETERMINAR SUS ENTRADAS Y POR LO TANTO SUS SALIDAS FUTURAS, ES DECIR, POSEEN MEMORIA.

TÍPICO COMPONENTE CON MEMORIA:



CLASIFICACION DE F.F.:

SEGUN SINCRONISMO:

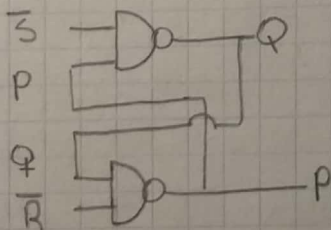
- ASINCRONICOS (NO HAY ENTRADA DE RELOJ)
- SINCRONICOS → LATCHES: CAMBIAN AL DETECTAR EL NIVEL DEL RELOJ
- FLIP-FLOPS: CAMBIAN AL DETECTAR UN FLANCO DE BAJADA O SUBIDA DEL RELOJ DE ENTRADA

SEGUN FUNCIÓN:

- ASINCRONICOS →  $\bar{S} \bar{R}$   
→ RS

- SINCRONICOS → "D"  
→ "T"  
→ "JK"

$\bar{S} \bar{R}$



$\bar{S}$	$\bar{R}$	$Q(n+1)$	$1/Q(n+1)$
0	0	PROHIBIDO	PROHIBIDO
0	1	1	0
1	0	0	1
1	1	$Q(n)$	$Q(n)$

$\overline{S}\overline{R} = 00$  DA  $Q/\overline{Q} = 11$  LO QUE NO ES ADMISIBLE.

EL ESTADO  $\overline{S}\overline{R} = 11$  ES LA CAPACIDAD DE MEMORIA

RS INVERSO A  $\overline{S}\overline{R}$ , BASADO EN COMPUERTAS NOR

R	S	$Q(n+1)$	$\overline{Q}(n+1)$
0	0	$Q(n)$	$\overline{Q}(n)$
0	1	1	0
1	0	0	1
1	1	PROHIBIDO	PROHIBIDO

$RS = 11$  DA  $Q/\overline{Q} = 00$  LO QUE NO ES ADMISIBLE

JK:

CLK	J	K	$Q_{n+1}$	$\overline{Q}_{n+1}$
↑	0	0	$Q_n$	$\overline{Q}_n$
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	$\overline{Q}_n$	$Q_n$
0,1,↓	X	X	$Q_n$	$Q_n$

↓  
CUANDO EL CLK SUBE,  $Q_{n+1}$  CAMBIA

D:

CLK	D	$Q_{n+1}$	$\overline{Q}_{n+1}$
↑	0	0	1
↑	1	1	0
0,1,↓	X	$Q_n$	$\overline{Q}_n$

T:

CLK	T	$Q_{n+1}$	$\overline{Q}_{n+1}$
↑	0	$Q_n$	$\overline{Q}_n$
↑	1	$\overline{Q}_n$	$Q_n$
0,1,↓	X	$Q_n$	$\overline{Q}_n$

TAMBIEN EXISTEN FF QUE RESPONDEN A FLANCOS DESCENDENTES DE CLK.



TIEMPO DE SET UP: TIEMPO EN QUE LA ENTRADA D DEBE ESTAR ESTABLE ANTES QUE LLEGUE EL FLANCO DEL CLK. SI NO SE ESPERA, SE PUEDE TOMAR MAL EL DATO.

TIEMPO DE HOLD: TIEMPO MINIMO QUE LA ENTRADA DEBE MANTENER SU VALOR LUEGO QUE HAYA PASADO EL FLANCO ACTIVO DE RELOS.

ENTRADAS EXTRA:

$\overline{1SD}$  (SET) y  $\overline{1RD}$  (RESET): - Si  $\overline{1SD} = 0$  y  $\overline{1RD} = 1 \rightarrow Q = 1$

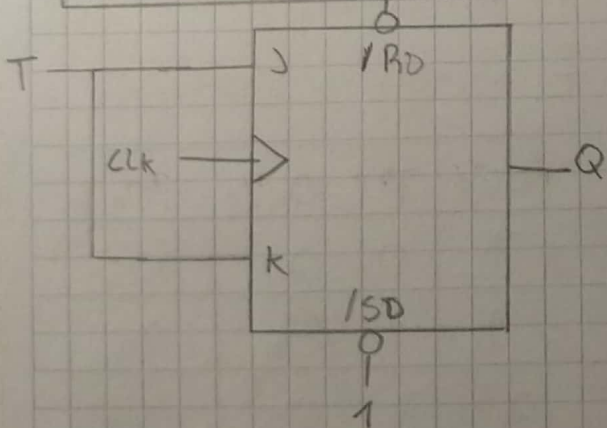
- Si  $\overline{1SD} = 1$  y  $\overline{1RD} = 0 \rightarrow Q = 0$

- Si  $\overline{1SD} = 1$  y  $\overline{1RD} = 1 \rightarrow Q \rightarrow$  FUNCIONA  
MIENTO NORMAL

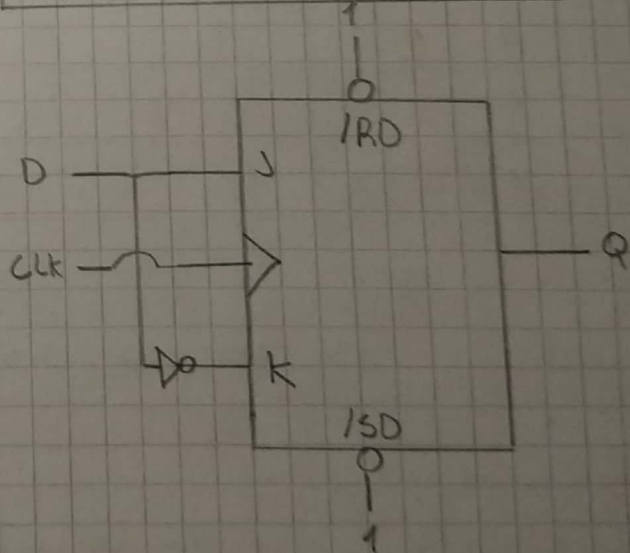
- Si  $\overline{1SD} = 0$  y  $\overline{1RD} = 0 \rightarrow$  NO HAY  
GARANTIA DE NADA

CONSTRUCCIÓN DE FF A PARTIR DE OTROS FF:

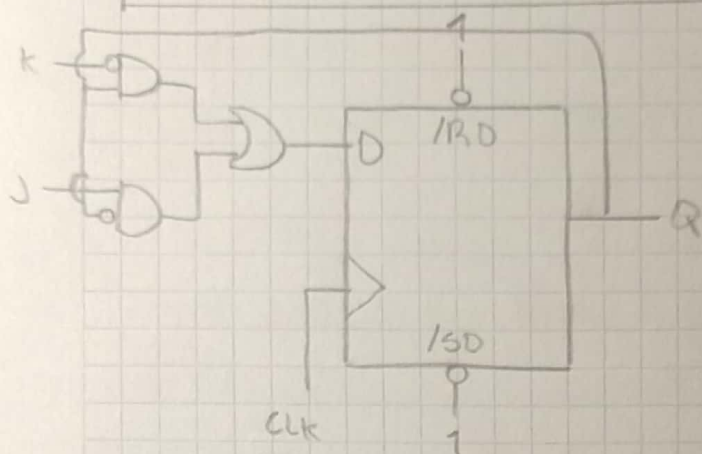
T A PARTIR DE JK:



D A PARTIR DE JK:



## JK A PARTIR DE FF TIPO D



## VHDL

VERY HIGH SPEED INTEGRATED CIRCUITS HARDWARE DESCRIPTION LANGUAGE

### VENTAJAS:

- POSIBILITA LA DOCUMENTACION DE LOS PROYECTOS Y SU REUTILIZACION.
- PERMITE MODALIZAR EL CONCEPTO DE TIEMPO.
- SE EJECUTA DE FORMA CONCURRENTE NORMALMENTE, PERO SE PUEDE EJECUTAR SECUENCIALMENTE DENTRO DE UN PROCESS
- ES PORTABLE

### TIPOS DE DATOS:

ASIGNACION A CTE

CONSTANTES (RETARDO := 3ns)

VARIABLES: LOCALES EN UN PROCESO. SE EJECUTAN SECUENCIALMENTE

(A <= CLK) ASIGNACION A VARIABLE

SEÑALES: SON DE USO GLOBAL, PERO SE ASIGNAN AL FINAL DE UN PROCESS. (SIGNAL A: STD LOGIC (<= 0))

SE UTILIZAN LOS OPERADORES ANTERIORMENTE NOMBRADOS  
(AND, OR, NOT, XOR, NAND, ETC)

PARTES DE UN PROGRAMA:

### 1) LIBRERIAS:

BASICA: library ieee; → LIBRERIA

Use ieee.std\_logic\_1164.all; → PAQUETE

### 2) ENTITY: ESPECIFICA LAS ENTRADAS, SALIDAS, Y TODA AQUELLA INFORMACION DE ELLAS.

```
ENTITY mux IS
    Port (
        CLK : in std_logic;
        Reset : in std_logic_vector (3 downto 0);
    );
END ENTITY mux;
```

↑ VARIABLE LOGICA

↑ VECTOR DE VARIABLES LOGICAS

### 3) ARCHITECTURE: DESCRIBE LA FUNCIONALIDAD DE LA ENTIDAD. LAS SEÑALES SE DEFINEN AQUI.

```
Architecture JK OF mux IS
    Signal qaux : std_logic := '1';
    Begin
        Process (CLK)
        IF CLK = '1' THEN
            qaux <= CLK;
        ELSE
            qaux <= 0;
        END IF;
        END PROCESS;
    End JK;
```

→ LISTA DE SENSIBILIDAD (VARIABLES LEIDAS)

SE EJECUTA DE FORMA SECUENCIAL, UTILIZANDO SENTENCIAS TIPO IF, FOR O CASE

NOTA



DENTRO DE ARCHITECTURE, LUEGO DE LA SENTENCIA BEGIN,  
SE PUEDE TRABAJAR DE DOS FORMAS POSIBLES: SECUENCIALMEN  
TE O CONCURRENTEMENTE:

#### - CONCURRENTE:

TODO LO QUE SE ESCRIBA LUEGO DEL BEGIN, FUERA DE  
UN PROCESS, SE EJECUTARÁ AL MISMO TIEMPO. SE UTILIZAN  
SENTENCIAS COMO EL WHEN Y EL WITH SELECT

EJ 1: SALIDA <= ENTRADA WHEN ENTRADA = '1' ELSE  
SALIDA <= 0;

EJ 2: WITH ENTRADA SELECT  
SALIDA <= "110" WHEN "000",  
"111" WHEN "001",  
"000" WHEN "010",  
"001" WHEN OTHERS;

#### - SECUENCIAL:

SE ESCRIBEN SENTENCIAS DENTRO DE UN PROCESS QUE  
TIENEN ORDEN DE PRIORIDAD DESDE ARRIBA HACIA ABAJO. SE  
UTILIZAN LAS SENTENCIAS: IF, THEN, ELIF, ELSE, CASE,  
LOOP.

EJ 1: IF A = '1' THEN B = '1';  
ELIF A = '0' THEN B = 0;  
ELSE B = 2;  
END IF;

NOTA

ED 2 :

```
FOR I IN 0 TO 8 LOOP
  IF (A=I) THEN
    Z(I) <= '1';
  END IF;
END LOOP;
```

PREVIAMENTE DEBEMOS COPIAR LA ENTITY DEL JK ORIGINAL  
Y PEGARLA EN EL PRINCIPIO (ANTES DE BEGIN) DE LA ARQUITECTURA  
CAMBIANDO 'ENTITY' POR 'COMPONENT'.

## CONTADORES:

### CLASIFICACIONES:

#### SEGÚN COMPORTAMIENTO CON CLOCK:

- ASINCRONICOS
- SINCRONICOS

#### SEGÚN EL FORMATO DE SALIDA DEL CONTEO:

- BINARIO
- BCD
- ARBITRARIO

#### SEGÚN DE CONTEO:

- ASCENDENTE O PROGRESIVO
- DESCENDENTE O REGRESIVO

**ASINCRÓNICO:** UN SOLO CLOCK CONECTADO A 'N' FLIP FLOPS TIPO T, DONDE 'N' ES EL NUMERO DE BITS DEL CONTADOR. TIENE PROBLEMAS DE RETARDO ACUMULADO. TIENE BAJA VELOCIDAD DE RESPUESTA

**SINCRÓNICO:** UN CLOCK POR CADA FF TIPO T, LO QUE CASI ELIMINA EL RETARDO ACUMULADO AL PASAR DE UN FF A OTRO. TIENE MAYOR VELOCIDAD DE RESPUESTA QUE EL ASINCRONICO, PERO ES RELATIVAMENTE MAS COMPLEJO CIRCUITALMENTE.

NOTA



MAXIMA FRECUENCIA DE TRABAJO: SI EL CLOCK EMPLEADO TRABAJA A FRECUENCIAS MAYORES DE ESTA, EL FABRICANTE NO GARANTIZA LA FUNCIONALIDAD PLENA DEL DISPOSITIVO UTILIZADO

$$F_{MAX} = \frac{1}{T_{SETUP} + T_{CLK \rightarrow Q} + T_{COMPUERTAS\ EXTRA}}$$

TIEMPO DE SKEW: DIFERENCIA ENTRE TIEMPO DE RETARDO MAS LENTO Y MAS RAPIDO

PARA HACER QUE UN CONTADOR CUENTE HASTA DETERMINADO NUMERO, SE CONECTA UN NAND A LAS SALIDAS DE CADA BIT, Y ESTA VA AL RESET DE CADA FLIP FLOP.

CONTADOR ANILLO: TIENE UN CICLO DE  $N$  CONTEOS:  
0001 - 0010 - 0100 - 1000.

CONTADOR JOHNSON: TIENE UN CICLO DE  $2N$  CONTEOS

0000 - 0001 - 0011 - 0111 - 1111 - 1110 -  
- 1100 - 1000 - 0000