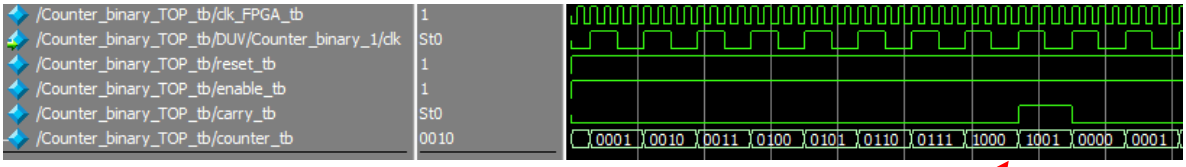
Contador binario

```
always@(posedge clk or negedge reset) begin
   if (reset == 1'b0)
      counter_reg <= {NBITS{1'b0}};
   else begin
      if(enable == 1'b1) begin
         if(counter_reg == MAXIMUM_VALUE)

            counter_reg <= 0;
         else
            counter_reg <= counter_reg + 1'b1;
      end
   end
end

assign counter = counter_reg;
```

```
always@(counter_reg)begin
   if(counter_reg == MAXIMUM_VALUE)
      MaxValue_Bit = 1;
   else
      MaxValue_Bit = 0;
end
```

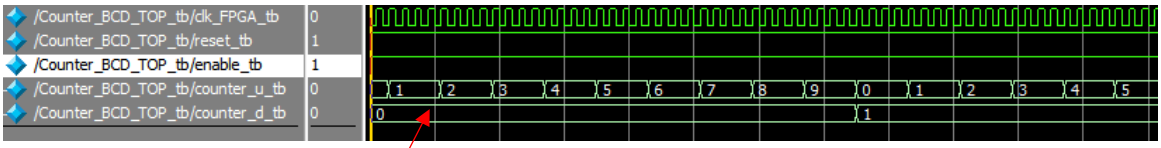


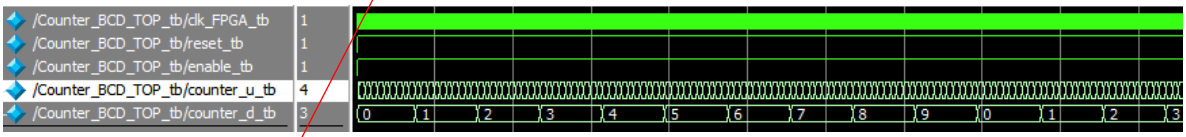

```
initial begin
   forever #10 clk_FPGA_tb = !clk_FPGA_tb;
end

initial begin
   #0 reset_tb = 1;
end

initial begin
   #0 enable_tb = 1;
end
```

Se usa este carry para usar este contador como celda unitaria para el BCD. Carry = 1 cuando se alcanza el conteo máximo

Contador BCD

```
Counter_Units
(
   .clk(Output_clk_1Hz_wire),
   .reset(reset),
   .enable(enable),

   .carry(Output_carry_wire),
   .counter(counter_u)
);

Counter_binary
#(
   .MAXIMUM_VALUE(MAXIMUM_VALUE),
   .NBITS(NBITS)
)
Counter_Tens
(
   .clk(Output_clk_1Hz_wire),
   .reset(reset),
   .enable(Output_carry_wire),

   .counter(counter_d)
);
```



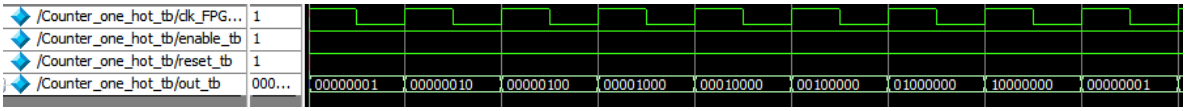


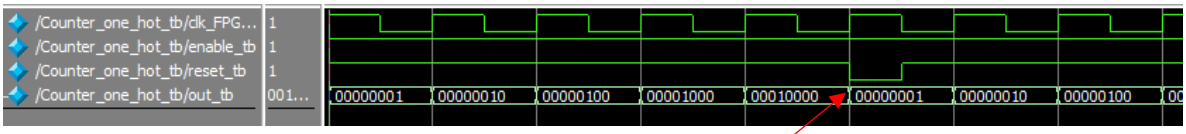
Aquí se aprecian las unidades

Aquí se aprecian las decenas

Contador one hot

```
always @ (posedge clk)
   if (!reset)
   begin
      out <= 8'b0000_0001;
   end
   else
   begin
      if (enable)
         out <= {out[6],out[5],out[4],out[3],out[2],out[1],out[0],out[7]};
   end
```



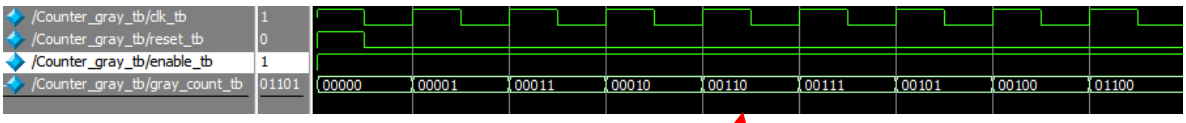



```
initial begin
    #0 reset_tb = 1;
    #100 reset_tb = 0;
    #10 reset_tb = 1;
end
```

Se observa la función de reinicio

Contador GRAY

Para este se utilizó un el contador gray que viene incluido en los template y solo se realizó el test bench.

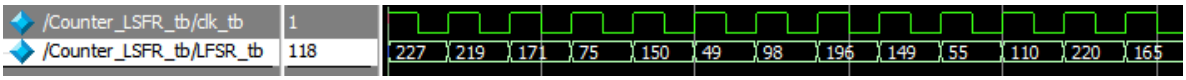



```
initial begin
   forever #10 clk_tb = !clk_tb;
end

initial begin
   #0 reset_tb = 1;
   #10 reset_tb = 0;
end

initial begin
   #0 enable_tb = 1;
end
```

Se observa que el cambio solo re realiza de un bit

Contador LFSR

```
always @(posedge clk)
begin
  LFSR[0] <= feedback;
  LFSR[1] <= LFSR[0];
  LFSR[2] <= LFSR[1] ^ feedback;
  LFSR[3] <= LFSR[2] ^ feedback;
  LFSR[4] <= LFSR[3] ^ feedback;
  LFSR[5] <= LFSR[4];
  LFSR[6] <= LFSR[5];
  LFSR[7] <= LFSR[6];
end
```



Se puede observar la generación de los números random