COMPUTADORES I

Contador de cuenta arbitraria

Índice

Contador arbitrario:

El objetivo de este trabajo es crear un contador de cuenta arbitraria, un circuito de tipo secuencial. Por ello, sus salidas dependen de las entradas, porque pueden almacenar información.
Debido a esto, hay que tener en cuenta el estado en el que estaba el circuito antes (es decir, si estaba en 0, 1 o x), y el estado al que lo queremos llevar.

En nuestro caso realizaremos el contador de la siguiente sucesión numérica:

4, 14, 6, 3, 12, 11, 13, 0 y comienza de nuevo.

A la hora de realizar el circuito son necesarios cuatro pasos:

- Crear un diagrama de transición.
- Realizar la tabla de transiciones.
- Hacer los mapas de Karnaugh para sintetizar las funciones.
- Programar y diseñar el circuito, analizar los cronogramas y resultados.

Nuestro contador va a llevar biestables JK, que tienen 2 entradas (J y K), y dos salidas (Q y QNEG), las cuales son opuestas, es decir, si Q vale 1, QNEG vale 0.
Debido a su capacidad de almacenar información, cada ciclo de reloj los biestables almacenarán una cifra en binario del número a tratar en ese momento.

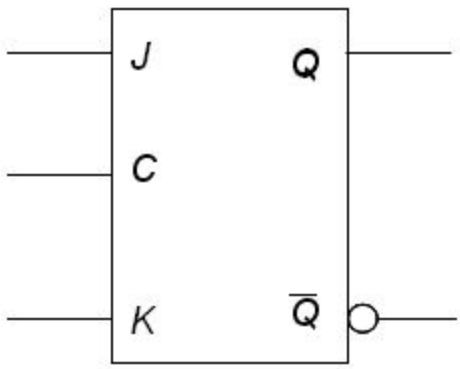


Diagrama de Transición:

El diagrama de transición es un esquema que se hace para saber cuál es el valor anterior en el que está el circuito, y cuál va a ser el valor siguiente al que tenemos que ir, que es la pregunta más importante que hay que hacerse cuando se trabaja con circuitos secuenciales.
Para ello hay que tener en cuenta dos cosas:

- La sucesión de números que queremos que siga.
- El valor que le vamos a asignar a los números que no estén en la sucesión.

El punto más importante es el último, puesto que nuestro circuito va a generar todos los valores posibles dentro del rango que le hemos dado por su construcción lógica. Como tiene cuatro biestables, va a poder ir del 0 al 15 (el rango se halla haciendo (2^número de biestables) - 1).
Debido a esto, los valores que no estén dentro de la sucesión que tenemos que generar, tenemos que recolocarlos en un valor que pertenezca a la sucesión que imprimiremos por pantalla.

Para crear el diagrama dispusimos nuestra serie de números en el orden establecido. Después asignamos el resto de números que no pertenecían a la sucesión, siguiendo dos criterios:

- A cada número ya posicionado solo se le asignará un único número.
- Cambiar el mínimo número de bits: es decir, si tenemos el número 2(0010), se lo asignaremos al número 3(0011), ya que solo hay que cambiar el último cero por un uno.

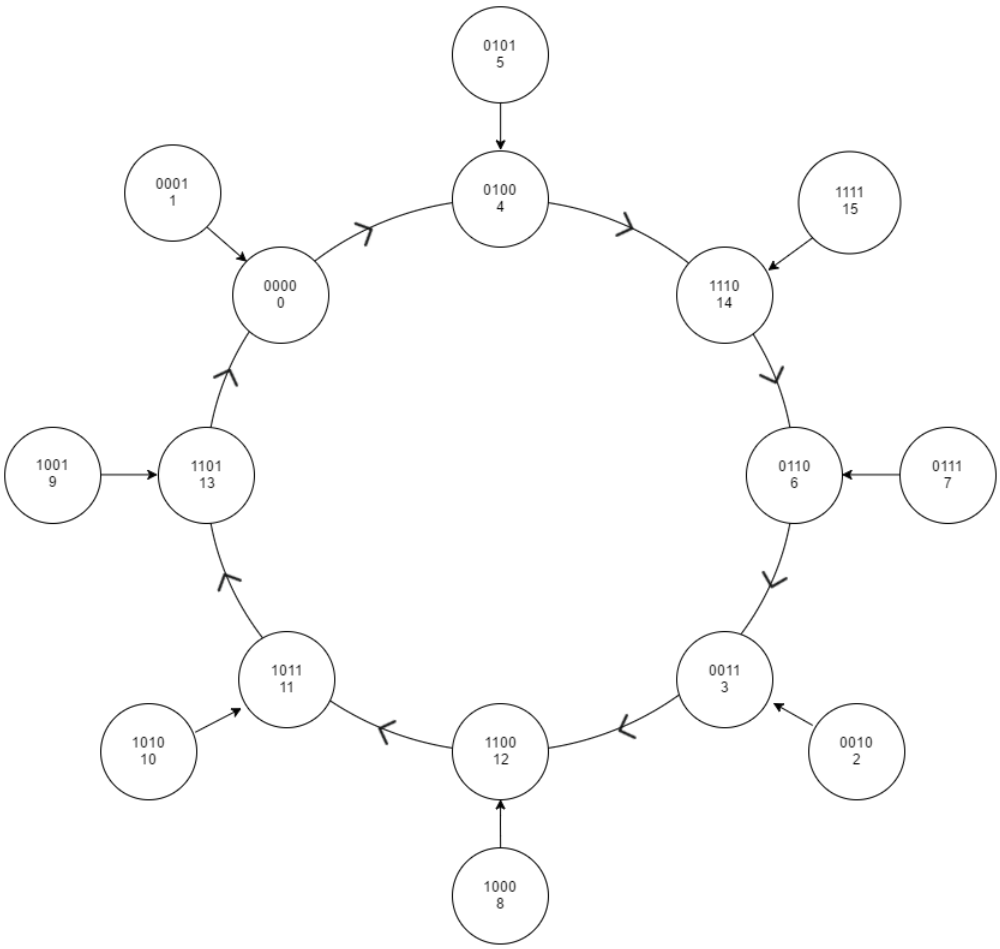


Tabla de transiciones:

Transiciones		J3	K3	J2	K2	J1	K1	J0	K0
0 → 4	0000 → 0100	0	x	1	x	0	x	0	x
1 → 0	0001 → 0000	0	x	0	x	0	x	x	1
2 → 3	0010 → 0011	0	x	0	x	x	0	1	x
3 → 12	0011 → 1100	1	x	1	x	x	1	x	1
4 → 14	0100 → 1110	1	x	x	0	1	x	0	x
5 → 4	0101 → 0100	0	x	x	0	0	x	x	1
6 → 3	0110 → 0011	0	x	x	1	x	0	1	x
7 → 6	0111 → 0110	0	x	x	0	x	0	x	1
8 → 12	1000 → 1100	x	0	1	x	0	x	0	x
9 → 13	1001 → 1101	x	0	1	x	0	x	x	0
10 → 11	1010 → 1011	x	0	0	x	x	0	1	x
11 → 13	1011 → 1101	x	0	1	x	x	1	x	0
12 → 11	1100 → 1011	x	0	x	1	1	x	1	x
13 → 0	1101 → 0000	x	1	x	1	0	x	x	1
14 → 6	1110 → 0110	x	1	x	0	x	0	0	x
15 → 14	1111 → 1110	x	0	x	0	x	0	x	1

La tabla de transiciones es llevar a la práctica el diagrama de transición, para ello, como cada biestable almacena de una cifra y nuestros números (en binario) van a tener 4 cifras, vamos a analizar cifra por cifra cada número y observar el modelo de comportamiento del biestable JK, y según esto, darle unos valores a las entradas de cada biestable (J y K), para poder generar esa transición.

Actual	Sgte.	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

La tabla de transiciones la realizamos como fue propuesta en clase:

- Se colocan los números en orden descendente, y a su lado colocamos al número que le hemos asignado como estado siguiente.

- Analizamos según el modelo de comportamiento del biestable JK, ya que es el recomendado para este trabajo, además que es el más sencillo para nuestro propósito, ya que tiene dos salidas, Q y Q negada por lo que nos ahorraremos varias puertas NOT, ademas de que a

diferencia del biestable RS, responde de forma positiva cuando J=1 y K=1 (porque es síncrono y el reloj anula la función complemento), por lo que se que facilitan bastante las cosas.

- La construcción de la tabla se realizó de la siguiente manera: por ejemplo, el 1 va a 0, o lo que es lo mismo, 0001 va a 0000. Como la primera cifra va de 0 a 0, según el modelo de comportamiento del biestable JK, para que esto ocurra, J debe ser 0 y K cualquier valor, es decir, x; debido a esto asignamos estos valores para J3 y K3 en ese momento; este proceso se repetiría con la segunda cifra, tercera y cuarta, asignando los correspondientes valores a las casillas J2 y K2, J1 y K1 y J0 y K0.

Mapas de Karnaugh:

J3				
Q3Q2 Q1Q0	0 0	0 1	1 1	1 0
0 0	0	1	X	X
0 1	0	0	X	X
1 1	1	0	X	X
1 0	0	0	X	X

J3 = $\overline{Q_0}\,\overline{Q_1}\,Q_2$ + $Q_0\,Q_1\,\overline{Q_2}$

J2				
Q3Q2 Q1Q0	0 0	0 1	1 1	1 0
0 0	1	X	X	1
0 1	0	X	X	1
1 1	1	X	X	1
1 0	0	X	X	0

J2= $\overline{Q_0}\,\overline{Q_1}$ + $Q_0\,Q_1$ + $\overline{Q_1}\,Q_3$
(el amarillo es intersección entre verde y azul)

J1				
Q3Q2 Q1Q0	0 0	0 1	1 1	1 0
0 0	0	1	1	0
0 1	0	0	0	0
1 1	X	X	X	X
1 0	X	X	X	X

J1 = $\overline{Q_0}\,Q_2$

K3				
Q3Q2 Q1Q0	0 0	0 1	1 1	1 0
0 0	X	X	0	0
0 1	X	X	1	0
1 1	X	X	0	0
1 0	X	X	1	0

K3 = $Q_0\,\overline{Q_1}\,Q_2$ + $\overline{Q_0}\,Q_1\,Q_2$

K2				
Q3Q2 Q1Q0	0 0	0 1	1 1	1 0
0 0	X	0	1	X
0 1	X	0	1	X
1 1	X	0	0	X
1 0	X	1	0	X

K2= $\overline{Q_1}\;\;Q_3$ + $\overline{Q_0}\,Q_1\,\overline{Q_3}$

K1				
Q3Q2 Q1Q0	0 0	0 1	1 1	1 0
0 0	X	X	X	X
0 1	X	X	X	X
1 1	1	0	0	1
1 0	0	0	0	0

K1 = $Q_0\,\overline{Q_2}$

J0				
Q3Q2 Q1Q0	0 0	0 1	1 1	1 0
0 0	0	0	1	0
0 1	X	X	X	X
1 1	X	X	X	X
1 0	1	1	0	1

J0 = $Q_1 \overline{Q_3}$ + $\overline{Q_1} Q_2 Q_3$ + $Q_1 \overline{Q_2}$
(el amarillo es intersección entre verde y azul)

K0				
Q3Q2 Q1Q0	0 0	0 1	1 1	1 0
0 0	X	X	X	X
0 1	1	1	1	0
1 1	1	1	1	0
1 0	X	X	X	X

K0 = Q_2 + $\overline{Q_3}$ = $\overline{Q_2} Q_3$
(el morado es intersección entre rojo y azul)

Los mapas de Karnaugh son un mecanismo para crear y simplificar funciones lógicas.
Para realizarlos se disponen los valores para esas funciones lógicas (en este caso los hallados en la tabla de transiciones), y se buscan (y se señalan) grupos de 1s y Xs lo más grandes posibles (porque ahora trabajamos por miniterminos dado que nos pareció lo más óptimo para el desarrollo del proyecto); pero esos grupos deben ser del tamaño de una potencia de 2; es decir, de 2,4,8 o 16 1s.
Luego se cogen los grupos extraídos uno por uno, y se mira que variables no cambian de valor en dicho grupo; por ejemplo en J0 en el grupo azul, Q_1 y Q_3 son las únicas variables que se mantienen constantes; Q_1 con 1s asique la ponemos como está, y Q_3 con 0s, asique la ponemos negada ($\overline{Q_3}$). Tras esto, se pondrían Q_1 y $\overline{Q_3}$ multiplicándose, y se sumaría esta multiplicación a el resultado de aplicar este proceso a todos los grupos de cada mapa.

Ecuaciones:

- J3 = $\overline{Q_0}\,\overline{Q_1}\,Q_2$ + $Q_0\,Q_1\,\overline{Q_2}$
- K3 = $Q_0\,\overline{Q_1}\,Q_2$ + $\overline{Q_0}\,Q_1\,Q_2$

- J2= $\overline{Q_0}\,\overline{Q_1}$ + $Q_0\,Q_1$ + $\overline{Q_1}\;Q_3$
- K2= $\overline{Q_1}\;Q_3$ + $\overline{Q_0}\,Q_1\,\overline{Q_3}$

- J1 = $\overline{Q_0}\,Q_2$
- K1 = $Q_0\,\overline{Q_2}$

- J0 = $Q_1\,\overline{Q_3}$ + $\overline{Q_1}\,Q_2\,Q_3$ + $Q_1\,\overline{Q_2}$
- K0 = Q_2 + $\overline{Q_3}$

Tras todo el proceso de realizar la tabla de transiciones y los mapas de Karnaugh, nos han quedado estas ecuaciones, que en la síntesis del circuito se quedan en 19 puertas lógicas.

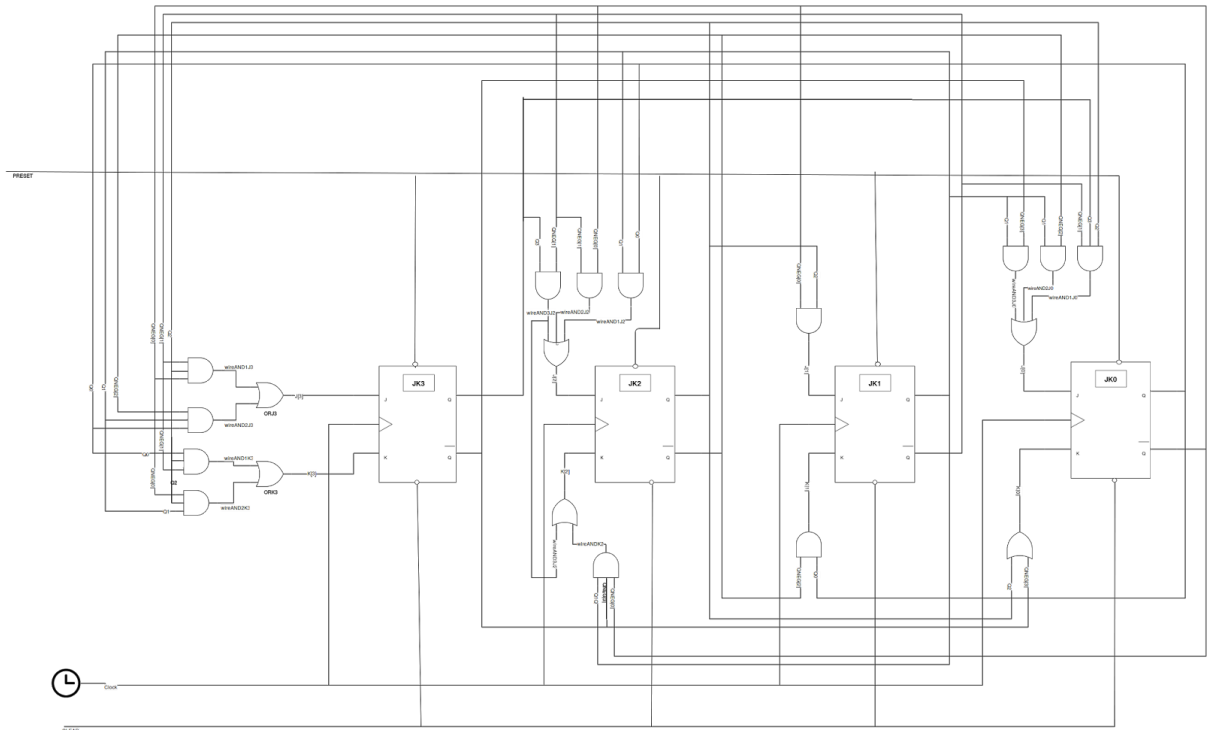
Tras este proceso estuvimos intentando reducirlas mediante los postulados de Huntington, las propiedades del algebra de Boole y las leyes de Morgan y Shannon, pero no conseguimos agrupar las variables de ninguna manera que nos permitiera aplicarlas.

La única optimización que pudimos encontrar en estas funciones fue el usar en J2 y K2 la siguiente función $\overline{Q_1}\;Q_3$, ya que formaba parte de ambas.
También nos dimos cuenta de que K0= Q_2 + $\overline{Q_3}$ = $\overline{Q_2}\,Q_3$, porque creímos que $\overline{Q_2}\,Q_3$ estaría presente en alguna otra parte del circuito sintetizado como función, pero no fue el caso.

También estuvimos buscando en las ecuaciones alguna agrupación de incógnitas que tuviera el siguiente patrón: $Q_a\,\overline{Q_b}$ + $\overline{Q_a}\,Q_b$, para sustituirlo por una puerta XOR (que haría la misma función); pero de nuevo, no encontramos ninguna.

Por último intentamos simplificar las ecuaciones observando los mapas de Karnaugh e intentando cambiar algún 0 por 1 o viceversa, viendo cómo podíamos cambiar los números de arriba, pero lo único que conseguimos fue rizar el rizo, porque en cuanto simplificamos un mapa, el resto de mapas se convertía en un desastre total; así que dejamos pasar esa simplificación también.

Circuito:

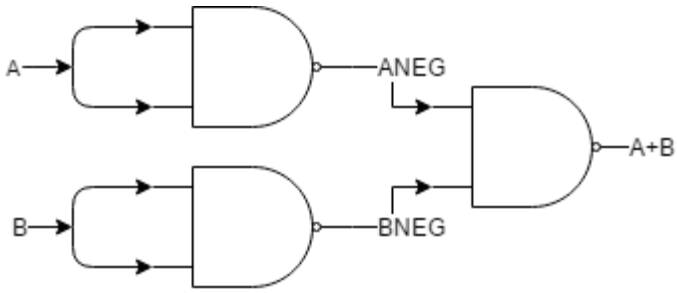


Adjuntamos este mismo circuito en los formatos PDF, HTML y PNG , en los que se puede ampliar, y con ello se pueden observan los nombres impuestos a cada conexión.

Esta parte del proceso se llama implementar el circuito, para ello se colocan los 4 biestables JK, y se traducen las ecuaciones a puertas lógicas.
En este caso solo hay sumas y multiplicaciones asique para la suma usaremos la puerta OR, y para la multiplicación usaremos la puerta AND;
Tras colocar las puertas lógicas y darles nombre a los cables que las unen (para poderlo codificar posteriormente), se unen a los cables correspondientes.

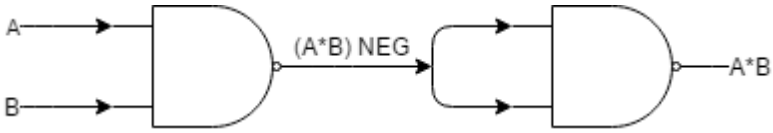
En este caso hemos usado solo puertas AND y OR, utilizando un total de 19. No obstante, para ahorrar costes en la fabricación del circuito (si se hubiera realizado en la vida real) se podrían haber sustituido estas por un solo tipo de puerta, como puede ser la puerta NAND:

Puerta OR con NAND



- Primero se niegan A y B, porque la NAND hace la multiplicación y saca el valor negado, por lo que en el caso de la primera puerta (por la que pasa A):
  A→ A*A=A--> ANEG
- Luego se pasan por la puerta NAND, que va a generar ANEG * BNEG, todo ello negado, que aplicando la D de morgan es A + B

Puerta AND con NAND:



- Primero se realiza la operación, y después se niega, ya que por la propiedad de la complementariedad, una variable negada dos veces, es esa misma variable.

Para haber realizado esto se habria que haber tenido en cuenta que cuando una puerta AND va conectada a una puerta OR, se eliminan las puertas NOT, generadas por AND, ya que si las conservamos estaríamos haciendo el negado del negado, y por la ya mencionada ley de complementariedad, el negado del negado es la variable sin negar.
Al final no realizamos sobre papel, ni sobre código el circuito solo con puertas NAND, ya que este tenía 23 puertas lógicas, lo que habría hecho que el circuito saliera incluso menos rentable.

Código:

Módulo “BiesableJK”:

```
//modulo del biestable JK
module biestableJK (output reg Q, output wire NQ, input wire J, input wire K,   input wire C);
  //hacemos Q negado, que es la salida secundara del biestable JK
  not(NQ,Q);
  initial
  begin
    //inicializamos Q a 0 para evitar errores
    Q='b0;
  end
  //codificamos los biestables (por flanco de bajada), y su modelo de comportamiento
  always @(negedge C)
    case ({J,K})
      2'b10: Q='b1; //set
      2'b01: Q='b0; //reset
      2'b11: Q=~Q;  //complemento
    endcase
endmodule
```

Este es el módulo más importante de todo el código, ya que es la base del contador. Lo llamamos cuatro veces para representar cada bit del número a contar en un momento determinado. Está modulado por comportamiento por lo que su funcionamiento es entendible leyendo el código que lo forma. Se activa cada vez que hay un flanco de bajada en la señal de reloj C (negedge).

Módulo “Contador”:

```
//modulo del contador
module contador (inout wire [3:0] Q, input wire C);
    //Declaramos arrays de tipo wire para poder almacenar la informacion que sale del llamamiento a los modulos de  biestable
    wire [3:0] QNEG; //salidas negadas
    wire [3:0] J; // entradas J
    wire [3:0] K; //entradas K
    //declaramos todos los cables que van a ser necesarios para unir las puetas con el siguiente esquema: wire      wire_tipoDePuerta_Numero_
     //nota: en el esquema del circuito se especifica cual es la funcion de cada cable
        //J3
            wire wireAND1J3, wireAND2J3;
        //K3
            wire wireAND1K3, wireAND2K3;
        //J2
            wire wireAND1J2,wireAND2J2,wireAND3J2;
        //K2
            wire wireANDK2;
  //aqui no hizo falta implementar una parte de la funcion, ya que estaba implementada en wireAND3J2
        //J0
            wire wireAND1J0, wireAND2J0, wireAND3J0;
    //implementamos las puetas para generar J y K con el siguiente esquema: puerta_numero_InputAlQueConecta
  //nota:si es de tipo OR no lleva numero ya que solo hay una por funcion logica
  //nota: las conexiones van en orden numerico de 0 a 3
        //J3
            and and1J3 (wireAND1J3,QNEG[0],QNEG[1],Q[2]);
            and and2J3 (wireAND2J3,Q[0],Q[1],QNEG[2]);
            or orJ3 (J[3],wireAND1J3,wireAND2J3);
        //K3
            and and1K3 (wireAND1K3,Q[0],QNEG[1],Q[2]);
            and and2K3 (wireAND2K3,QNEG[0],Q[1],Q[2]);
            or orK3 (K[3],wireAND1K3,wireAND2K3);
        //J2
            and and1J2 (wireAND1J2,Q[0],Q[1]);
            and and2J2 (wireAND2J2,QNEG[0],QNEG[1]);
            and and3J2 (wireAND3J2,QNEG[1],Q[3]);
            or orJ2 (J[2],wireAND1J2,wireAND2J2,wireAND3J2);
        //K2
            and and1K2 (wireAND1K2,QNEG[0],Q[1],QNEG[3]);
            or orK2 (K[2],wireAND3J2,wireAND1K2);
        //J1
            and andJ1 (J[1],QNEG[0],Q[2]);
        //K1
            and andK1 (K[1],Q[0],QNEG[2]);
        //J0
            and and1J0 (wireAND1J0,QNEG[1],Q[2],Q[3]);
            and and2J0 (wireAND2J0,Q[1],QNEG[2]);
            and and3J0 (wireAND3J0,Q[1],QNEG[3]);
            or orJ0 (J[0],wireAND1J0, wireAND2J0, wireAND3J0);
        //K0
            or orK0 (K[0],Q[2],QNEG[3]);
    //Hacemos la llamada al biestable JK, y le damos las inputs J,K y C, y sacamos los ouputs Q
    //(llamada al modulo) (nombre) (Salida Q, Salida Q negada, entrada J, entrada K, reloj)
    biestableJK JK3 (Q[3],QNEG[3],J[3],K[3],C);
    biestableJK JK2 (Q[2],QNEG[2],J[2],K[2],C);
    biestableJK JK1 (Q[1],QNEG[1],J[1],K[1],C);
    biestableJK JK0 (Q[0],QNEG[0],J[0],K[0],C);
  initial
     begin
     end
endmodule
```

Esta parte del código sería el circuito en general, ya que hace la llamada al biestable JK cuatro veces e implementa las diversas puertas lógicas necesarias para que que realice la secuencia numérica deseada.

Es necesario crear tres registros, para las salidas negadas, las Js y las Ks. Implementamos las puertas lógicas con nombres acordes a su orden en la función (tipo de puerta y su número), al igual que los cables (wires) correspondientes.

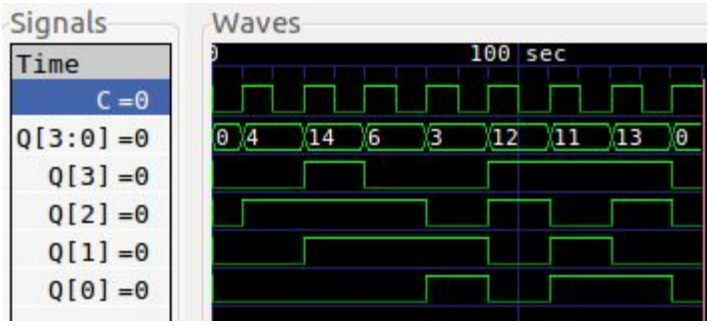
Finalmente llamamos al JK cuatro veces y le conectamos los inputs del circuito (J, K y C) y sacamos los outputs (Q).

Módulo “Test”:

```
//modulo de test
module test;
    wire [3:0] Q; //salidas de los biestables
    reg C; //reloj
    //llamada al modulo del contador (llamada al modulo) (nombre) (salidas Q, reloj)
    contador CONT (Q,C);
    //generamos elreloj: negamos C continuamente
    always #10 C=~C;
  //instrucciones para la ejecucion del modulo test
  initial
  begin
    //declaramos la monitorizacion del cronograma y creamos
    $dumpfile("cronograma.dmp");
    $dumpvars(1,CONT);
    $dumpon;
    C='b0;
    //sacamos por pantalla los resultados
    $monitor($time, " C=%b | Q=%d| Q=%b%b%b%b \n",C,Q,Q[3],Q[2],Q[1],Q[0]);
  //finalizamos la ejecucion a los 160 tics (hay 8 numeros y vamos a 10 tics por numero, asique hacemos dos ciclos)
  #160 $finish;
  //se finaliza el cronograma
  $dumpoff;
  end
endmodule
```

Esta es la parte final del programa, donde se comprueba que funciona el circuito y que muestra en pantalla la selección de números ofrecida y en el orden determinado. Consta de las variables de entrada del circuito del contador, la función $monitor que imprime los resultados del contador en la pantalla y las líneas de código necesarias para crear el cronograma que se adjunta del circuito.

Cronograma:



El cronograma es un diagrama que no representa gráficamente el funcionamiento de un circuito binario; es decir, que expresa los ceros con una señal baja, y los unos con una señal alta.
como se puede apreciar, si vamos leyendo de 10 ms en 10 ms, de forma descendente Q[3] → Q[2] → Q[1] → Q[0] , se ve un número en binario, que coincide con el valor que tiene que tomar el circuito en cada momento.
Esto también se aprecia mirando el registro Q, que lo pone en decimal directamente.
La sucesión de números que salen, son la serie de números que teníamos que reproducir, lo que nos indica que el proceso de creación del circuito esta bien hecho.

Fuentes:

- Draw.io : En esta pagina hemos realizado todos los diagramas.
- http://avellano.fis.usal.es/~compi/ : la hemos usado para mirar algunas cosas sobre sintaxis y sobre el desarrollo de este tipo de contadores.