Bionano

Digital System Design Course

El Logaritmo

Diseño:
Álvaro José Caicedo Beltrán

1. Temática del Logaritmo

- La temática a trabajar se divide en las siguientes secciones:
  - Definición de la operación
  - Propiedades del logaritmo
  - Algoritmo para el cálculo en base entera > 1
    - Obtención de la característica
    - Obtención de la mantisa
  - Logaritmo en base 2
    - Datapath
    - Algoritmo de cálculo
    - Diagrama de estados
    - Tablas de transición y de salida
  - Implementación en AHDL para Quartus II
    - Programa
    - Simulaciones

2. Definición de Logaritmo

- El logaritmo es la séptima operación aritmética. Es la operación inversa a la exponencial o antilogaritmo. El logaritmo se realiza con dos parámetros, la base (b) y el número dado (X).

$$Y = \log_b X \quad \Rightarrow \quad X = b^Y$$

- Donde Y es la potencia a la que se eleva la base *b* para obtener X.
- El logaritmo Y se compone de una parte entera llamada característica (C) y de una fracción decimal conocida como mantisa (M).

$$Y = C + M$$

- Para un sistema con base entera mayor a 1 se tiene que la posición de la cifra de mayor peso diferente de cero corresponde a la característica.
- La mantisa siempre es el logaritmo de un número normalizado N de la forma 1≤N<b y 0 ≤M ≤1.

$$M = \log_b N$$

- La complejidad del cálculo del logaritmo radica en la mantisa. Se pueden encontrar tablas con las mantisas en las bases más conocidas: base 10 y base *e*. Ej:

$$\log_{10} 152.3 = 2,1827 \quad \Rightarrow \quad 10^{2,1827} = 152.3$$

$$X = 152.3, \quad Y = 2.1827, \quad b = 10$$

$$C = 2, \quad M = 0.1827, \quad N = 1.523$$

3. Propiedades del logaritmo

- El logaritmo goza de propiedades que facilitan el cálculo de operaciones aritméticas como la multiplicación, la división, la radicación y la potenciación. Es complicado para remplazar la adición y sustracción. El logaritmo en base positiva sólo está definido para X>0. Se tiene para b>1, b→{Z}:

1) $\log_b b = 1$	5) $\log_b(X \cdot Y) = \log_b X + \log_b Y$
2) $\log_b 1 = 0$	6) $\log_b(X / Y) = \log_b X - \log_b Y$
3) $\log_b 0 = -\infty$	7) $\log_b(X)^n = n \log_b X$
4) $\log_b a = NaN$ $a < 0$	8) $\log_k X = \dfrac{\log_b X}{\log_b k}$

4. Algoritmo para cálculo en base entera > 1

- El cálculo de logaritmo se divide en dos procedimientos, el primero calcula la característica a medida que normaliza el número dado, para luego calcular la mantisa.
- Antes de calcular se solicita el número dado, y al final del cálculo se combinan la característica y la mantisa mediante la suma.
- Si se desea el logaritmo en una base diferente debe acudirse a la propiedad (8).

4.1 Obtención de la característica

- El cálculo de la característica se realiza haciendo divisiones sucesivas (X>b) o multiplicaciones sucesivas (0<X<1) por la base con el fin de convertir X en un número normalizado (1≤N<b).

$$N = \frac{X}{b^C}$$

- Usando divisiones C>0, usando multiplicaciones C<0, si no es necesario siendo X un número normalizado entonces C=0. Esto se basa en la recurrencia de la siguiente propiedad:

$$\log_b X = \log_b\left(\frac{X}{b}b\right) = \log_b(b) + \log_b\left(\frac{X}{b}\right) = 1 + \log_b\left(\frac{X}{b}\right)$$

- Para un número X de *n* cifras parte entera y *m* cifras parte decimal se tiene el siguiente algoritmo:
  - 1) C = n-1, X = X0 (Número dado)
  - 2) Mientras cifra mayor de X sea cero, hacer:
    - C = C-1, X = X/b
    - Si C es –m-1, entonces Y = -inf, Salir.
  - 3) Calcular M.
- Como se observa, basta un estado para calcular C, ya que los otros dos estados son de iniciación y para la mantisa.

4.2 Obtención de la mantisa

- El problema de la mantisa radica en el hecho que el número ya no tiene característica. Entonces debe hallarse una identidad que permita modificar el normalizado de forma que podamos calcular subcaracterísticas C_{-i}. Dicha identidad es:

$$M = \log_b N_0 = \log_b (N_0)^{\frac{W}{W}}$$

$$M = \frac{1}{W} \log_b N_o{}^W = W^{-1}\left(C_{-1} + \log_b N_{-1}\right)$$

- Si W>1 es probable que N_0 sea mayor que b y se pueda obtener C_{-1}. Esto hace que aparezca nuevamente un normal denominado ahora N_{-1}

- Este procedimiento se puede realizar continuamente un número de veces Q sobre cada N_{-i}. El último N sin descomponer se convierte en el residuo R y de él depende la precisión del resultado.

$$M = \log_b N_0 = \left(\sum_{i=1}^{Q} W^{-i} C_{-i} \right) + \frac{1}{W^Q} \log_b N_{-Q}$$

$$R = \frac{1}{W^Q} \log_b N_{-Q}$$

$$E_{\max} \leq W^{-Q}$$

- El error máximo se halla asumiendo $\log_b N_{-Q} \approx 1$.

- Si W=b (Con b entero) cada C-i se vuelve una cifra de la mantisa. Pero N^b es muy grande para *b* grande (Ej: Sistema hexadecimal y decimal).
- La operación más sencilla se hace con W=2.
- Cómo el error depende inversamente de W y de Q entonces a mayor potencia W y mayor número de iteraciones Q, se obtiene un menor error y por tanto mayor exactitud en el cálculo.

- El algoritmo para calcular la mantisa para Q iteraciones y potencia W se estructura como:
  - 1) $i = 0$, $M=0$, $X = N_0$ (Número normalizado)
  - 2) $X = X$^W
  - 3) Normalizar X (obtener C_{-i} y $X=N_{-i}$)
  - 4) $M = M + C_{-i}*W$^$(-i)$, $i = i+1$
    - Si i=Q entonces (5), sino (3)
  - 5) Presentar Y=C + M. Salir.

5. Logaritmo binario (b=2)

- El algoritmo para base 2 tiene varias simplificaciones a causa del reducido alfabeto, sólo dos símbolos: 0 y 1.
- La potencia más simple es a la vez la base, W=b=2, luego cada subcaracterística corresponde a un bit en la mantisa.
- Con W=2 y como la parte entera de N es 1, se consigue que N^2 siempre sea menor que 4, es decir, con dos bits se representa la parte entera de N^2: 01,10,11; donde la subcaracterística es el bit 1: 0,1,1. Este bit es el MSB de X.

5.1 Datapath

- Para normalizar X se debe dividir, entonces este debe estar almacenado en un registro de desplazamiento. Como siempre se revisa el MSB entonces debe ser desplazamiento a la izquierdo, sin perder la ubicación del punto decimal. X tiene tamaño $n+m$.
- Para el flujo de datos se debe contar con un multiplicador para el cuadrado de X. La salida es de tamaño $2(n+m)$.
- Para seleccionar entre cargar el dato inicial en X o el cuadrado se requiere un multiplexor. El ancho del bus es $n+m$.
- La acumulación de bits en M se hace encolando desde el de mayor peso, luego M se almacena en un registro desplazador a la izquierda. El tamaño de M es de *q bits*.
- Un contador proporciona el valor de C, cuyo tamaño es el mayor entre log2 (n+1) y log2(m) más el bit de signo.

- Un sumador combina C con M para Y. Se puede evitar con lógica de complemento si C<0.
- Un contador Q se usa para las iteraciones en el cálculo en la mantisa. El tamaño es de log2(*q*).
- Las señales que proporciona el datapath al controlador son:
  - MSB: Bit más significativo de X.
  - C_m: Contador C ha alcanzado el valor –*m*-1. Sirve de indicador de -infinito (log2 0 = -∞).
  - Q0: Contador Q ha alcanzado el cero.
- La señal LSB es el msb de la parte despreciada de X^2.

- Las señales que proporciona el controlador para regulación de operaciones deben permitir:
  - Recargar contadores C y Q, y habilitar decremento.
  - Desplazamiento de M.
  - Selección entre X0 y X^2 para cargar en X.
  - Cargar X.
  - Desplazar X con cero o con el mayor bit despreciado de X^2.

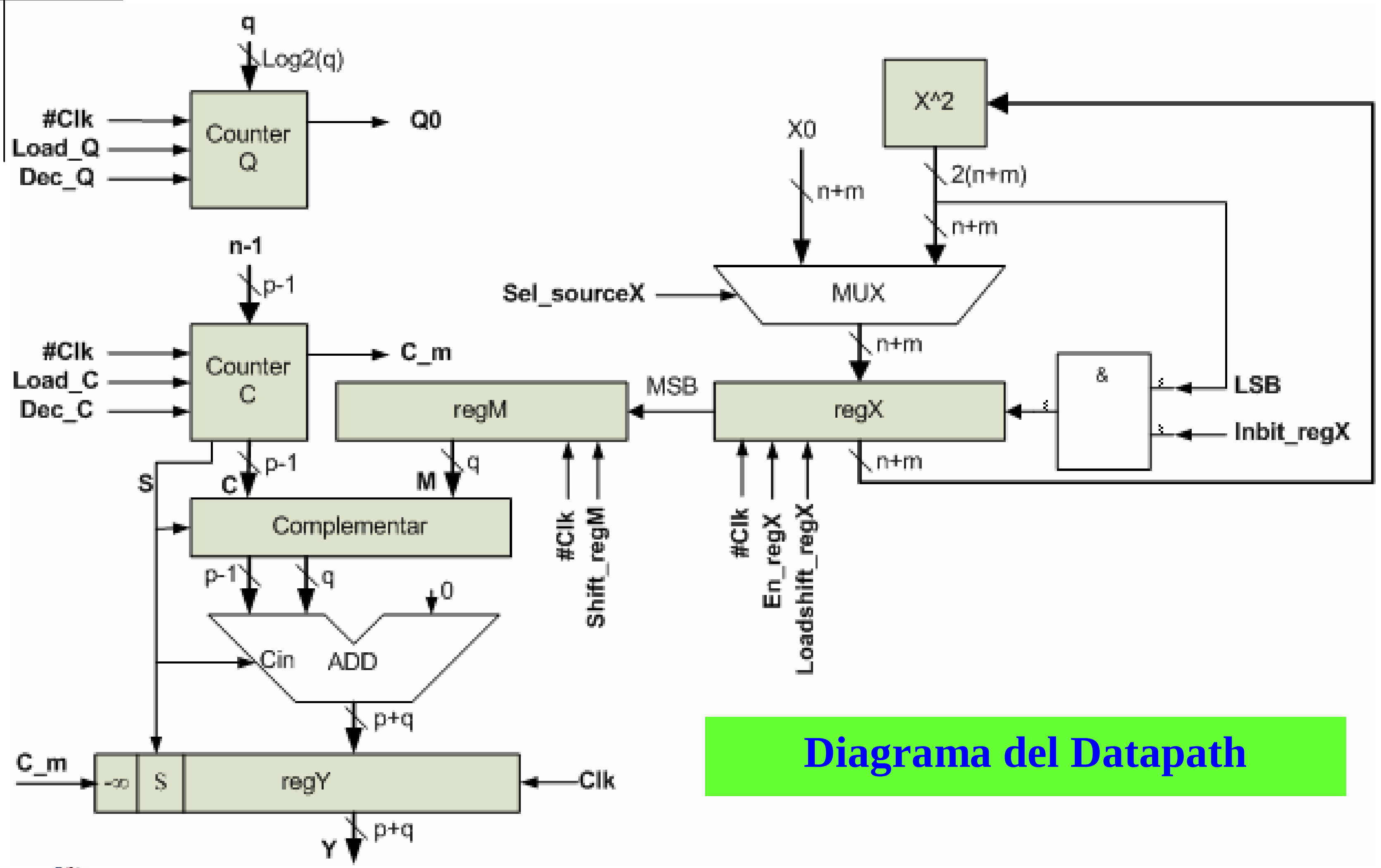
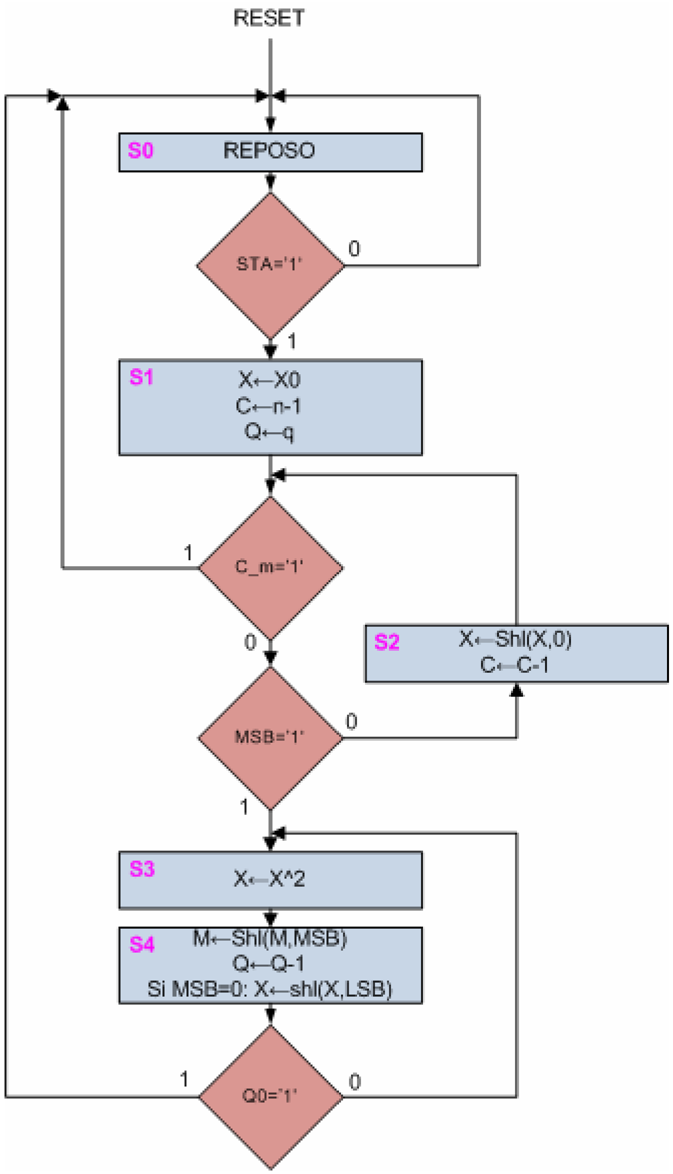
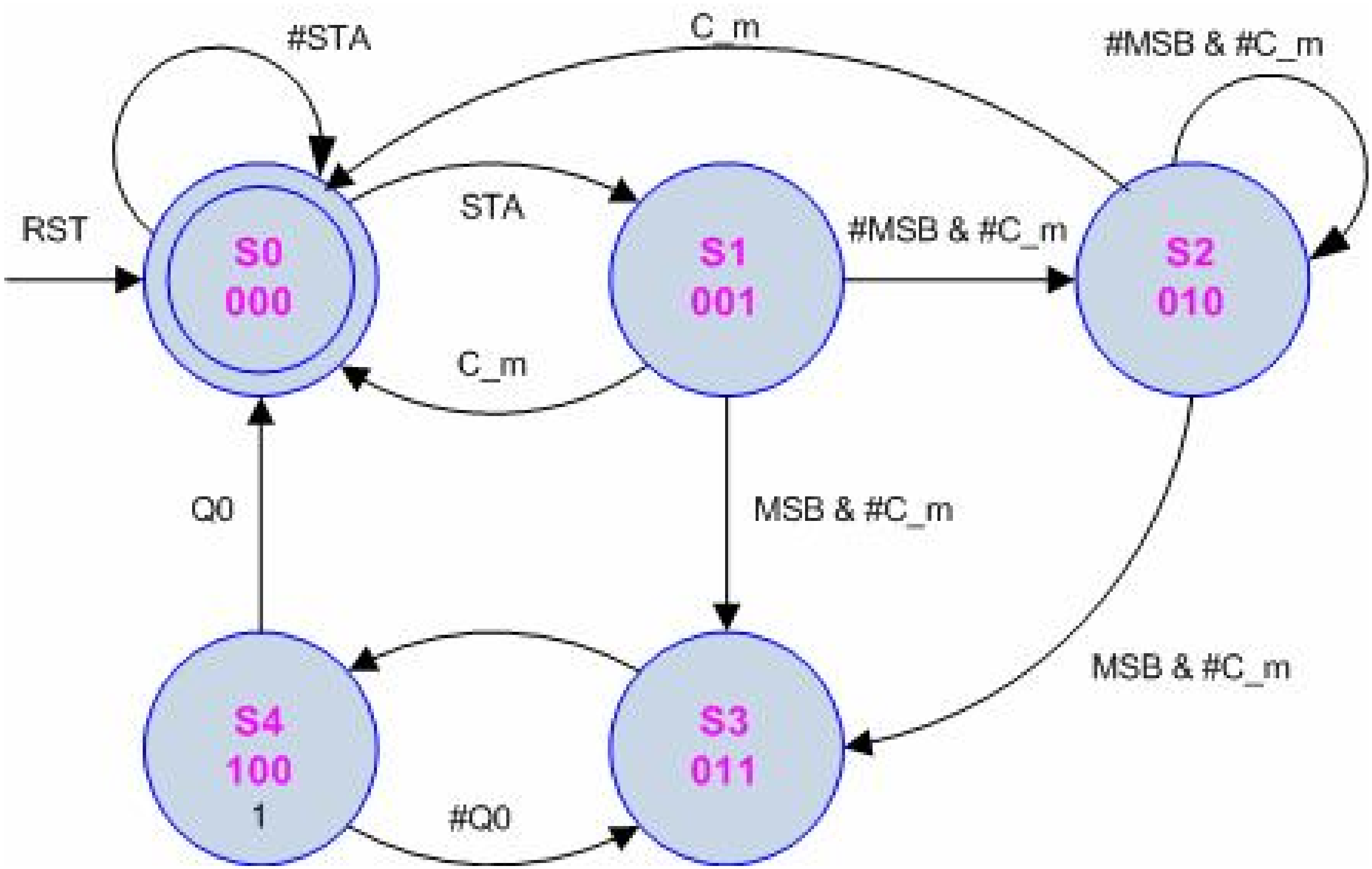


Diagrama del Datapath

5.2 Algoritmo del controlador



5.3 Diagrama de Estados del controlador



5.4 Tabla de transición de estados

- La transición de estados viene definida por la siguiente tabla:

SS	STA	Q0	C_m	MSB	SS*
S0	0	X	X	X	S0
S0	1	X	X	X	S1
S1	X	X	0	0	S2
S1	X	X	0	1	S3
S1	X	X	1	X	S0
S2	X	X	0	0	S2
S2	X	X	0	1	S3
S2	X	X	1	X	S0
S3	X	X	X	X	S4
S4	X	0	X	X	S3
S4	X	1	X	X	S0

5.5 Tabla de salida

	S0	S1	S2	S3	S4
Dec_C	0	0	1	0	0
Load_C	0	1	0	0	0
Dec_Q	0	0	0	0	1
Load_Q	0	1	0	0	0
Shift_regM	0	0	0	0	1
Sel_source_X	X	0	X	1	X
En_regX	0	1	1	1	0
Loadshift_regX	X	1	0	1	0
Inbit_regX	X	X	0	X	1
Ready	1	0	0	0	0

6. Implementación en AHDL

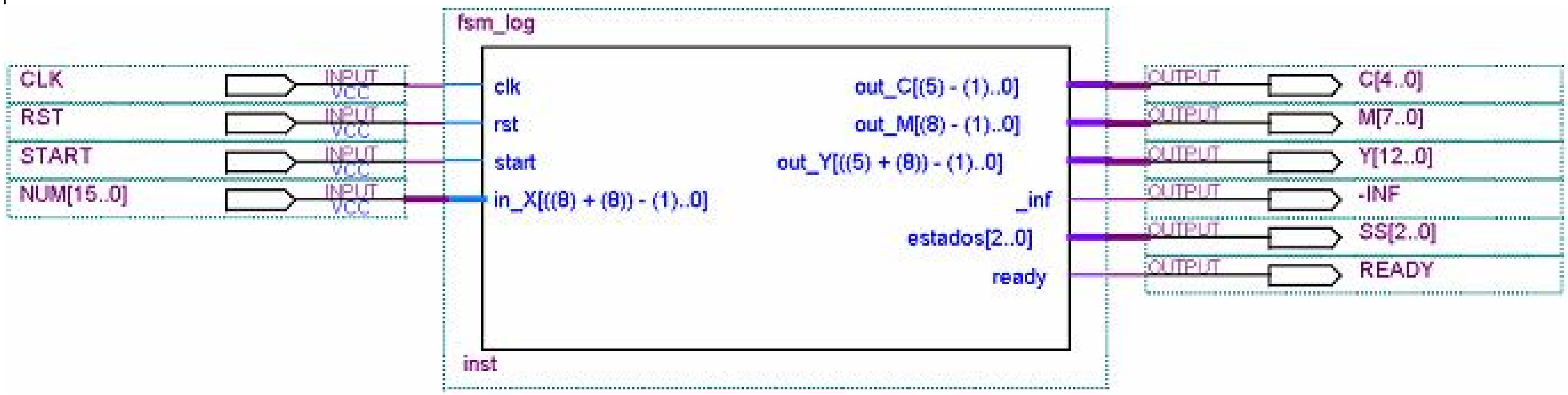
```
% Adaptación para N=8, M=8, Q=8;                                        %

TITLE "Logaritmo en base 2";

% Librerías de altera utilizadas - Megafunciones parametrizadas         %
INCLUDE "lpm_counter.inc";
INCLUDE "lpm_mux";
INCLUDE "lpm_shiftreg";
INCLUDE "lpm_mult";

CONSTANT N = 8;          % #bits de la parte entera de X                 %
CONSTANT M = 8;          % #bits de la parte decimal de X                %
CONSTANT Q = 8;          % #bits de la mantisa                           %
CONSTANT P = 5;          % #bits de la característica con signo          %

% Definición de puertos de i/o                                           %
SUBDESIGN fsm_log
(
    % Señal de reloj, reinicio, y arranque de la fsm                     %
    clk, rst, start                          :INPUT = GND;
    % Dato de entrada para calcularsele el logarítmo                     %
    in_X[(N+M-1)..0]                         :INPUT = GND;
    % Datos de salida: Característica (C2) y 0.Mantisa                   %
    out_C[(P-1)..0],out_M[(Q-1)..0]          :OUTPUT;
    % Salida Y=log2 X (SM) y señal para -inf                             %
    out_Y[(P+Q-1)..0], _inf                  :OUTPUT;
    % Estados de fsm y señal que indica listo para recalcular            %
    estados[2..0], ready                     :OUTPUT;
)

VARIABLE
    % Contador de P bits descendente para la característica              %
    contador_C: lpm_counter WITH (
        LPM_WIDTH = P,
        LPM_DIRECTION = "DOWN"
    );
    % Contador modulo Q descendente para bits de mantisa                 %
    contador_Q: lpm_counter WITH (
        LPM_WIDTH = LOG2(Q)+1,
        LPM_DIRECTION = "DOWN"
    );
```

```
% Registro de desplazamiento para la mantisa                   %
shifter_M: lpm_shiftreg WITH (
    LPM_WIDTH = Q,
    LPM_DIRECTION = "LEFT"
);
% Registro de desplazamiento para la variable X                %
shifter_X: lpm_shiftreg WITH (
    LPM_WIDTH = N+M,
    LPM_DIRECTION = "LEFT"
);
% Multiplexor para cargar X con valor inicial o con X^2         %
mux_X: lpm_mux WITH (
    LPM_WIDTH = N+M,
    LPM_SIZE = 2,
    LPM_WIDTHS = 1
);
% Multiplicador para calcular X^2. Salida truncada para X       %
cuadrado_X: lpm_mult WITH (
    LPM_WIDTHA = N+M,
    LPM_WIDTHB = N+M,
    LPM_WIDTHS = 1,
    LPM_WIDTHP = (N+M)+1
    % Bits más significativos de a*b + s                        %
    % Mostrando sólo los bits para X más el próximo LSB         %
);
msb         : NODE;         % Bit de salida del registro X      %
q_min       : NODE;         % Q alcanza valor 0                 %
c_min       : NODE;         % C alcanza valor -M-1              %

ss: MACHINE
    % Señales que dependen del estado presente                  %
    OF BITS(
    dec_C,          % Controla el decremento de C               %
    load_C,         % Recargar contador C                       %
    dec_Q,          % Decrementar el contador Q                 %
    load_Q,         % Recarga del contador Q                    %
    shift_regM,     % Desplazar mantisa a la izquierda          %
    sel_sourceX,    % Cargar X con in_X o con X^2               %
    en_regX,        % Habilitar carga/desplazamiento de X       %
    loadshift_regX, % Cargar/desplazar X si está habilitado     %
    inbit_regX, % Habilitar carga del msb de parte baja de X^2  %
    estados[2..0],  % Codificación de estados                   %
    ready           % Indicador de dato listo                   %
    )
```

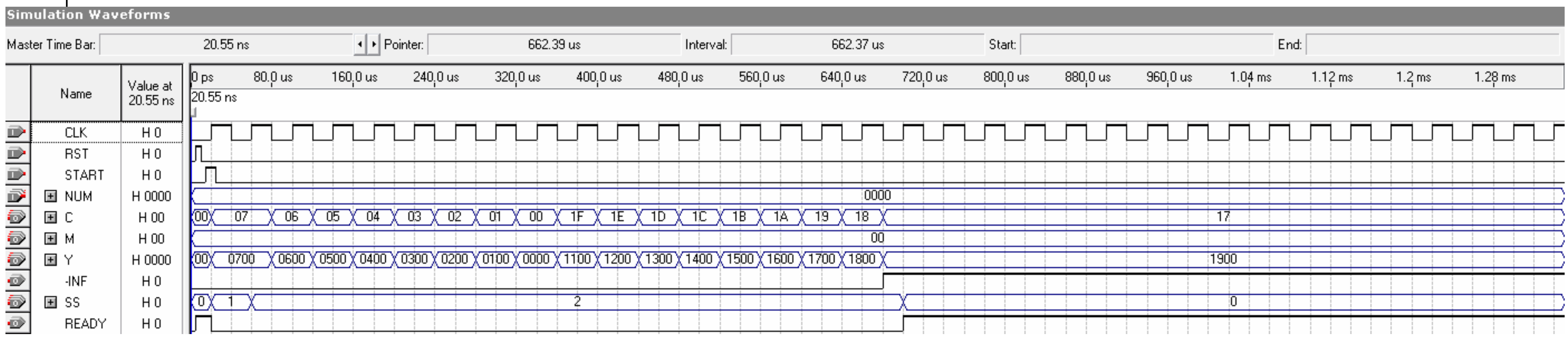
```
        WITH STATES(
            s0=B"00000X0XX0001",     % Entrega resultado              %
            s1=B"01010011X0010",     % Recarga C, Q y X=in_X          %
            s2=B"10000X1000100",     % Desplaza X,0 y decrementa C    %
            s3=B"00000111X0110",     % Hacer X = X^2 (parte alta+1)   %
            s4=B"00101X0011000"      % Desplazar regM, decrementar Q %
                                     % y normalizar X si MSB=0        %
        );
BEGIN
    DEFAULTS
        ss.reset = VCC;
    END DEFAULTS;

    % Selección de fuente de carga para X                              %
    mux_X.data[0][] = in_X[];
    mux_X.data[1][] = cuadrado_X.result[(N+M)..1];
    mux_X.sel[0] = sel_sourceX;

    % Manejo del contador de característica                            %
    contador_C.clock  = !clk;
    contador_C.clk_en = dec_C;
    contador_C.aload  = load_C;
    contador_C.data[] = N-1;
    out_C[] = contador_C.q[];

    % Manejo del contador Q                                            %
    contador_Q.clock  = !clk;
    contador_Q.clk_en = dec_Q;
    contador_Q.aload  = load_Q;
    contador_Q.data[] = Q;

    % Manejo del registro de desplazamiento X                          %
    shifter_X.clock = !clk;
    shifter_X.enable = en_regX # (inbit_regX & !msb);
    shifter_X.shiftin = inbit_regX & cuadrado_X.result[0];
    shifter_X.load = loadshift_regX;
    shifter_X.data[] = mux_X.result[];
    cuadrado_X.dataa[] = shifter_X.q[];
    cuadrado_X.datab[] = shifter_X.q[];
    msb = shifter_X.shiftout;
```

```
% Manejo del registro de desplazamiento de mantisa                    %
shifter_M.clock = !clk;
shifter_M.enable = shift_regM;
shifter_M.shiftin = msb;
out_M[] = shifter_M.q[];

% Señales de control de la máquina de estados                         %
ss.clk = clk;
ss.reset = rst;

% Expresiones de las señales de control o indicadores                 %
q_min = (contador_Q.q[] == 0);
c_min = (contador_C.q[] == -M-1);

% Formateando la salida (signo, característica, mantisa)              %
IF out_C[P-1] THEN
    out_Y[] = !(out_C[], out_M[])+1;        % -out_C - out_M    %
    out_Y[P+Q-1] = VCC;
ELSE
    out_Y[] = (out_C[], out_M[]);
END IF;

_inf = c_min;

% Tabla de transición de estados                                      %
TABLE
    ss, start,  q_min,  c_min,  msb =>  ss;
    s0, 0,      x,      x,      x   =>  s0;
    s0, 1,      x,      x,      x   =>  s1;
    s1, x,      x,      0,      0   =>  s2;
    s1, x,      x,      0,      1   =>  s3;
    s1, x,      x,      1,      x   =>  s0;
    s2, x,      x,      0,      0   =>  s2;
    s2, x,      x,      0,      1   =>  s3;
    s2, x,      x,      1,      x   =>  s0;
    s3, x,      x,      x,      x   =>  s4;
    s4, x,      0,      x,      x   =>  s3;
    s4, x,      1,      x,      x   =>  s0;
END TABLE;
END;
```

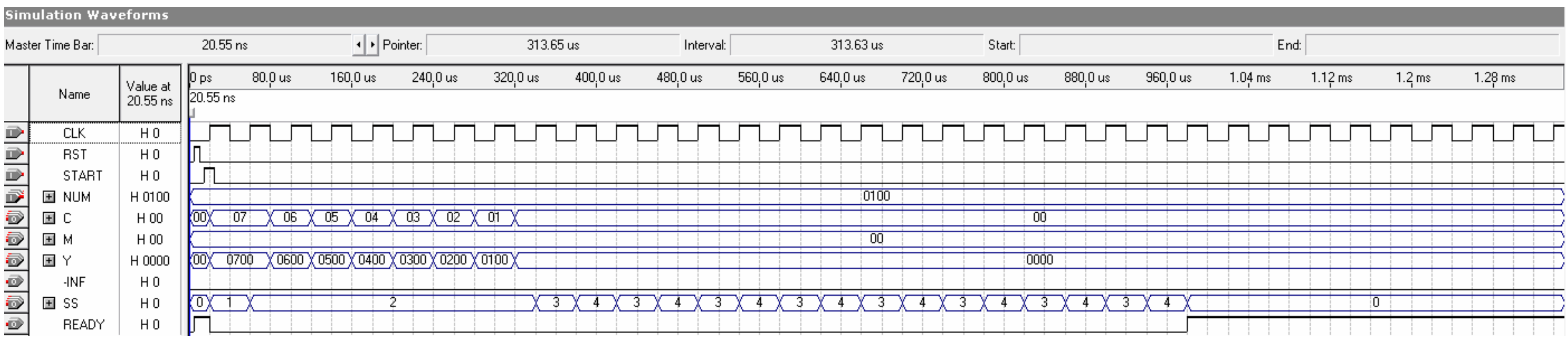
Bloque en Quartus II para n=8, m=8, q=8.



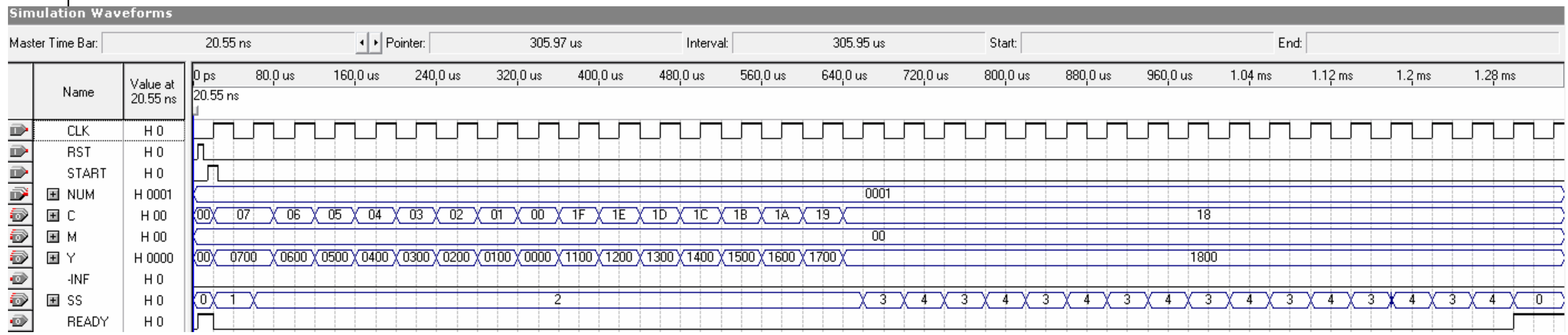
6.2 Simulación



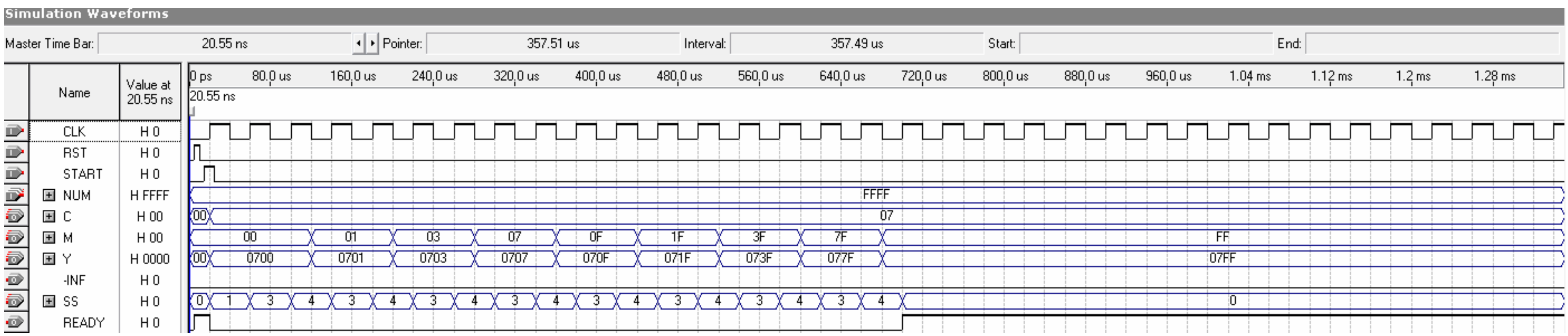
Log2 (0) = -inf



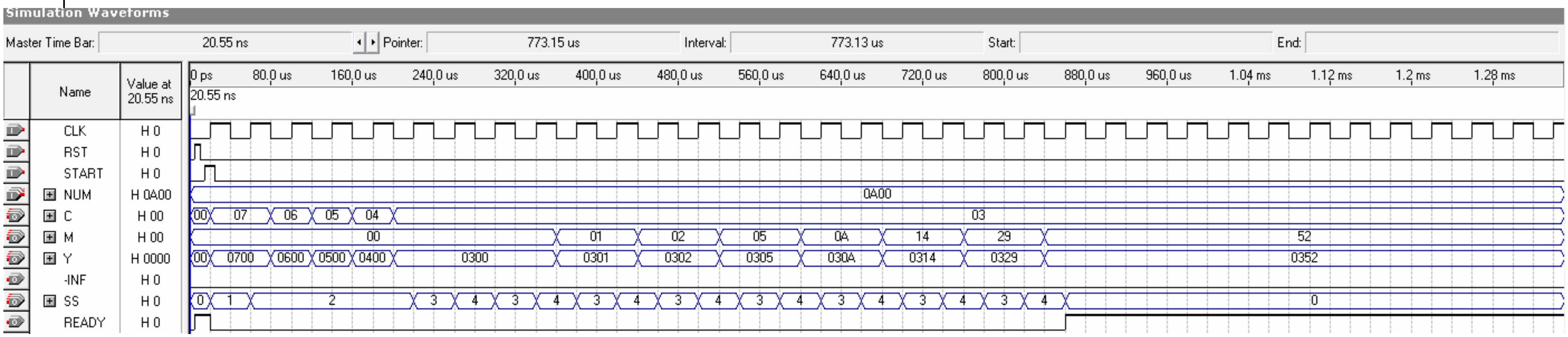
Log2 (1) = 0



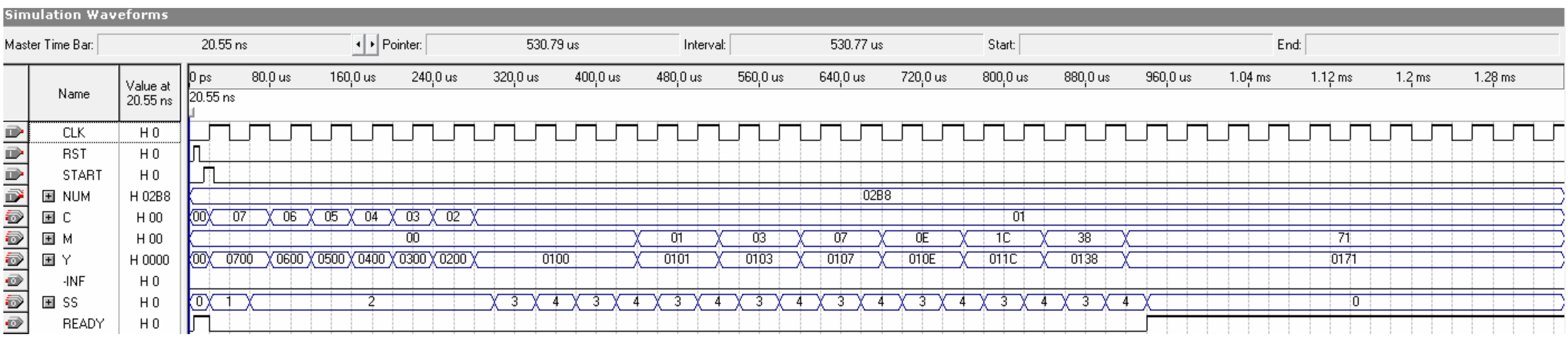
Log2 (2^-8) = -8



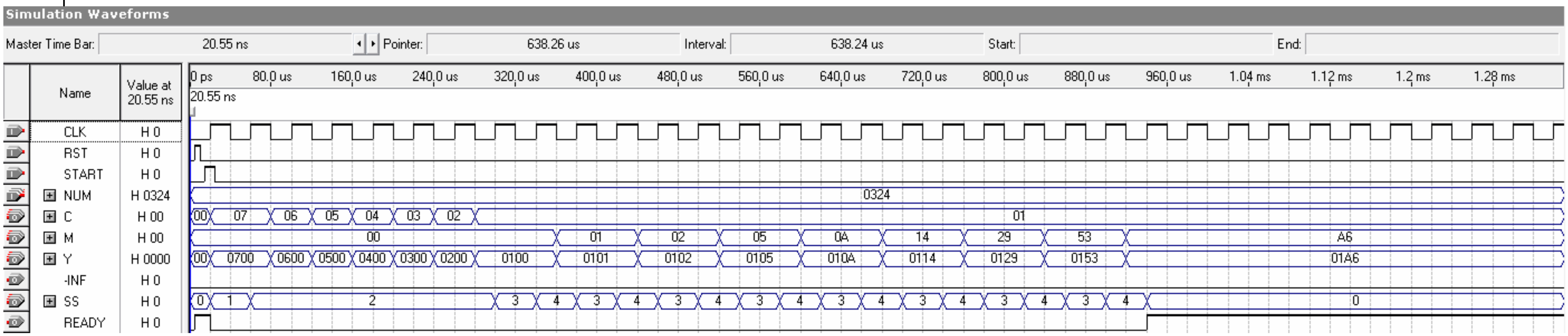
Log2 (255+255/256) = 7+255/256



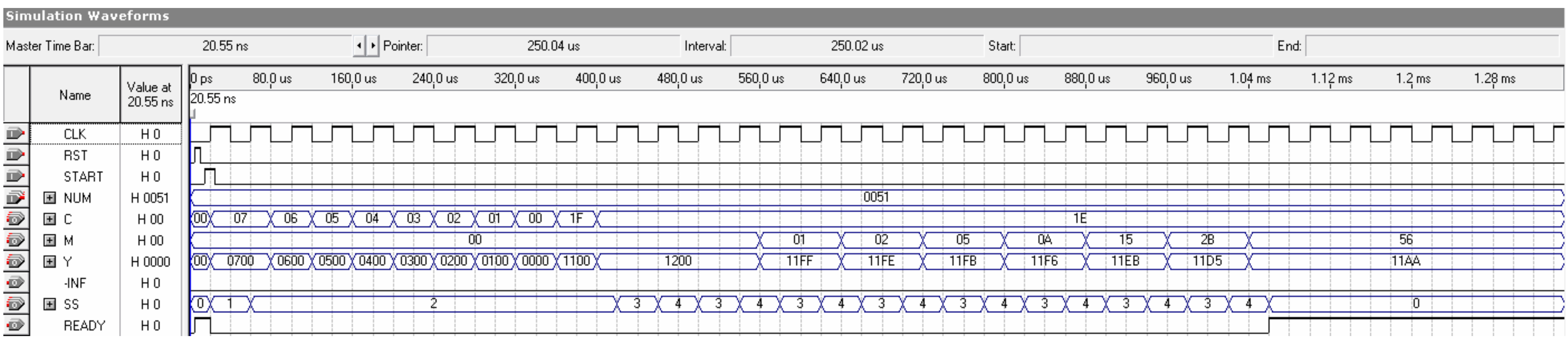
Log2 (10) = 3+82/256



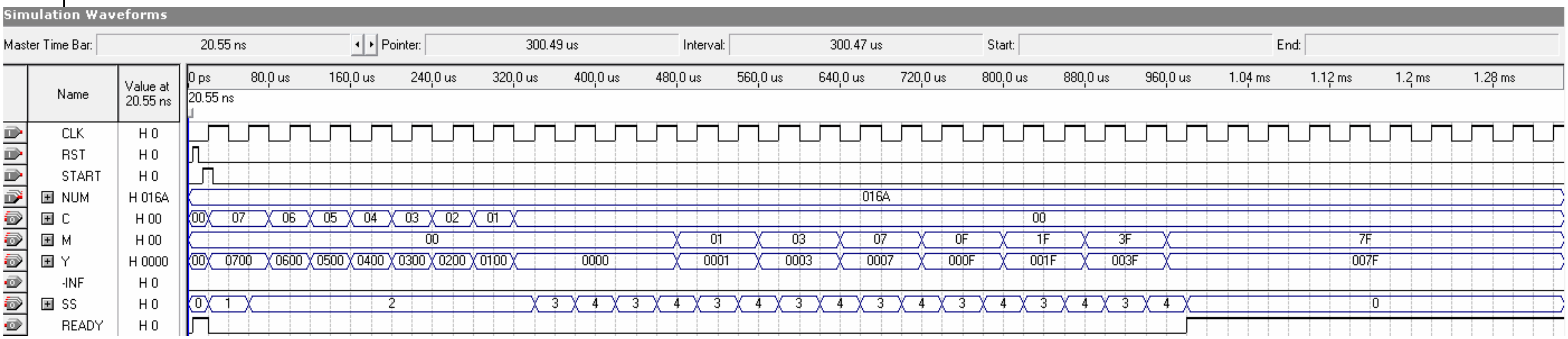
Log2 (e) ≈ Log2 (2+184/256) = 1+113/256



$$\text{Log2}\ (\Pi) \approx \text{Log2}\ (3+36/256) = 1+166/256$$



$$\text{Log2}\ (1/\Pi) \approx \text{Log2}\ (81/256) = -(1+170/256)$$



$$\text{Log2}(\sqrt{2}) \approx \text{Log2}(1+106/256) = 127/256$$


¿Inquietudes?


Gracias