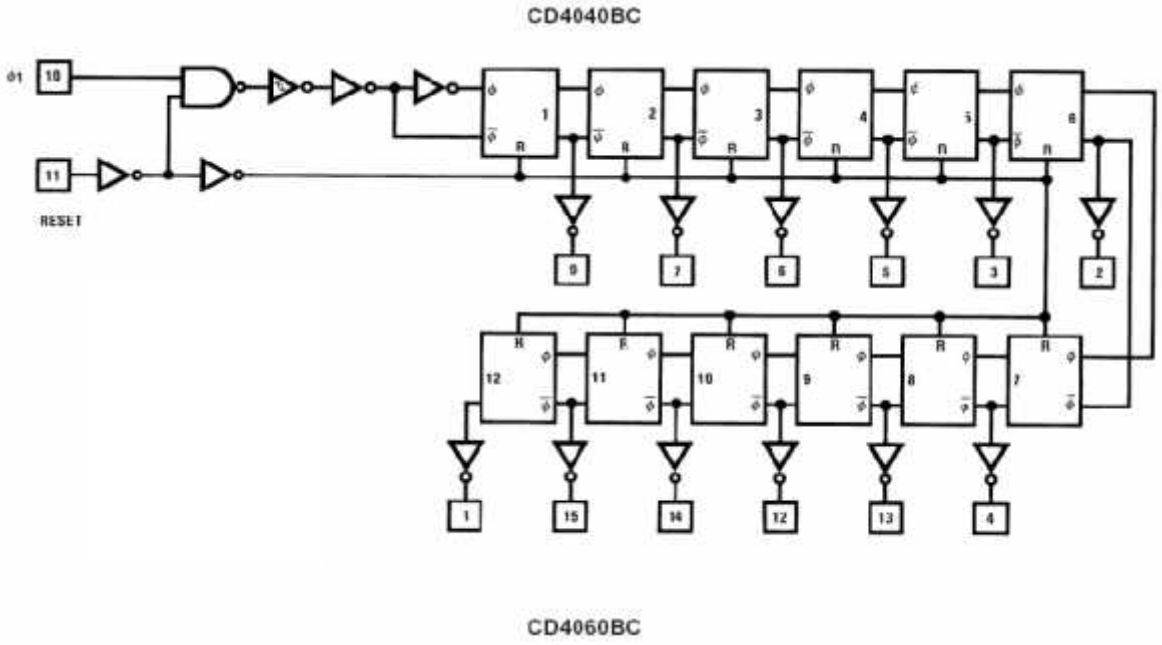
Módulo 2:
Circuitos contadores

Contenidos del módulo 2

- Qué es un contador?
- Qué significa “contar”?
- Qué sucede si lo que se cuenta es una señal donde lo importante es la frecuencia y no la cantidad de pulsos?
- Alternativas de implementación de circuitos contadores sincrónicos (Johnson, LFSR, sincrónicos, Gray).
- Ventajas y desventajas de cada uno.
- Diseño de macrofunciones para el diseño de LFSRs.

Contadores asincrónicos

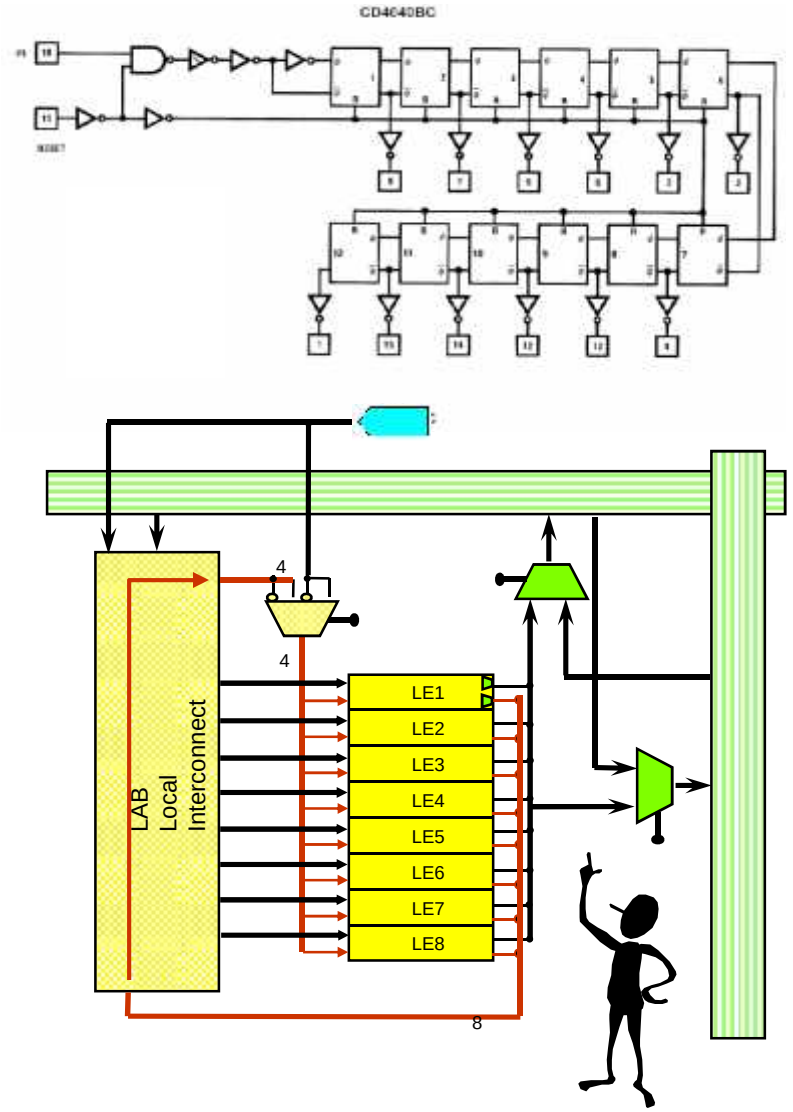
- Llamados también Ripple Carry Binary Counters, un ejemplo clásico es el CD4040BC, de 12 etapas
- Es una sucesión de 12 flipflops T donde el reloj de cada etapa proviene de la salida del flipflop previo
- Cada etapa conmuta entonces a la mitad de frecuencia de la etapa previa



Analizar la relación temporal entre las salidas
Si se lo desea usar como contador, cuál es el caso peor desde que ingresa la señal de reloj hasta que todas las salidas están estables?
Qué ventajas tiene este circuito?

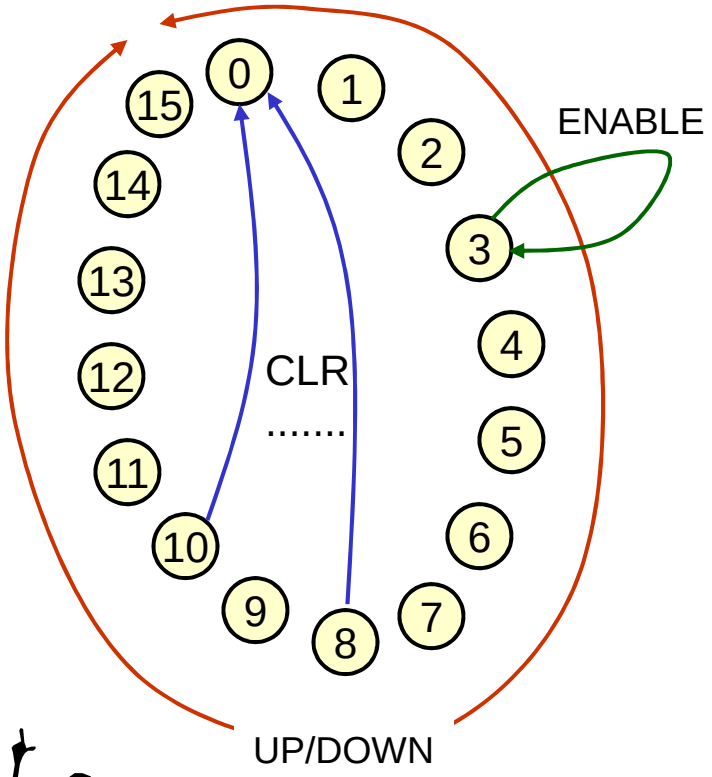
Limitaciones de implementación de contadores asincrónicos en FPGAs

- En una FPGA el camino de llegada de la señal de reloj a una macrocelda suele estar asociado a redes globales de distribución, de número limitado, con mínimo skew
- En ciertas FPGAs existen caminos de reloj alternativos, usando escasas y limitadas vias de ruteado convencional, que son lentas
- Un contador asincrónico tiene tantas señales de reloj como etapas o flipflops, con lo que el retardo de estos caminos se suma
- En el caso de la familia FLEX de ALTERA, por ejemplo, cada LAB de 8 macroceldas sólo posee dos posibles caminos de reloj, lo que limita la cantidad de etapas de un contador asincrónico por cada LAB
- Estos retardos generan el efecto “Ripple Carry”



Contadores sincrónicos binarios

- Son circuitos sincrónicos (un único reloj común a todos los biestables)
- Respetan la definición estándar de una FSM: en función del estado actual y las señales de control calcula el estado futuro, al que conmuta en el siguiente flanco activo de reloj
- Sus salidas presentan valores compatibles con la representación de números binarios consecutivos

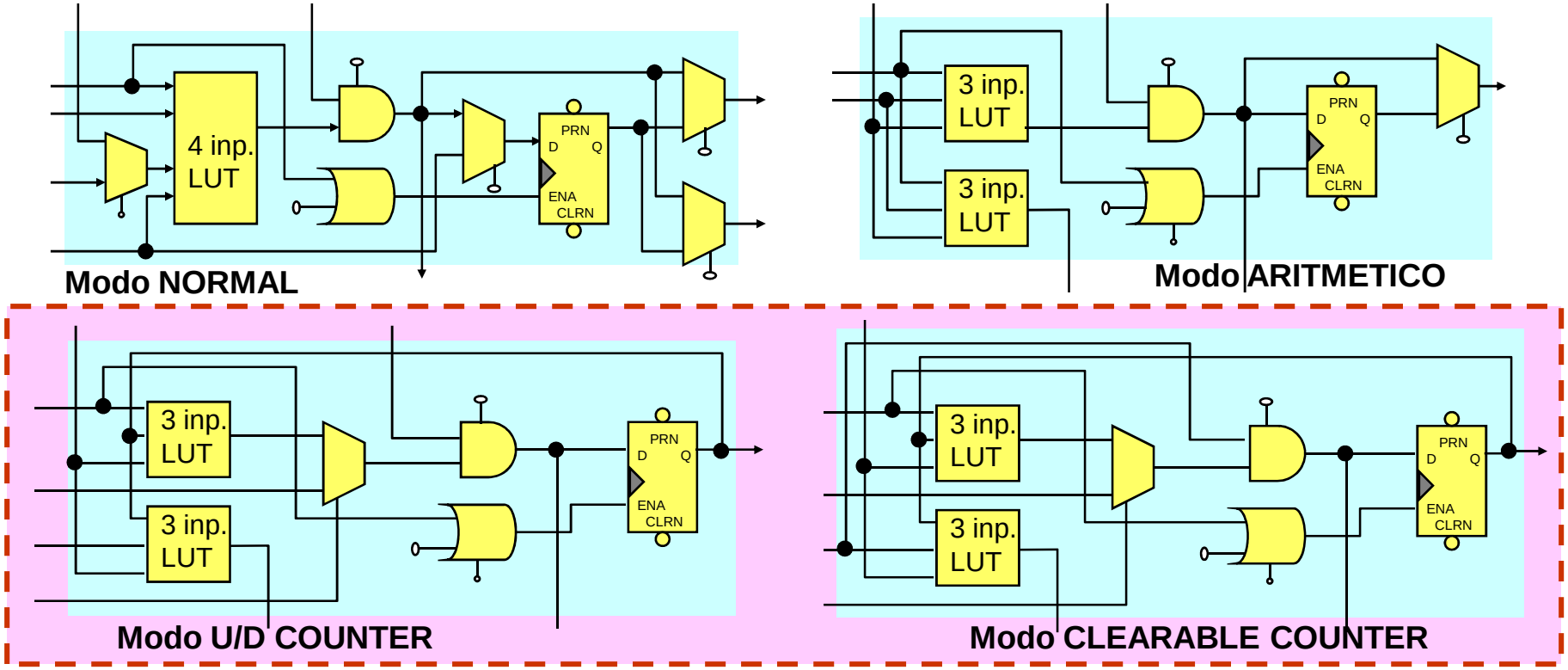


Analizar ventajas y desventajas

Imaginar cómo será la respuesta en frecuencia en función de la cantidad de etapas

Modos de operación para contadores sincrónicos en FPGAs

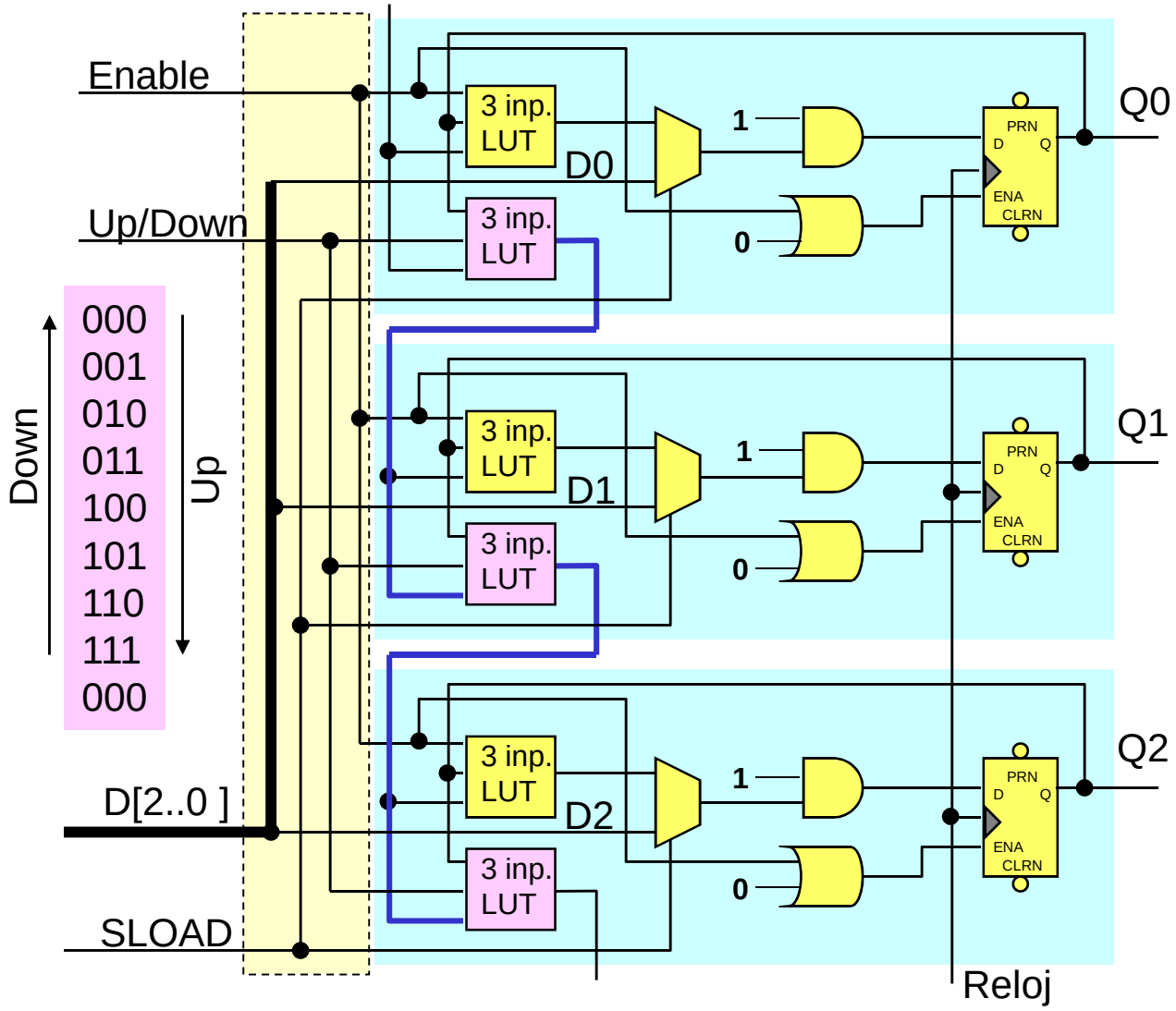
- Los elementos lógicos de las FLEX10K pueden operar en cuatro distintos modos, y cada modo usa los recursos del LE en modo diferente
- El software detecta automáticamente la función lógica y configura al LE



Contador binario U/D programable usando Carry Chain

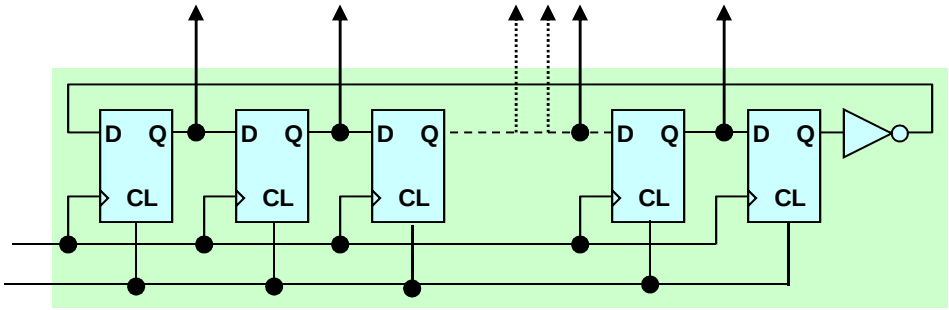
- Para implementar contadores, se emplea feedback desde el registro del LE, la cadena de Carry, un multiplexor disponible a la salida de la LUT, y señales de control independientes (Enable e Up/Down) para cada una de las sub-LUTs.

El retardo asociado al camino AZUL e directamente propocional a la cantidad de etapas



Contadores Johnson

- Los contadores Johnson son ineficientes en el uso de registros, porque su cantidad de estados es sólo el doble de los registros usados
- En diseños de chips de RF suelen ser usados como *prescalers* en altas frecuencias por su simplicidad circuital y velocidad de operación
- Estos beneficios no son tales al usar FPGAs, donde tienen igual performance que un contador LFSR
- La secuencia (para 4 FFs) es 0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001 ... y nuevamente 0000

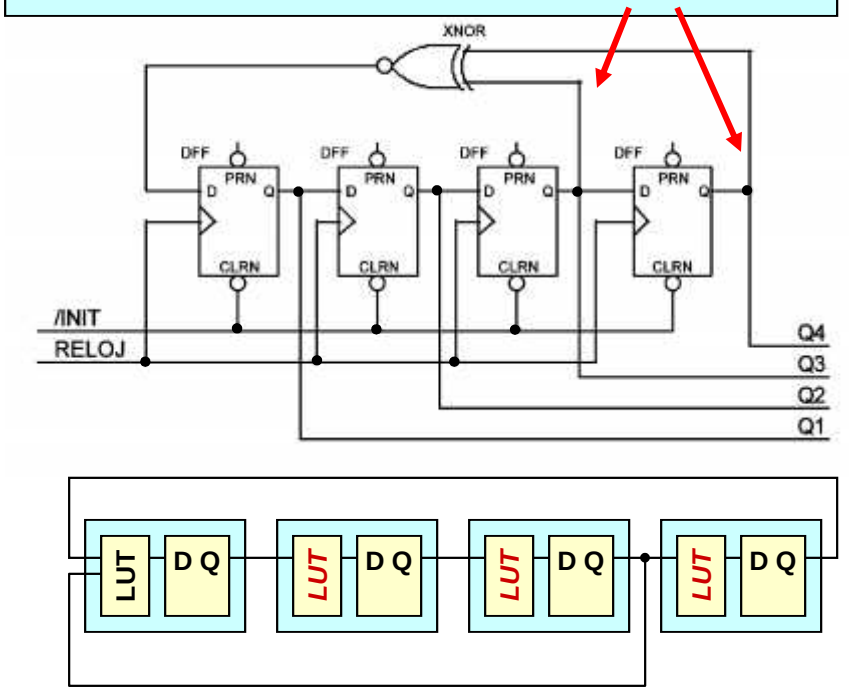



```
ENTITY johnson IS GENERIC (n:INTEGER:= 8);
 PORT (reloj,init: IN BIT;
       q: BUFFER BIT_VECTOR (n DOWNTO 1));
END ENTITY johnson;

ARCHITECTURE a OF johnson IS
BEGIN
sync: PROCESS (reloj,init)
 BEGIN
  IF init='1' THEN q <=(others =>'0');
  ELSIF reloj'EVENT AND reloj='1' THEN
   q(n DOWNTO 2) <= q (n-1 DOWNTO 1);
   q(1) <= NOT (q(n));
  END IF;
 END PROCESS sync;
END ARCHITECTURE a;
```

Divisores enteros con contadores
Linear Feedback Shift Register

- Un ***Linear Feedback Shift Register (LFSR)*** es una máquina sincrónica de enorme simplicidad circuital
- Se basa en un shift register de N etapas, donde el valor que entra a la primer etapa se calcula como el XNOR de 2 o 4 etapas intermedias (donde siempre está la última etapa).
- Si las realimentaciones son bien elegidas, la secuencia generada se repite luego de 2^N-1 ciclos de reloj, lo que aproxima su eficiencia de uso de flipflops al de un contador binario puro (2^N estados)
- Pero los recursos de cableado que usa son mucho menores, por lo que es más veloz y más fácil de rutear

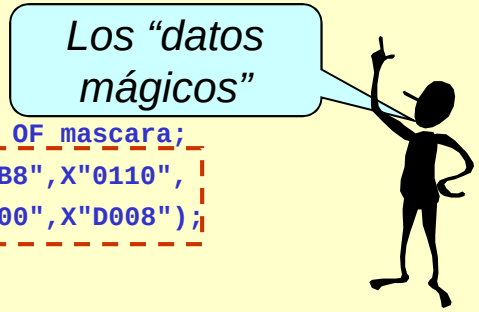
Realimentación desde 2 etapas: secuencia 0000, 1000, 1100, 1110, 0111,1011, 1101, 0110, 0011, 1001, 0100, 1010, 0101, 0010, 0001 ... y vuelve a 0000



Al ser función de 2 o 4 realimentaciones, bastan N elementos lógicos para un LFSR de N bits

LFSR genérico, de 3 a 16 etapas

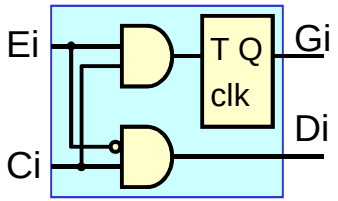
```
ENTITY lfsrn IS GENERIC (LFSRTAPS : INTEGER RANGE 3 TO 16 :=3); -- cantidad de etapas, entre 3 y 16
  PORT (n_init,reloj : IN BIT; taps : OUT BIT_VECTOR (LFSRTAPS DOWNTO 1));
END ENTITY lfsrn;
ARCHITECTURE a OF lfsrn IS
 SUBTYPE mascara IS BIT_VECTOR (16 DOWNTO 1); TYPE tabla IS ARRAY (3 TO 16) OF mascara;
 CONSTANT tablaxor : tabla := (X"0006",X"000C",X"0014",X"0030",X"0060",X"00B8",X"0110",
                               X"0240",X"0500",X"0829",X"100D",X"2015",X"6000",X"D008");
 SIGNAL  fftaps : BIT_VECTOR (LFSRTAPS DOWNTO 1); SIGNAL  xorout : BIT;
 BEGIN
p0: PROCESS (reloj,n_init) BEGIN
   IF n_init = '0' THEN fftaps <= (others=>'0');
   ELSIF reloj'EVENT AND reloj = '1' THEN
    FOR i IN 2 TO LFSRTAPS LOOP fftaps(i) <= fftaps (i-1); END LOOP;
    fftaps (1) <= xorout;
   END IF;
  END PROCESS p0;
p1: PROCESS (fftaps)
   VARIABLE temp : BIT; VARIABLE mask : BIT_VECTOR (16 DOWNTO 1);
   BEGIN
    temp := '0'; mask := tablaxor (LFSRTAPS);
    FOR i IN 1 TO LFSRTAPS LOOP if (mask (i)='1') THEN temp := temp XOR fftaps(i); END IF; END LOOP;
    xorout <= NOT (temp);
  END PROCESS p1;
 taps <= fftaps;
END ARCHITECTURE a;
```



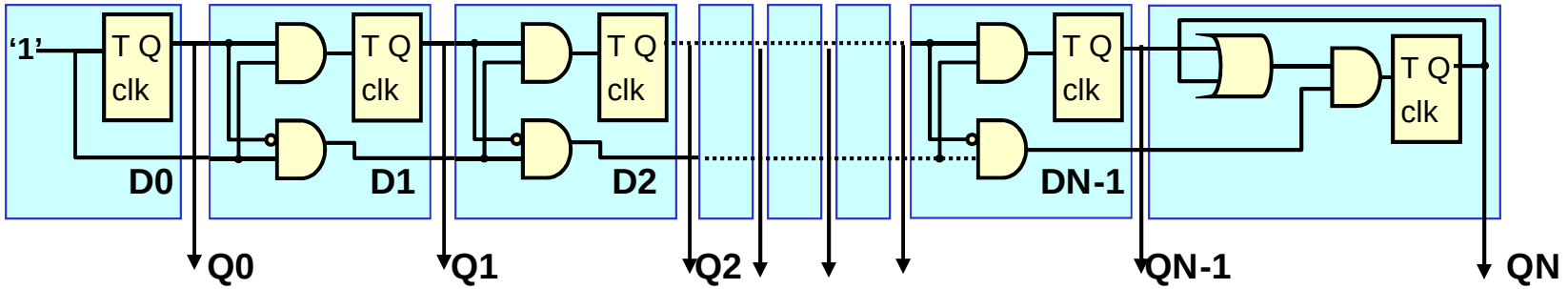
Divisores enteros mediante contadores Código GRAY

- Cuando las salidas de un divisor son decodificadas en forma combinatoria para generar señales de control, el empleo de contadores GRAY garantiza la ausencia de glitches y simplifica la decodificación
- Una solución eficiente permite realizar un contador GRAY de N bits usando sólo N+1 macroceldas, que en base al bit actual “Gi”, al previo Gi-1 y a una función Di-1 que viene desde la etapa previa calcula el siguiente Gi y la función Di
- Excepto la primer etapa (generación de D-1) y la última etapa, las demás etapas tienen una arquitectura como sigue

Evaluar los posibles usos de los “modos de operacion”



- En base a este “módulo” un contador GRAY de N bits se configura

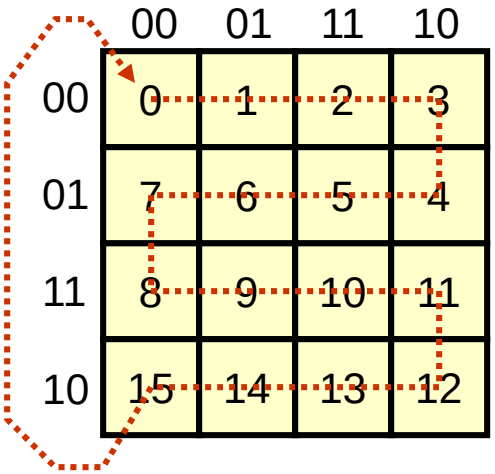


Contador Código GRAY genérico

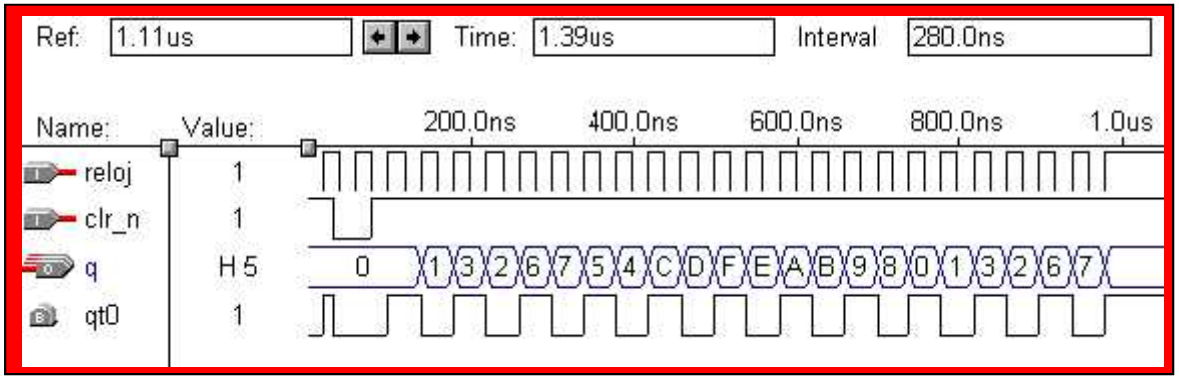
```
ENTITY graycont IS GENERIC (NBITS : INTEGER := 4);
 PORT (reloj,clr_n: IN BIT; q: OUT BIT_VECTOR (NBITS downto 1));
END ENTITY graycont;

ARCHITECTURE a OF graycont IS
 SIGNAL d: BIT_VECTOR (NBITS-1 downto 0); SIGNAL qt: BIT_VECTOR (NBITS downto 0);
BEGIN
sync: PROCESS (reloj,clr_n)
 BEGIN
  IF clr_n='0' THEN qt <=  (others =>'0');
  ELSIF reloj'EVENT AND reloj='1' THEN
   qt(0) <= NOT(qt(0));
   FOR i IN 1 TO NBITS-1 LOOP qt(i) <= qt(i) XOR (qt(i-1) AND d(i-1)); END LOOP;
   qt(NBITS) <= qt(NBITS) XOR ( (qt(NBITS-1) OR qt(NBITS)) AND d(NBITS-1));
  END IF;
 END PROCESS sync;
 d(0) <= '1';
 dloop: FOR i IN 1 TO NBITS-1 GENERATE d(i) <= NOT(qt(i-1)) AND d(i-1);
        END GENERATE;
 q <= qt (NBITS downto 1);
END ARCHITECTURE a;
```

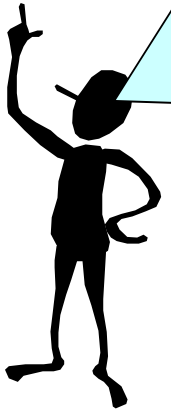
Contadores Código GRAY

Ciclo	Código GRAY	Dummy qt0	Ciclo	Código GRAY	Dummy qt0
0	0000=0	1	8	1100=C	1
1	0001=1	0	9	1101=D	0
2	0011=3	1	A	1111=F	1
3	0010=2	0	B	1110=E	0
4	0110=6	1	C	1010=A	1
5	0111=7	0	D	1011=B	0
6	0101=5	1	E	1001=9	1
7	0100=4	0	F	1000=8	0



- Resultados de la simulación funcional





Con hasta 4 bits, contadores ***GRAY*** de qué largo de secuencia serán realizables?
Cómo eran los diagramas de Karnaugh?
Qué era la *adyacencia*?
Usar la idea para diseñar contadores que no sean potencia de dos

Fmax y complejidad de las distintas soluciones

- La tabla muestra las frecuencias estimadas de distintas soluciones que empleen N flipflops, suponiendo en el caso Binario y Gray que se emplean FPGAs que poseen cadenas de carry. De no ser así las frecuencias de operación son muchísimo menores
- La máxima cantidad de estados es ofrecida por el binario, seguido casi excatamente por el LFSR
- La solución binaria es la única que genera una secuencia natural de números
- La única ventaja del contador GRAY es el cambio de un solo bit por vez, la cantidad de estados es la mitad de uno binario o LFSR
- En el caso del Johnson no se pone tcomb porque un FF tiene naturalmente la salida negada

Tipo	Estados	Frecuencia
Binario	2^N	1/(N*tcarry)
Johnson	2*N	1/(tco+tsetup)
Gray	$2^{(N-1)}$	1/(N*tcarry)
LFSR	2^N-1	1/(tco+tcomb+tsetup)