UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
Facultad de Ingeniería – Ingeniería Electrónica
Análisis y diseño de microprocesadores

Integrantes:

- Leidy Katherine Castelblanco Romero 20151005571
- Nicolás David Pastran Zamora 20151005087

Planteamiento del problema

Se desea implementar un tablero de basketball, el cual debe tener el registro de los puntos de cada equipo, del cuarto de partido, de los minutos y decimas de segundos restantes para que termine cada cuarto. Además, debe tener contar 10 segundos para que empiece el siguiente cuarto. Cada registro debe visualizarse según la guía.

Solución

Se plantea diseñar 3 bloques con entradas en común el temporizador, el marcador y el contador de posesión. Además, un divisor de frecuencia para condicionar la frecuencia de la FPGA.

El **temporizador** lo modelaremos como una máquina de estados, la cual debe tener 4 registros, uno para los segundos, uno para las décimas de segundo, uno para minutos y uno para el periodo. La combinación los estados actuales de cada registro será el estado presente de la máquina y estos serán realimentados a entrada de un bloque combinacional, determinara junto con las entradas pausa, aumento minuto y aumentos segundos, el estado siguiente. Sin embargo, en nuestro caso ya que se plantea la solución en alto nivel el estado actual y las entradas determinaran la acción que debe realizar cada uno de los registros correspondientes.

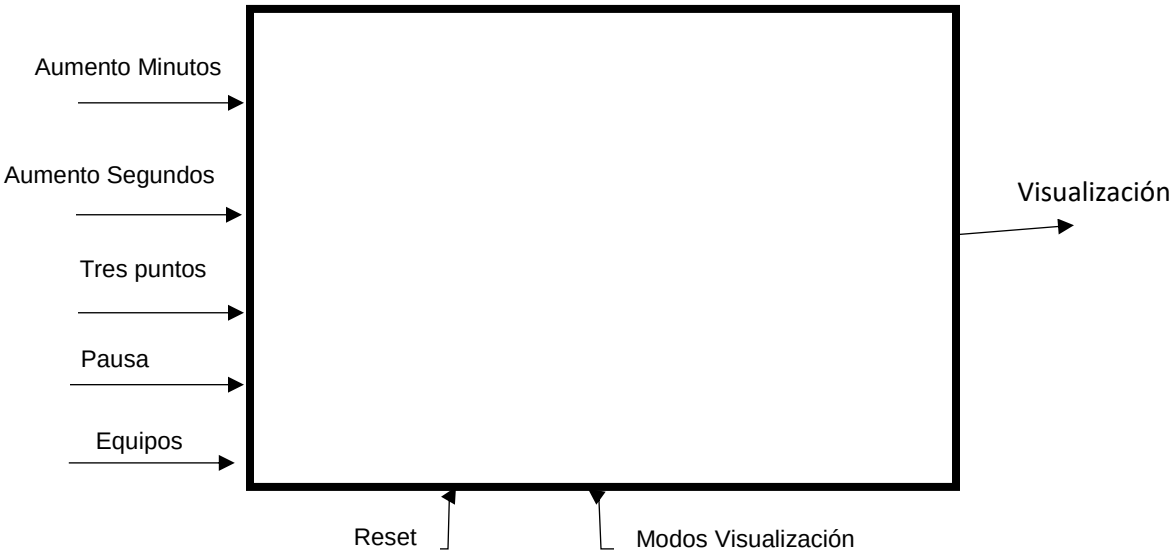
El **marcador** será una máquina de estados sin embargo a su entrada no tendrá una Rom sino un multiplexor que determinara si mantiene el estado o lo actualiza con la suma de dos o tres puntos.

El **Contador de Posesión** debe ser un contador que cuando llegue a 0 señal que active un efecto y que cuando se detecte el cambio de jugador se reinicie.

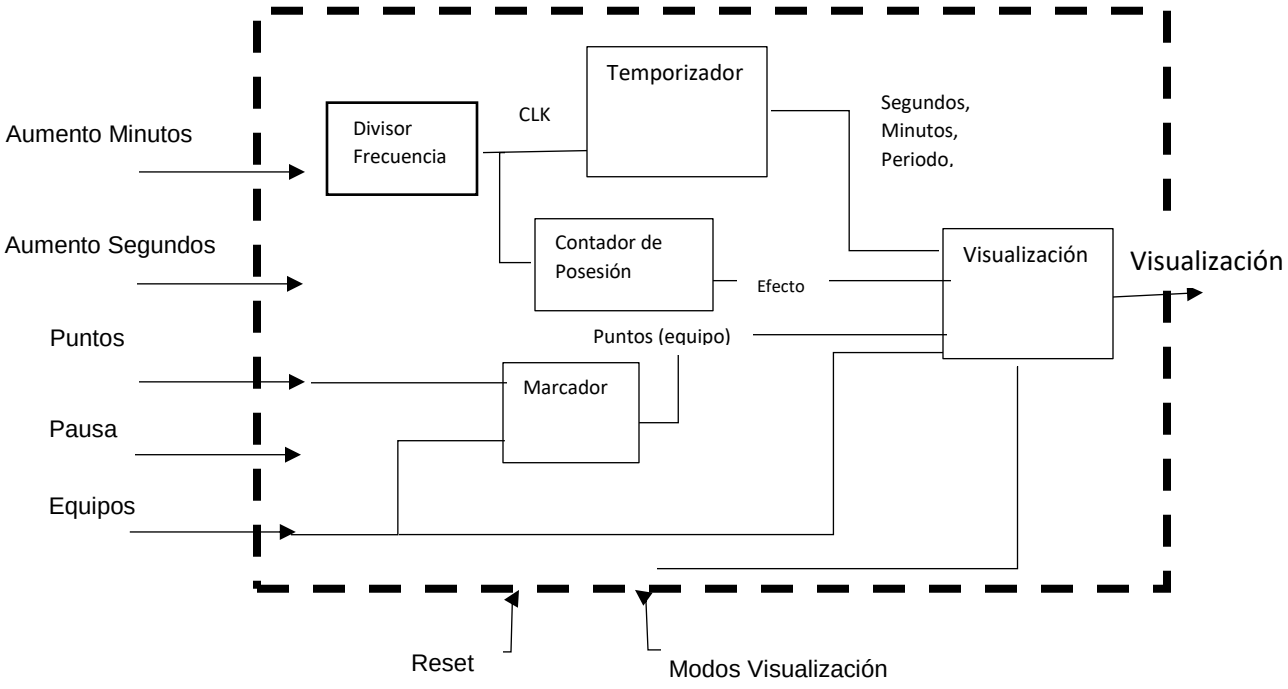
La **visualización** estará compuesta con deberá tener un multiplexor y una función lógica que dependiendo del modo de visualización active solo un tri-state mientras que los demás tengan estado de alta impedancia, las salidas de los tri-state deberán estar unidas y estas serán las salidas del circuito global.

Diagrama de Bloques

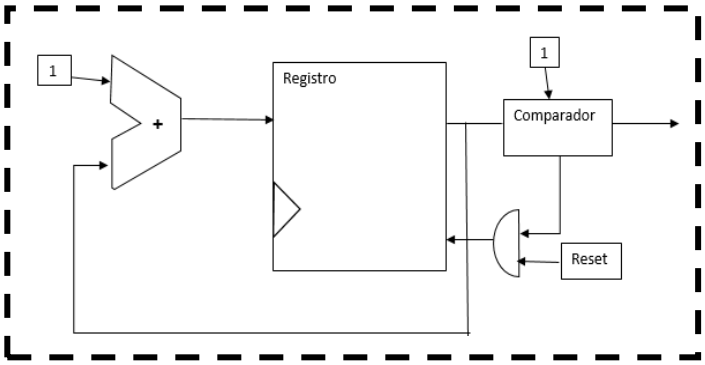
Caja Negra:



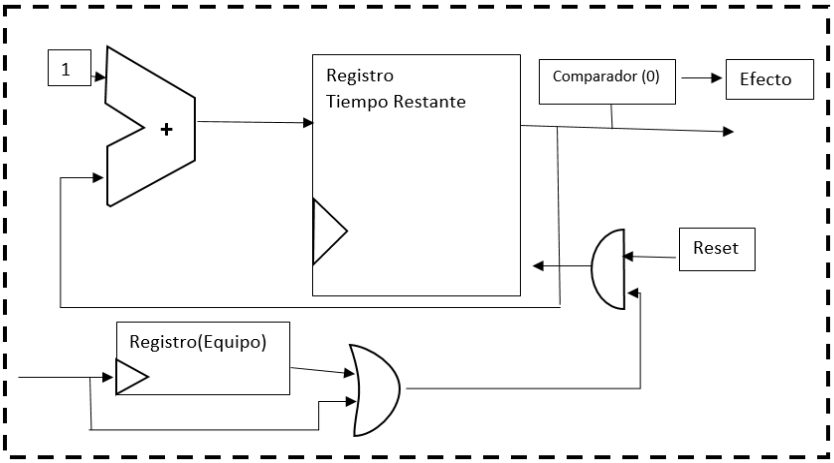
Caja Gris:



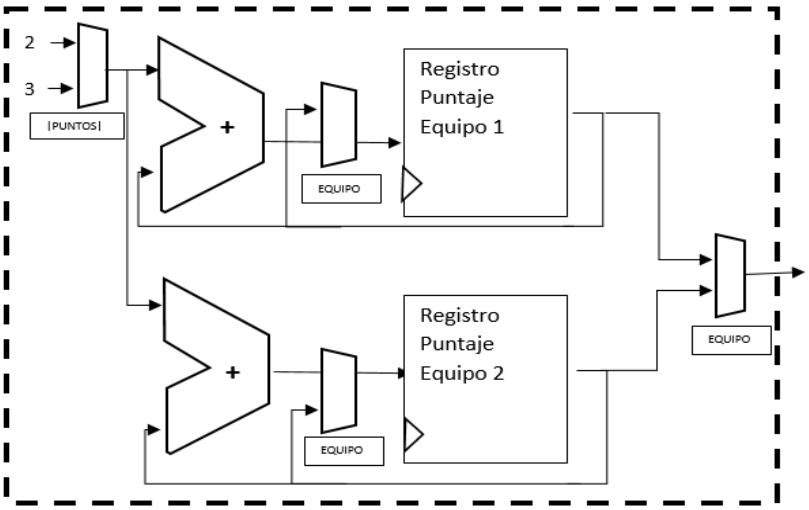
Divisor de Frecuencia:



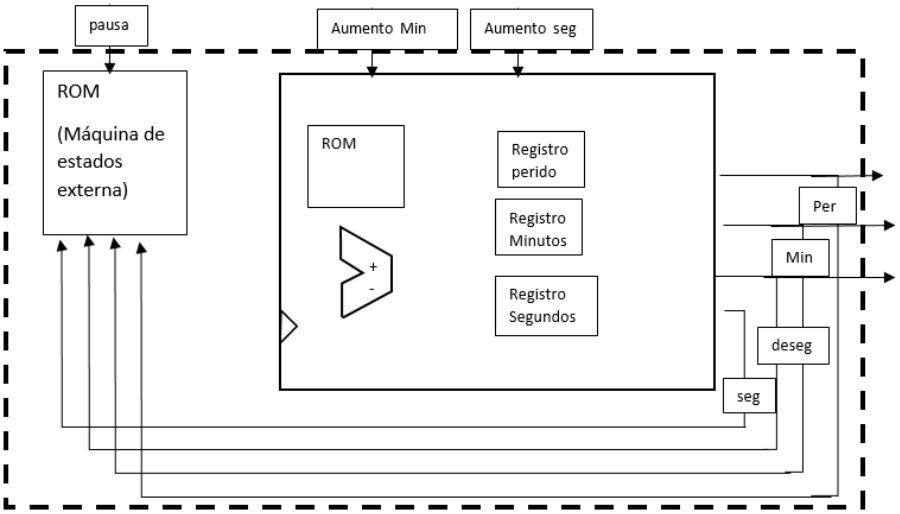
Contador Posesión



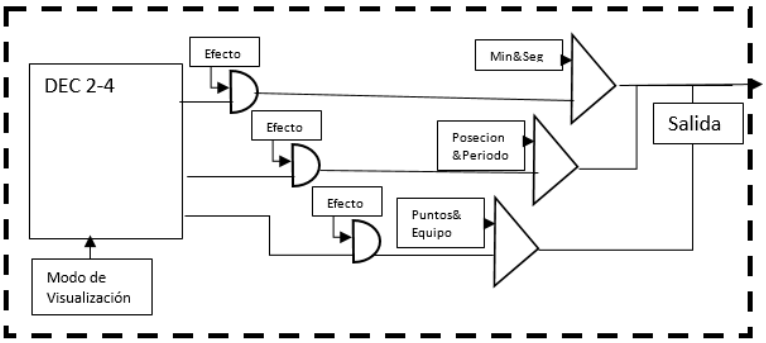
Marcador



Temporizador

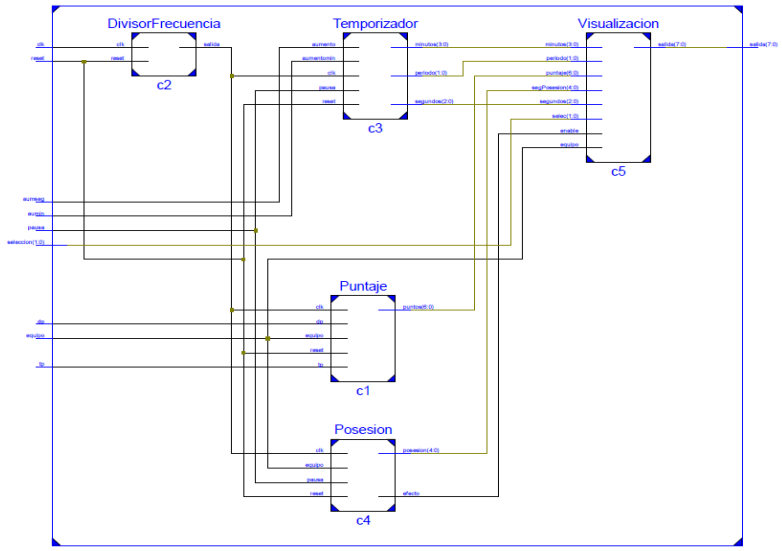


Visualización:

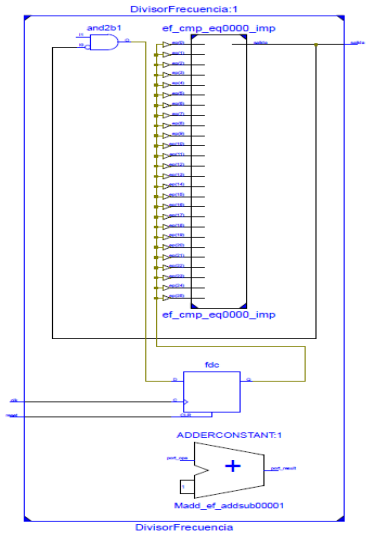


RTL

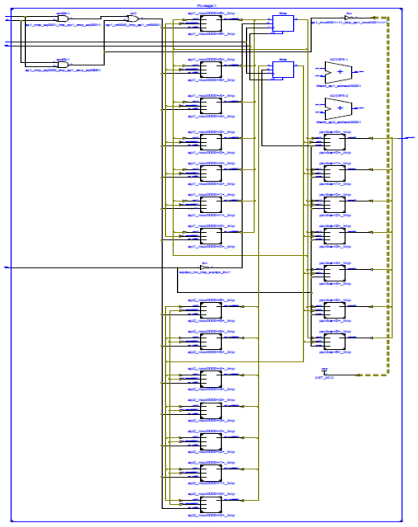
Caja Negra:



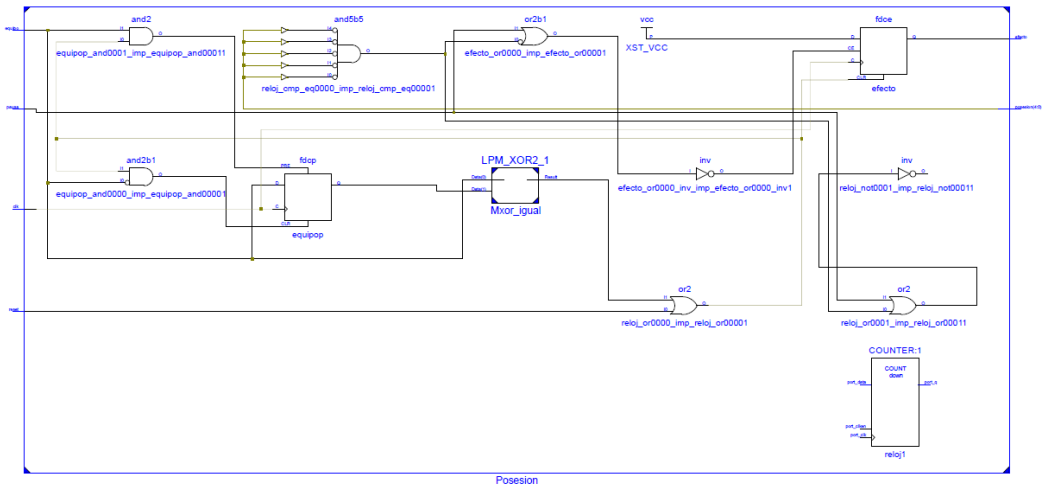
Divisor de Frecuencia:



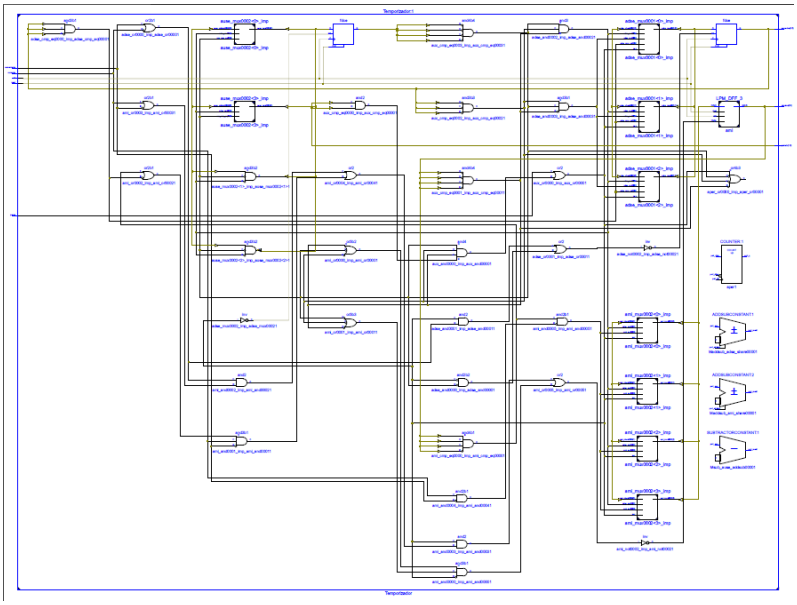
Marcador:



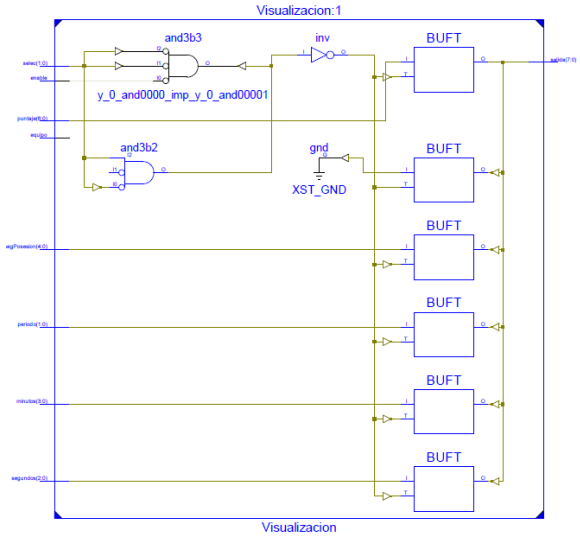
Contador de Posesión:



Temporizador:



Visualización:



Diseño

Como mencionamos anteriormente se implementará el tablero de basketball por usando 4 bloques principalmente y el divisor de frecuencia que adaptará el reloj de 50MHz a un 1Hz para el cual usamos un sumador, un comparador y un registro el cual tendrá 25 F’F ya que debe ser igual a cantidad de bits a la que corresponde 50.000 en binario, en cada ciclo de reloj le sumaremos al estado actual 1 y en el momento en que él estado llegue a 50.000 el comparador envía una señal que resetea el registro.

Cada bloque lo empaquetaremos y los uniremos en un archivo por medio de por maps. Los demás bloques los explicaremos a continuación:

Marcador:

Para empezar el diseño del marcador debemos definir sus entradas y su forma de captura. Con pulsador tomaremos La entrada que determine dos puntos y tres puntos y con interruptor tomaremos el equipo y la pausa (No debe sumar puntos a ninguno si esta pausado). La tabla de verdad teniendo en cuenta las condiciones anteriores es la siguiente:

Pausa	Equipo	Dos Puntos	Tres Puntos	Puntos E1(P1)	Puntos E2(P2)
1	x	x	x	P1	P2
0	0	1	0	P1+2	P2
0	0	0	1	P1+3	P2
0	1	1	0	P1	P2+2
0	1	0	1	P1	P2+3
0	x	0	0	P1	P2

Como vemos en la tabla el bloque cambia su estado cuando no está pausado de modo que la pausa cumple una condición similar al reloj de tal manera que será conveniente que el equipo y el clk vayan a una función lógica y que la salida de esta sea el clk de los registros. Identificamos los componentes que debe tener el circuito a continuación y su funcionamiento:

- 2 registros de 7 bits cada uno, ya que el valor puntaje máximo debe estar entre 0 y 100
- 2 sumadores para sumar los puntos correspondientes a cada equipo.
- 3 Multiplexores uno para identificar si se le suma 2 o 3 puntos sus llaves será Dos puntos y Tres puntos, mientras que los otros dos serán para identificar si el registro debe actualizar el dato con la suma o mantener el dato estado anterior, de modo que uno una de sus entradas será el estado actual y la otra el estado actual más dos o tres y su llave será el equipo.

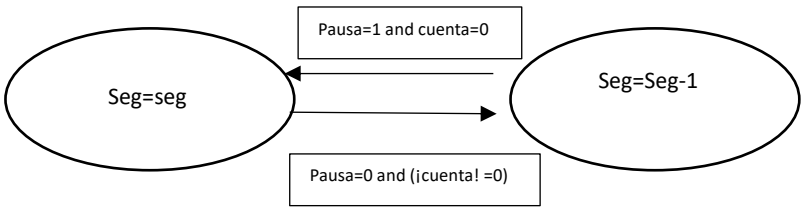
Para facilitar el bloque de visualización la salida del circuito dependerá del equipo que este seleccionado, esto generará otro multiplexor, siendo sus entradas los dos puntajes y la llave el equipo. De la descripción en VHDL tenemos:

```
architecture Behavioral of Puntaje is

signal ap1,ap2: std_logic_vector(6 downto 0) ;
signal a: std_logic_vector(1 downto 0);
signal aclk:std_logic;

aclk<=(not(pausa))or(clk);
begin
a<=tp&dp;
process(equipo,dp,tp,reset,clk)
begin
if(reset='1')then
      ap1<=(others=>'0');
      ap2<=(others=>'0');
elsif(aclk'event and aclk='1')then
   if(equipo='0') then
   case a is
      when "01"=> ap1 <= ap1+2;
      when "10"=>ap1<= ap1+3;
      when others => ap1<=ap1;
   end case;
   else
   case a is
      when "01"=> ap2 <=ap2+2;
      when "10"=>ap2<=ap2+3;
      when others=>ap2<=ap2;
   end case;
   end if;
end if;
end process;
puntos<=ap1 when equipo='0' else
       ap2;
end Behavioral;
```

Como se puede apreciar el del registro depende de la pausa y del clk y del reset. El case representa si suma dos o tres puntos o mantiene y el if representa si le suma al equipo 1 o al 2. Y fuera del process vemos el multiplexor 2 a 1 para la salida.

Contador de posesión

El contador de posesión es el encargado de enviar un efecto cuando un jugador tiene el balón por más de 24 s, que tiene dos entradas sin contar el Reset y el clk que es uno para todos los bloques, la entrada pausa la cual debe pausar el conteo si esta activada, la entrada jugador nos indicara en que momento el balón cambia de dueño, sin embargo como las entradas de la FPGA no son suficientes usaremos la entrada equipo la cual no es tan relevante en los demás bloques de modo que si equipo es diferente a sus estado anterior el contador de 24 a 0 debe reiniciarse y si llega a 0 debe enviar una señal que ponga efecto en 1. Podemos ver el funcionamiento anterior con el siguiente diagrama:



Necesitaremos un registro para guardar a que equipo cuando este sea diferente del estado anterior debe reiniciar la máquina de estados anterior a 24 s. Este circuito lo podemos hacer con un F'F tipo D y una compuerta XOR entre la equipo presente y equipo pasado, y la salida de este al Reset de modo que si son diferentes se reinicie el contador.

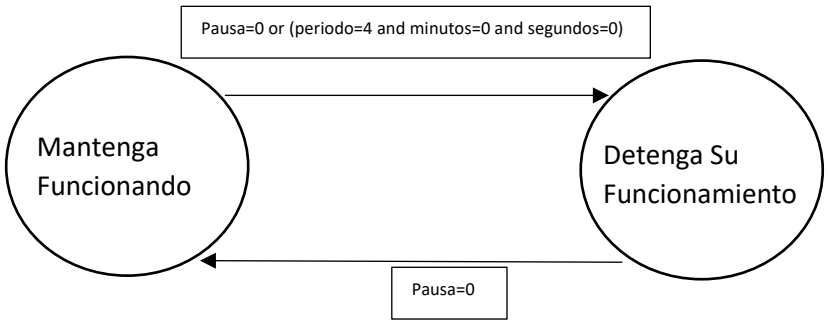
Y a la salida máquina de estados que actúa como contador podemos poner un comparador que cuando el estado de la sea 0 se detenga el contador y la salida sea 1, y en la visualización se produzca el efecto. De la descripción en VHDL tenemos:

```
architecture Behavioral of Posesion is
signal reloj: STD_LOGIC_VECTOR (4 downto 0);
signal equipop: STD_LOGIC;
signal igual :std_logic;

begin
igual<=((equipop)xor(equipo));
process(clk,reset,igual,equipo)
   begin
   if((reset='1')or(igual='1'))then
      reloj<="11000";
      efecto<='0';
      equipop<=equipo;
   elsif(clk'event and clk='1')then

   if(pausa='0')then
      if(reloj="00000")then
         efecto<='1';
      else
         reloj<=reloj-1;
      end if;
   else
         reloj<=reloj;
   end if;

   equipop<=equipo;

   end if;
end process;
posesion<=reloj;
end Behavioral;
```

Podemos ver la funcion igual reinicia la maquina de estados y que bajo el mismo clk esta esta el contador y el F'F que guarda el estado de equipo, ademas dentro para que el reloj cuente este debe estar despausado y la cuenta debe ser diferente a 0, lo cual corresponde al comparador. Y que en este caso se marca el efecto = 1 ademas que cuando se reinica la maquina de estado el el estado o es 0 de modo que el efecto =0.

Temporizador

Las entradas para este bloque serán aumentar minutos, aumentar segundos y la de pausa. De modo que sus salidas serán el cuarto del partido el minuto y el segundo en el que está. El diseño del temporizador será en alto nivel sin embargo lo modelaremos como una máquina de estados para la cual un estado será que entre en funcionamiento otra. El funcionamiento de la primera máquina de estados la podemos interpretar con el siguiente diagrama de estados:



Como vemos las entradas de esta máquina de estados será la pausa y el momento en el cual se encuentre el juego, pero como necesitamos que el cambio de funcionamiento será inmediato entonces esta máquina de estados actuara de forma inmediata esta solo tendrá un solo tendrá parte combinacional. Debemos tener en cuenta que no el aumento segundos o minutos no funciona si no está pausado, de modo que la entrada aumento y pausa son

las entradas de una función y aumento segundos o aumento minutos serán sus salidas correspondientes.

Para la máquina de estados interna tenemos 4 registros uno para unidades de segundo, uno para décimas de segundo, uno para minuto y uno para periodo. El registro de unidades de segundo debe tener una parte combinacional que cumpla con la siguiente tabla estado presente-Futuro:

Minutos Presentes(min)	**D. Segundo Presentes(dse)**	**U. segundos Presentes(use)**	**U. Segundos Futuros**
X	not (0)	not (0)	use=use-1
X	0	0	use=9

Para el registro de decenas de segundos debemos tener una parte combinacional que se encargue debe responder con la siguiente tabla de estado:

Aumento Segundos	**D. Segundo Presentes(dse)**	**U. segundos Presentes(use)**	**D. segundos Futuros**
0	X	not (0)	dse=dse
0	not (0)	0	dse=des-1
0	0	0	dse=5
1	x	not (5)	dse=dse+1
1	x	5	des=0

El registro que corresponde a minutos debe tener a su entrada una parte combinacional que cumpla con la siguiente tabla de estados:

Aumento Minutos	**Aumento Segundos**	**Minutos presentes(min)**	**D. Segundo Presentes(dse)**	**U. segundos Presentes(use)**	**Minutos Futuros**
0	0	not (0)	not (0)	x	min=min
0	0	not (0)	0	x	min=min-1
0	0	0	0	0	Min=12
1	0	x	x	x	min=min+1
0	1	x	not (5)	x	min=min
0	1	x	5	x	min=min+1

Y la tabla de estados para el periodo no tendremos en cuenta el periodo actual ya que la única condición es que, si el periodo el 4 se detenga, pero ya la tenemos presente en la máquina de estados principal, será la siguiente:

Minutos presentes(min)	D. Segundo Presentes(dse)	U. segundos Presentes(use)	Periodo(per)
not (0)	not (0)	not (0)	per=per
0	0	0	per=per+1

El diseño se realizó teniendo en cuenta las condiciones anteriores en alto nivel para mayor facilidad. De la descripción en VHDL, para la primera máquina de estados con parte netamente combinacional tenemos:

```
process(pausa,ami,adse,ause,aper)
begin
      if(((aper="11")and(ami="0000")and(adse="000")and(ause="0000"))or(pausa='1')) then
         aux<='1';
         else aux<='0';
      end if;
end process;
```

Para la maquina se estados segundaria tenemos:

```
process(clk,reset,ami,ause,aux,adse,aper)
   begin
   if(reset='1')then
      ause<="0000";
      adse<="000";
      ami<="1100";
      aper<="00";
   elsif(clk'event and clk='1')then
   case aux is
   when('0') =>
      if(ause="0000")then
         ause<="1001";
         adse<=adse-1;
      else ause<=ause-1;
      end if;
      if((adse="000")and(ause="0000")and(not(ami="0000")))then
         adse<="101";
         ami<=ami-1;
      end if;
      if(ami="0000" and adse="000" and ause="0000") then
         ause<="1001";
         ami<="1100";
         adse<="000";
         aper<=aper+1;
      end if;
   when others=>
   if((aumento='1')and(aumentomin='0'))then
         if(adse="101")then adse<="000";
         if(ami="1011") then ami<=ami;
         else           ami<=ami+1;
         end if;
         else
            adse<=adse+1;
         end if;
   elsif((aumentomin='1')and(aumento='0'))then
         if(ami="1011")then  ami<="0000";
         else ami<=ami+1;
         end if;
   else
      ause<=ause;
      adse<=adse;
      ami<=ami;
      aper<=aper;
   end if;
   end case;
end if;
```

Se puede observar que existen 4 registros ya que en el Reset los inicializamos y que la salida de del bloque combinacional anterior determinar si cuenta o mantiene su estado y que si mantiene su estado este puede aumentar el tiempo, además podemos ver que cuando cuenta las asignaciones del estado futuro de cada registro dependerá de los demás registros como se puede analizar en las tablas de estados anteriores

Visualización

Según las condiciones del diseño nos indica que la visualización debe ser por medio de TRISTATE de modo que cada uno de los modos de visualización será la salida de los 3 tristate de 8 bits, entonces mientras que uno de estos tiene el enable en 1 los demás lo deben de tener en 0 para que funcione de forma correcta. Otra condición del diseño es que cuando el efecto este en 1 en la visualización debe estar producirse un efecto diferente a la visualización normal. De modo que el circuito debe cumplir con la siguiente tabla de verdad

Efecto	Modo de visualización	Salida
0	00	Minutos & Segundos
0	01	Equipo & Puntos
0	10	Contador de Posesión
1	xx	00000000

Para que esto se cumpla al Habilitador de cada threestate debe de ir la salida correspondiente de un decodificador 2-4 or (not(efecto)) ya que si efecto es 1 las salidas de todas las compuertas or es 0 e inhabilitara todos los threestate. Del código en VHDL tenemos:

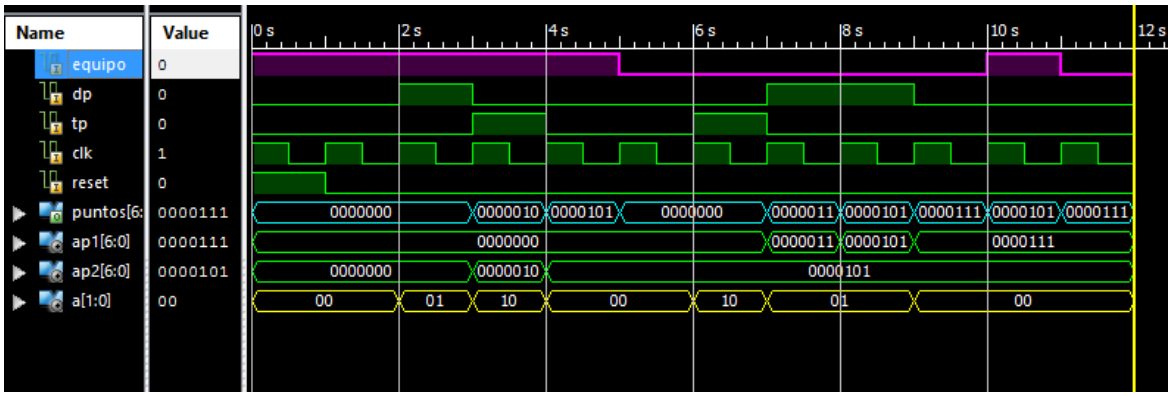
```
architecture Behavioral of Visualizacion is
signal y: std_logic_vector(2 downto 0);
begin
y(0)<=(not(enable)and(not(selec(0))))and(not(selec(1)));
y(1)<=(not(enable)and(selec(0))and(not(selec(1))));
y(2)<=(not(enable)and(not(selec(0))))and(selec(1));

salida<=('0')&(minutos&segundos) when y(0)='1' else
        "ZZZZZZZZ";
salida<=('0')&(segPosesion&periodo) when y(1)='1' else
        "ZZZZZZZZ";
salida<=puntaje&equipo when y(2)='1' else
         "ZZZZZZZZ";

end Behavioral;
```

Como el solo necesitamos 3 salidas del decodificador lo hacemos usando compuertas lógicas y los threestate por medio when- else similar a un multiplexor

Resultados y Análisis

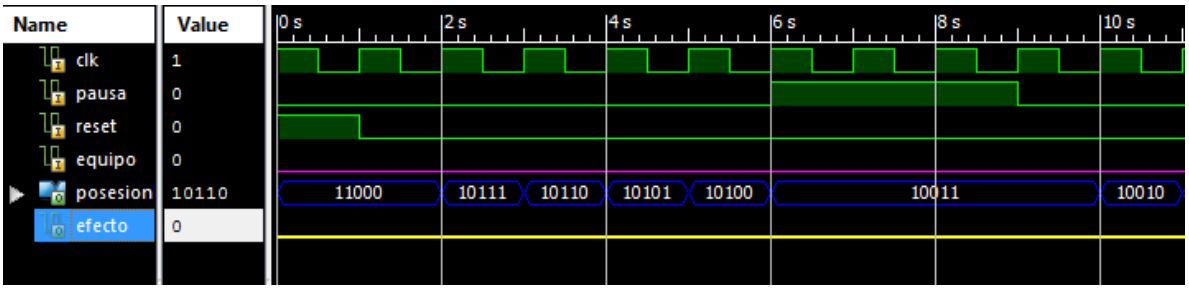
Simulación del circuito

- **Simulación para el bloque de Marcador**

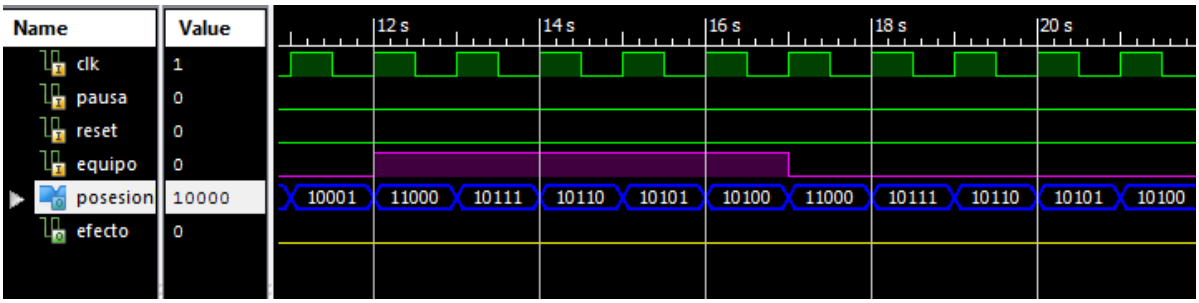


Vemos en la simulación como el reset inicializa los puntajes en cero para ambos equipos, siendo ap1 el puntaje del equipo ‘0’ y ap2 el puntaje del equipo ‘1’; sabemos que la máquina de estados que se implementó para este bloque, depende de las entradas ‘equipo’ (señal morada) y ‘a’ (señal amarilla) donde es simplemente la con concatenación de ‘dp’ (dos puntos) y ‘tp‘ (tres puntos), de esta forma la salida ‘puntos’ (señal azul claro) se cargara con el puntaje del equipo que este seleccionado en ese momento. Ya que inicialmente equipo es ‘1’ y después ‘dp’ y ‘tp’ tuvieron un pulso cada una al puntaje del equipo 1 se cargó con cinco puntos es decir ap2=’101’, y la salida puntos también. Luego equipo era ‘0’ y después ‘tp’ tuvo un pulso y ‘dp’ un pulso pero más largo que alcanzo a tomar dos flancos del reloj, haciendo que se cargara el puntaje del equipo 0 a siete puntos, es decir ap1=’111’ y la salida puntos también. Vemos como la salida se carga con el puntaje del equipo que se selecciona, cuando equipo es 0, puntos=ap1 y cuando equipo es 1, puntos =ap2.

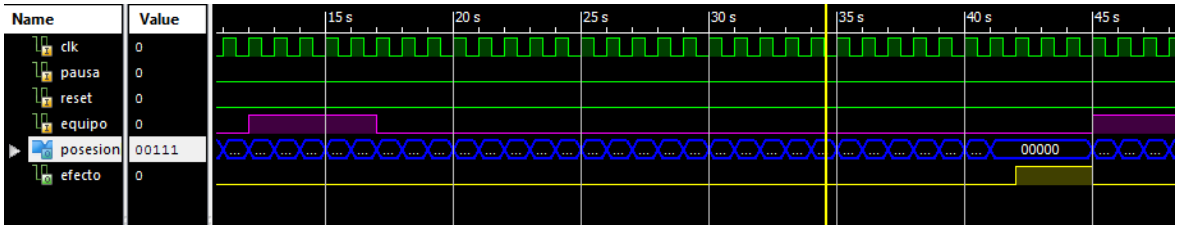
- **Simulación para el bloque de contador de posesión**



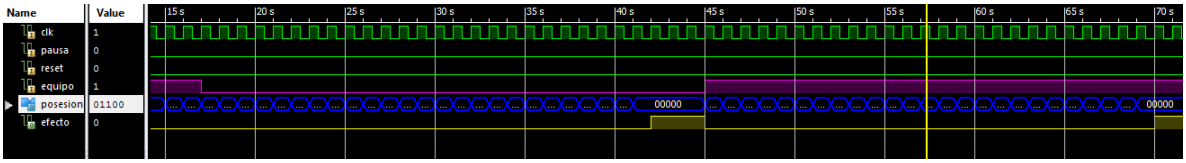
Observamos en la simulación que el reset inicializa el reloj de posesión (señal azul) en ‘11000’ que es 24 en binario, luego vemos como en cada segundo va decreciendo, hasta que pausa es ‘1’, en ese momento el reloj se detiene y se mantiene en ese valor hasta que pausa se vuelve ‘0’ de nuevo.



De igual forma cuando hay un cambio de estado en la entrada equipo (señal morada) el reloj se inicializa en ‘11000’ y empieza a decrecer hasta que halla una pausa o un cambio de equipio nuevamente.



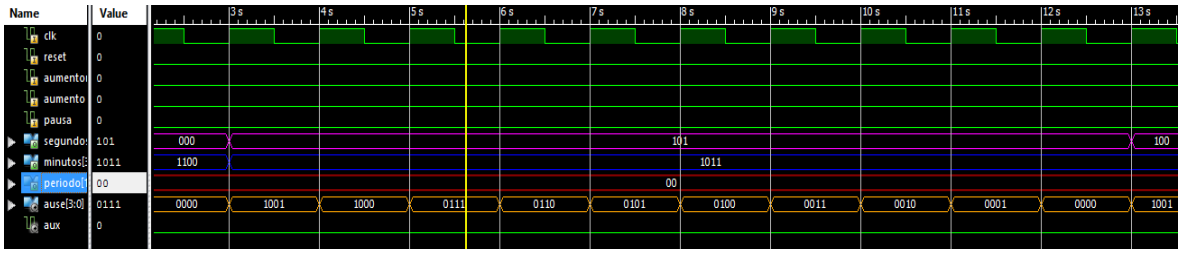
Vemos como el reloj por fin llega a ‘0’ por no haber un cambio de posesión de balón por 24 segundos y en ese valor se mantendrá hasta que haya un cambio de equipo.



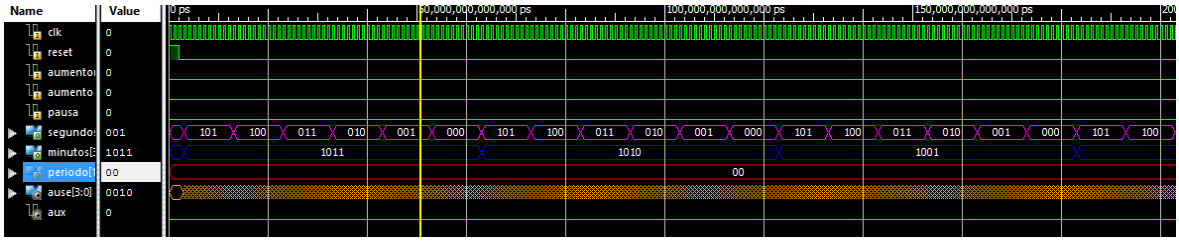
También se observa que cuando el reloj esta en cero la salida efecto (señal amarilla) toma el valor de ‘1’ independientemente de cual sea el equipo, y se mantendrá en ese estado hasta que haya un cambio de equipo donde se reinicia el reloj en ‘11000’ nuevamente. Después veremos cómo nos funcionara la salida efecto para el bloque de visualización.

- **Simulación del bloque del Temporizador**

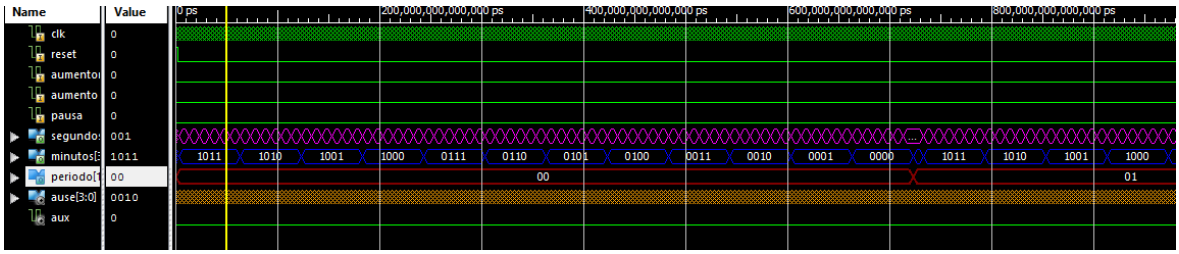
Para un mejor entendimiento de esta simulación: La señal naranja llamada ause es una señal que se refiere a las unidades de segundos, la señal morada llamada segundos es la señal que se refiere a las decenas de segundos, la señal azul llamada minutos efectivamente se refiere a los minutos que van de 0 a 12, la señal roja se refiere al cuarto, finalmente la entrada que esta de terceras llamada ‘aumentomin’ se refiere al aumento de los minutos y la cuarta llamada aumento se refiere al aumento de las decenas de segundos, parece que se llamaran igual pues el siguiente bloque recorto uno de los nombres.



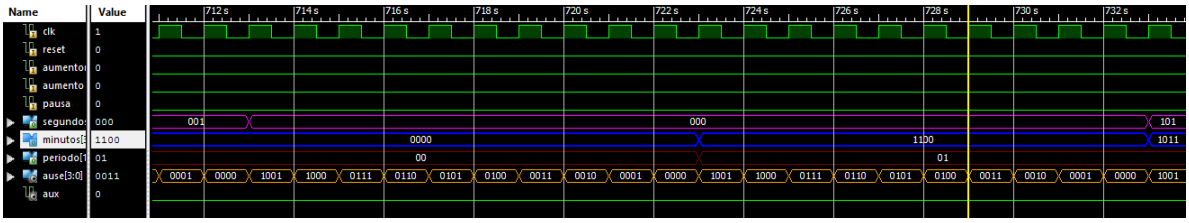
Primeramente vemos que los minutos, las decenas de segundos y las unidades de segundos, se inicializan en 12,0 y 0 respectivamente, y en el siguiente pulso cambian a 11,5 y 9 respectivamente, después se observa que las unidades de segundos van decreciendo en dada pulso mientras que las decenas de segundos y los minutos se mantienen en 5 y 11 eso hasta que las unidades de segundos llegan a 0 en ese instante las decenas de segundo cambian de valor a 4 y las unidades de segundo vuelven y toman el valor de 9.



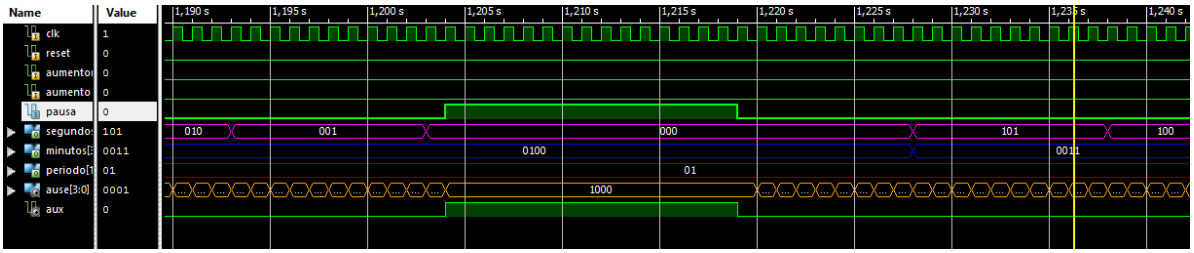
Ahora en esta simulacion se puede apreciar mejor lo que ya se dijo antes, cada 10 segundos las decenas de segundos van decreciendo desde 5, pero cuando llegan a cero, ahora son los minutos los que cambian de valor de 11 pasan a 10 y en ese mismo instante las decenas de segundos vuelven a tomar el valor de 5 y empizan a decrecer de nuevo y se repite el acontecimeinto, pero ahora los minutos pasan de 10 a 9.



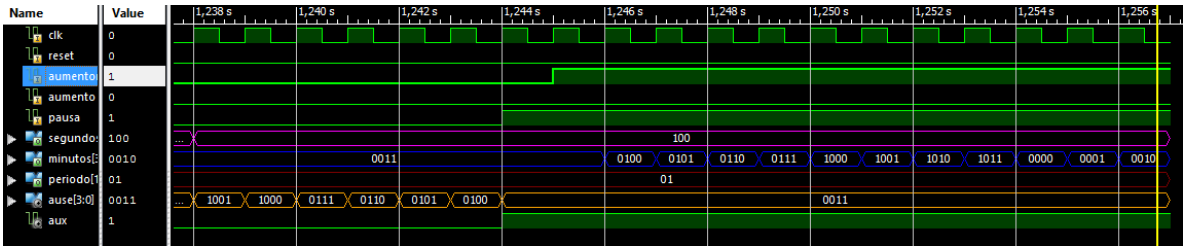
Con esta simulación se puede evidenciar mejor que los minutos van decreciendo cada 60 segundos empezando desde 12 hasta cero, y vemos que un poco después de que los minutos llegaron a cero el cuarto cambia y pasa de ‘00’ a ‘01’.



Acá podemos ver más de cerca cuando el cuarto hace el cambio, y se observa que hace el cambio justo cuando tanto los minutos, las decenas de segundos y las unidades de segundos están en cero, y cuando el cuarto cambia de valor, nuevamente los minutos, las decenas de segundos y las unidades de segundos vuelven a su estado inicial 12, 0 y 0 respectivamente. Ademas se aprecia que se cumple el requerimiento de que haya una espera de 10 segundos durante el cambio de cuarto, pues los minutos y las decenas de segundos se mantienen en ese valor mientras que las unidades de segundos hace la cuenta de los 10 segundo, y cuando por fin las unidades de segundos llegan a cero, los minutos y las decenas de segundos toman el valor de 11 y 5 respectivamente, tomando su comportamiento habitual, donde las decenas de segundos decrecen cada 10 segundos, y los minutos cada 60 segundos.

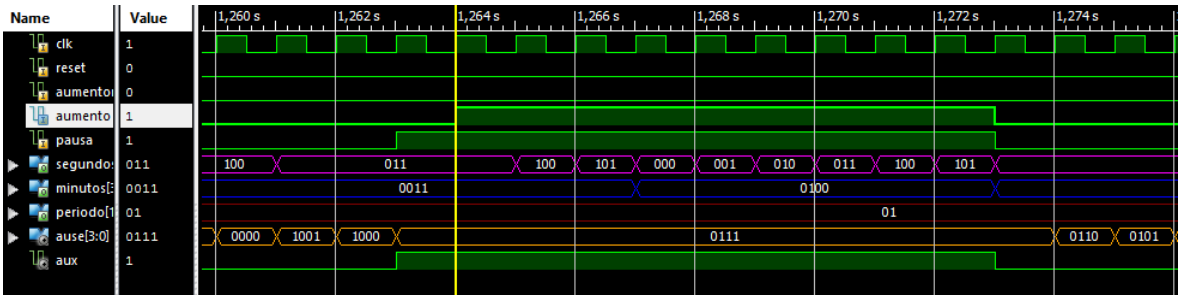


Ya que la máquina de estados que se implementó para este bloque depende de muchas entradas y señales, una de ellas es la pausa, pues ahora vemos que cuando la pausa es ‘1’ todo se detiene, los minutos, las decenas de segundos y hasta las unidades de segundos se mantiene su estado hasta que la pausa vuelva a ser cero.

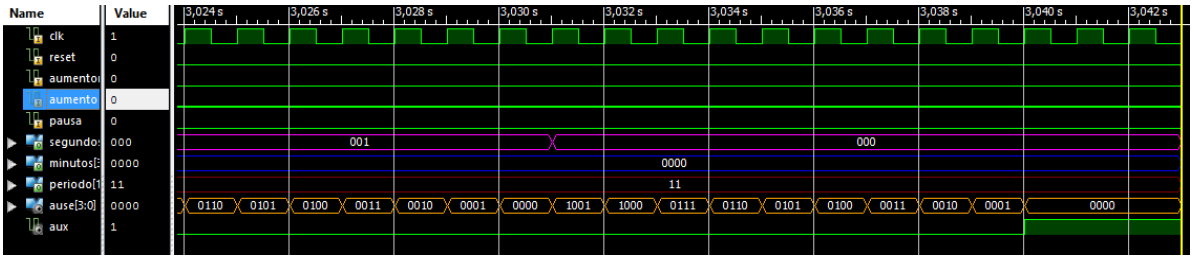


Acá podemos visualizar cómo funciona la opción de aumentar los minutos, primeramente para poder aumentar los minutos el juego debe estar pausado, o sea pausa=’1’ y el aumento

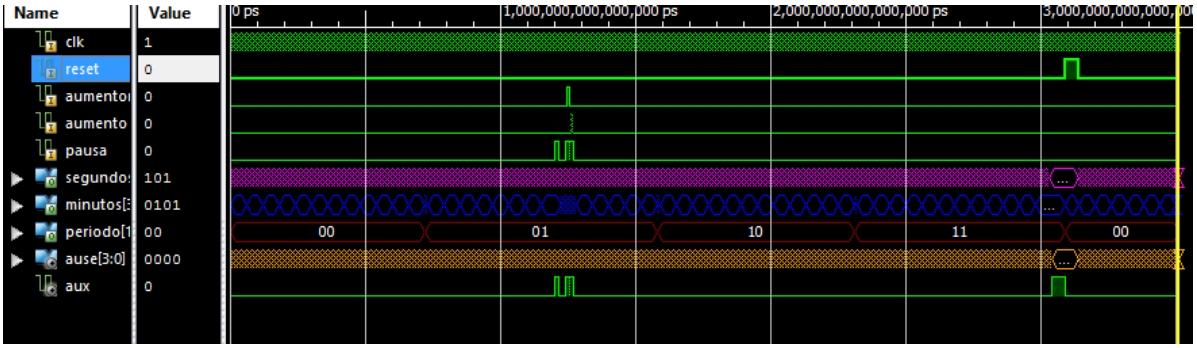
de minutos también en ‘1’, vemos como los minutos van aumentando en cada pulso de reloj mientras se mantenga tanto la pausa y el aumento de minutos en ‘1’, además se observa que el único que está cambiando son los minutos, las decenas de segundos y las unidades de segundos se mantienen intactas, pero cuando los minutos llegan a 11 no pasan a 12 ya que cada cuarto es de 12:00 min no de 12:43 min, cuando están en 11 pasan a cero y sigue aumentando, pero esto no genera que haya un cambio de cuarto.



Ahora vemos en esta simulacion como funciona la opcion de aumentar las decenas de segundos, primeramente para poder aumentar las decenas de segundos el juego debe estar pausado, o sea pausa=’1’ y el aumento de decenas de segundos también en ‘1’, vemos como las decenas de segundos van aumentando en cada pulso de reloj mientras se mantenga tanto la pausa y el aumento de las decenas de segundos en ‘1’, además se observa que el único que está cambiando son las decenas de segundos, los minutos y las unidades de segundos se mantienen intactas, pero cuando las decenas de segundos llegan cero no solo cambia las decenas de segundos de cero a cinco, sino que ahora los minutos cambian y se incrementan de tres a cuatro minutos.

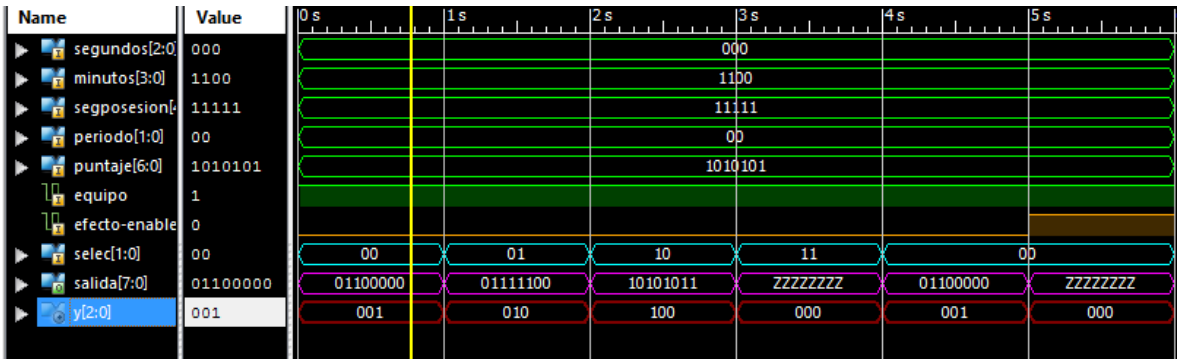


En esta simulación lo que podremos ver es que sucede cuando se acaba el último cuarto, y cómo podemos ver el proceso de conteo se pausa, manteniendo el reloj en ceros por completo.



Finalmente, la única forma para poder iniciar un nuevo es poniendo un ‘1’ en el reset por un momento y así todo los valores se reiniciaran, poniendo el reloj nuevamente en 12:00 y el cuarto volverá a ser ‘00’.

- **Simulación del bloque de Visualización**



Vamos en la simulación que la salida (señal morada) que es nuestra visualización, depende solamente de la entrada de selección (señal azul clara) y la entrada efecto (señal amarilla, salida del bloque de reloj de posesión), y que este bloque se implementó con un DEC 2-3 y tres three state, cuando la selección toma el valor de ‘00’ está eligiendo el primer modo de visualización, pues está haciendo que la salida del DEC que es ‘Y’ (señal roja) en este caso tome el valor de ‘001’ y como ‘Y’ es el habilitador de los three state, de esta forma está habilitando el primer three state y la salida se carga con ‘0’&minutos&segundos, después la selección toma el valor de ‘01’ escogiendo el segundo modo de visualización, habilitando el segundo three state y la salida se cargara con ‘0’&segPosesion&periodo, luego la selección toma el valor de ‘10’ haciendo que el modo de visualización sea el tercero habilitando el ultimo three state y de esta manera la salida se cargara con puntaje&equipo, cuando se selección toma el valor de ‘11’ la salida se vuelve ‘Z’ ya que de esta manera no está habilitando ninguno de los three state pues ‘Y’ está en ‘000’. Finalmente vemos que efecto durante todo este tiempo estuvo en ‘0’, pero cuando este toma el valor de ‘1’, sin

importar la selección la salida se cargara con ‘Z’ ya que efecto funciona como el enable del DEC y como ya se sabe cuándo el enable de un DEC es ‘1’ todas sus salidas son ‘000’, por eso en ese caso ‘Y’=’000’ y esto provoca que ningún three state este habilitado.

Paramentos de desempeño

En bajo nivel

- **Numero de recursos:**

```
Device utilization summary:
---------------------------

Selected Device : 3s700anfgg484-5

 Number of Slices:                       84  out of   5888      1%
 Number of Slice Flip Flops:             60  out of  11776      0%
 Number of 4 input LUTs:                157  out of  11776      1%
 Number of IOs:                          18
 Number of bonded IOBs:                  18  out of    372      4%
 Number of GCLKs:                         2  out of     24      8%
```

- **Tiempo de retardo:**

```
=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 49 / 8
-------------------------------------------------------------------------
Delay:               8.127ns (Levels of Logic = 4)
  Source:            equipo (PAD)
  Destination:       salida<6> (PAD)

  Data Path: equipo to salida<6>
                              Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            20   0.824   1.003  equipo_IBUF (equipo_IBUF)
     LUT4:I1->O            1   0.562   0.423  salida<6>_MLTSRCEDGELogicTrst31 (salida<6>_MLTSRCEDGELogicTrst31)
     LUT3:I1->O            1   0.562   0.357  salida<6>_MLTSRCEDGELogicTrst33 (salida<6>_MLTSRCEDGE)
     OBUFT:I->O                4.396          salida_6_OBUFT (salida<6>)
    ----------------------------------------
    Total                      8.127ns (6.344ns logic, 1.783ns route)
                                       (78.1% logic, 21.9% route)

=========================================================================
```

Conclusiones

- Podemos ver que las máquinas de estados no solo sirven para ver el estado actual del circuito sino también las podemos usar para indicar la función que debe hacer un circuito.
- Es importante identificar claramente sí que parte del circuito puede ser combinacional y que parte puede ser secuencial antes de hacer una descripción de hardware.
- Con el método de camino de datos y unidad de control es posible hacer el temporizador ya que la unidad de control será una máquina de estados que le dirá a una combinación de registros, ALU's, contadores o máquinas de estados su tarea a realizar.
- Podemos usar Threestate en Ves de Multiplexores para la visualización dinámica ahorrando así algunos recursos
- Cuando tenemos una condición que reinicia al estado inicial lo mejor para evitar posibles errores o aumentar la complejidad de la máquina de estados que se está describiendo es hacer un bloque combinacional que vaya al Reset
- A la hora de describir hardware lo más óptimo, es poner todas los condicionales o asignaciones que pueden ser combinacionales a fuera de la condición del clock, ya que nos puede generar más FF's de los necesarios.