Circuitos Integrados y Microelectrónica
Grado en Ingeniería en Tecnologías de Telecomunicación
Examen Parcial 1. Curso 2022-2023
21 marzo de 2023

Ejercicio 1 (40 min, 4 pto). SOLUCIÓN

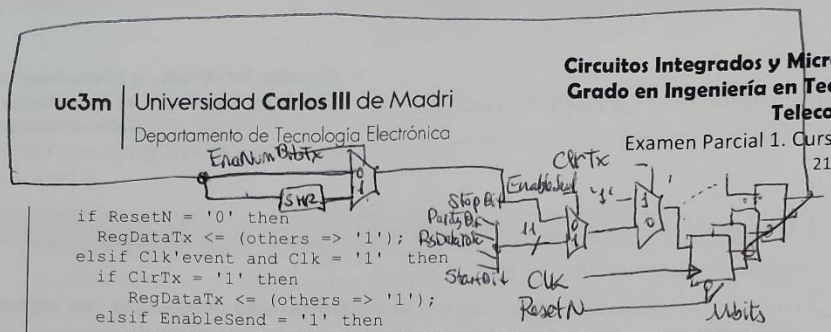
Para el circuito digital descrito en VHDL, se pide:

1. Complete las listas de sensibilidad para que las simulaciones y las síntesis lógica sean correctas y coherentes
2. Complete las declaraciones de las señales indicadas en la arquitectura
3. Dibuje el hardware resultante de la síntesis lógica del proceso P2. ¿Cuántos biestables serán necesarios?
4. Dibuje el diagrama de estados de la máquina de estados que describen P3 y P4

```
library IEEE;
use IEEE.std_logic_1164.all;
entity RS232_Tx is
  generic(
    gDataWidth     : natural := 8;  -- bits for Data
    gTotalNumBit   : natural := 11; -- WordWidth= Start(1)+Data(8)+Parity(1)+Stop(1)
    gBaudRate      : natural := 57600;      -- bps
    gClkFrequency  : natural := 100000000; -- Hz
  port(
    Clk            : in  std_logic;
    ResetN         : in  std_logic;
    RSDataToSend   : in  std_logic_vector(gDataWidth - 1 downto 0);
    EnableSend     : in  std_logic;
    TxD            : out std_logic;
    TxBusy         : out std_logic
  );
end RS232_Tx;
architecture Behavioural of RS232_Tx is
  constant StartBit        : std_logic := '0';
  constant StopBit         : std_logic := '1';
  ------------------------------------------------------------------
  constant cBaudRateCycles : natural := gClkFrequency/gBaudRate;
  signal EnaBaudRate       : std_logic;
  signal ClrBaudRate       : std_logic;
  signal CountBaudRate     : ;
  signal EndBaudRate       : std_logic;
  ------------------------------------------------------------------
  signal EnaNumBitsTx      : std_logic;
  signal ClrNumBitsTx      : std_logic;
  signal CountNumBitsTx    : natural range 0 to gTotalNumBit-1;
  signal EndNumBitsTx      : std_logic;
  ------------------------------------------------------------------
  type tStateTx is (Idle, Sending);
  signal CurrentState      : tStateTx;
  signal NextState         : TstateTx;
  signal RegDataTx         : std_logic_vector(gTotalNumBit-1 downto 0);
  signal ClrTx             : std_logic;
  signal ParityBit         : std_logic;
begin
  TxD          <= RegDataTx(0);
  TxBusy       <= NOT(ClrTx);
  ------------------------------------------------------------------------------
  P1: process( RSDataToSend )
    variable aux_Parity: std_logic;
  begin
    aux_Parity:= '0';
    for I in RSDataToSend'range loop
      aux_Parity:= aux_Parity XOR RSDataToSend(I);
    end loop;
    ParityBit     <= aux_Parity;
  end process P1;
  ------------------------------------------------------------------------------
  P2: process( ResetN, Clk )
  begin
```

natural 0 to cBaudRateCycles − 1;

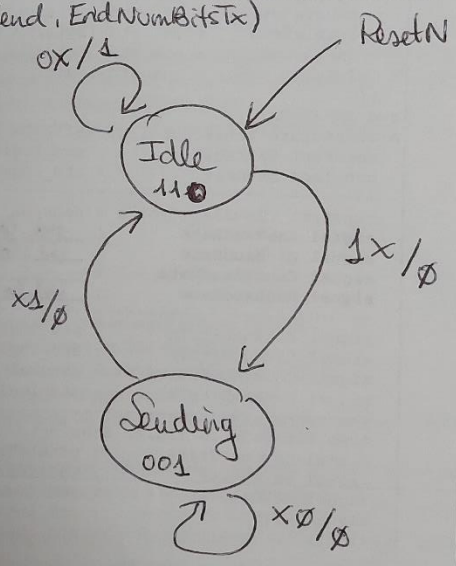
11bits




```
  if ResetN = '0' then
    RegDataTx <= (others => '1');
  elsif Clk'event and Clk = '1'  then
    if ClrTx = '1' then
       RegDataTx <= (others => '1');
    elsif EnableSend = '1' then
      RegDataTx <= StopBit & ParityBit & RSDataToSend & StartBit;
    elsif EnaNumBitsTx = '1' then
      RegDataTx <= '1' & RegDataTx(gTotalNumBit-1 downto 1);
    end if;
  end if;
end process P2;
-------------------------------------------------------------------------------
P3: process( ResetN, Clk )
begin
  if ResetN = '0' then
    CurrentState <= Idle;
  elsif Clk'event and Clk = '1' then
    CurrentState <= NextState;
  end if;
end process P3;
-------------------------------------------------------------------------------
P4: process( CurrentState, EnableSend, EndNumBitsTx)
begin
  ClrTx         <= '0';
  ClrBaudRate   <= '0';
  ClrNumBitsTx <= '0';
  EnaBaudRate    <= '0';
  case CurrentState is
    when Idle =>
      ClrBaudRate   <= '1';
      ClrNumBitsTx <= '1';
      ClrTx          <= '1';
      if EnableSend = '1' then
        NextState <= Sending;
        ClrTx   <= '0';
      else
        NextState <= Idle;
      end if;
    when Sending =>
      EnaBaudRate   <= '1';
      if EndNumBitsTx = '1'  then
        NextState <= Idle;
      else
        NextState <= Sending;
      end if;
    when others =>
      NextState <= Idle;
  end case;
end process P4;
-------------------------------------------------------------------------------
P5: process( ResetN, Clk )
begin
  if ResetN = '0' then
    CountBaudRate <= 0;
  elsif Clk'event and Clk = '1' then
    if ClrBaudRate = '1' then
       CountBaudRate <= 0;
    elsif  EnaBaudRate = '1' then
      if CountBaudRate = cBaudRateCycles - 1 then
        CountBaudRate <= 0;
      else
       CountBaudRate <= CountBaudRate + 1;
      end if;
    end if;
```

Salidas
ClrBaudRate, ClrNumBitsTx, ClrTx → Moore, EnaBaudRate
Ready

Entradas
EnableSend
EndNumBitsTx



```
    end if;
  end process P5;
  EndBaudRate <= '1' when ((CountBaudRate = cBaudRateCycles - 1) AND
                           (EnaBaudRate = '1'))
                 else '0';
  ------------------------------------------------------------------------------
  P6: EnaNumBitsTx <= EndBaudRate;
  process(                                    )
  begin
    if ResetN = '0' then
      CountNumBitsTx <= 0;
    elsif Clk'event and Clk = '1' then
      if ClrNumBitsTx = '1' then
          CountNumBitsTx <= 0;
      elsif  EnaNumBitsTx = '1' then
        if CountNumBitsTx = gTotalNumBit - 1 then
          CountNumBitsTx <= 0;
        else
          CountNumBitsTx <= CountNumBitsTx + 1;
        end if;
      end if;
    end if;
  end process P6;
  EndNumBitsTx <= '1' when ((CountNumBitsTx = gTotalNumBit - 1) AND
                            (EnaNumBitsTx = '1'))
                  else '0';
end Behavioural;
```

Criterios

(Pi)

(a) LS proc. comb { OK → 0'5; incompletas → Ø; excesivas → 0'25 }

LS proc seq. { incompletas → Ø; excesivas → 0'25; OK → 0'5 }

(b) Declaraciones 0'25 / dec → { std_logic_vector Ø; natural sin sup 0'05; unsigned " " 0'05 }

CountBaudRate

(c) HW → No #bits -0'2
No realimentar Registro -0'2
No Preset N -0'2
Reset secuencial -0'2
Clear realimentado -0'2
AND para concatenar -0'2
AND para Shift -0'2

No HW para Shift y/o Concat -0'2

(d) FSM
2 FSM -0'25
3 estados -0'25
ResetN serial -0'2
ClrTx Moore -0'4
No todas transic -0'2
No todas Salidas -0'2
No ResetN a estado inicial -0'1
Mal valor Clear -0'1
Comparadores para bit Num -0'1
No Muxes -0'2

Ejercicio 2 (50 min, 6 pto)

Se quiere diseñar un circuito digital que procesa datos, realizando la función exponencial. El circuito deberá ejecutar la operación e^X. Para diseñar el circuito se va a utilizar su representación en serie de Taylor, truncando en el cuarto elemento.

$$Y(X) = 1 + X + \frac{X^2}{2!} + \frac{X^3}{3!} \qquad (1)$$

Para que esta aproximación sea válida, los datos de entrada (X) deben estar en el rango entre -1 y +1. Para poder implementar el algoritmo utilizando números enteros, se realiza un cambio de variable (Z=X*128), de modo que Z sea un número entero y se represente con 8 bits (en CA2). Nótese que con este cambio se descarta el valor +1 en el rango de X. Con este cambio, la ecuación queda:

$$Y(Z) = 1 + \frac{Z}{2^7} + \frac{Z^2}{2 \cdot 2^{14}} + \frac{Z^3}{6 \cdot 2^{21}} \qquad (2)$$

Con el rango de entradas que se desea, el resultado de la función resulta entre e^{-1} y $e^{127/128}$. Se quiere expresar dicho resultado como un número entero de 10 bits, de modo que hay que multiplicar la ecuación anterior por 128 (2^7), quedando la ecuación que se quiere implementar mediante un circuito:

$$W(Z) = 2^7\left(1 + \frac{Z}{2^7} + \frac{Z^2}{2 \cdot 2^{14}} + \frac{Z^3}{6 \cdot 2^{21}}\right) = 2^7 + Z + \frac{Z^2}{2^8} + \frac{Z^3}{3 \cdot 2^{15}} \qquad (3)$$

El sistema cuenta con un reloj (Clk) de 100 kHz para realizar las operaciones, una señal de inicialización asíncrona (Reset) y un Enable de carga del dato de entrada, que se activa durante un ciclo de reloj para que el circuito cargue el dato de entrada (frecuencia de muestreo 1kHz). La entidad del circuito es:

```
entity EXP_23 is
  port (
    Clk       : in  std_logic;
    Reset     : in  std_logic;
    Enable    : in  std_logic;
    Data_in   : in  signed(7 downto 0);  -- Z
    Data_out  : out signed(9 downto 0)); -- W(Z)
end EXP_23;
```

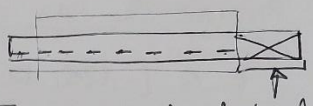
Para todas las preguntas, razone sus respuestas.

1. Compruebe los rangos de las variables y las funciones que se van a utilizar, es decir:
   Si el rango de la variable Z es [-128, 127] (número entero en CA2 con 8 bits):
   - 0'3 *a.* Especifique el rango de la variable X que se corresponde con el rango de Z
   - 0'6 *b.* Especifique los rangos de Y(Z) y W(Z) según el rango de Z
   - 0'1 c. Compruebe que todos ellos están dentro de las especificaciones (Z es signed de 8 bits y W(Z) es signed de 10 bits).
2. Exprese la función W(Z) de modo que se pueda realizar con números enteros, y sólo con divisiones entre potencias de 2. Los coeficientes necesarios deben ser de 8 bits en complemento a 2.
3. Se quiere hacer una arquitectura paralela.
   - 0'6 a. Dibújela en nivel de transferencia de registros (RTL)
   - 0'2 b. Determine razonadamente el número de bits de cada sumando de la ecuación.
   - 0'2 c. Indique el número de bits que se necesitan para representar el resultado final, sin despreciar ningún dígito. Si el resultado final debe tener 10 bits, indique los bits que se deben despreciar.
4. Asumir que la entrada se captura con la señal Enable, y la salida está registrada, disponible en el ciclo de reloj siguiente.
   - 0'2 a. Describa la declaración de las señales que se necesitan para almacenar los resultados parciales y final
   - 0'2 b. Describa el proceso secuencial que registra el dato de entrada
   - 0'4 c. Describa las operaciones intermedias mediante asignaciones concurrentes o un proceso combinacional
   - 0'2 d. Describa el proceso secuencial que registra el resultado final.
5. Si cada multiplicador tarda 40 ns en producir un resultado y cada sumador 20 ns. Indique el retardo máximo del circuito y la frecuencia necesaria para la señal de reloj. ¿Cómo modificaría la arquitectura del apartado anterior para triplicar la frecuencia de reloj?
   - 0'5
   - 0'5

Coeficientes 0'2
Expresión 0'4
+ Registros (-0'2)
+ Reg's = 1 (-0'2)
No potencias (-0'2)
No truncar (-0'2)

Solución. Ejercicio 2. Parcial VHDL. Curso 22123. CIM-GITT-GITI

$$e^x \simeq 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} \quad (x \in [-1,1])$$



$z = 128 \cdot x$; z: 8 bits en CA2 $\Rightarrow$ $[-128, 127]$ (despreciamos los decimales que haya a partir del bit LSB)

$$Y(z) = 1 + \frac{z}{128} + \frac{z^2}{2(128)^2} + \frac{z^3}{6\cdot(128)^3} = 1 + \frac{z}{2^7} + \frac{z^2}{2^{15}} + \frac{z^3}{3\cdot 2^{22}}$$

$$Y(z) \in [e^{-1}, e^{127/128}] \overset{(*)}{=} [0'3678794412..., 2'697127991...]$$

Para que sea entero, lo multiplicamos por $2^7 = 128$

(*) RESULTADO IDEAL

$$\Rightarrow Y(z)\cdot 128 = W(z) = (47,088\ldots ; 345'2323828492...]$$

estos decimales se despreciarán

$$W(z) = 2^7 \cdot Y^z = 1 + z + \frac{z^2}{2^8} + \frac{z^3}{3\cdot 2^{15}}$$

Se necesitan 9 bits para representar este número. Si asumimos que la salida es CA2 $\Rightarrow$ 10 bits son necesarios para Data Out

El resultado será un número entre 47 y 345

Para saber el valor real hay que dividir por 128 $\Rightarrow$

$(0'3671875 ; 2'6953125)$ $\rightarrow$ ERROR frente al resultado ideal (*)

$\rightarrow$ 0'06731 %

$\rightarrow$ 0'188 % | ERROR Relativo

Aunque, como las entradas Yb se están truncando, los resultados ~~sobre~~ para datos entre $(-1,0)$ y $(0,1)$ tendrán más error. Y, además, el error será mayor porque estamos truncando la serie de Taylor en el tercer término.

Respuestas al enunciado

① Rangos de las variables

$z \in [-128, 127]$ (8 bits en CA2)

a) ¿X? $\left(X = \frac{z}{128}\right) \Rightarrow X \in \left[-1, \frac{127}{128}\right]$

b) $Y(z)$ $\quad z = -128 \quad Y(-128) = 1 + \frac{z^7}{2^7} + \frac{z^{14}}{2 \cdot 2^{14}} + \frac{z^{21}}{6 \cdot 2^{21}}$

$Y(-128) = 1 + 1 + 0'5 + 0'16\hat{6} = 0'8$

$W(-128) = 128 \cdot Y(-128) = 43$ porque despreció los decimales (redondeo)

$z = \frac{127}{128} \quad Y\left(+\frac{127}{128}\right) = 1 + \frac{127}{128} + \frac{(127)^2}{2(128)^2} + \frac{(127)^3}{6(128)^3}$

$Y(+127) = 1 + 0'992187\overline{5} + 0'492218\overline{0176} + 0'162740\overline{8542}$

$Y(+127) = 2'647196\overline{3724}$

$z(127) = 338$ porque desprecio los decimales (redondeo)

Se puede observar que el error es mayor que el obtenido anteriormente.

② Si W(z) es igual de 10 bits, puede representar números enteros entre -512 y 511 ¿luego sí se puede representar el resultado anterior

Si z es igual de 8 bits puede representar números enteros entre -128 y 127 luego ~~no~~ vale.

2) Expresa W(z) en digital. Coeficientes 8 bits en CA2

$$W(z) = 2^7 + z + \frac{z^2}{128} + \frac{z^3}{2^{15}} \cdot \frac{1}{3} = 2^7 + a_0 z + a_1 \frac{z^2}{128} + a_2 \frac{z^3}{2^{15}}$$

$128 = 2^8$ $\quad$ $128 = 2^8$

Coeficientes a_0, a_1, a_2

Los productos parciales se están dividiendo por potencias de 2 (equivalente a despreciar bits de la derecha)

$a_0 = 1$; este subproducto se suma tal cual al resultado

$a_1 = 1$; además, este subproducto se trunca quitando los 7 LSB

$a_2 = \frac{1}{3}$; además, este subproducto se trunca quitando los 15 LSB

CTE $= 2^7$; si se representa en CA2 se necesitan 9 bits

$$\boxed{0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0} = 128 = 2^7$$

$a_0 = 1$ } en CA2 8 bits hay que multiplicar por 26
$a_1 = 1$ } $\boxed{0\ 1\ 0\ 0\ 0\ 0\ 0\ 0}$.
$a_2 = 0'333 \rightarrow$ Multiplicamos por $2^6 = 64$ — Pq si no se desborda

$$a_2 = 2^6 \cdot \frac{1}{3} = 21'3 \simeq 21$$

$$\boxed{a_2 = 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1}$$

$\frac{2^6}{3} = \frac{64}{3} \simeq 21$

$$W(z) = 2^7 + \frac{2^6}{2^6}\left(z + \frac{z^2}{128} + \frac{z^3}{2^{15}} \cdot \frac{1}{3}\right)$$

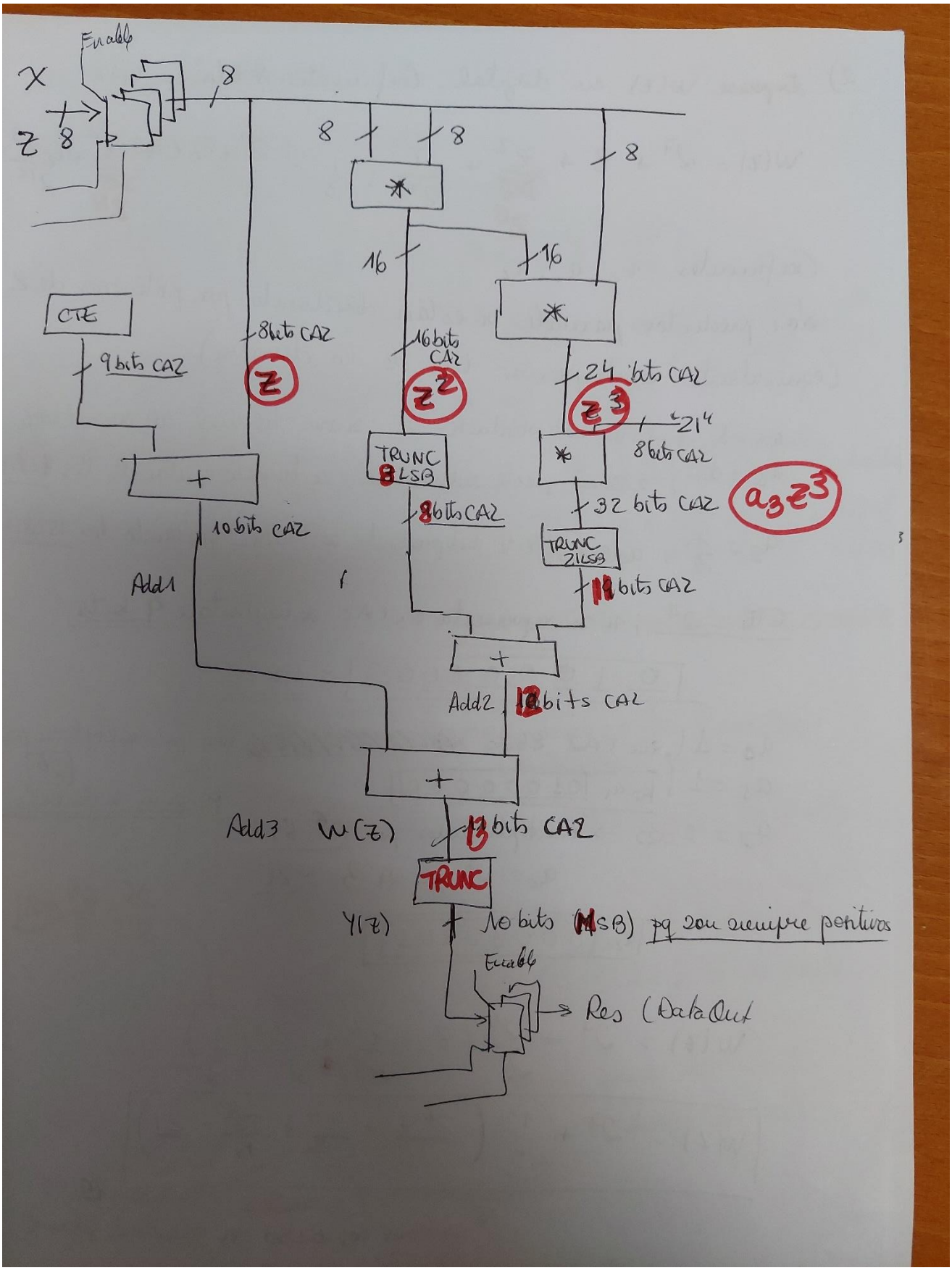
$128 = 2^8$

$$\boxed{W(z) = 2^7 + \frac{1}{2^6}\left(2^6 \cdot z + \frac{z^2}{2^7} + \frac{z^3}{2^{15}} \cdot 21\right)}$$

quitamos 7 LSB $\quad$ quitamos 15 LSB

quitamos los 6 LSB del resultado parcial

$$W(z) = 2^7 + z + \frac{1}{2^6}\left(\frac{z^2}{2^7}\right) + \frac{1}{2^{21}} \cdot 21 \cdot z^3$$


Enable
X
Z
8
8
8
8
8
*
16
16
*
CTE
9 bits CA2
8 bits CA2
16 bits CA2
24 bits CA2
Z
Z²
Z³
TRUNC 8 LSB
21
8 bits CA2
*
+
10 bits CA2
8 bits CA2
32 bits CA2
a3z3
TRUNC 21 LSB
Add1
bits CA2
+
Add2
bits CA2
+
Add3
W(z)
bits CA2
TRUNC
Y(z)
bits (MSB) pq sou sempre positivo
Enable
Res (DataOut

(4)

```
signal RegDataIn : signed (7 downto 0); -- x
signal RegDI2 : signed (15 downto 0);
signal RegDI3 : signed (23 downto 0);
signal Add1 : signed (9 downto 0);

constant C1 : signed (8 downto 0) := "010000000";
constant C3 : signed (7 downto 0) := "00010101";
signal Add2 : signed (9 downto 0);
signal AuxDI3 : signed (31 downto 0);
signal Add3 : signed (10 downto 0);

signal Aux Y : signed (9 downto 0);
begin
process (Clk, Reset)
begin
   if Reset = '1' then
      RegDataIn <= (others => '0');
   elsif Clk'event and Clk = '1' then
      if Enable = '1' then
         RegDataIn <= DataIn;
      end if;
   end if;
end process;
Add1 <= resize (C1, 10) + resize (RegDataIn, 10);
Add2 <= resize ( (RegDataIn * RegDataIn) (8 downto 0), 9) +
        resize (
RegDI2 <= (RegDataIn * RegDataIn ), (8 downto 0)
RegDI3 <= RegDI2 * RegDataIn;
AuxDI3 <= RegDI3 * C3;
Add2 <= resize ( RegDI2 (8 downto 0), 9) + resize (RegDI3 (8 downto 0), 9);
Add3 <= resize (Add1, 11) + resize (Add2, 11);
```

```
proces (Clk, Reset)
begin
  if Reset = '1' then
    Aux Y <= (others => '0');
  elsif Clk'event and Clk = '1' then
    if Enable = '1' then
      Aux Y <= Add3 (10 downto 1);
    end if;
  end if;
end proces;
Data Out <= Aux Y;
```

pq la salida es CA2

(5) Retardo máximo

Camino + largo $x \to x^2 \to x^3 \to a_3 x^3 \to Add2 \to Add3$

⊛ ⊛ ⊛ ⊕ ⊕

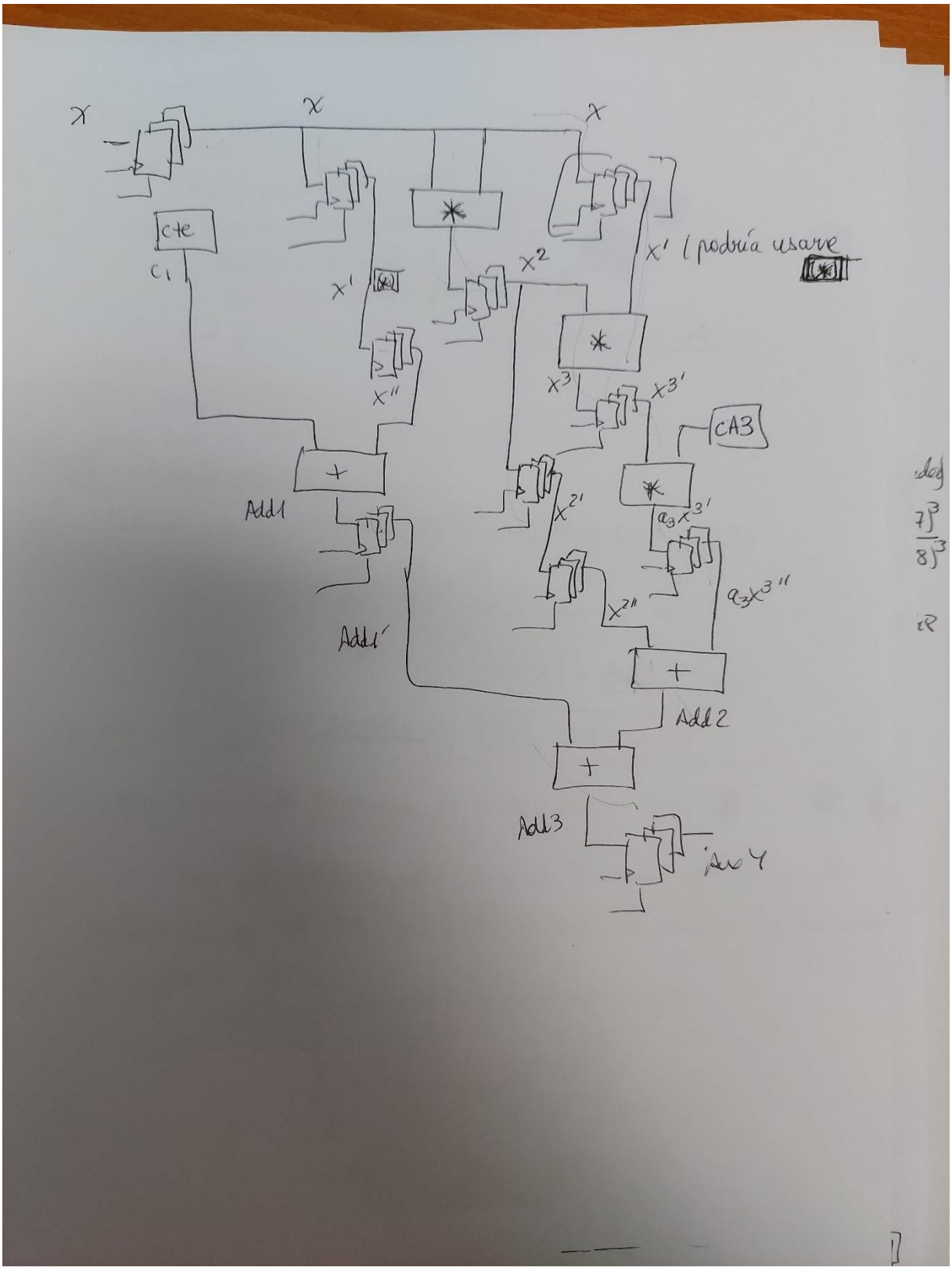
40 40 40 20 20

160ns + t_{setup} + t_{reloj}

Mayor cota $T = \frac{1}{160} = 6'25\ MHz \Rightarrow$ 6 MHz $\Rightarrow T = 166'6\ ns$

Para triplicar la frecuencia → 18 MHz ⇒ T = 55'5 ns

Hay que poner registros después de cada operación de multiplicar ✱. No hace falta después de sumar pq ya está el registro de salida


x
x
x
cte
c1
x'
x''
x²
x' (podría usarse
x³
x³'
CA3
Add1
x²'
a3x3'
x²''
a3x3''
Add1'
Add2
Add3
Add4