# Proyecto
# Maquina de Turing en Hardware con VHDL

***INTEGRANTES:***
*Ángel López Manríquez*
***Tema:*** *Maquina de Turing*
***Fecha:*** *17 de junio de 2019*
***Grupo:*** *2CV1*

M. en C. Luz María Sánchez García

Instituto Politecnico Nacional
Escuela Superior de Cómputo

# Índice

# Maquina de Turing

Lopez Manriquez Angel 2CV1

JUNIO 2018

# 1. Introducción

En este proyecto se desarrollará la simulación de una Máquina de Turing clásica en C++, con el fin de aplicarla a un convertidor de decimal a binario.

## 1.1. Definición

La Máquina de Turing (MT de aquí en adelante) es el modelo de autómata con máxima capacidad computacional, pues podemos desplazarnos tanto a la izquierda como a la derecha y y sobreescribir símbolos en la cinta de entrada. Una MT $M$ es una séptupla $M = (Q, q_0, F, \Sigma, \Gamma, \text{ƀ}, \delta)$, donde:

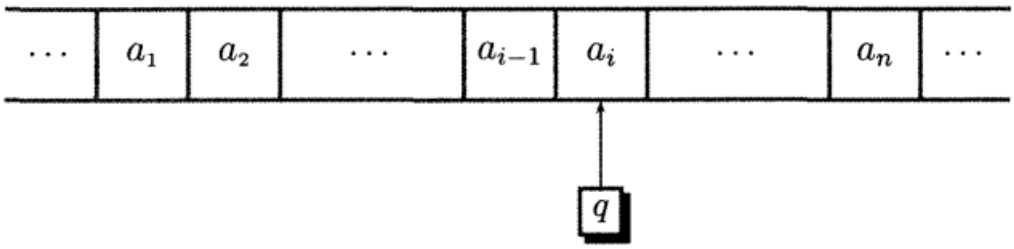
- $Q$ es el conjunto finito de estados internos.
- $q_0 \in Q$ es el estado inicial.
- $F \neq \phi$ es el conjunto finito de estados de aceptación, donde $F \subseteq Q$.
- $\Sigma$ es el alfabeto de entrada.
- $\Gamma$ es el alfabeto de cinta tal que $\Sigma \subseteq \Gamma$.
- $\text{ƀ} \in \Gamma$ es el símbolo *blanco* tal que $\text{ƀ} \notin \Sigma$.
- $\delta$ es la función de transición, donde $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, -, \rightarrow\}$, es decir, recibe un estado y un símbolo de la cinta para devolver otro estado, otro símbolo y una dirección de movimiento.

  $\delta$ es una función parcial, pues puede que no esté definida en algunos elementos del dominio. La transición $\delta(q, a) = (p, b, d)$ significa que estando en el estado $q$ escaneando el símbolo $a$: borramos $a$, escribimos $b$ y nos movemos al estado $p$, además avanzamos a la cinta hacia la izquierda (si $d =\leftarrow$), hacia la derecha (si $d =\rightarrow$) o nos quedamos ahí (si $d = -$).

La MT $M$ procesará cadenas $w \in \Sigma^*$, donde $w$ se colocará en la cinta al principio del cómputo. La cinta es en esencia infinita en ambas direcciones. La MT comenzará con el primer símbolo de $w$ estando en el estado $q_0$. Las demás casillas de la cinta contienen el símbolo blanco ƀ.

## 1.2. Descripción instantánea

Es una expresión de la forma $a_1 a_2 \cdots a_{i-1} q a_i \cdots a_n$, donde $a_1, \ldots, a_n \in \Gamma$ y $q \in Q$. Significa que estamos en el estado $q$ escaneando el símbolo $a_i$. Se supone que las casillas a la izquierda de $a_1$ y a la derecha de $a_n$ contienen el símbolo blanco ƀ.




La descripción inicial es $q_0w$, donde $w$ es la cadena de entrada. $w$ puede ser colocada en cualquier parte de la cinta, pues esta es infinita.

Definimos a un **paso computacional** como el paso de una descripción instantánea a otra por medio de una transición definida por $\delta$. Se denota como $u_1qu_2 \vdash v_1pv_2$, donde $u_1, u_2, v_1, v_2 \in \Gamma^*$ y $p, q \in Q$. Esto es equivalente a decir que existe una transición $\delta(q, a) = (p, b, d)$.

Por último, la notación $u_1qu_2 \vdash^* v_1pv_2$ significa que $M$ puede pasar de la descripción instantánea $u_1qu_2$ a $v_1pv_2$ en cero o más pasos computacionales.

Durante el procesamiento de la cadena de entrada puede darse los siguientes casos especiales:

- El cómputo termina porque no existe una transición. Es decir, si estamos leyendo el símbolo $a$ estando en el estado $q$, la transición $\delta(q, a)$ no existe.
- El cómputo no termina, es decir, entra en un bucle infinito. Se denota como $u_1qu_2 \vdash^* \infty$, que indica que el cómputo que inicia en la descripción $u_1qu_2$ no se detiene nunca.

## 1.3. Lenguaje aceptado por una MT

Una cadena de entrada $w$ es aceptada por $M$ si el cómputo que inicia desde la descripción $q_0w$ termina en una descripción $w_1pw_2$, donde $p \in F$ y $M$ se detiene completamente. Por lo tanto, podemos definir el lenguaje aceptado por $M$ como:

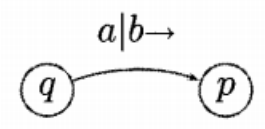
$$L(M) = \{w \in \Sigma^* \mid q_ow \vdash^* w_1pw_2, p \in F, w_1, w_2 \in \Gamma^*, M \text{ se detiene completamente en la descripción } w_1pw_2\}$$

.

Vemos que, a diferencia de los autómatas que hemos estado desarrollando antes, una cadena no tiene forzosamente que ser leída en su totalidad para que sea aceptada, solo se requiere que la máquina se detenga completamente en algún momento en algún estado de aceptación. Para ello (y para simplificar) no se permitirán transiciones $\delta(p, a)$ cuando $p \in F$.

## 1.4. Notación en grafos

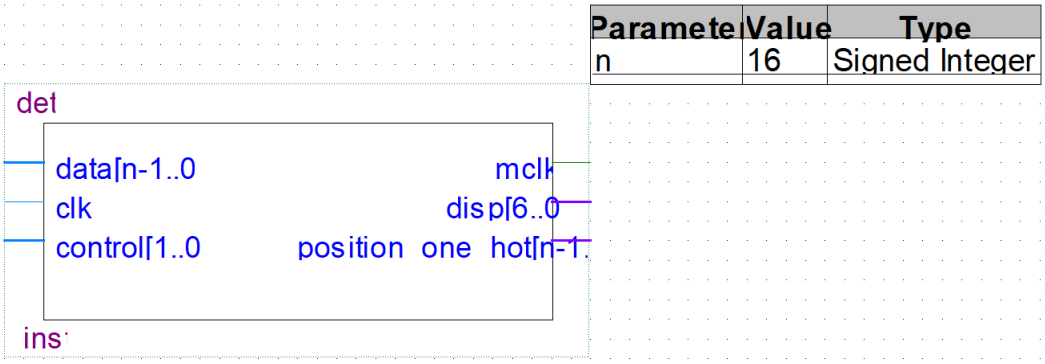
La función de transición $\delta$ de $M$ se puede representar como un dígrafo etiquetado. Así, la transición $\delta(q, a) = (p, b, \rightarrow)$ se puede representar como:

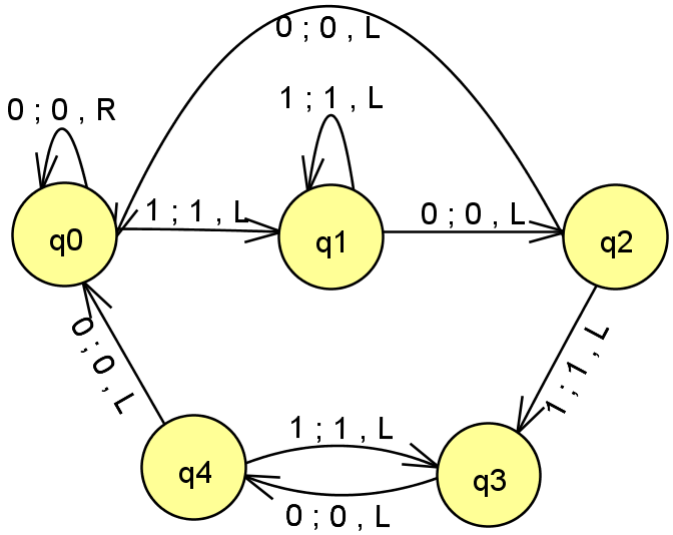


## 2. Planteamiento del problema

Simular el comportamiento de una Maquina de Turing, realizando un programa que determine la aceptación de cadenas $w = (0|1)^*(1010)^+$.

## 3. Diseño de la solución




| qi | Γ(qi,0) | Γ(qi,1) | Γ(qi,B) |
|---|---|---|---|
| ->q0 | (q0,0,R) | (q1,1,R) | |
| q1 | (q2,0,R) | (q1,1,R) | |
| q2 | (q0,0,R) | (q3,1,R) | |
| q3 | (q4,0,R) | | |
| q4* | (q0,0,R) | (q3,1,R) | (q5,B,F) |

# 4. Implementación de la solución

## det.vhd

Código en el cual manejamos las funcionalidades mas relevantes.

```
-- Proyecto: Maquina de Turing
-- Integrantes del equipo
-- Lopez Manriquez Angel
-- Grupo: 2CV1
--    ================================================================================

--    HDL (Hardware Design Language) que determina el lenguaje
--        L = (0|1)*(1010)+
--    usando el concepto de maquina de Turing, cuando nos encontremos en el estado
--    final se muestra en un display una C de correcto y una E de error en otro caso.

--    Como efecto secundario, este puede ser usado como un detector de secuencia para
--    una cadena w = 1010.

-- Caracteristicas ------------------------------------------------------------------
--    Hacemos uso de la palabra reservada type que nos provee VHDL para la creacion
--    de un estado y usamos un vector logico, el cual simula la cinta. Para saber en
--    que posicion estamos, hacemos uso de un entero.

library ieee; -- biblioteca ieee (Institute of Electrical and Electronics Engineers)

use ieee.std_logic_1164.all; -- para usar std_logic y std_logic_vector
use ieee.numeric_bit.all; -- para usar enteros

entity det is -- detector de secuencia
    generic (
        n : integer := 16); -- numero de bits para la entrada de la cadena
    port (
        data: in std_logic_vector(n-1 downto 0); -- palabra a probar
        clk: in std_logic; -- senial de reloj
        mclk: inout std_logic; -- senial de reloj maestra
        control: in std_logic_vector(1 downto 0); -- entradas de control
        disp: out std_logic_vector(6 downto 0); -- display de anodo comun
        position_one_hot: out std_logic_vector(n-1 downto 0) -- posicion de la cinta
    );
end entity;

architecture behave of det is

    type state is (d0, d1, d2, d3, d4); -- definicion de edos.
    constant final_state: state := d4; -- edo. final

    signal current_state, next_state: state; -- seniales auxiliares

    signal position: integer range 0 to n - 1 := 0;
```

```
46      signal accepted: std_logic := '0'; -- sera '1' cuando nos encontremos en el estado
        ↪  final
47
48      signal tape: std_logic_vector(n-1 downto 0);
49
50      constant disp_anode_c: std_logic_vector(6 downto 0) := "0110001";
51      constant disp_anode_e: std_logic_vector(6 downto 0) := "0110000";
52
53      component bin2one_hot -- retorna la codificacion de un entero en one hot
54          generic (bin_vec_len: integer := 8);
55          port ( entry: in integer;
56                 result: out std_logic_vector(bin_vec_len-1 downto 0)    );
57      end component;
58
59      component clk_div -- divisor de frecuencia
60          generic (   freq: integer := 50e6;
61                      freq_out: integer := 1 );
62          port (  clk: in std_logic;
63                  o: out std_logic );
64      end component;
65
66  begin
67
68      u0: bin2one_hot generic map (n) -- instancia del componente bin2one
69          port map(position, position_one_hot);
70      --u1: clk_div port map(clk, mclk); -- divisor de frecuencia
71
72      mclk <= clk;
73
74      disp <= (others => '1') when control = "11" else
75                  disp_anode_c when accepted = '1' else
76                  disp_anode_e;
77
78      accepted <= '1' when next_state = final_state else
79                      '0';
80
81      management: process(control, mclk)
82      begin
83          if rising_edge(mclk) then
84              if control = "00" then -- save
85                  tape <= tape;
86              elsif control = "01" then -- load
87                  tape <= data;
88              elsif control = "10" then -- enable
89                  position <= position + 1; -- right
90              else -- clr
91                  tape <= (others => '0'); -- limpiamos la cinta
92                  position <= 0;
93              end if;
94          end if;
```

```
95      end process;
96
97      -- proceso encargado de cambiar de estados
98      turing: process (current_state, tape, position)
99      begin
100         case current_state is
101             when d0 =>      if tape(position) = '1' then
102                             next_state <= d1;
103                         else
104                             next_state <= d0;
105                         end if;
106
107             when d1 =>     if tape(position) = '0' then
108                             next_state <= d2;
109                         else
110                             next_state <= d1;
111                         end if;
112
113             when d2 =>      if tape(position) = '1' then
114                             next_state <= d3;
115                         else
116                             next_state <= d0;
117                         end if;
118
119             when d3 =>     if tape(position) = '0' then
120                             next_state <= d4;
121                         else
122                             next_state <= d0;
123                         end if;
124
125             when d4 =>     if tape(position) = '1' then
126                             next_state <= d3;
127                         else
128                             next_state <= d0;
129                         end if;
130         end case;
131
132     end process ;
133
134     -- actualiza el estado actual por cada flanco de reloj
135     update_state: process(mclk)
136     begin
137         if rising_edge(mclk) then
138             current_state <= next_state;
139         end if;
140     end process ;
141
142 end architecture;
```

## bin2one_hot.vhd

Decodificador de entrada binaria al formato one hot

```
library ieee;

use ieee.std_logic_1164.all;
use ieee.math_real.all; -- log2

entity bin2one_hot is
    generic (bin_vec_len: integer := 8);
    port ( entry: in integer;
           result: out std_logic_vector(bin_vec_len-1 downto 0)    );
end entity;

architecture arch of bin2one_hot is

begin

    process (entry)
        variable i: integer range 0 to bin_vec_len - 1;
        variable ans: std_logic_vector(bin_vec_len - 1 downto 0);
    begin
        for i in 0 to bin_vec_len - 1 loop
            if i = entry then
                ans(i) := '1';
            else
                ans(i) := '0';
            end if;
        end loop;
        result <= ans;
    end process;

end architecture;
```

## clk_div.vhd

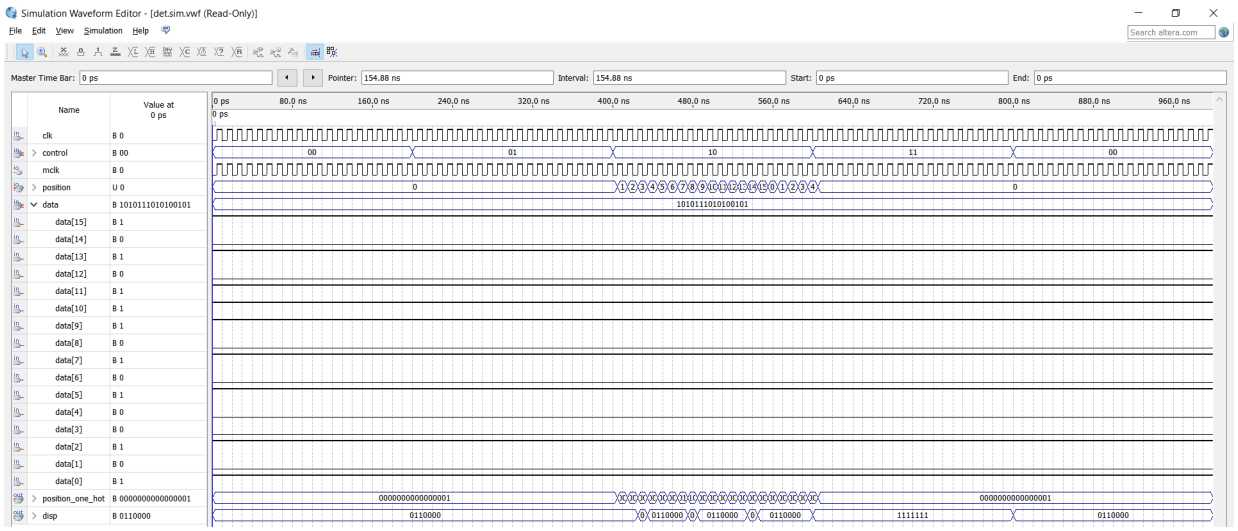
Divisor de frecuencia

```
library ieee;

use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

-- internal oscilator of fpga cyclone ii EP2C5T144C8N is 50Mhz

entity clk_div is
    generic (
        freq: integer := 50e6;
        freq_out: integer := 1 );
    port (
```

```
13          clk: in std_logic;
14          o: out std_logic );
15  end entity;
16
17  architecture angel of clk_div is
18      signal os: std_logic;
19  begin
20
21      process (clk)
22          variable i: integer range 0 to freq;
23      begin
24          if rising_edge(clk) then
25              if i = (freq * freq_out) / 2 then
26                  os <= not os;
27                  i := 0;
28              else
29                  i := i + 1;
30              end if;
31          end if;
32      end process;
33
34      o <= os;
35
36  end architecture ; -- arch
```

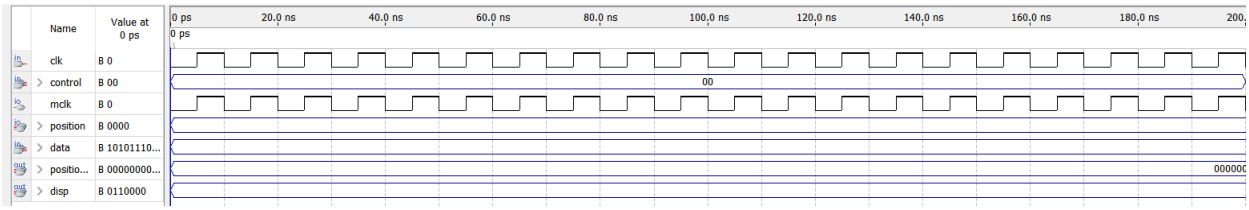
# 5. Funcionamiento

## 5.1. Vista general



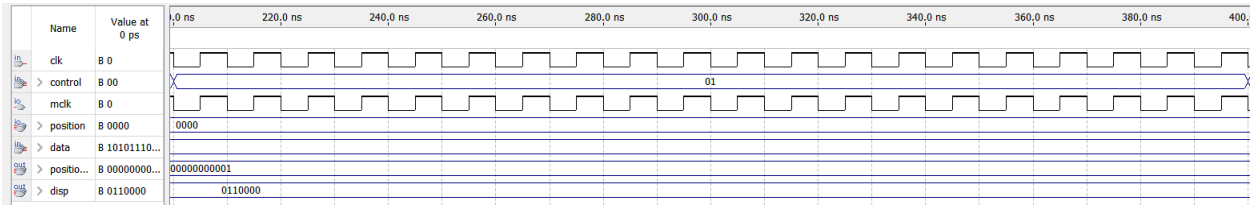

## 5.2. Cuando $c = 00$

Se esta en un estado de retención, por lo que la cinta no se ve afectada.



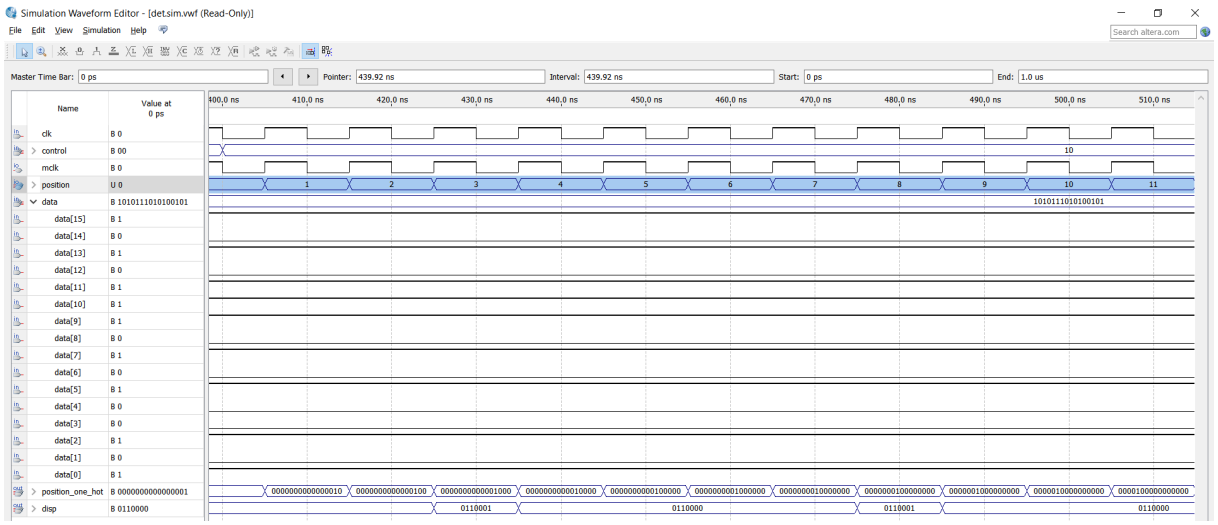

## 5.3. Cuando $c = 01$

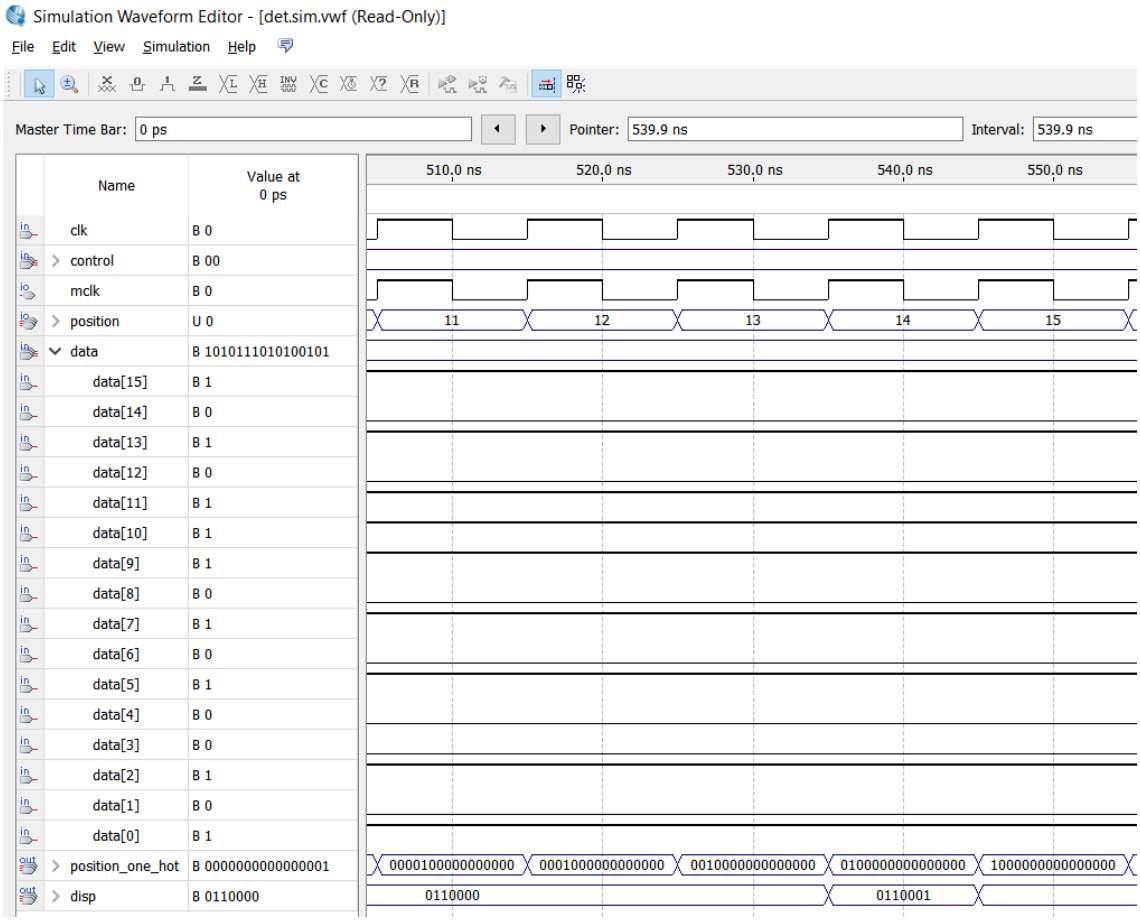
La cinta ha sido cargada y toma el valor de la variable data.




## 5.4. Cuando $c = 10$

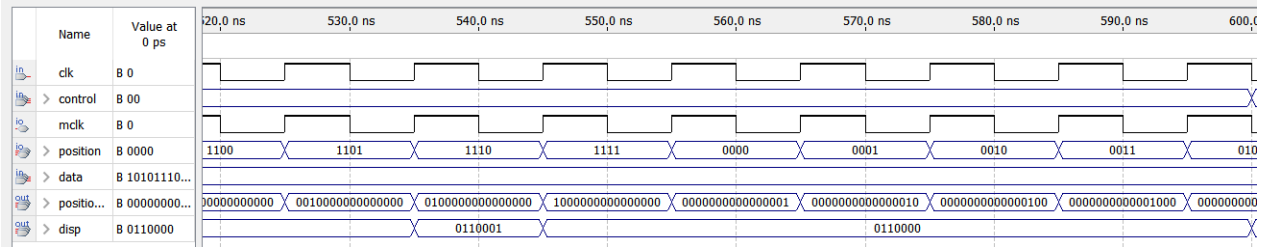
La maquina de Turing se activa, como vemos, la palabra data = 1010010101110101 tiene las subcadenas 1010, estaremos en el estado de aceptación cuando la variable posición sea $3, 8, 14$ y esto se aprecia en la variable disp.






## 5.5. Cuando $c = 11$

La cinta es retirada, establecemos la posición en 0.




# 6. Conclusiones

- **López Manríquez Ángel:** La Máquina de Turing es el autómata más poderoso, pues a pesar de agregarle elementos (más cintas, pistas o no determinismo) no aumenta ni disminuye su capacidad computacional. Tuvimos la oportunidad de adaptar un detector de secuencia en Hardware. De esta forma verificamos una vez más que la tesis de Church-Turing está lejos de ser falsa, pues todo algoritmo computable tiene su equivalente en una MT, claro, asumiendo que la memoria y el tiempo de ejecución tienden a infinito, mira que hacer este proyecto bajo la definición de Turing nos dio una forma sencilla de determinar subcadenas

a nivel de Hardware en HDL donde la falta de métodos tal como `substring` es sustentada por este modelo matemático.