



Centro de Investigación y de Estudios Avanzados del IPN

Iván Cabrera Altamirano

Tarea # 3

Multiplicador Binario Parametrizado

Tópicos Selectos en Sistemas Digitales: VHDL

Dr. Arturo Díaz Pérez

Dr. Adriano de Luca Pennacchia

Diseño general del multiplicador

El diseño del multiplicador, sigue la arquitectura del expuesto en la figura 1.

Para esta arquitectura de 32 Bits, se tiene un registro multiplicador de 32 bits, una unidad aritmético-lógica de 32 bits, un registro para almacenar el producto de 64 bits y 32 iteraciones.

Siguiente la arquitectura anteriormente descrita, para un multiplicador de N bits, requeriríamos:

- Registro para el Multiplicador de N bits.
- Unidad Aritmético-Lógica de N Bits.
- Registro para el Producto de N bits.
- N iteraciones.

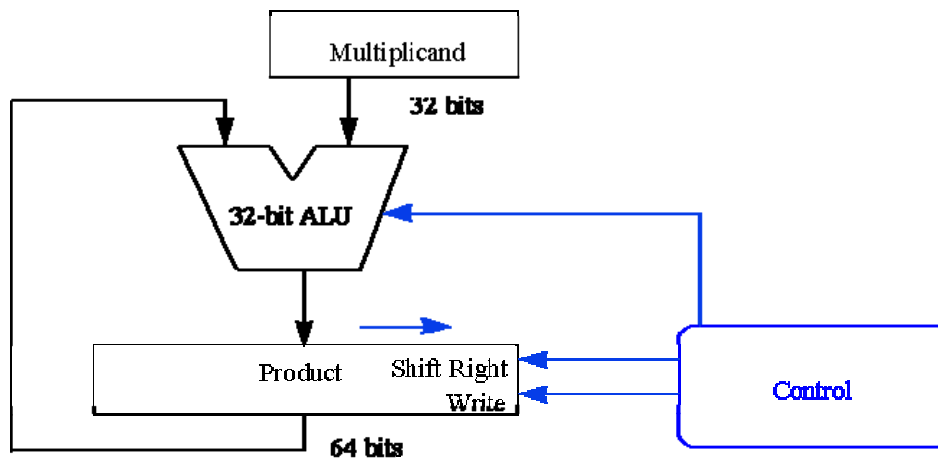


Figura 1. Multiplicador de 32 Bits.

El código fuente que implementa el multiplicador de N bits se muestra a continuación. Este es una adaptación del código presentado en clase. Las principales modificaciones realizadas son:

- Parametrizar el tamaño de los vectores: MULT_IN, MULT_OUT, CA, COUNTER, A, B y Q.
- Agregar 2 parámetros de tipo GENERIC para especificar N y S.
 - N: Tamaño de Palabra en Bits.
 - S: Bits necesarios para representar N.
- Agregar a la entidad principal una señal que indica fin del algoritmo.
- Agregar un proceso "z_process" que determina cuando se han terminado las iteraciones.
- Definir sentencias de tipo "Aggregate" los siguientes vectores:
 - COUNTER: Este vector lleva la cuenta del número de iteraciones.
 - CONST: Este vector es una constante de tamaño N inicializado en 0.
 - A: Este vector lleva resultados parciales de la multiplicación.

```

1     LIBRARY ieee;
2     USE ieee.std_logic_1164.ALL;
3     USE ieee.std_logic_unsigned.ALL;
4
5     -- Iván Cabrera Altamirano, Tarea #3, Programa Principal.
6
7     -- Parametros para GENERIC:
8     -- N:          Numero de bits de los operandos.
9     -- S:          Cantidad de bits necesarios para representar N.
10
11    -- Parametros para PORT:
12    -- CLK:        Señal de Reloj.
13    -- RST:        Señal de Reset Asincrono.
14    -- START:      Señal de Inicio de Algoritmo.
15    -- LOADB:      Señal de Carga de Operando B.
16    -- LOADQ:      Señal de Carga de Operando Q.
17    -- RESULT:     Bandera de Resultado, activa cuando la operación ha terminado.
18    -- MULT_IN:    Entrada de Operandos B y Q de acuerdo a señales correspondientes.
19    -- MULT_OUT:   Resultado de la multiplicación.
20
21    ENTITY binary_multiplier IS
22        GENERIC ( N: INTEGER := 64; S: INTEGER := 6);
23        PORT( CLK, RST, START:      IN    std_logic;
24              LOADB, LOADQ:        IN    std_logic;
25              RESULT:              OUT   std_logic;
26              MULT_IN:             IN    std_logic_vector((N-1) DOWNTO 0);
27              MULT_OUT:            OUT   std_logic_vector(((2*N)-1) DOWNTO 0));
28    END binary_multiplier;
29
30    ARCHITECTURE behavior_N OF binary_multiplier IS
31        TYPE state_type IS (IDLE, MUL0, MUL1);
32        SIGNAL state, next_state : state_type;
33        SIGNAL A, B, Q:          std_logic_vector(N-1 DOWNTO 0);
34        SIGNAL COUNTER:         std_logic_vector(S-1 DOWNTO 0) := (others => '1');
35        CONSTANT CONST:        std_logic_vector(S-1 DOWNTO 0) := (others => '0');
36        SIGNAL Z,C:            std_logic;
37    BEGIN
38
39        MULT_OUT <= A & Q;
40
41        -- Este proceso activa Z y RESULT cuando el algoritmo ha terminado.
42        z_process: PROCESS (COUNTER)
43            BEGIN
44                IF COUNTER = CONST THEN
45                    Z <= '1';
46                    RESULT <= '1';
47                END IF;
48            END PROCESS;
49
50        -- Este proceso efectua un reset cuando recibe la señal correspondiente
51        -- y de hacer la transición al estado siguiente.
52        state_register: PROCESS (CLK, RST)
53            BEGIN
54                IF (RST = '1') THEN
55                    state <= IDLE;
56                ELSIF (CLK'event AND CLK = '1') THEN
57                    state <= next_state;
58                END IF;
59            END PROCESS;
60
61

```

```

62  -- Este proceso efectua la asignación del estado siguiente de acuerdo
63  -- a la tabla de transiciones.
64  next_state_chart: PROCESS (START, Z, state)
65  BEGIN
66      CASE state IS
67          WHEN IDLE =>
68              IF START = '1' THEN
69                  next_state <= MUL0;
70              ELSE
71                  next_state <= IDLE;
72              END IF;
73          WHEN MUL0 =>
74              next_state <= MUL1;
75          WHEN MUL1 =>
76              IF Z = '1' THEN
77                  next_state <= IDLE;
78              ELSE
79                  next_state <= MUL0;
80              END IF;
81          END CASE;
82  END PROCESS;
83
84  -- Este proceso realiza la carga de los operandos y
85  -- efectua la multiplicaciones de acuerdo al valor del estado actual.
86  datapath_func: PROCESS (CLK)
87  VARIABLE CA: std_logic_vector(N DOWNT0 0);
88  BEGIN
89      IF (CLK'event AND CLK = '1') THEN
90          IF LOADB = '1' THEN
91              B <= MULT_IN;
92          END IF;
93          IF LOADQ = '1' THEN
94              Q <= MULT_IN;
95          END IF;
96          CASE state IS
97              WHEN IDLE =>
98                  IF START = '1' THEN
99                      C <= '0';
100                     A <= (others => '0');
101                     COUNTER <= (others => '1');
102                 END IF;
103             WHEN MUL0 =>
104                 IF Q(0) = '1' THEN
105                     CA := ('0' & A) + ('0' & B);
106                 ELSE
107                     CA := C & A;
108                 END IF;
109                 C <= CA(N);
110                 A <= CA(N-1 DOWNT0 0);
111             WHEN MUL1 =>
112                 C <= '0';
113                 A <= C & A(N-1 DOWNT0 1);
114                 Q <= A(0) & Q(N-1 DOWNT0 1);
115                 COUNTER <= COUNTER - "1";
116             END CASE;
117          END IF;
118  END PROCESS;
119  END behavior_N;

```

```
1      -- Iván Cabrera Altamirano, Tarea #3, Testbench for 4 bits.
2      LIBRARY ieee;
3      USE ieee.std_logic_1164.ALL;
4      USE ieee.std_logic_unsigned.all;
5      USE ieee.numeric_std.ALL;
6
7      ENTITY TB4BitM1 IS END TB4BitM1;
8
9      ARCHITECTURE behavior OF TB4BitM1 IS
10         COMPONENT binary_multiplier
11         PORT(CLK, RST, START, LOADB, LOADQ :      IN      std_logic;
12              RESULT :                          OUT    std_logic;
13              MULT_IN :                         IN     std_logic_vector(3 DOWNTO 0);
14              MULT_OUT :                       OUT    std_logic_vector(7 DOWNTO 0));
15         END COMPONENT;
16
17         -- Entradas
18         SIGNAL CLK, RST, START :                std_logic;
19         SIGNAL LOADB, LOADQ, RESULT :           std_logic;
20         SIGNAL MULT_IN :                       std_logic_vector(3 DOWNTO 0);
21         -- Salidas
22         SIGNAL MULT_OUT :                     std_logic_vector(7 DOWNTO 0);
23
24         CONSTANT PERIOD :                     TIME := 100 PS;
25         CONSTANT DUTY_CYCLE :                 REAL := 0.5;
26         CONSTANT OFFSET :                    TIME := 50 PS;
27
28     BEGIN
29         -- Instanciando la unidad a probar.
30         uut: binary_multiplier PORT MAP (
31             CLK      => CLK,      RST      => RST,      START      => START,
32             LOADB    => LOADB,    LOADQ    => LOADQ,    RESULT    => RESULT,
33             MULT_IN  => MULT_IN,  MULT_OUT  => MULT_OUT);
34
35         clk_process: PROCESS
36         BEGIN
37             WAIT FOR OFFSET;
38             CLOCK_LOOP : LOOP
39                 CLK <= '0';   WAIT FOR (PERIOD - (PERIOD * DUTY_CYCLE));
40                 CLK <= '1';   WAIT FOR (PERIOD * DUTY_CYCLE);
41             END LOOP CLOCK_LOOP;
42         END PROCESS;
43
44         asig_process: PROCESS
45         BEGIN
46             WAIT FOR 200 PS; RST <= '1';
47             WAIT FOR 200 PS; RST <= '0';
48             WAIT FOR 200 PS; LOADB <= '1'; LOADQ <= '0'; MULT_IN <= "1010";
49             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '0'; MULT_IN <= "0000";
50             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '1'; MULT_IN <= "0101";
51             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '0'; MULT_IN <= "0000";
52             WAIT FOR 200 PS; START <= '1';
53             WAIT FOR 200 PS; START <= '0';
54             WAIT FOR 3400 PS;
55         END PROCESS;
56     END;
```

```

1      -- Iván Cabrera Altamirano, Tarea #3, Testbench for 8 bits.
2      LIBRARY ieee;
3      USE ieee.std_logic_1164.ALL;
4      USE ieee.std_logic_unsigned.all;
5      USE ieee.numeric_std.ALL;
6
7      ENTITY TB8BitM1 IS END TB8BitM1;
8
9      ARCHITECTURE behavior OF TB8BitM1 IS
10         COMPONENT binary_multiplier
11         PORT(CLK, RST, START, LOADB, LOADQ :      IN      std_logic;
12              RESULT :                            OUT     std_logic;
13              MULT_IN :                           IN      std_logic_vector(7 DOWNTO 0);
14              MULT_OUT :                          OUT     std_logic_vector(15 DOWNTO 0));
15         END COMPONENT;
16
17         -- Entradas
18         SIGNAL CLK, RST, START :                  std_logic;
19         SIGNAL LOADB, LOADQ, RESULT :             std_logic;
20         SIGNAL MULT_IN :                          std_logic_vector(7 DOWNTO 0);
21         -- Salidas
22         SIGNAL MULT_OUT :                         std_logic_vector(15 DOWNTO 0);
23
24         CONSTANT PERIOD :                        TIME := 100 PS;
25         CONSTANT DUTY_CYCLE :                   REAL := 0.5;
26         CONSTANT OFFSET :                       TIME := 50 PS;
27
28     BEGIN
29         -- Instanciando la unidad a probar.
30         uut: binary_multiplier PORT MAP (
31             CLK      => CLK,      RST      => RST,      START      => START,
32             LOADB    => LOADB,    LOADQ    => LOADQ,    RESULT    => RESULT,
33             MULT_IN  => MULT_IN,  MULT_OUT  => MULT_OUT);
34
35         clk_process: PROCESS
36         BEGIN
37             WAIT FOR OFFSET;
38             CLOCK_LOOP : LOOP
39                 CLK <= '0';   WAIT FOR (PERIOD - (PERIOD * DUTY_CYCLE));
40                 CLK <= '1';   WAIT FOR (PERIOD * DUTY_CYCLE);
41             END LOOP CLOCK_LOOP;
42         END PROCESS;
43
44         asig_process: PROCESS
45         BEGIN
46             WAIT FOR 200 PS; RST <= '1';
47             WAIT FOR 200 PS; RST <= '0';
48             WAIT FOR 200 PS; LOADB <= '1'; LOADQ <= '0'; MULT_IN <= "10101010";
49             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '0'; MULT_IN <= "00000000";
50             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '1'; MULT_IN <= "01010101";
51             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '0'; MULT_IN <= "00000000";
52             WAIT FOR 200 PS; START <= '1';
53             WAIT FOR 200 PS; START <= '0';
54             WAIT FOR 3400 PS;
55         END PROCESS;
56     END;

```

```

1      -- Iván Cabrera Altamirano, Tarea #3, Testbench for 16 bits.
2      LIBRARY ieee;
3      USE ieee.std_logic_1164.ALL;
4      USE ieee.std_logic_unsigned.all;
5      USE ieee.numeric_std.ALL;
6
7      ENTITY TB16BitM1 IS END TB16BitM1;
8
9      ARCHITECTURE behavior OF TB16BitM1 IS
10         COMPONENT binary_multiplier
11         PORT(CLK, RST, START, LOADB, LOADQ :      IN      std_logic;
12              RESULT :                          OUT    std_logic;
13              MULT_IN :                         IN      std_logic_vector(15 DOWNTO 0);
14              MULT_OUT :                       OUT    std_logic_vector(31 DOWNTO 0));
15         END COMPONENT;
16
17         -- Entradas
18         SIGNAL CLK, RST, START :                std_logic;
19         SIGNAL LOADB, LOADQ, RESULT :           std_logic;
20         SIGNAL MULT_IN :                       std_logic_vector(15 DOWNTO 0);
21         -- Salidas
22         SIGNAL MULT_OUT :                      std_logic_vector(31 DOWNTO 0);
23
24         CONSTANT PERIOD :                      TIME := 100 PS;
25         CONSTANT DUTY_CYCLE :                 REAL := 0.5;
26         CONSTANT OFFSET :                    TIME := 50 PS;
27
28     BEGIN
29         -- Instanciando la unidad a probar.
30         uut: binary_multiplier PORT MAP (
31             CLK      => CLK,      RST      => RST,      START      => START,
32             LOADB    => LOADB,    LOADQ    => LOADQ,    RESULT    => RESULT,
33             MULT_IN  => MULT_IN,  MULT_OUT => MULT_OUT);
34
35         clk_process: PROCESS
36         BEGIN
37             WAIT FOR OFFSET;
38             CLOCK_LOOP : LOOP
39                 CLK <= '0';   WAIT FOR (PERIOD - (PERIOD * DUTY_CYCLE));
40                 CLK <= '1';   WAIT FOR (PERIOD * DUTY_CYCLE);
41             END LOOP CLOCK_LOOP;
42         END PROCESS;
43
44         asig_process: PROCESS
45         BEGIN
46             WAIT FOR 200 PS; RST <= '1';
47             WAIT FOR 200 PS; RST <= '0';
48             WAIT FOR 200 PS; LOADB <= '1'; LOADQ <= '0'; MULT_IN <= "1010101010101010";
49
49             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '0'; MULT_IN <= "0000000000000000";
50             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '1'; MULT_IN <= "0101010101010101";
51             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '0'; MULT_IN <= "0000000000000000";
52             WAIT FOR 200 PS; START <= '1';
53             WAIT FOR 200 PS; START <= '0';
54             WAIT FOR 3400 PS;
55         END PROCESS;
56     END;

```

```

1      -- Iván Cabrera Altamirano, Tarea #3, Testbench for 32 bits.
2      LIBRARY ieee;
3      USE ieee.std_logic_1164.ALL;
4      USE ieee.std_logic_unsigned.all;
5      USE ieee.numeric_std.ALL;
6
7      ENTITY TB32BitM1 IS END TB32BitM1;
8
9      ARCHITECTURE behavior OF TB32BitM1 IS
10         COMPONENT binary_multiplier
11         PORT(CLK, RST, START, LOADB, LOADQ :      IN      std_logic;
12              RESULT :                          OUT      std_logic;
13              MULT_IN :                          IN      std_logic_vector(31 DOWNTO 0);
14              MULT_OUT :                         OUT      std_logic_vector(63 DOWNTO 0));
15         END COMPONENT;
16
17         -- Entradas
18         SIGNAL CLK, RST, START :                std_logic;
19         SIGNAL LOADB, LOADQ, RESULT :           std_logic;
20         SIGNAL MULT_IN :                        std_logic_vector(31 DOWNTO 0);
21         -- Salidas
22         SIGNAL MULT_OUT :                       std_logic_vector(63 DOWNTO 0);
23
24         CONSTANT PERIOD :                       TIME := 100 PS;
25         CONSTANT DUTY_CYCLE :                   REAL := 0.5;
26         CONSTANT OFFSET :                       TIME := 50 PS;
27
28     BEGIN
29         -- Instanciando la unidad a probar.
30         uut: binary_multiplier PORT MAP (
31             CLK      => CLK,      RST      => RST,      START      => START,
32             LOADB    => LOADB,    LOADQ    => LOADQ,    RESULT    => RESULT,
33             MULT_IN  => MULT_IN,  MULT_OUT => MULT_OUT);
34
35         clk_process: PROCESS
36         BEGIN
37             WAIT FOR OFFSET;
38             CLOCK_LOOP : LOOP
39                 CLK <= '0';  WAIT FOR (PERIOD - (PERIOD * DUTY_CYCLE));
40                 CLK <= '1';  WAIT FOR (PERIOD * DUTY_CYCLE);
41             END LOOP CLOCK_LOOP;
42         END PROCESS;
43
44         asig_process: PROCESS
45         BEGIN
46             WAIT FOR 200 PS; RST <= '1';
47             WAIT FOR 200 PS; RST <= '0';
48             WAIT FOR 200 PS; LOADB <= '1'; LOADQ <= '0';
49             MULT_IN <= "10101010101010101010101010101010";
50             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '0';
51             MULT_IN <= "00000000000000000000000000000000";
52             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '1';
53             MULT_IN <= "01010101010101010101010101010101";
54             WAIT FOR 200 PS; LOADB <= '0'; LOADQ <= '0';
55             MULT_IN <= "00000000000000000000000000000000";
56             WAIT FOR 200 PS; START <= '1';
57             WAIT FOR 200 PS; START <= '0';
58             WAIT FOR 6500 PS;
59         END PROCESS;
60     END;

```


Resultado de las simulaciones

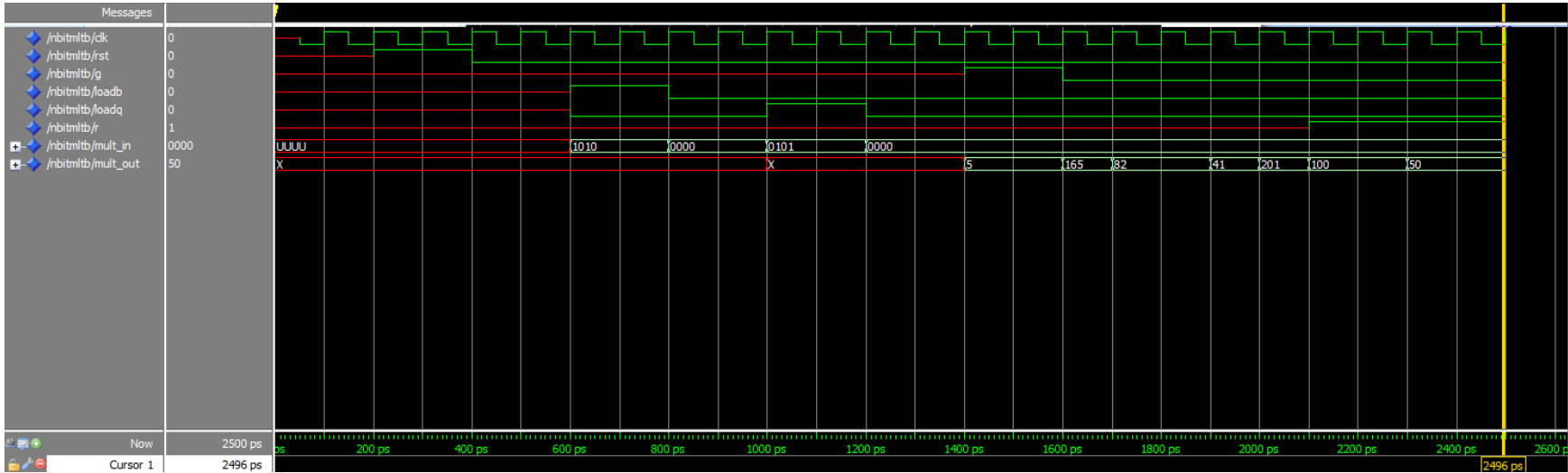


Figura 2. Resultado de la simulación con N = 4 bits.

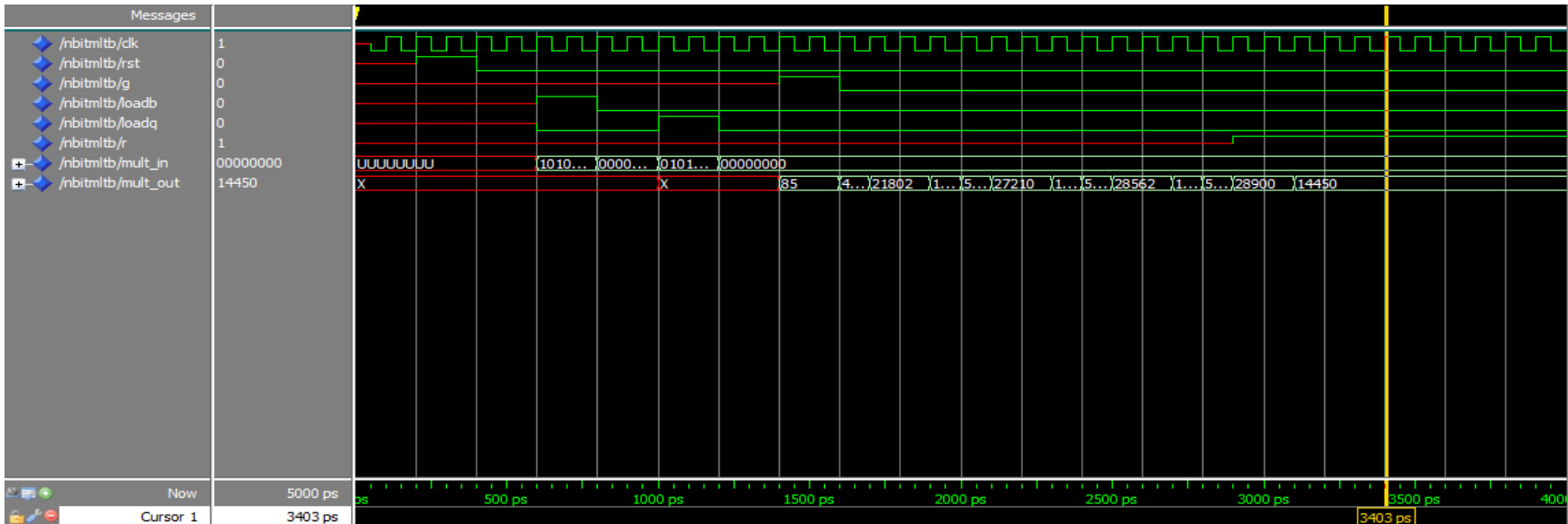


Figura 3. Resultado de la Simulación con N = 8 bits.

Medidas de Desempeño:

A continuación se presenta a manera de resumen las medidas de desempeño para la implementación con diferentes longitudes de palabras. (N se encuentra expresado en bits).

	N =4	N = 8	N = 16	N = 32	N = 64
Numero de Slices	18	32	61	116	227
Ciclos de Reloj para obtener el resultado	9	17	33	65	129
Frecuencia Máxima de Operación	235.255 Mhz	226.652 Mhz	207.295 Mhz	176.098 Mhz	136.488 Mhz

Tabla 1. Desempeño del multiplicador para diferentes valores de N.

- Los ciclos de reloj son contados a partir de que comienza el algoritmo de multiplicación.
- El numero de slices, se tomo después de la fase de implementación, dado que en la fase de síntesis el dato proporcionado por el entorno de desarrollo es estimado.

