

# Sistemas Digitales 1

## Sistema de Puntuación Automático para un Juego de Baloncesto

<b>Integrante 1</b>	Andrea Elizabeth Mendoza Valarezo
<b>Integrante 2</b>	Marcos Xavier Veliz Moran
<b>Paralelo</b>	112
<b>Fecha</b>	23 de enero del 2024

### 1. Introducción

Como parte del sistema de reglamentación y arbitraje en los partidos de baloncesto, los tableros forman parte importante para saber qué equipo lleva ventaja con relación al otro en el desarrollo del partido, es por esto que la detección del ingreso del balón al aro debe ser contabilizada para así ir acumulando puntos y saber el resultado final del partido en cuestión. Como parte de la solución, se propone un sistema controlado basado en lógica combinatoria donde el conteo de puntos depende netamente del sensor de presencia la cual por cada aro correctamente encestando el contador va aumentando.

### 2. Antecedentes/Descripción del proyecto

Ante la necesidad de tener un conteo claro y visible para todos los equipos y el público en general, las personas reguladoras y gestoras en los partidos de baloncesto plantearon el uso de paneles para llevar el puntaje. Es por esto que con la ayuda de los dos estudiantes de Sistemas Digitales 1 se plantean distintos requerimientos como:

- Se utilizarán sensores de presencia (S) que enviarán una señal "1" que se guardará en el contador de cada equipo.
- Para mostrar el puntaje que cada equipo tiene se utilizará una pantalla de 7 segmentos para así visualizar si aumenta o disminuye la cantidad de puntos por cada equipo.

- El sistema funcionará siempre y cuando el botón de arranque haya sido accionado ( $START.H = "1"$ ), en caso de que se accione el botón de reinicio ( $RESET.L = "0"$ ), el sistema se apagará borrando todo registro almacenado.
- Cada equipo tendrá un botón de anulación ( $ANULAR1.H = "1"$ ) y ( $ANULAR2.H = "1"$ ) la cual será accionado por el árbitro para restar el puntaje actual en el contador.
- Un led verde se encenderá para saber qué equipo ha ganado y un led rojo se encenderá al haber una anulación por parte del árbitro.

### **3. Objetivo General**

Mostrar el puntaje respectivo de cada equipo en un partido de baloncesto, mediante un sistema controlador basado en lógica combinatoria por contadores para identificar de manera rápida y eficiente el equipo vencedor.

### **4. Objetivos Específicos**

- Identificar e interpretar todas las variables de interés.
- Definir las variables de entradas, salidas y habilitadores que permitan el correcto funcionamiento del sistema.
- Diseñar un diagrama de bloques que permita visualizar de mejor manera el sistema.
- Identificar los casos la cual son los casos posibles y cuáles no mediante una tabla de verdad del sistema controlador.
- Simular los datos obtenidos para su correcta validación y funcionamiento.

### **5. Descripción de la solución**

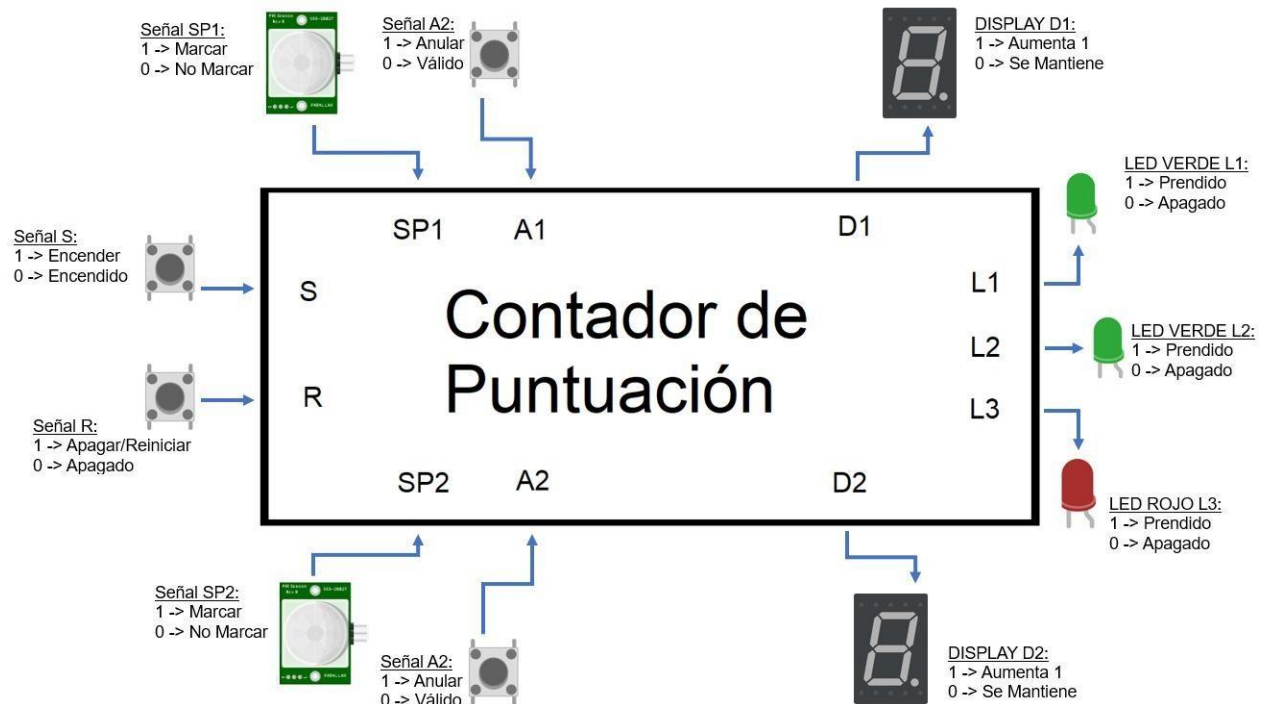
El diseño se basa en un sistema lógico combinatorial de 5 entradas y 5 salidas, la cual, dos salidas recibirán señales de 7 bits para así lograr ser mostradas en un display y el resto serán solo de 1 bit. Para poder iniciar el sistema se debe presionar el botón de inicio (S.H)

## 5.1. Entradas y salidas

Entradas	
S.H	Esta señal funciona como habilitador, la cual envía la señal “1” al ser accionada.
R.L	Esta señal funciona como deshabilitador, que carga la señal de “0” al ser accionada.
SP1.H	Señal enviada por el sensor de presencia del equipo 1 la cual será “1” cuando el balón ingrese al aro y “0” cuando no ingrese.
SP2 .H	Señal enviada por el sensor de presencia del equipo 2 la cual será “1” cuando el balón ingrese al aro y “0” cuando no ingrese.
A1.H	Esta señal indica la anulación del punto al equipo 1 si su señal lógica es “1”, en caso de ser “0” el punto cuenta como válido.
A2.H	Esta señal indica la anulación del punto al equipo 2 si su señal lógica es “1”, en caso de ser “0” el punto cuenta como válido.

Salidas	
D1.H	Es la señal de 7 bit que recibe el display del equipo 1 por parte del contador
D2.H	Es la señal de 7 bit que recibe el display del equipo 2 por parte del contador
L1.H	Esta señal al estar en estado lógico “1” encenderá el led de color verde del equipo 1 señalando que ganó o enviará la señal de “0” y el led no encenderá
L2.H	Esta señal al estar en estado lógico “1” encenderá el led de color verde del equipo 2 señalando que ganó o enviará la señal de “0” y el led no encenderá
L3.H	Al estar en esta lógico “1” encenderá el led rojo indicando anulación de puntos o la señal “0” para no encender el led y el punto es válido

## 5.2. Diagrama de bloques de la solución



## 5.3. Descripción de bloques

Entra la señal de encendido, se activa el funcionamiento del contador, se tiene también los sensores de presencia para cada equipo donde en cada cesta se sumará un punto y se mostrará en el display correspondientes.

En caso de anular alguna puntuación, se pulsa el anulador para restar el puntaje y al mismo tiempo se activará el led 3 de color rojo.

Para que se activen los leds 1 y 2 de color verde se compara el resultado de cada equipo, donde se activa uno de los dos leds.

### 5.3.1. Tabla de verdad del controlador

entradas						salidas				
S.H	R.L	SP1.H	SP2.H	A1.H	A2.H	D1.H	D2.H	L1.H	L2.H	L3.H
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0
0	0	1	1	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0
0	1	0	0	1	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0
0	1	0	1	0	1	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	0
0	1	0	1	1	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	0	1	0	0	0	0	0	0
0	1	1	0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0
0	1	1	1	0	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0
0	1	1	1	1	1	0	0	0	0	0
1	0	0	0	0	0	1	1	Ø	Ø	0

1	0	0	0	0	1	1	1	Ø	Ø	1
1	0	0	0	1	0	1	1	Ø	Ø	1
1	0	0	0	1	1	1	1	Ø	Ø	1
1	0	0	1	0	0	1	1	Ø	Ø	0
1	0	0	1	0	1	1	1	Ø	Ø	1
1	0	0	1	1	0	1	1	Ø	Ø	1
1	0	0	1	1	1	1	1	Ø	Ø	1
1	0	1	0	0	0	1	1	Ø	Ø	0
1	0	1	0	0	1	1	1	Ø	Ø	1
1	0	1	0	1	0	1	1	Ø	Ø	1
1	0	1	0	1	1	1	1	Ø	Ø	1
1	0	1	1	0	0	1	1	Ø	Ø	0
1	0	1	1	0	1	1	1	Ø	Ø	1
1	0	1	1	1	0	1	1	Ø	Ø	1
1	0	1	1	1	1	1	1	Ø	Ø	1
1	1	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	0	0	1	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	0
1	1	0	1	1	1	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0
1	1	1	0	1	1	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	0	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0

### 5.3.2. Expresión lógica del comportamiento del controlador

$$\begin{aligned}
 D1.H &= S.H * \overline{RL} \\
 D2.H &= S.H * \overline{RL} \\
 L1.H &= S.H * \overline{RL} \\
 L2.H &= S.H * \overline{RL} \\
 L3.H &= (S.H * \overline{RL} * A1.H) + (S.H * \overline{RL} * A2.H)
 \end{aligned}$$

## 6. Ejemplos de funcionamiento

### Ejemplo 1

	S.H	R.L	SP1.H	SP2.H	A1.H	A2.H	D1.H	D2.H	L1.H	L2.H	L3.H
CASO 1	1	0	0	1	1	1	1	1	Ø	Ø	1
CASO 2	1	0	1	0	0	0	1	1	Ø	Ø	0

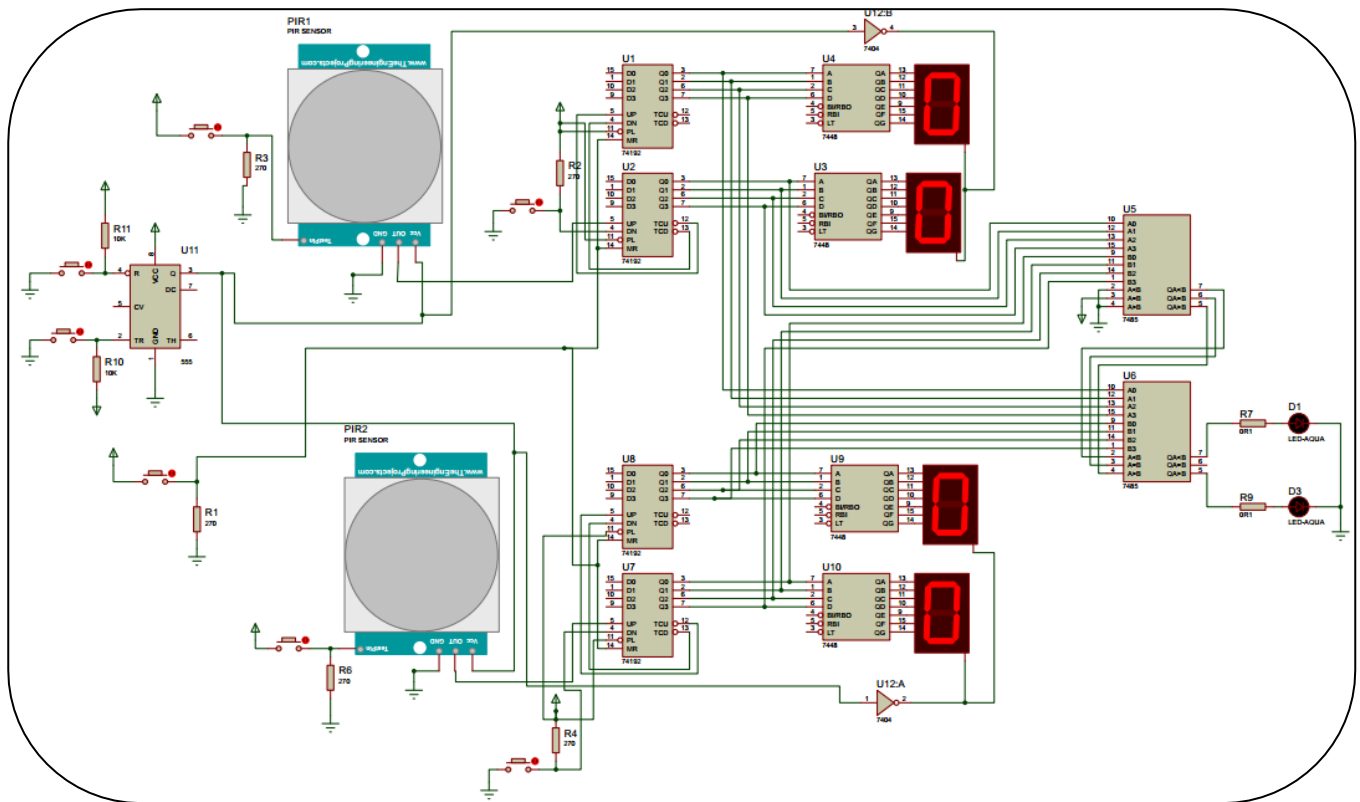
- Caso 1: El habilitador esta activado, entonces los display marcan el valor del puntaje y los leds se activarán según qué equipo vaya ganando. Los pulsadores de anulación 1 y 2 están activados por lo que el led 3 también se activa.
- Caso 2: Sucede casi lo mismo que en el caso 1, a excepción de los pulsadores de anulación 1 y 2, en este caso, se encuentran desactivados entonces el led 3 se mantiene apagado.

### Ejemplo 2

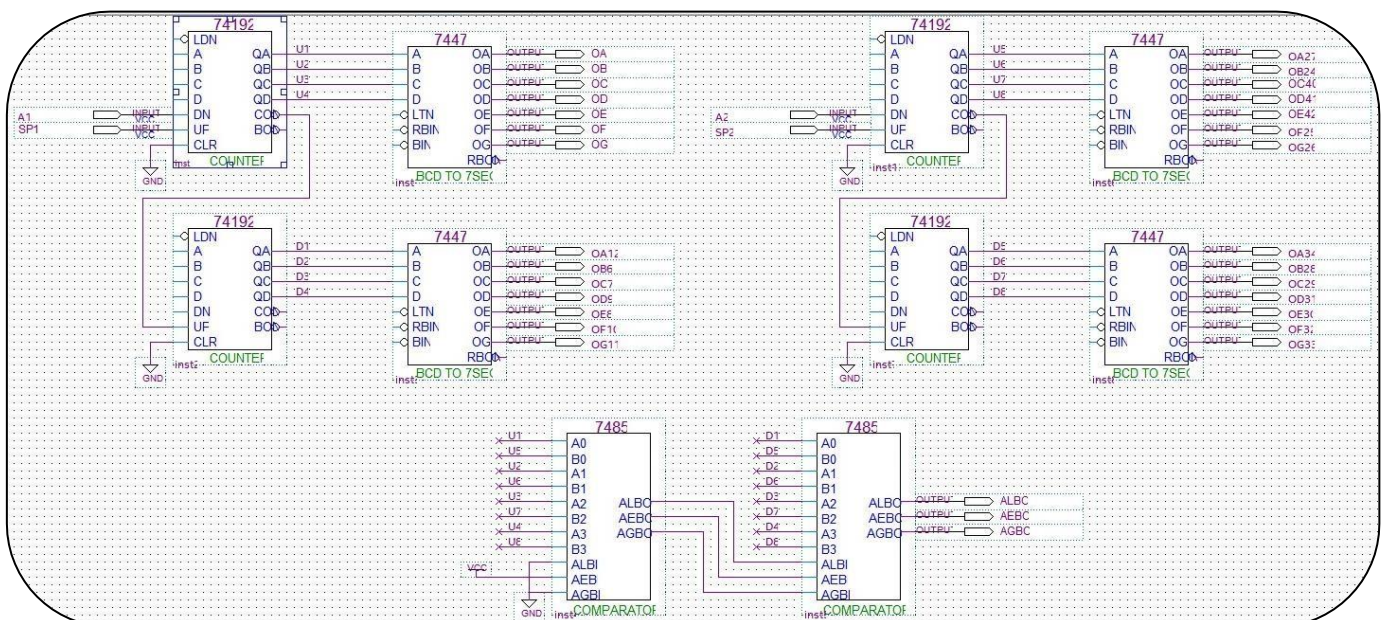
	S.H	R.L	SP1.H	SP2.H	A1.H	A2.H	D1.H	D2.H	L1.H	L2.H	L3.H
CASO 1	0	0	1	1	1	1	0	0	0	0	0
CASO 2	0	1	0	0	0	0	0	0	0	0	0

- Caso 1: No se activan las salidas, debido a que el habilitador S.H esta apagado, es decir los displays no marcan ningún valor, y los leds tampoco se encienden
- Caso 2: Es igual al caso 1, tampoco influye el reset en las salidas.

## 7. Diagrama de Bloques



*Ilustración 1*



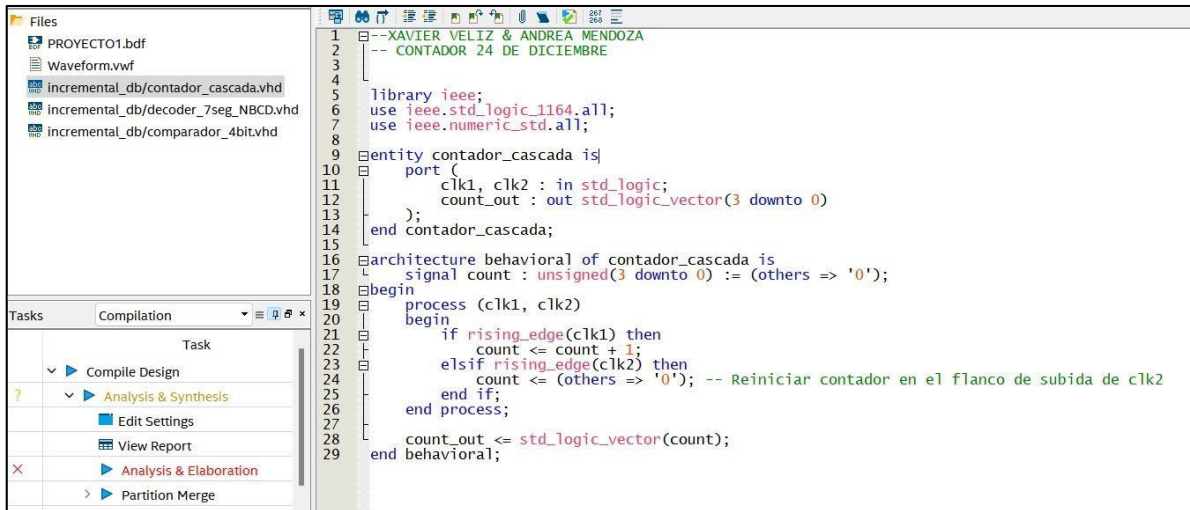
*Ilustración 2*



## 8. VHDL y Compilación

### 8.1. CONTADOR CODIGO

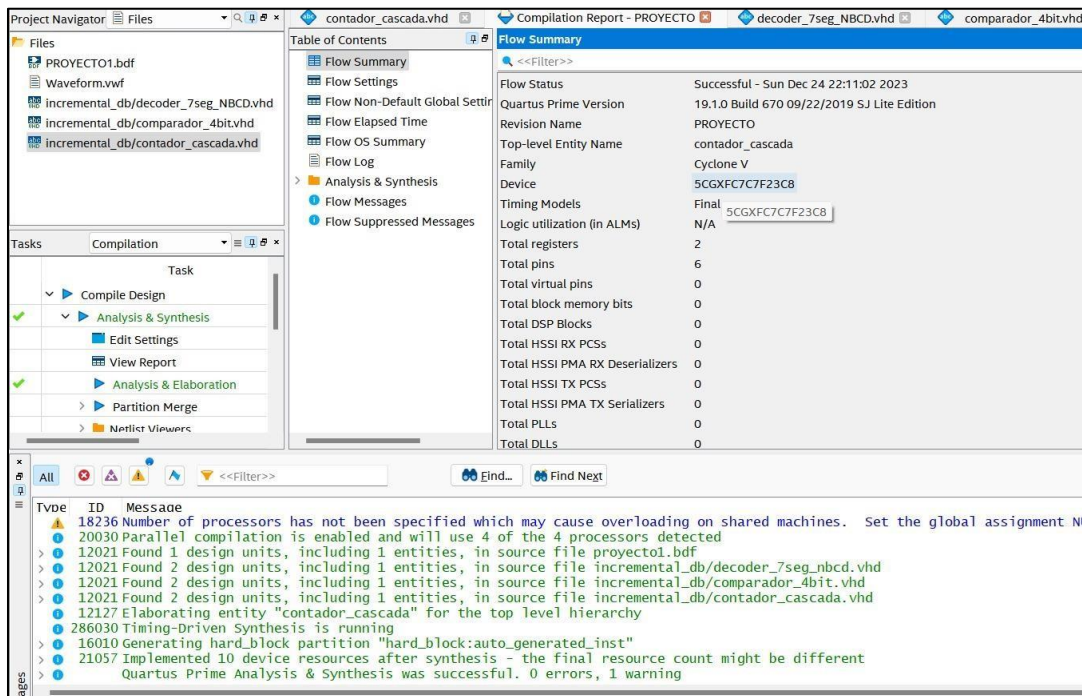
#### CODIGO



```
1  --XAVIER VELIZ & ANDREA MENDOZA
2  -- CONTADOR 24 DE DICIEMBRE
3
4  library ieee;
5  use ieee.std_logic_1164.all;
6  use ieee.numeric_std.all;
7
8  entity contador_cascada is
9  port (
10     clk1, clk2 : in std_logic;
11     count_out : out std_logic_vector(3 downto 0)
12 );
13 end contador_cascada;
14
15 architecture behavioral of contador_cascada is
16     signal count : unsigned(3 downto 0) := (others => '0');
17 begin
18     process (clk1, clk2)
19     begin
20         if rising_edge(clk1) then
21             count <= count + 1;
22         elsif rising_edge(clk2) then
23             count <= (others => '0'); -- Reiniciar contador en el flanco de subida de clk2
24         end if;
25     end process;
26
27     count_out <= std_logic_vector(count);
28 end behavioral;
```

Ilustración 3

#### COMPILACION



The screenshot displays the Quartus IDE interface during the compilation phase. The 'Project Navigator' on the left shows the project files, including 'PROYECTO1.bdf', 'Waveform.vwf', and several VHDL files in the 'incremental\_db' directory. The 'Tasks' pane on the left indicates that the 'Analysis & Synthesis' task is completed, marked with a green checkmark.

The main window shows the 'Flow Summary' report for the 'contador\_cascada' entity. The report details the compilation process, including the flow status, version, and various resource utilization metrics.

Flow Status	Successful - Sun Dec 24 22:11:02 2023
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	PROYECTO
Top-level Entity Name	contador_cascada
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	2
Total pins	6
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

The 'Messages' pane at the bottom shows a list of messages, including a warning about the number of processors and a message indicating that the synthesis was successful with 0 errors and 1 warning.

Ilustración 4

## 8.2. DECODER DE 7 SEGMENTOS

### CODIGO

```

1  --XAVIER VELIZ & ANDREA HENAOZA
2  -- DECODER 24 DE DICIEMBRE
3
4  library IEEE;
5  use IEEE.std_logic_1164.all;
6
7  entity decoder_7seg_NBCD is
8      port (
9          input_nbcd : in std_logic_vector(3 downto 0); -- Entrada en NBCD (dos dígitos BCD)
10         seg_out : out std_logic_vector(6 downto 0); -- Salida a los segmentos del display de siete segmentos
11     );
12 end decoder_7seg_NBCD;
13
14 architecture behavioral of decoder_7seg_NBCD is
15 begin
16     process(input_nbcd)
17     begin
18         case input_nbcd is
19             when "0000" =>
20                 seg_out <= "1111110"; -- Valor correspondiente al número 0 en siete segmentos
21             when "0001" =>
22                 seg_out <= "0110000"; -- Valor correspondiente al número 1 en siete segmentos
23             when "0010" =>
24                 seg_out <= "1101101"; -- Valor correspondiente al número 2 en siete segmentos
25             when "0011" =>
26                 seg_out <= "1111001"; -- Valor correspondiente al número 3 en siete segmentos
27             when "0100" =>
28                 seg_out <= "1011001"; -- Valor correspondiente al número 4 en siete segmentos
29             when "0101" =>
30                 seg_out <= "1011011"; -- Valor correspondiente al número 5 en siete segmentos
31             when "0110" =>
32                 seg_out <= "1011111"; -- Valor correspondiente al número 6 en siete segmentos
33             when "0111" =>
34                 seg_out <= "1110000"; -- Valor correspondiente al número 7 en siete segmentos
35             when "1000" =>
36                 seg_out <= "1111111"; -- Valor correspondiente al número 8 en siete segmentos
37             when "1001" =>
38                 seg_out <= "1111011"; -- Valor correspondiente al número 9 en siete segmentos
39             when "1010" =>
40                 seg_out <= "1111011"; -- Valor correspondiente al número 9 en siete segmentos
41             when "1011" =>
42                 seg_out <= "1111011"; -- Valor correspondiente al número 9 en siete segmentos
43             when "1100" =>
44                 seg_out <= "1111011"; -- Valor correspondiente al número 9 en siete segmentos
45             when "1101" =>
46                 seg_out <= "1111011"; -- Valor correspondiente al número 9 en siete segmentos
47             when "1110" =>
48                 seg_out <= "1111011"; -- Valor correspondiente al número 9 en siete segmentos
49             when "1111" =>
50                 seg_out <= "1111011"; -- Valor correspondiente al número 9 en siete segmentos
51             when others =>
52                 seg_out <= "0000000"; -- Si la entrada no es válida, apagar todos los segmentos
53         end case;
54     end process;
55 end behavioral;

```

Ilustración 5

## COMPILACION

**Flow Summary**

Flow Status	Successful - Sun Dec 24 22:11:35 2023
Quartus Prime Version	19.1.0 Build 670 09/22/2019 SJ Lite Edition
Revision Name	PROYECTO
Top-level Entity Name	decoder_7seg_NBCD
Family	Cyclone V
Device	5CGXFC7C723C8
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	0
Total pins	11
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

**Messages**

18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM\_PARALLEL to 20030. Parallel compilation is enabled and will use 4 of the 4 processors detected.

12021 Found 1 design units, including 1 entities, in source file proyecto1.bdf

12021 Found 2 design units, including 1 entities, in source file incremental\_db/decoder\_7seg\_nbcd.vhd

12021 Found 2 design units, including 1 entities, in source file incremental\_db/comparador\_4bit.vhd

12021 Found 2 design units, including 1 entities, in source file incremental\_db/contador\_cascada.vhd

12127 Elaborating entity "decoder\_7seg\_NBCD" for the top level hierarchy

286030 Timing-Driven Synthesis is running

16010 Generating hard\_block partition "hard\_block:auto\_generated\_inst"

21057 Implemented 18 device resources after synthesis - the final resource count might be different

Quartus Prime Analysis & Synthesis was successful. 0 errors, 1 warning

Ilustración 6

### 8.3. COMPARADOR DE 4 BITS CODIGO

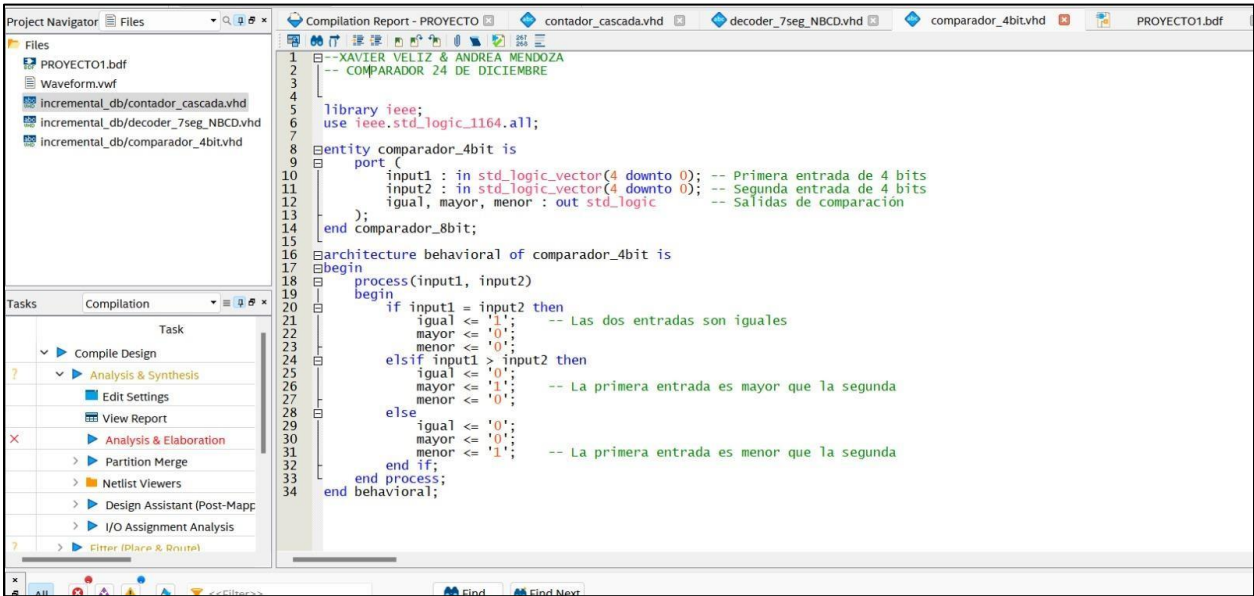


Ilustración 7

### COMPILACION

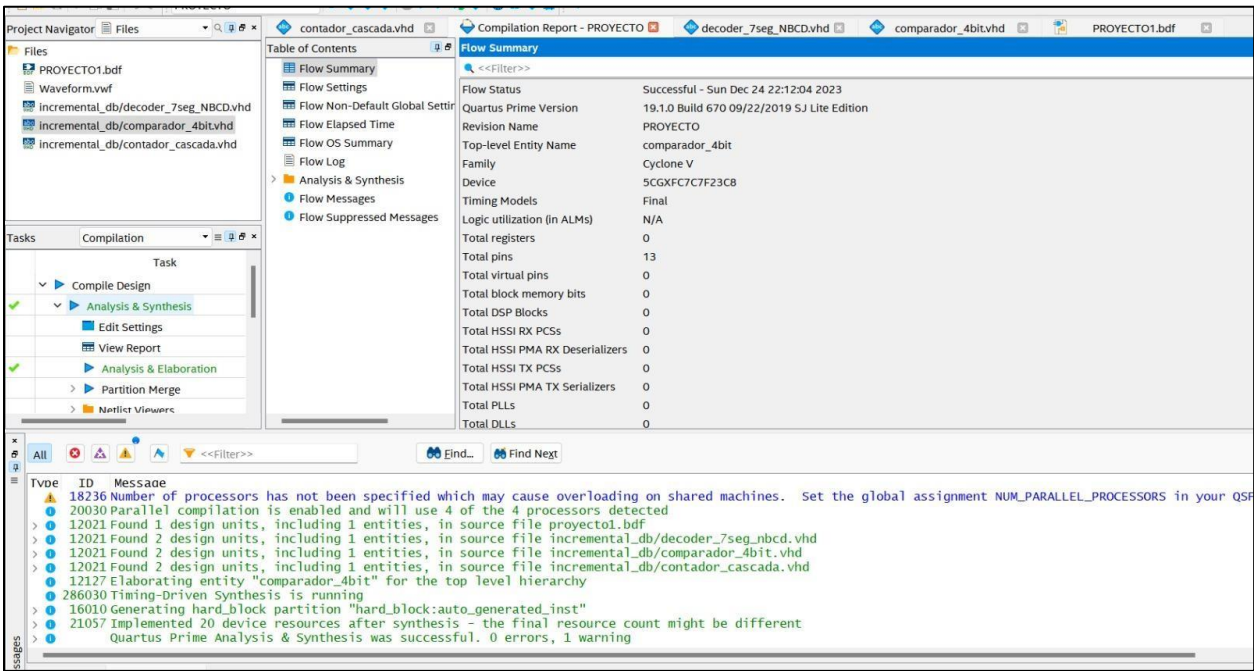


Ilustración 8

*Como acotación a lo realizado en el avance, en nuestro proyecto hemos decidido usar los integrados, enseñados en clase (a excepción de los contadores, puesto que, son temas de la siguiente unidad) para hacer la implementación física. Solo realizo los bloques de los integrados que usamos en el diagrama de bloque, como son los contadores los decoders y los comparadores; la implementación en el diagrama de bloques se la realizo con integrados comerciales.*

## **9. RESULTADOS**

- Entrada de la señal del sensor, marca la cantidad de balones encestandos, se puede visualizar estos resultados en los display de 7 segmentos
- Comparación entre los puntajes de los jugadores se visualiza según los led verdes para identificar el ganador.
- Se efectúa la resta uno a uno del puntaje en caso de fallo por medio de un botón restador.
- En caso de fallo en el puntaje, la entrada del botón restador, se puede visualizar si funciona correctamente según el led rojo.

## **10. CONCLUSIONES**

- Se logro identificar las entradas necesarias para el correcto funcionamiento del contador de puntaje para baloncesto.
- El uso de los bloques escogidos fue el adecuado para el correcto conteo al igual que la resta del puntaje.
- La conexión de los integrados fue la acertada permitiendo que se pueda visualizar correctamente el puntaje de cada equipo a su vez la activación de los botones y por último la señal de los leds del equipo ganador.