DISEÑO AVANZADO DE HARDWARE
FACULTAD DE INGENIERÍA ELECTRÓNICA
UNIVERSIDAD PONTIFICIA BOLIVARIANA
BUCARAMANGA

Sebastian Augusto Baquero Peña ID: 244379

"Numero de Fibonacci"

OBJETIVO GENERAL

Implementación del número de "Fibonacci" utilizando máquina de estados.

OBJETIVOS ESPECÍFICOS

- Entender el funcionamiento del número de "Fibonacci".
- Aplicar los conceptos de FSM.
- Aplicar los conceptos de FSMD.
- Utilizar la plataforma de ModelSim para verificar funcionamiento.

MARCO TEÓRICO

En matemáticas, se le conoce como Sucesión de Fibonacci la cual está compuesta por una sucesión de infinitos números naturales. Esta sucesión inicia con los números cero y uno, que a partir de estos números el resultado del número siguiente es la suma de los dos resultados anteriores.

A los elementos que conforma esta sucesión se le conoce con el nombre de "números de Fibonacci" cuyo inventor fue el matemático italiano del siglo XIII Leonardo de Pisa cuyo nombre conocido fue el mismo al que se le llamó a dicha sucesión.

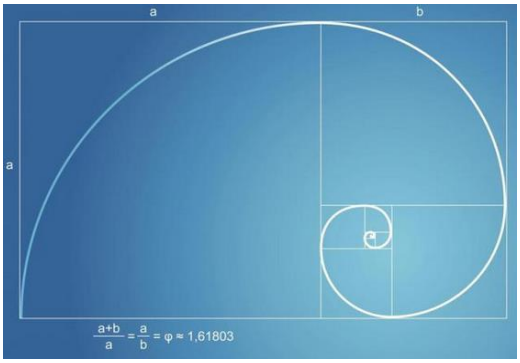

Figura 1. Representación del espiral de Fibonacci

Con esta seria se pudo observar que esta se aplica o se encuentra en múltiples configuraciones biológicas presentes en la naturaleza y también se encuentran inclusive en el espacio.

Figura 2. Sucesión de Fibonacci en la naturaleza.

Los números de Fibonacci constituyen una secuencia definida como:

$$fib(i) = \begin{cases} 0 \rightarrow & if\ i = 0 \\ 1 \rightarrow & if\ i = 1 \\ fib(i-1) + fib(i-2) \rightarrow & if\ i > 1 \end{cases}$$

Una forma para calcular "$fib(i)$" es construir la función por medio de iteraciones, de 0 hasta el valor de i deseado. Este enfoque requiere dos registros temporales para almacenar los dos valores más recientemente calculados ("$fib(i-1) + fib(i-2)$") y un registro índice para realizar un seguimiento de los números de las iteraciones. El gráfico ASMD se mostrara en la Figura 3. en la cual "$t0$" y "$t1$" son registros almacenados temporalmente y "n" el registro índice. Adicionalmente para los datos regulares de señales de entrada y salida, i y f, incluimos una señal de comando "start", cuales señales de inicio de operación, y dos señales de estado: "ready", cuales indican que el circuito es "IDLE2 y lista para tomar una nueva entrada, y "donde_tick", cual es afirmada para un ciclo de reloj cuando la operación es completada.

Desde este circuito, como muchos otros diseños de FSMD, es probablemente una parte de un sistema largo, estas señales son necesitadas para interactuar con otros sistemas.

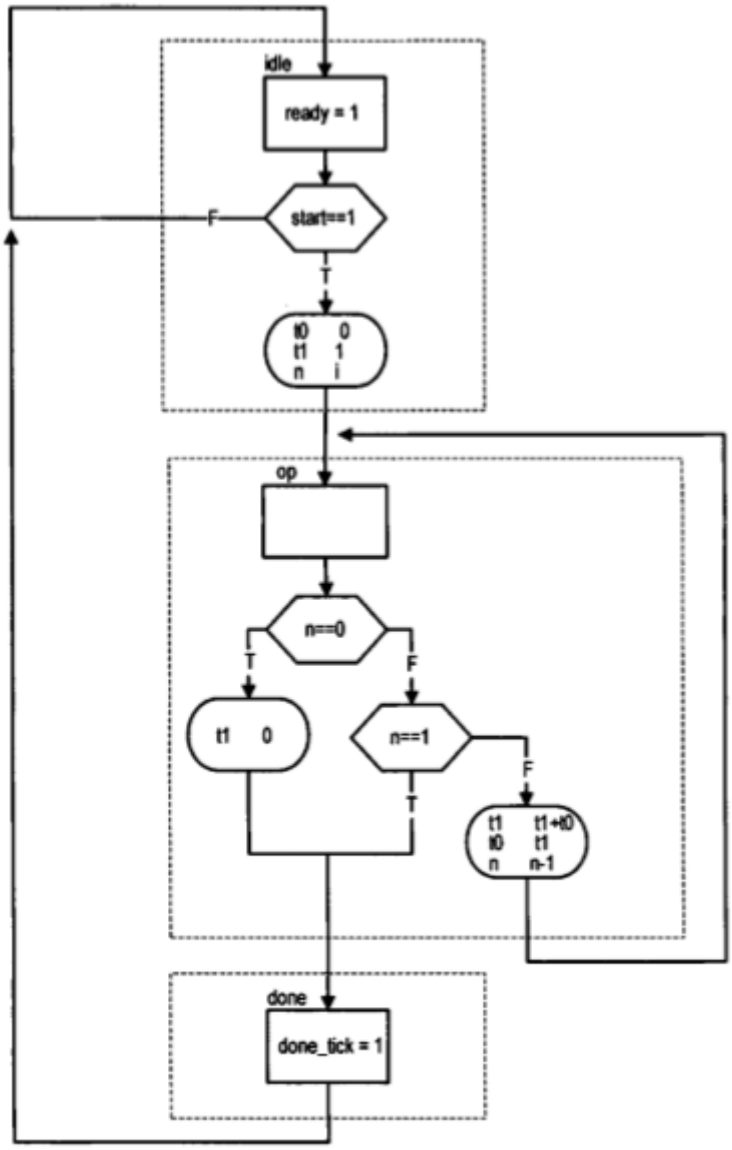

Figura 3. Diagrama de bloques para la sucesión de Fibonacci. Obtenida del libro de Pong Chu - "Embededd SoPC Design with Nios II Porcessor and Verilgo Examples".

El gráfico o diagrama de bloques tiene tres estados. El estado "IDLE" idica que el circuito es actualmente "IDLE". Cuando "start" es afirmado, el FSMD pasa al estado "op" y carga los valores iniciales a los tres registros. Los registro "$t0$" y "$t1$" son guardados con 0 y 1, que representa "$fib(0)$" y "$fib(1)$" respectivamente. El registro "n" es guardado con i, el número escogido de iteraciones.

La principal cálculo es iterado a través del estado "op" por tres operadores RT:

- $t1 \leftarrow t1 + t0$
- $t0 \leftarrow$ t1
- $n \leftarrow n - 1$

Las dos primeras operaciones RT obtienen un nuevo valor y los guardan en dos valores recientemente calculados en "$t0$" y "$t1$". El tercer operador RT disminuye las iteraciones del índice. La iteración termina cuan "n" alcanza 1 o su valor inicial es 0 ("$fib(0)$"). Diferente a diagrama de flujo regular, las operaciones en un bloque ASDM pude ser ejecutada concurrentemente en el mismo ciclo del clock. Ponemos todas las comparaciones y operaciones RT en el estado "op" para reducir el tiempo de cálculo. Note que los nuevos valor de los registro "$t1$" y "$t0$" son cargados al mismo tiempo cuando el FSMD existe el estado "op". Así, el valor original de "$t1$", no "$t1 + t0$", es almacenado en $t0$. El propósito del estado "done" es general un ciclo de señal reloj "$done\_tick$" para indicar terminación del cálculo. Este estado puede ser omitido si esta señal de estado no es necesitada.

A continuación se muestra el código realizado para la secuencia de Fibonacci.

```
1  module fibonaci_sec (
2      input wire [3:0]i,
3      input wire clock,
4      input wire reset,
5      input wire start,
6      output reg ready,done_tick,
7      output wire [15:0]f
8  );
9  //Se crean parametros locales para luego ser llamados
10 localparam idle=2'b00;
11 localparam op=2'b01;
12 localparam done=2'b10;
13
14 //se crean registros para hacer cada proceso
15 reg [1:0]state_r,state_n;
16 reg [15:0] t0_r, t0_n, t1_r,t1_n;
17 reg [3:0]n_r, n_n;
18
```

Figura 4. Parte 1 programa

```
24 //Definicion del proceso FSMD
25 always@(posedge clock, posedge reset)
26 begin
27     state_r<=state_n;
28     t0_r<=t0_n;
29     t1_r<=t1_n;
30     n_r<=n_n;
31
32     if (reset==1)
33         begin
34             state_r<=idle;
35             t0_r<=0;
36             t1_r<=0;
37             n_r<=0;
38         end
39 end
40
```

Figura 5. Parte 2 programa

```
41    //Imprementacion FSMD
42    always@ *
43    begin
44        state_n=state_r;
45        ready=1;
46        done_tick=1;
47        t0_n=t0_r;
48        t1_n=t0_r;
49        n_n=n_r;
50
51        case (state_r)
52
53           idle :  begin
54                    ready=1;
55                    if (start)
56                       begin
57                       t0_n=0;
58                       t1_n=1;
59                       n_n=i;
60                       state_n= op;
61                       end
62                  end
63           op :    begin
64                       t1_n=t1_r + t0_r;
65                       t0_n=t1_r;
66                       n_n=n_r -1;
67                     if (n_r==0)
68                       begin
69                          t1_n=0;
70                          state_n=done;
71                       end
72                     else if (n_r==1)
73                       begin
74                          state_n=done;
75                       end
76                  end
```

Figura 6. Parte 3 programa

```
77           done : begin
78                    done_tick = 1'b1;
79                    state_n=idle;
80                  end
81           default: state_n=idle;
82       endcase
83    end
84
85    assign f=t1_r;
86    endmodule
```

Figura 7. Parte 4 programa

ModelSim

Se forzó la entrada i = 11;

Clock= 20ns de periodo

Start=1;

Ready=1;

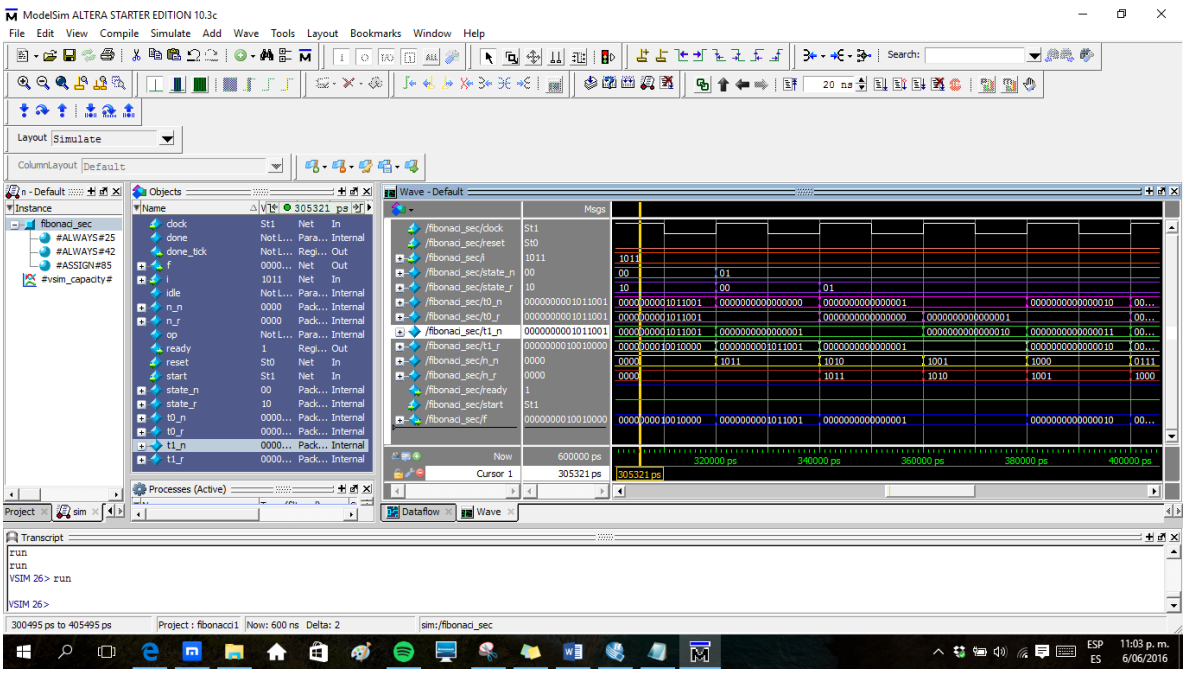

Figura 8. Clock 1 de 20 ns.

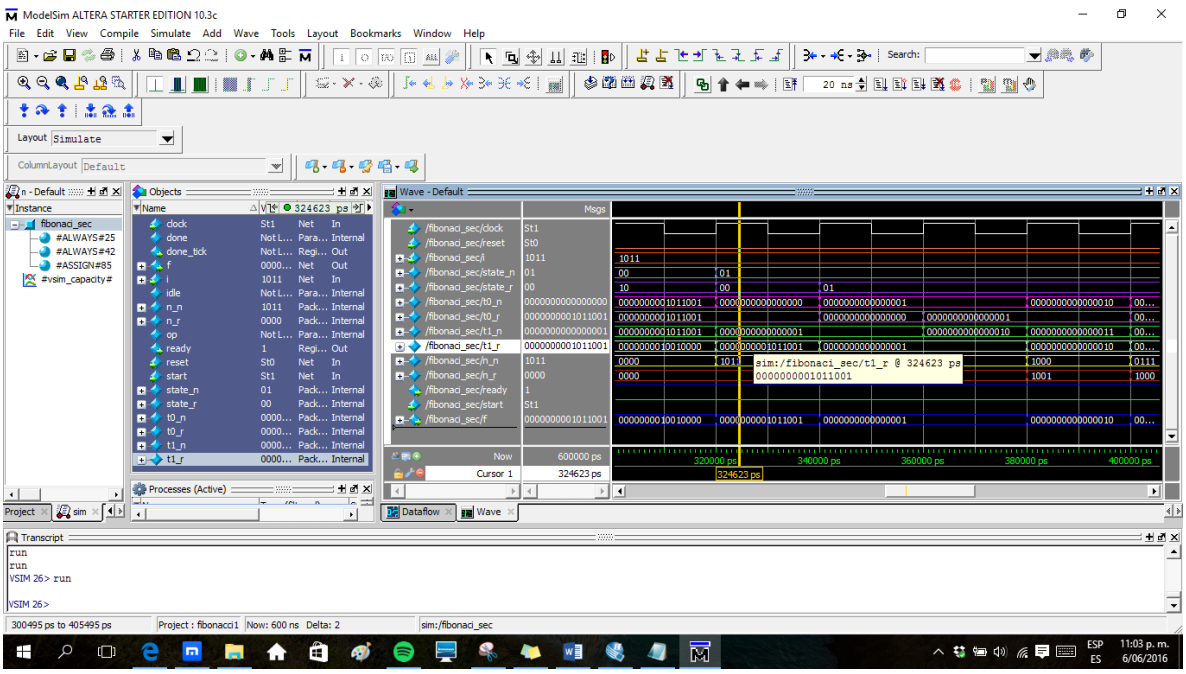

Figura 8. Clock 2 de 20 ns.

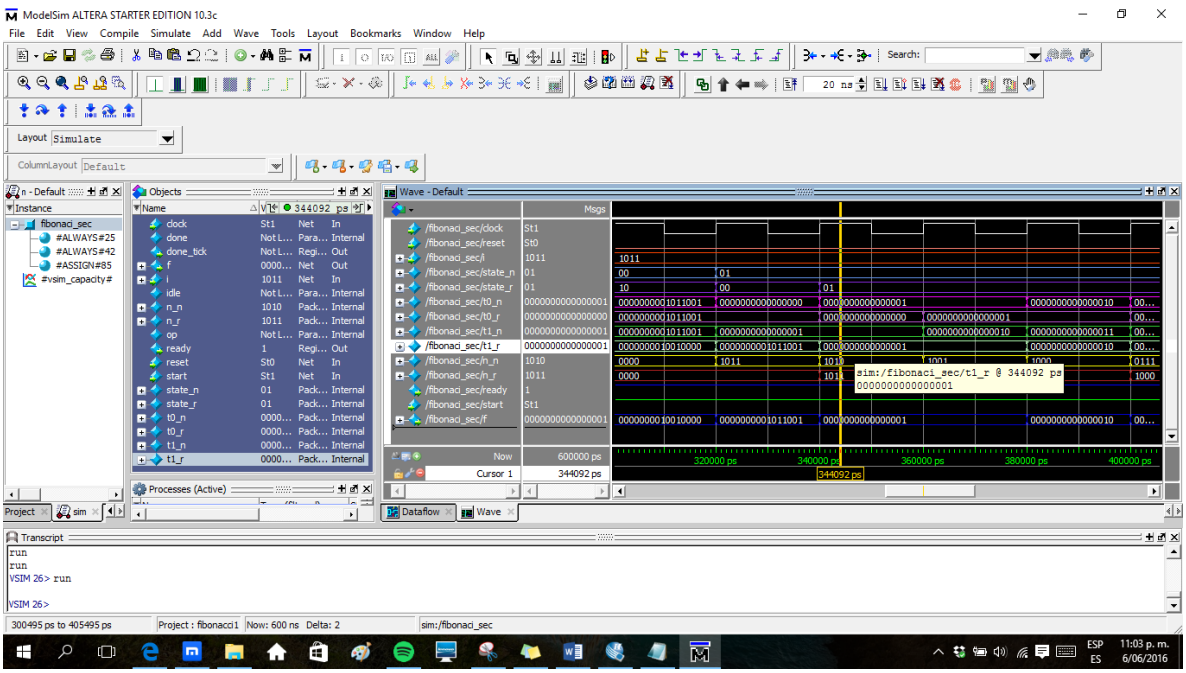

Figura 8. Clock 3 de 20 ns.

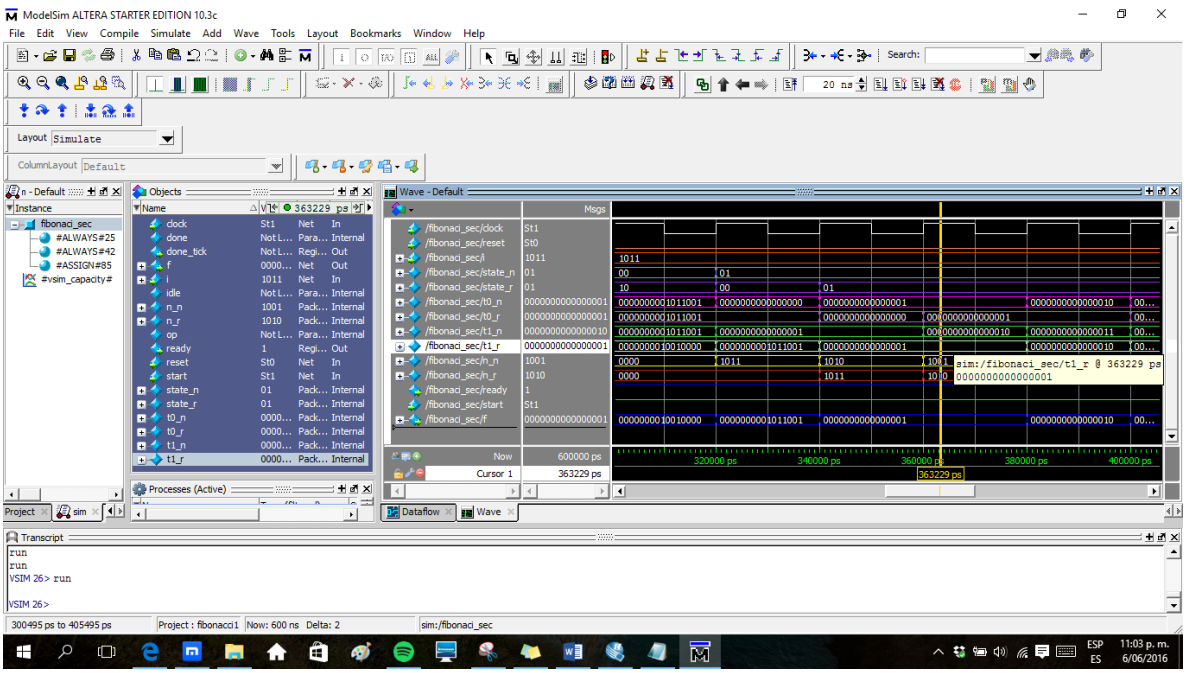

Figura 8. Clock 4 de 20 ns.

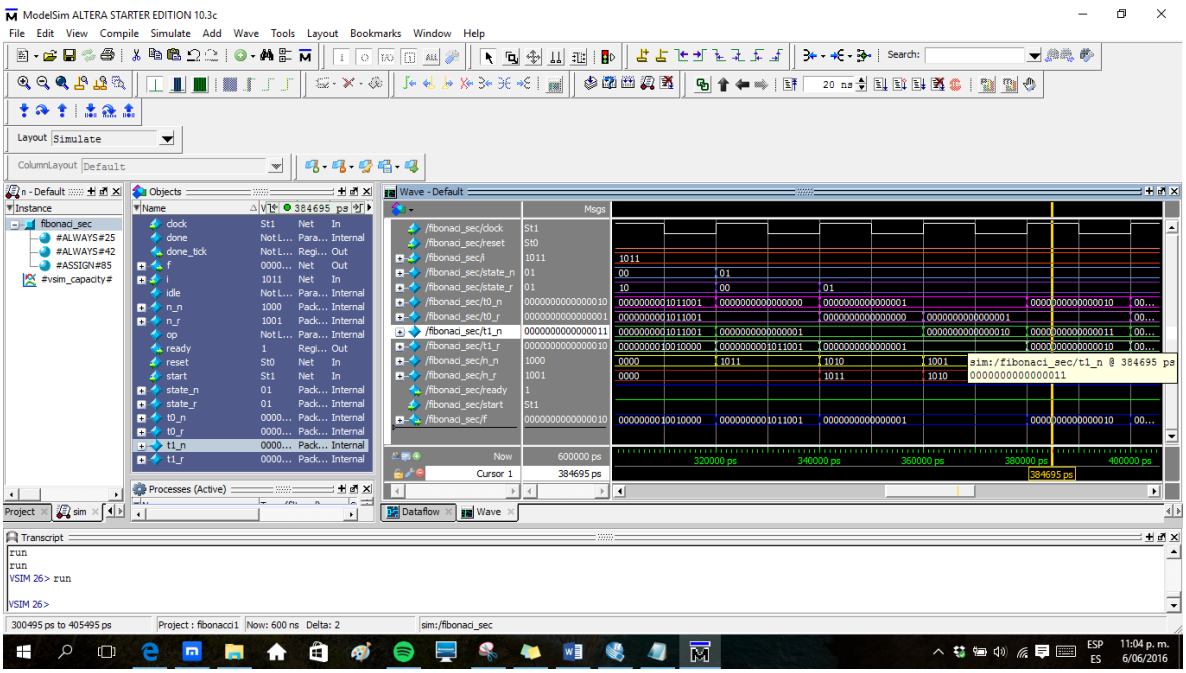

Figura 8. Clock 5 de 20 ns.

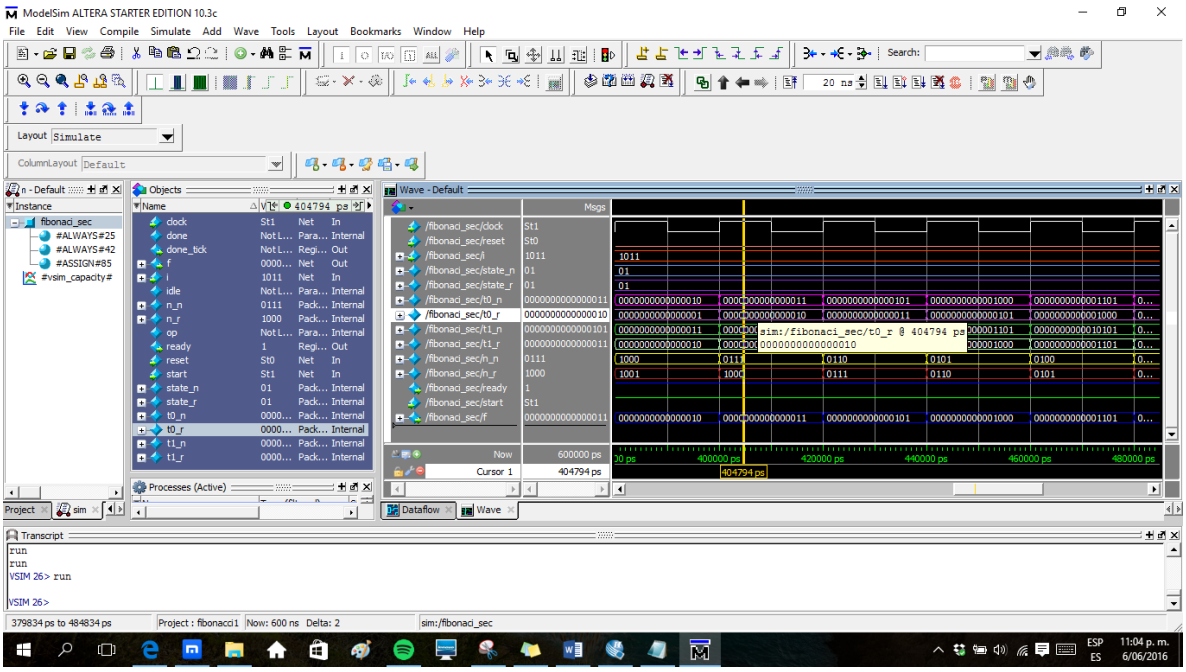

Figura 8. Clock 6 de 20 ns.

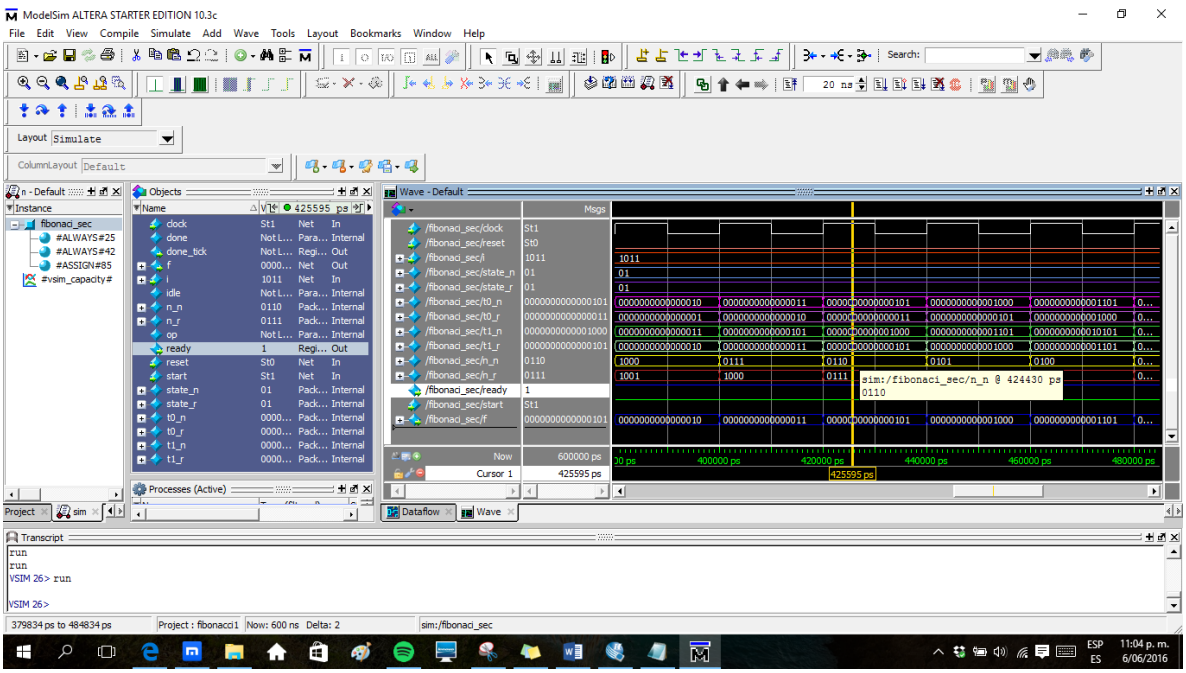

Figura 8. Clock 7 de 20 ns.

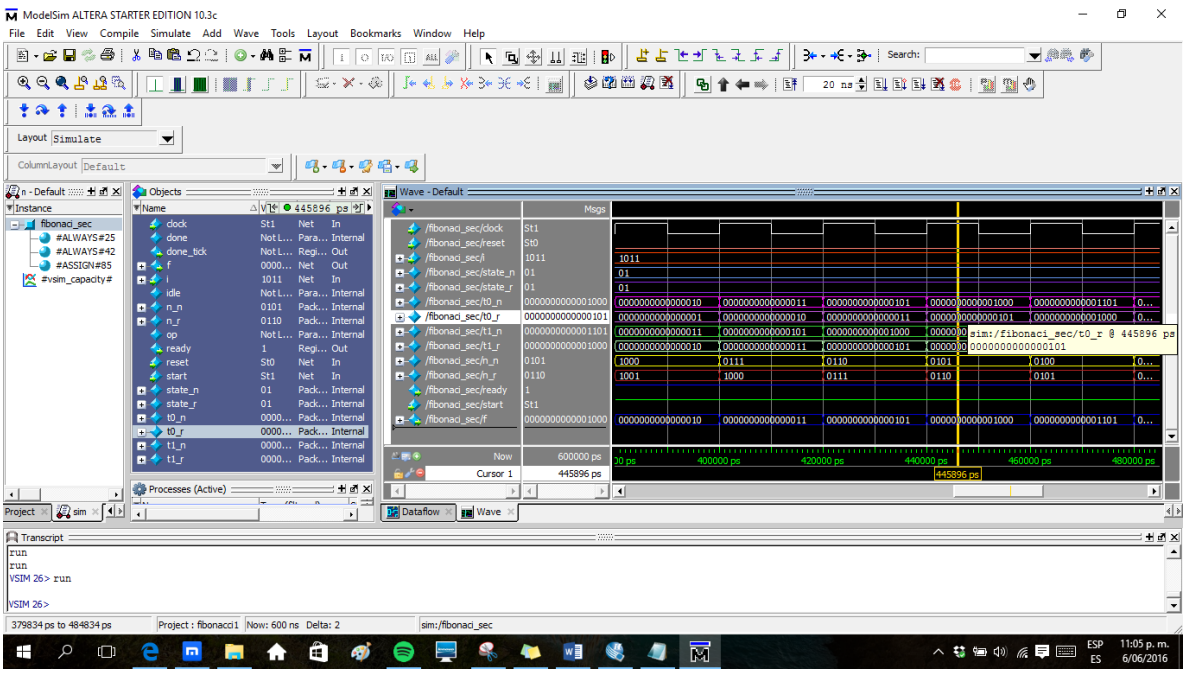

Figura 8. Clock 8 de 20 ns.

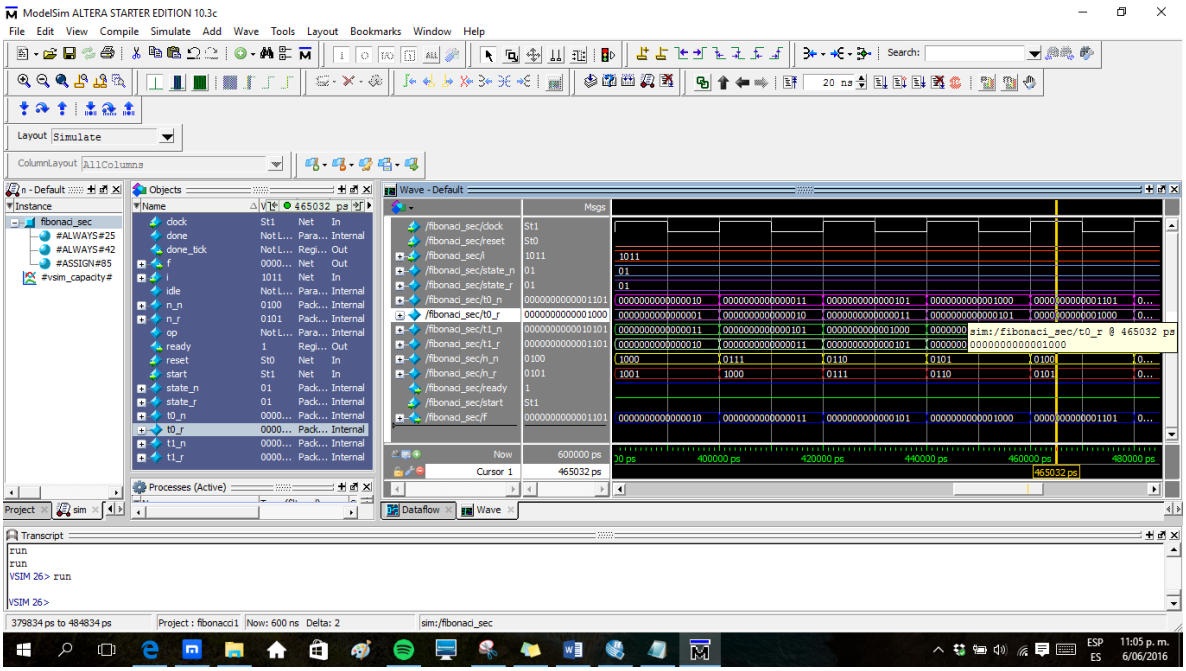

Figura 8. Clock 9 de 20 ns.

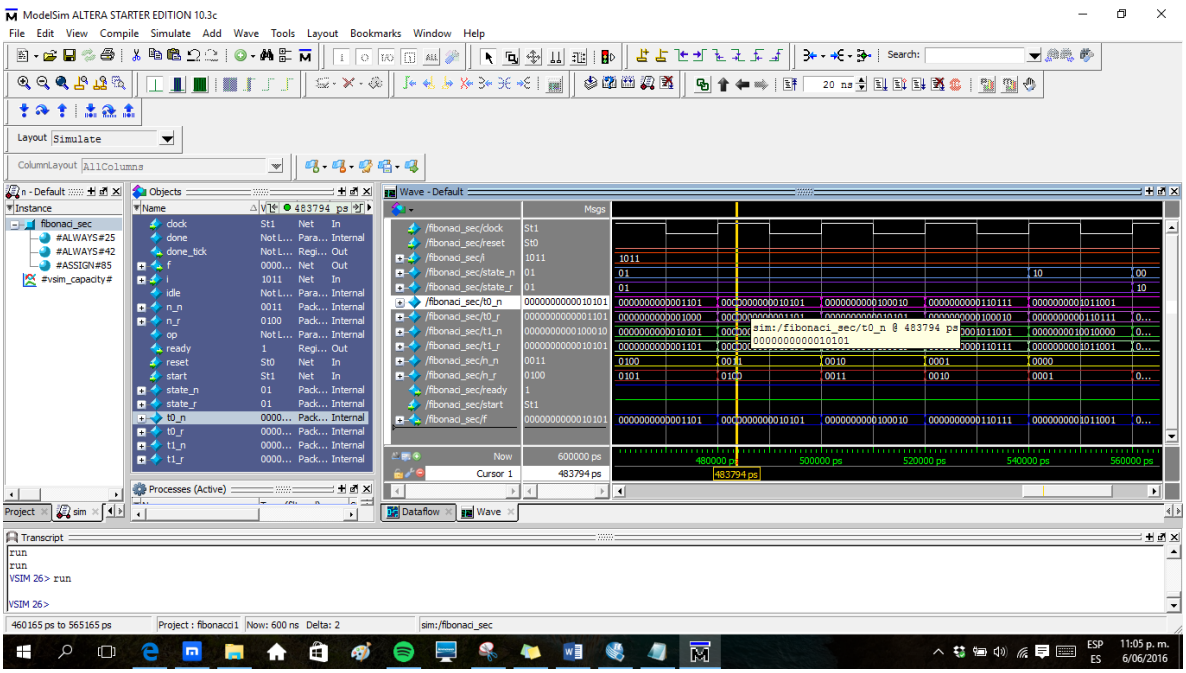

Figura 8. Clock 10 de 20 ns.

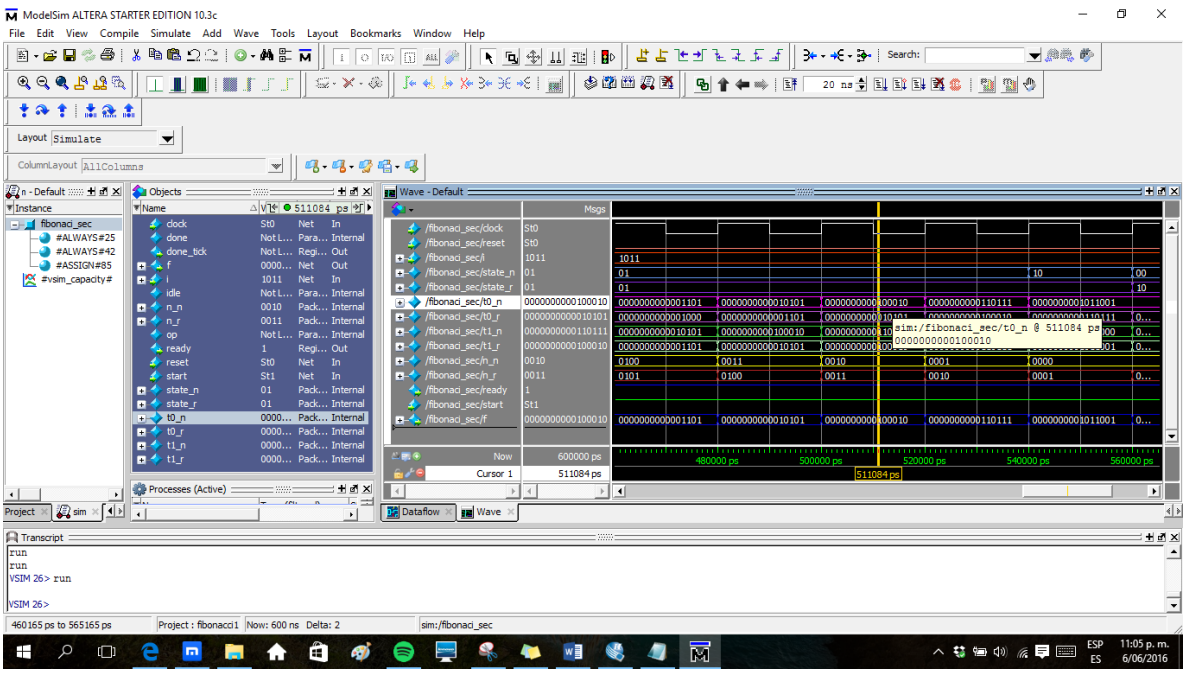

Figura 8. Clock 1 de 20 ns.

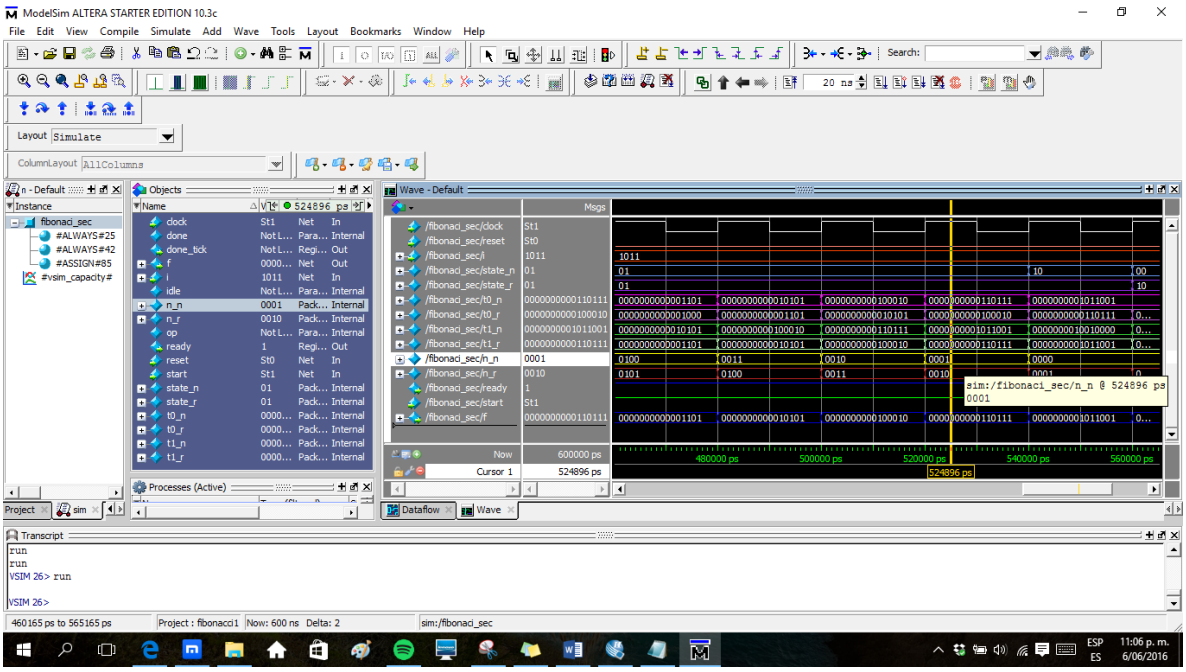

Figura 8. Clock 1 de 20 ns.

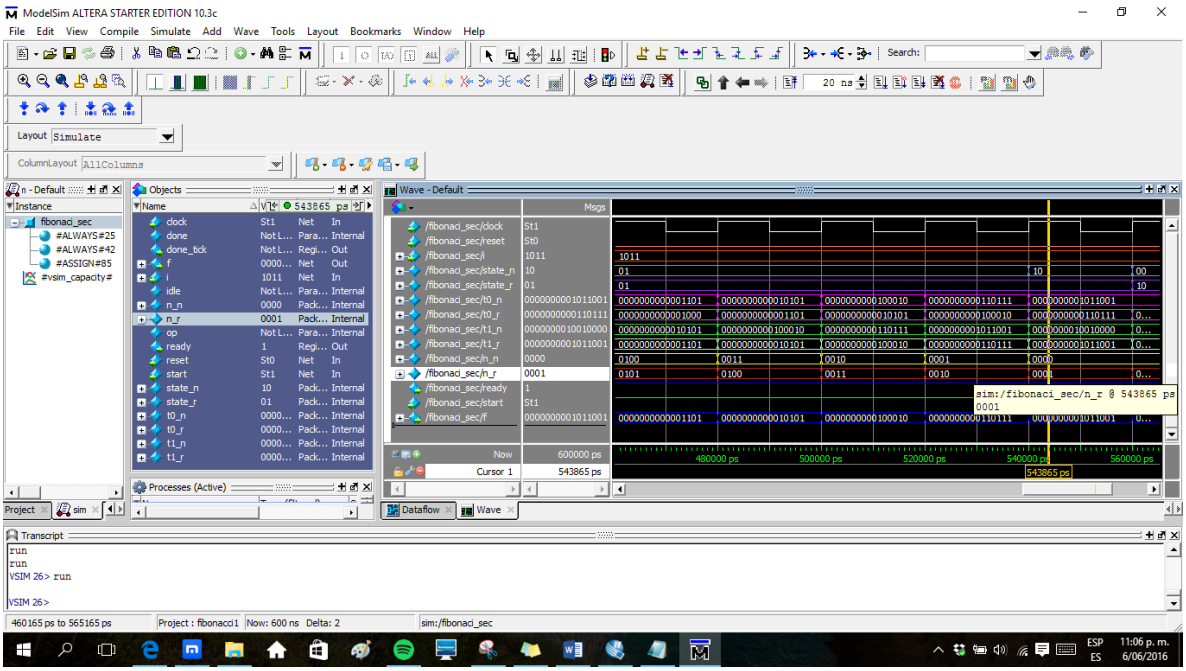

Figura 8. Clock 1 de 20 ns.