DISEÑO DE CIRCUITOS CONTROLADORES PARA LOS DATAPATH TIPO PROCESADOR UV 2007 Y UV2008 PARA LOS ALGORITMOS DE RAÍZ CUADRADA Y MULTIPLICACIÓN

Jenniffer Sidney Guerrero Prado

UNIVERSIDAD DEL VALLE

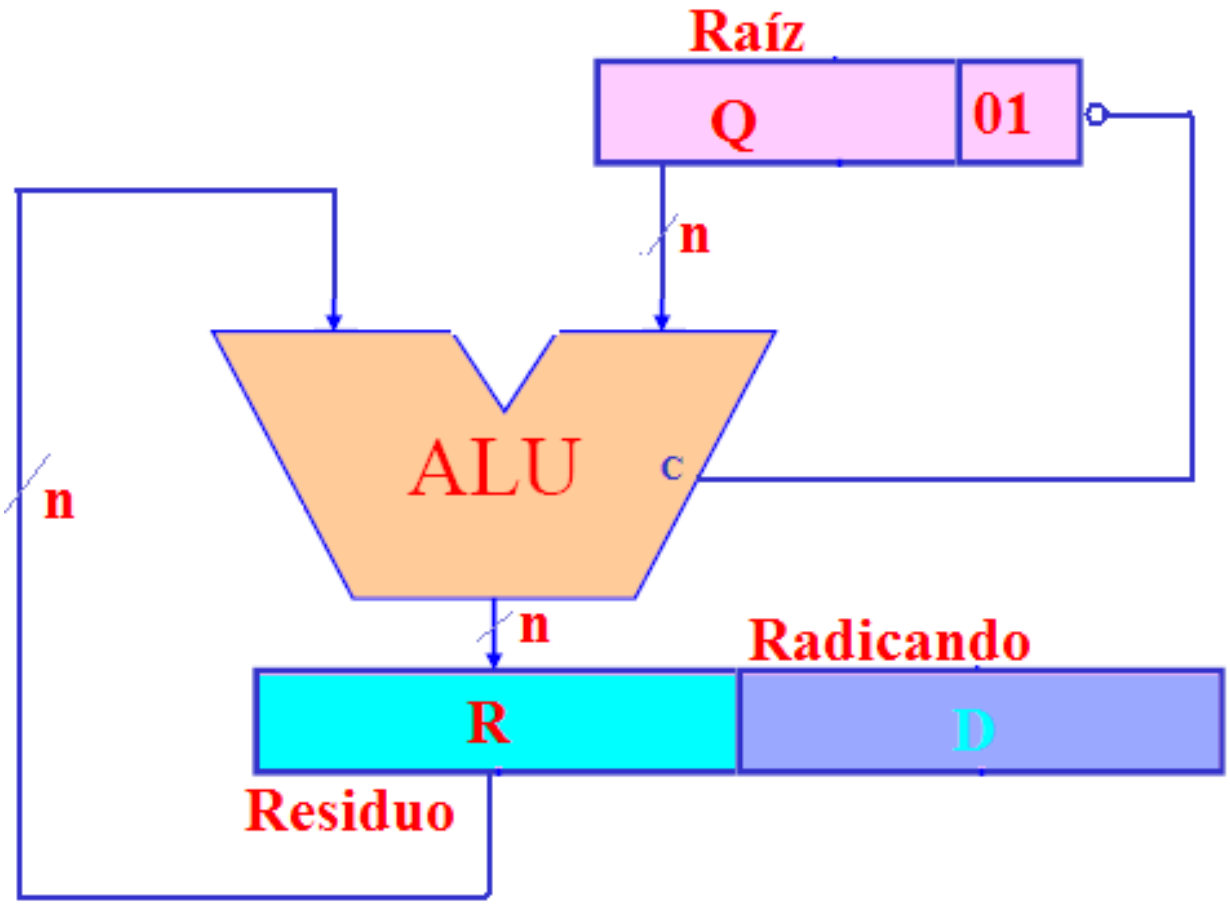
SISTEMAS DIGITALES II

INTRODUCCIÓN

A continuación se presentarán los datapath específicos para el cálculo de la raíz cuadrada y la multiplicación de números de 8 bits, así como los estados que describen el funcionamiento del circuito que controla dicho datapath.

También se implementará la raíz cuadrada en el procesador UV2008 y la multiplicación en el procesador UV2007, siguiendo el mismo algoritmo mostrado para los datapath específicos, pero con un cambio significativo en el número de estados, gracias a la estructura misma y a la diferencia que presentan estos procesadores con los datapath específicos mostrados inicialmente.

Datapath específico: raíz Cuadrada



Descripción de estados para el Datapath específico de la raíz Cuadrada

- E0: Cargar datos: radicando en D; limpiar R, Q.
- E1: Desplazar R,D dos veces a la izquierda con "0". Restar R-Q,01.
- E2:Guardar R=R-Q,01

SI C=0→E3

SI C=1→E4

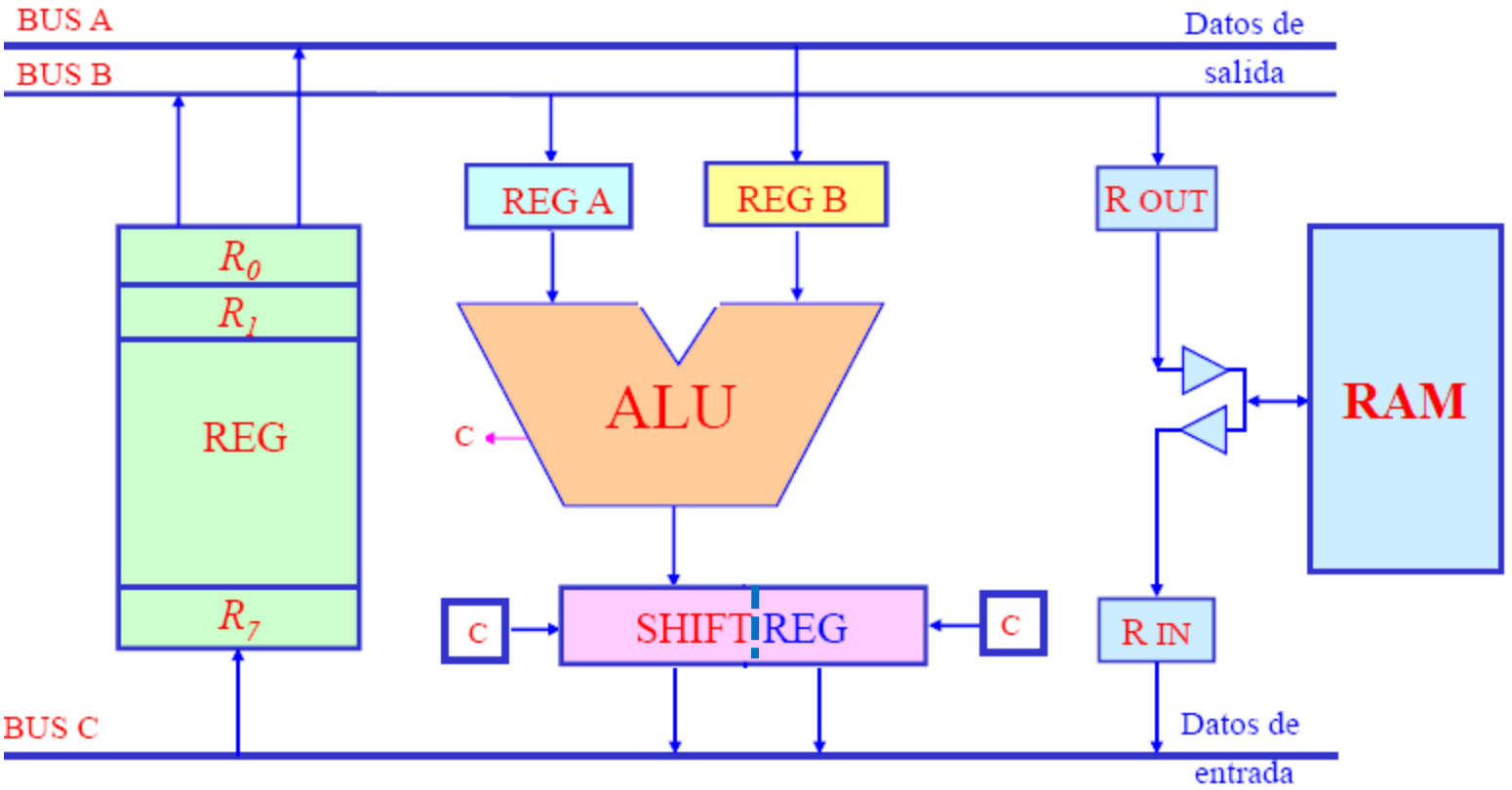
- E3:Desplazar Q a la izquierda con "1".
- E4: Guardar R=R+Q,01 (Restaurar)

  Desplazar Q a la izquierda con "0".

Repetir E1-E4 n/2 veces.

Problema propuesto

- Diseñar un circuito controlador para calcular la raíz cuadrada de un dato que se encuentra en el registro R4 y el resultado debe ser almacenado en MRAM[02].
- La operación debe realizarse en el procesador UV2008.



Procesador UV2008

Descripción de estados

- Cada estado del algoritmo descrito para el datapath específico de la raíz cuadrada descrito anteriormente, tendrá sus correspondientes estados para la FSM a diseñar para controlar el procesador UV2008.
- De esta manera, el algoritmo no se modificará, sino que se adaptará para funcionar en un datapath que ya no es específico, sino tipo procesador.
- A continuación se muestra dicha correspondencia de estados.

Correspondencia de estados:

- E0: Cargar datos: radicando en D; limpiar R, Q.
  - S0 – S1
- E1: Desplazar R,D dos veces a la izquierda con “0”.
  - S2 – S4
- E2: Restar R=R-Q,01
  - S5 – S12

SI C=0→S13

SI C=1→S15

- E3:Desplazar Q a la izquierda con “1”. Y
  E4:Desplazar Q a la izquierda con “0”.
  - S13 – S19
- NOTA: Gracias a la estructura del datapath, no fue necesario realizar la restauración propuesta en E4, ya que la resta **sólo** se guarda en el shift register si el Carry es “0”.

Descripción de estados

- S0:R0←0, R1←0, R2←0, RA←R4, Count =100.
- S1: RB←R2, RCa←RA, CA=1000.
- S2: RC←shR(0,RCa), CA=CA-1.

SI CA=0→S3

SI CA≠0→S2

- S3:RCa←RB
- S4:RC←shL(RC,0)
- S5:RC←shL(RC,0)
- S6:R2←RCa , RB←R0
- S7:R4←RCb, RA←R2, RCa←RB, CA=1000
- S8:RC←shR(0,RCa), CA=CA-1.

SI CA=0→S9

SI CA≠0→S8

- S9:RC←shL(RC,0)
- S10:RC←shL(RC,1)
- S11:R1←RCb
- S12:RB←R1. (ALU:restar, Activar reloj para FFC, para capturar el carry).

SI C=0→S13

SI C=1→S15

- S13:RCa ←RA-RB
- S14:R2←RCa
- S15:RB←R0
- S16:RCa ←RB, CA=1000.
- S17:RC←shR(0,RCa), CA=CA-1.

SI CA=0→S18

SI CA≠0→S17

- S18:RC←shL(RC,C)
- S19:R0←RCb, RA←R4, COUNT=COUNT-1

SI COUNT=0→S20

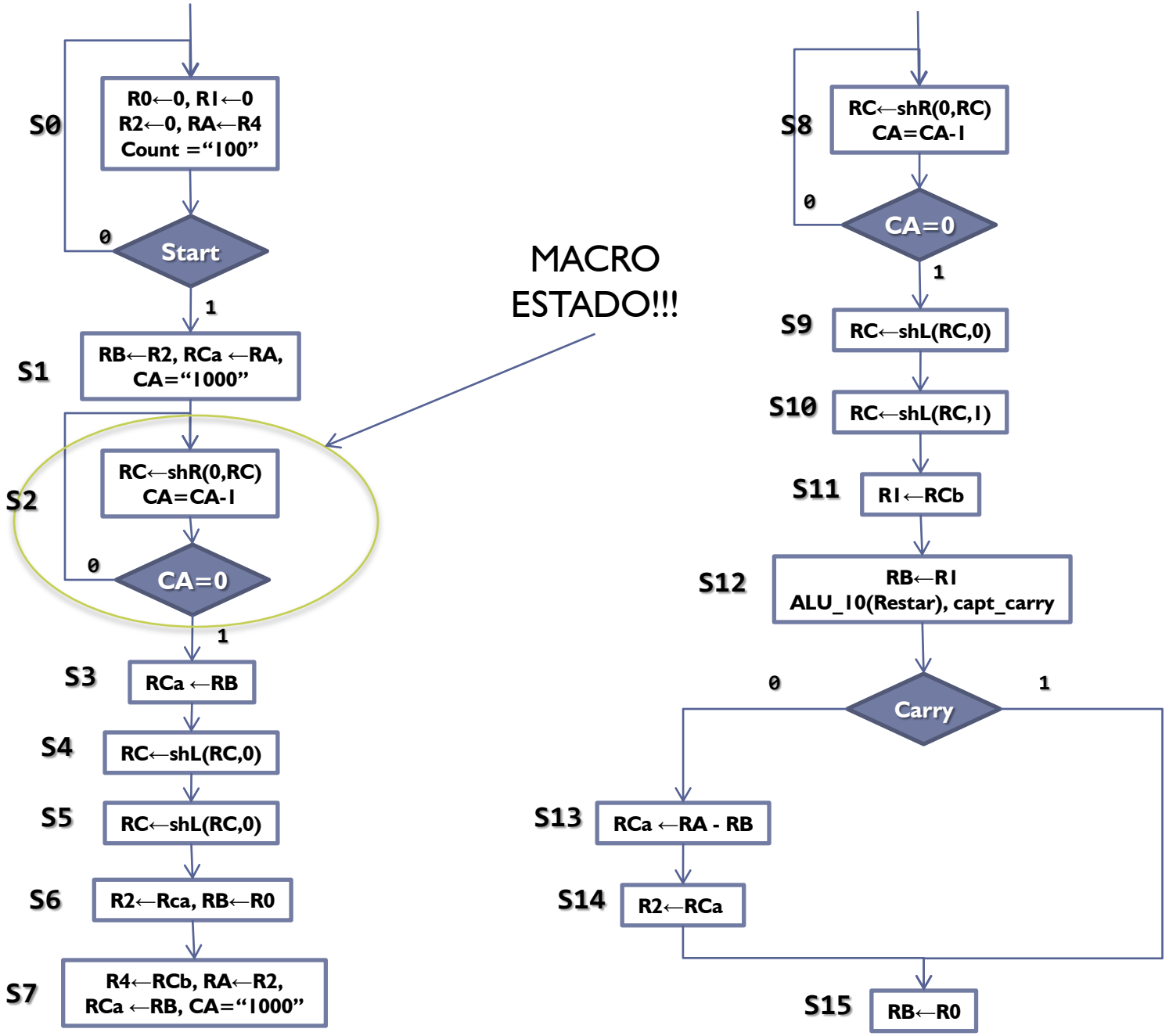
SI COUNT ≠0→S1

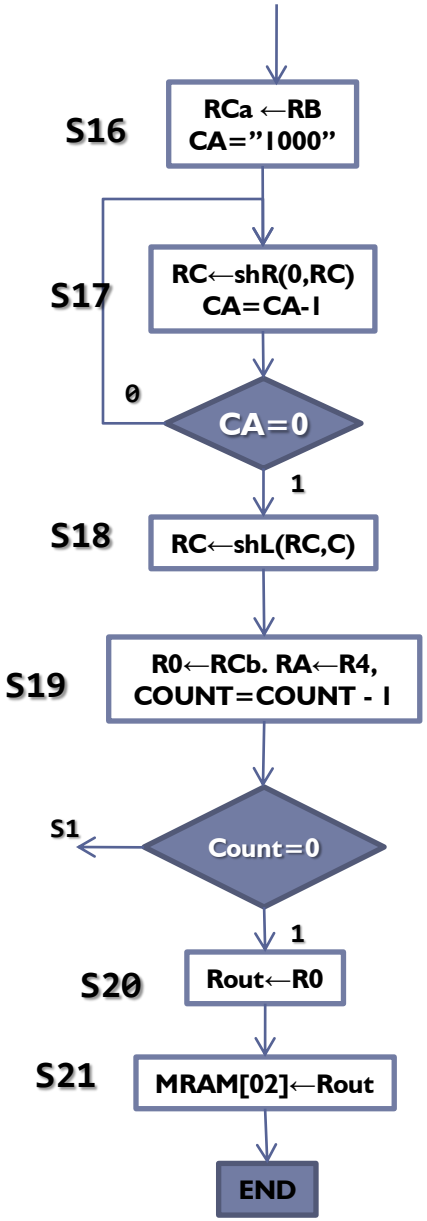
S20:Rout←R0

S21:MRAM[02]←Rout.

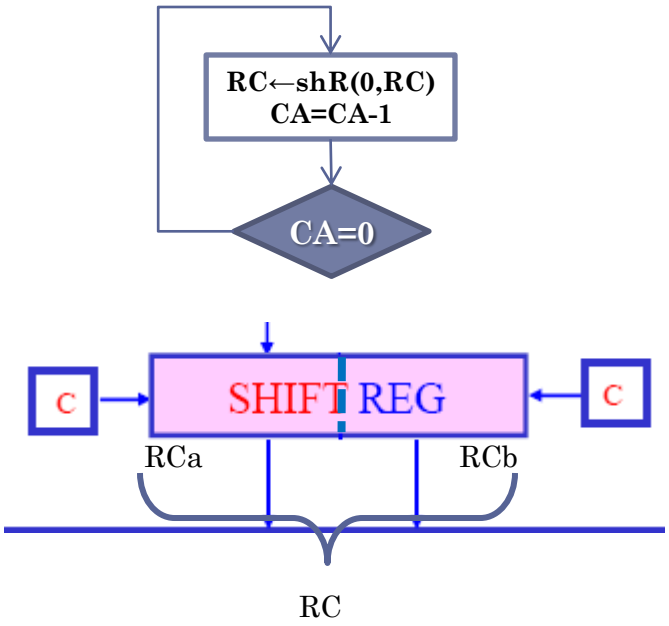
END

A continuación se muestra el ASM correspondiente a estos estados.


S0
R0←0, R1←0
R2←0, RA←R4
Count ="100"
Start
0
1
S1
RB←R2, RCa ←RA,
CA="1000"
S2
RC←shR(0,RC)
CA=CA-1
CA=0
0
1
S3
RCa ←RB
S4
RC←shL(RC,0)
S5
RC←shL(RC,0)
S6
R2←Rca, RB←R0
S7
R4←RCb, RA←R2,
RCa ←RB, CA="1000"
MACRO
ESTADO!!!
S8
RC←shR(0,RC)
CA=CA-1
CA=0
0
1
S9
RC←shL(RC,0)
S10
RC←shL(RC,1)
S11
R1←RCb
S12
RB←R1
ALU_10(Restar), capt_carry
Carry
0
1
S13
RCa ←RA - RB
S14
R2←RCa
S15
RB←R0



En que consiste el macro estado utilizado tres veces en el anterior ASM?



El shift register presentado en el UV2008, no presenta8 bits como los demás registros, sino 16, por lo que su funcionamiento se ha dividido en dos partes: RCa y RCb. Los bits que entrega la ALU se capturan con RCa, por lo cual, si se requiere rotar a la izquierda un dato que esta en RCa, primero debe pasarse este dato bit a bit a RCb, para poderse rotar a la izquierda. Este paso de datos de RCa a RCb, es el descrito mediante el macro estado presentado en este diseño.

Señales de control utilizadas

- EN_RA
- EN_RB
- CLR_R0
- EN_R0
- OE_R0
- CLR_R1
- EN_R1
- OE_R1
- CLR_R2
- EN_R2
- OE_R2
- EN_R4
- OE_R4
- EN_ROUT
- OE_ROUT
- EN_RCa
- OE_RCa
- OE_RCb
- W_RAM
- CLK_FFC
- ADR_RAM
- CLK_COUNTER
- CLK_CA
- LOAD_COUNTER
- LOAD_CA
- DECO_A (3 BITS)
- DECO_B (3 BITS)
- DECO_C (3 BITS)
- OP_ALU:
  - 00: Transferir A
  - 01: Transferir B
  - 10: restar A – B
- OP_SHIF REG
  - 000: No desplazar (transferir)
  - 001: Despl. Derecha con “0”
  - 010: Despl. Izquierda con carry
  - 011: Despl. Izquierda con “0”
  - 100: Despl. Izquierda con “1”

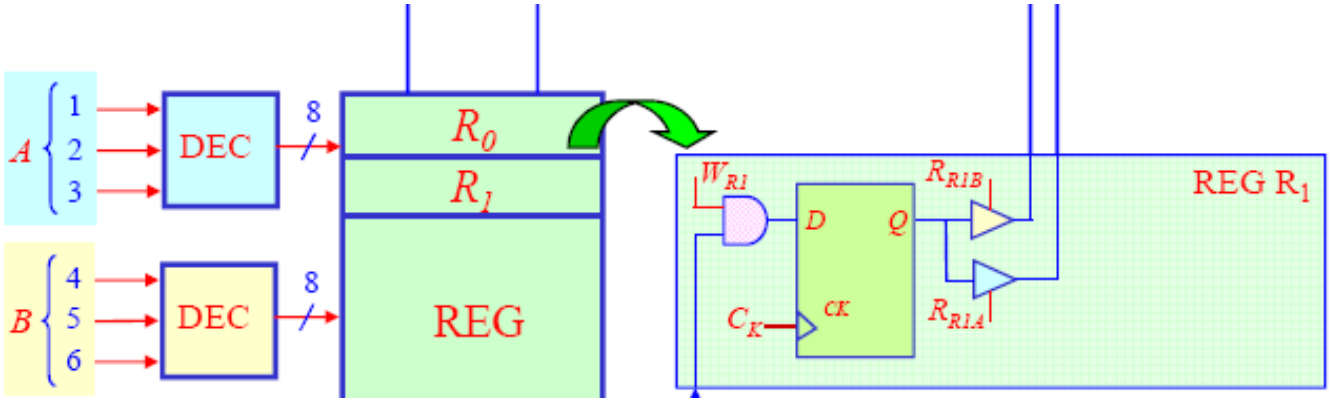
Señales De Control

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
EN_RA	1	0	0	0	0	0	0	1	0	0	0
EN_RB	X	1	0	0	0	0	1	0	0	0	0
CLR_R0	1	0	0	0	0	0	0	0	0	0	0
EN_R0	0	0	0	0	0	0	0	0	0	0	0
OE_R0	0	0	0	0	0	0	1	0	0	0	0
CLR_R1	1	0	0	0	0	0	0	0	0	0	0
EN_R1	0	0	0	0	0	0	0	0	0	0	0
OE_R1	0	0	0	0	0	0	0	0	0	0	0
CLR_R2	1	0	0	0	0	0	0	0	0	0	0
EN_R2	0	0	0	0	0	0	1	0	0	0	0
OE_R2	0	1	0	0	0	0	0	1	0	0	0
EN_R4	0	0	0	0	0	0	0	1	0	0	0
OE_R4	1	0	0	0	0	0	0	0	0	0	0
EN_RCA	0	1	0	1	0	0	0	1	0	0	0
OE_RCA	0	0	0	0	0	0	1	0	0	0	0
OE_RCB	0	0	0	0	0	0	0	1	0	0	0
W_RAM	0	0	0	0	0	0	0	0	0	0	0
CLK_FFC	0	0	0	0	0	0	0	0	0	0	0
Clk_Count	1	0	0	0	0	0	0	0	0	0	0
CLK_CA	0	1	1	0	0	0	0	1	1	1	1
LOAD_COUNT	1	0	0	0	0	0	0	0	0	0	0
LOAD_CA	0	1	0	0	0	0	0	1	0	0	0
EN_Rout	0	X	0	0	0	0	0	0	0	0	0
OE_Rout	0	X	0	X	0	0	0	0	0	0	0
DECO_A0	1	X	X	X	X	X	X	0	X	X	X
DECO_A1	0	X	X	X	X	X	X	1	X	X	X
DECO_A2	0	X	X	X	X	X	X	0	X	X	X
DECO_B0	X	0	X	X	X	X	0	X	X	X	X
DECO_B1	X	1	X	X	X	X	0	X	X	X	X
DECO_B2	X	0	X	X	X	X	0	X	X	X	X
DECO_C0	X	X	X	X	X	X	0	1	X	X	X
DECO_C1	X	X	X	X	X	X	1	0	X	X	X
DECO_C2	X	X	X	X	X	X	0	0	X	X	X
OP_ALU0	X	0	X	0	X	X	X	0	X	X	X
OP_ALU1	X	0	X	1	X	X	X	1	X	X	X
OP_SHIFT0	X	0	0	0	0	0	0	0	0	0	1
OP_SSHIFT_1	X	0	0	0	0	0	0	0	0	1	1
OP_SHIFT2	X	0	1	0	1	1	0	0	1	0	1

Señales De Control

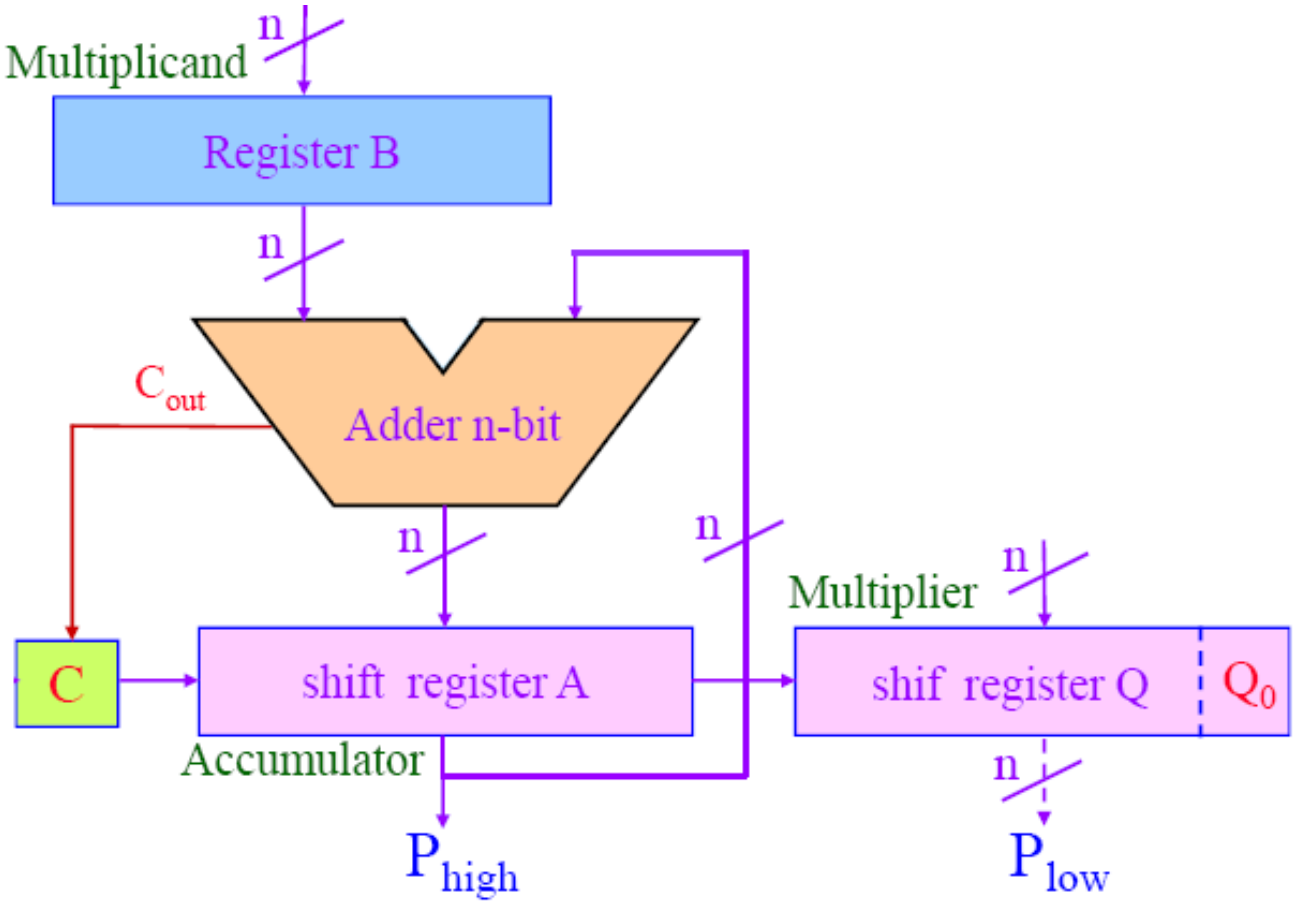
	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21
EN_RA	0	0	0	0	0	0	0	0	1	0	0
EN_RB	0	1	0	0	1	0	0	0	0	0	0
CLR_R0	0	0	0	0	0	0	0	0	0	0	0
EN_R0	0	0	0	0	0	0	0	0	1	0	0
OE_R0	0	0	0	0	1	0	0	0	0	1	0
CLR_R1	0	0	0	0	0	0	0	0	0	0	0
EN_R1	1	0	0	0	0	0	0	0	0	0	0
OE_R1	0	1	0	0	0	0	0	0	0	0	0
CLR_R2	0	0	0	0	0	0	0	0	0	0	0
EN_R2	0	0	0	1	0	0	0	0	0	0	0
OE_R2	0	0	0	0	0	0	0	0	0	0	0
EN_R4	0	0	0	0	0	0	0	0	0	0	0
OE_R4	0	0	0	0	0	0	0	0	1	0	0
EN_RCA	0	0	1	0	0	1	0	0	0	0	0
OE_RCA	0	0	0	1	0	0	0	0	0	0	0
OE_RCB	1	0	0	0	0	0	0	0	1	0	0
W_RAM	0	0	0	0	0	0	0	0	0	0	1
CLK_FFC	0	1	0	0	0	0	0	0	0	0	0
Clk_Count	0	0	0	0	0	0	0	0	1	0	0
CLK_CA	0	0	0	0	0	1	1	0	0	0	0
LOAD_COUNT	0	0	0	0	0	0	0	0	0	0	0
LOAD_CA	0	0	0	0	0	1	0	0	0	0	0
EN_Rout	0	0	0	0	0	0	0	0	0	1	0
OE_Rout	0	0	0	0	0	0	0	0	0	0	1
DECO_A0	X	X	X	X	X	X	X	X	1	X	X
DECO_A1	X	X	X	X	X	X	X	X	0	X	X
DECO_A2	X	X	X	X	X	X	X	X	0	X	X
DECO_B0	X	0	X	X	0	X	X	X	X	X	X
DECO_B1	X	0	X	X	0	X	X	X	X	X	X
DECO_B2	X	1	X	X	0	X	X	X	X	X	X
DECO_C0	0	X	X	0	X	X	X	X	0	X	X
DECO_C1	0	X	X	1	X	X	X	X	0	X	X
DECO_C2	1	X	X	0	X	X	X	X	0	X	X
OP_ALU0	X	1	1	X	X	0	X	X	X	X	X
OP_ALU1	X	0	0	X	X	1	X	X	X	X	X
OP_SHIFT0	0	0	0	0	0	0	0	0	0	X	X
OP_SSHIFT_1	0	0	0	0	0	0	0	1	0	X	X
OP_SHIFT2	0	0	0	0	0	0	1	0	0	X	X

- Las señales OE_RA, OE_RB y EN_RC, siempre estarán habilitadas, ya que gracias a su disposición en el datapath, no es necesario cambiar su nivel lógico en alguno de los estados. Vale la pena aclarar que es necesario utilizar decodificadores para escribir en los buses de datos, ya que los ocho registros del datapath tienen acceso a los dos buses, por lo cual se hace necesario especificar que registro está escribiendo y en que bus los hace:



- Los contadores CA y COUNT, son contadores descendentes módulo 8 y 4 respectivamente.
- La única dirección de la memoria RAM que se utilizará es la [02], por lo tanto no se incluye en el diseño como una salida de la FSM, ya que puede conectarse fijamente sin variar con ningún estado.

Datapath específico: multiplicación



Descripción de estados para el datapath específico de la multiplicación

- E0: cargar datos: Multiplicando en B,

  Multiplicador en Q, limpiar C, A.

SI Q0=1→E1

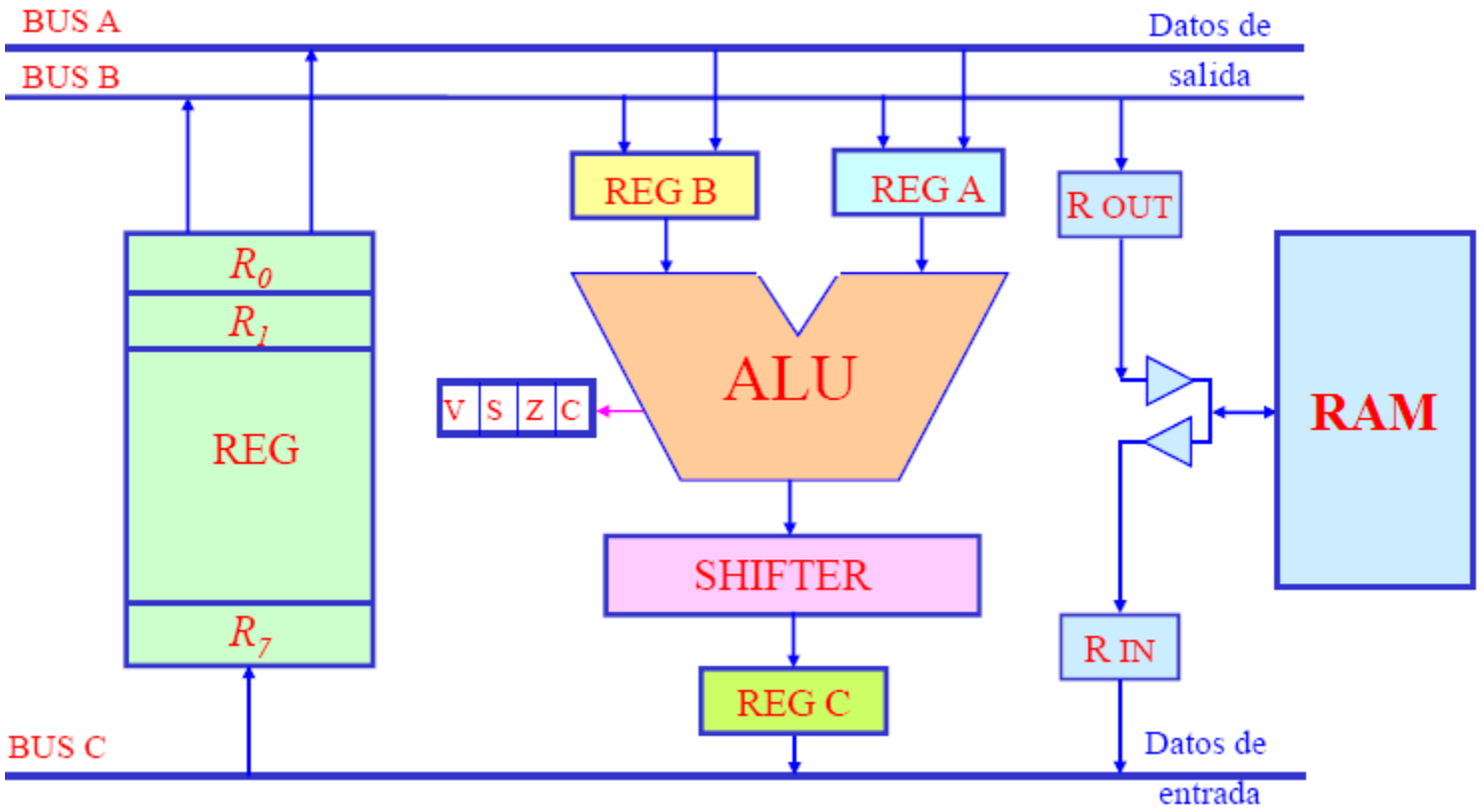
SI Q0=0→E2

- E1: Guardar C, A=A+B
- E2: Rotar a la derecha (C, A, Q) con “0”.

Repetir E1 y E2 N veces (N= Número de bits)

Problema propuesto

- Diseñar un circuito controlador para realizar la multiplicación de dos datos de ocho bits que se encuentran en la memoria RAM: multiplicando en MRAM[02] y multiplicador en MRAM[03]. Almacenar PH en MRAM [04] y PL en MRAM[05]
- La operación debe realizarse en el procesador UV2007.



Procesador UV2007

Descripción de estados

- Cada estado del algoritmo descrito para el datapath específico de la multiplicación descrito anteriormente, tendrá sus correspondientes estados para la FSM a diseñar para controlar el procesador UV2007.
- De esta manera, el algoritmo no se modificará, sino que se adaptará para funcionar en un datapath que ya no es específico, sino tipo procesador.
- A continuación se muestra dicha correspondencia de estados.

Correspondencia de estados:

- E0: cargar datos: Multiplicando en B, Multiplicador en Q, limpiar C, A.
  - S0 – S4

SI Q0=1→E1

SI Q0=0→E2

- E1: Guardar C, A=A+B
  - S5 – S6
- Rotar a la derecha (C, A, Q) con “0”.
  - S7 – S10

NOTA: Para identificar Q0, se hizo uso del enmascaramiento de dicho bit, que consiste en realizar el producto lógico del registro Q con un registro que tenga “1” solo en su último bit. Al realizar este producto lógico, se analizará el resultado de la bandera Z (Zero_detect), que se pondrá en “1” si Q0=0 o en “0” si Q0=1, gracias al producto lógico que se realizó.

Descripción de estados

- S0: Rin←MRAM[02], RB←”0”, COUNT=“1000”.
- S1:RC←RB+1, Rin←MRAM[03], R0←Rin
- S2:R3←RC
- S3: R1←Rin.
- S4:RA←R0, RB←R3 (ALU_prod lógico)

SI Z=0→S5

SIZ=1→S7

- S5:RA←R2, RB←R1, RC←RA+RB.
- S6: R2←RC, RC←shR(C,RA), S0=C
- S7:RA←R2, RB←”0”, RC←shR(C,A),S0=C
- S8: R2←RC, RB←R3 (ALU_prod lógico)
- S9:RB←R0, S0=/Z, RC=sh(/Z,R0).
- S10: R0←RC, COUNT=COUNT – 1

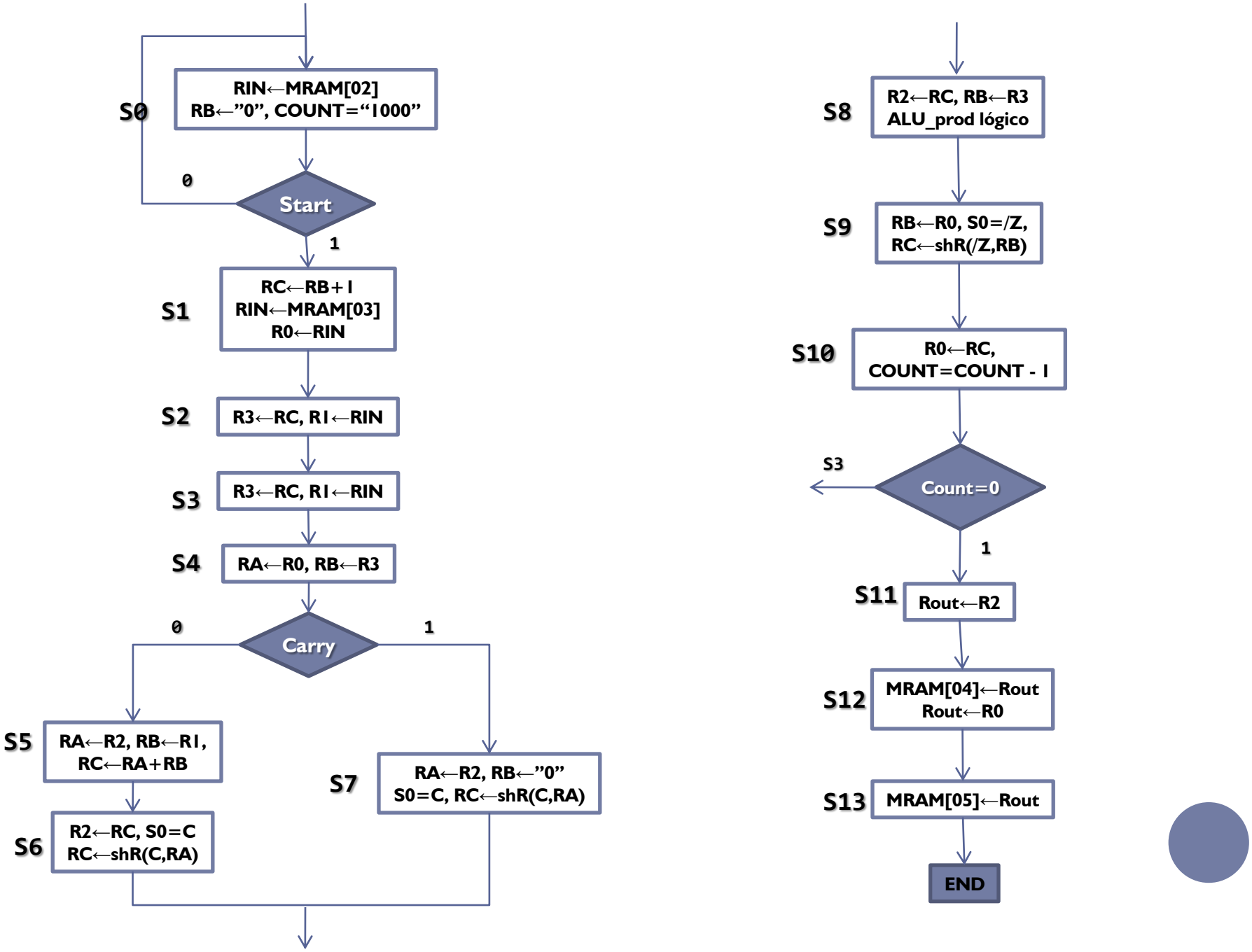
SI COUNT=0→S11

SI COUNT≠0→S4

- S11: Rout←R2
- S12: Rout←R0, MRAM[04]←Rout
- S13: MRAM[05]←Rout

END

A continuación se muestra el ASM correspondiente a dichos estados.


S0
RIN←MRAM[02]
RB←"0", COUNT="1000"
0
Start
1
S1
RC←RB+1
RIN←MRAM[03]
R0←RIN
S2
R3←RC, R1←RIN
S3
R3←RC, R1←RIN
S4
RA←R0, RB←R3
0
Carry
1
S5
RA←R2, RB←R1,
RC←RA+RB
S6
R2←RC, S0=C
RC←shR(C,RA)
S7
RA←R2, RB←"0"
S0=C, RC←shR(C,RA)
S8
R2←RC, RB←R3
ALU_prod lógico
S9
RB←R0, S0=/Z,
RC←shR(/Z,RB)
S10
R0←RC,
COUNT=COUNT - 1
S3
Count=0
1
S11
Rout←R2
S12
MRAM[04]←Rout
Rout←R0
S13
MRAM[05]←Rout
END

Señales de control utilizadas

- EN_RIN
- OE_RIN
- R_RAM
- W_RAM
- EN_RA
- EN_RB
- EN_R0
- OE_R0
- EN_R1
- OE_R1
- EN_R2
- OE_R2
- CLR_R3
- EN_R3
- OE_R3
- OE_RC
- EN_ROUT
- OE_ROUT
- CLK_COUNTER
- LOAD_COUNTER
- DECO_A (3 BITS)
- DECO_B (3 BITS)
- DECO_C (3 BITS)
- OP_ALU:
  - 000: Sumar A+B
  - 001: Transferir A
  - 010: Transferir B
  - 011: Incrementar B
  - 100: Producto lógico A and B
- OP_SHIF REG (S1 S0)
  - 00: Despl. Derecha con “0”
  - 01: Despl. Derecha con “1”
  - 1X: No desplazar (transferir)
- DECO_ADR_RAM
  - 00:MRAM[02]
  - 01:MRAM[03]
  - 10:MRAM[04]
  - 11:MRAM[05]

Señales De Control

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
EN_RIN	1	0	0	0	0	0	0	0	0	0	0	0	0	0
OE_RIN	0	1	0	1	0	0	0	0	0	0	0	0	0	0
R_RAM	1	0	0	0	0	0	0	0	0	0	0	0	0	0
W_RAM	0	0	0	0	0	0	0	0	0	0	0	0	1	1
EN_RA	0	0	0	0	1	1	0	1	0	0	0	0	0	0
EN_RB	0	0	0	0	1	1	0	0	1	1	0	0	0	0
CLR_RB	1	0	0	0	0	0	0	1	0	0	0	0	0	0
EN_R0	0	1	0	0	0	0	0	0	0	0	1	0	0	0
OE_R0	0	0	0	0	1	0	0	0	0	1	0	0	1	0
EN_R1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
OE_R1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
EN_R2	0	0	0	0	0	0	1	0	1	0	0	0	0	0
OE_R2	0	0	0	0	0	1	0	1	0	0	0	1	0	0
CLR_R3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EN_R3	0	0	1	0	0	0	0	0	0	0	0	0	0	0
OE_R3	0	0	0	0	1	0	0	0	1	0	0	0	0	0
OE_RC	0	0	1	0	0	0	1	0	1	0	1	0	0	0
EN_ROUT	0	0	0	0	0	0	0	0	0	0	0	1	1	0
OE_ROUT	0	0	0	0	0	0	0	0	0	0	0	0	1	1
CLK_COUNT	1	0	0	0	0	0	0	0	0	0	1	0	0	0
LOAD_COUNT	1	0	0	0	0	0	0	0	0	0	0	0	0	0
DECO_A0	X	X	0	0	0	0	X	0	0	X	X	X	X	X
DECO_A1	X	X	0	0	0	1	X	1	0	X	X	X	X	X
DECO_A2	X	X	0	0	0	0	X	0	0	X	X	X	X	X
DECO_B0	X	X	0	0	0	0	X	X	0	0	X	X	X	X
DECO_B1	X	X	0	0	1	0	X	X	1	0	X	X	X	X
DECO_B2	X	X	0	0	1	1	X	X	1	0	X	X	X	X
DECO_C0	X	0	0	0	X	X	0	0	0	X	0	X	X	X
DECO_C1	X	0	1	0	X	X	1	0	1	X	0	X	X	X
DECO_C2	X	0	1	1	X	X	0	0	0	X	0	X	X	X
OP_ALU0	X	0	X	X	X	0	0	0	1	0	X	X	X	X
OP_ALU1	X	1	X	X	X	0	0	0	0	1	X	X	X	X
OP_ALU2	X	1	X	X	X	0	1	1	0	0	X	X	X	X
OP_SHIFT1 (S1)	X	1	1	1	1	1	0	0	1	0	1	X	X	X
DEC_AD_RAM0	0	X	X	X	X	X	X	X	X	X	X	X	1	1
DEC_AD_RAM1	0	X	X	X	X	X	X	X	X	X	X	X	0	1
S0	*X*	*X*	*X*	*X*	*X*	*X*	*C*	*C*	*0*	*/Z*	*0*	*0*	*0*	*0*

- Las señales OE_RA, OE_RB y EN_RC, siempre estarán habilitadas, ya que gracias a su disposición en el datapath, no es necesario cambiar su nivel lógico en alguno de los estados. El bit menos significativo para la selección de la operación del shifter (S0) corresponde siempre a una de las banderas de la ALU, como se observa en los estado 5, 6 y 8 del ASM presentado anteriormente.
- La señal S0, es solo una señal **combinacional**, que toma el valor de una determinada entrada para completar el bit de operación del shifter; Por esta razón no se incluye en la tabla de señales de control
- También conviene aclarar que se utilizó un decodificador para direccionar la RAM, cuyo funcionamiento ya se aclaró en la lista de las señales de control.
- En este diseño, al igual que en el de la raíz cuadrada, se utilizaron decodificadores para la transferencia de datos entre registros a través de los buses del datapath.
- A continuación se muestra laFSM para la multiplicación implementada en AHDL

CODIGO AHDL…

```
SUBDESIGN FSM_mult(
CLK,RST,S,Z,COUNT : INPUT;
EN_RIN,OE_RIN,R_RAM,W_RAM,EN_RA,EN_RB,EN_R0,OE_R0,EN_R1,OE_R1: OUTPUT;
EN_R2,OE_R2, CLR_R3,EN_R3,OE_R3,OE_RC,EN_ROUT,OE_ROUT,CLK_COUNT: OUTPUT;
LOAD_COUNT,DEC_A0,DEC_A1,DEC_A2,DEC_B0,DEC_B1,DEC_B2,DEC_C0,DEC_C1,DEC_C2: OUTPUT;
OP_ALU0,OP_ALU1,OP_ALU2,S1,DEC_RAM0,DEC_RAM1: OUTPUT;
)
VARIABL
SS: MACHINE OF BITS(
EN_RIN,OE_RIN,R_RAM,W_RAM,EN_RA,EN_RB,EN_R0,OE_R0,EN_R1,OE_R1,
EN_R2,OE_R2, CLR_R3,EN_R3,OE_R3,OE_RC,EN_ROUT,OE_ROUT,CLK_COUNT,
LOAD_COUNT,DEC_A0,DEC_A1,DEC_A2,DEC_B0,DEC_B1,DEC_B2,DEC_C0,DEC_C1,DEC_C2,
OP_ALU0,OP_ALU1,OP_ALU2,S_1,DEC_RAM0,DEC_RAM1)
WITH STATES(
S1=B"10100010000000000011xxxxxxxxxxxxx00",
S2=B"01000001000000000000xxxxxx0000111xx",
S3=B"00000000000000101000000000011xxx1xx",
S4=B"01000000010000000000000000001xxx1xx",
S5=B"00001100100000010000000011xxxxxx1xx",
S6=B"00001100001010000000010001xxx0001xx",
S7=B"00000000001000010000xxxxxx0100010xx",
S8=B"00001010000010000000010xxx0000010xx“,
S9=B"00000100000100011000000010101001xx",
S10=B"00000100100000000000xxx000xxx0100xx",
```

```
S11=B"00000001000000010010xxxxxx000xxx1xx",
S12=B"00000000000100001000xxxxxxxxxxxxxx",
S13=B"00010000100000001100xxxxxxxxxxxx10",
S14=B"00010000000000000100xxxxxxxxxxxx11");
BEGIN
SS.CLK=CLK;
SS.RESET=RST;
TABLE
%estado entrada entrada entrada estado%
%actual actual  acutal actual siguiente%
SS, S, Z, COUNT => SS;
S1, 0, X, X      => S1;
S1, 1, X, X      => S2;
S2, X, X, X      => S3;
S3, X, X, X      => S4;
S4, X, 0, X      => S5;
S4, X, 1, X      => S7;
S5, X, X, X      => S6;
S6, X, X, X      => S8;
S7, X, X, X      => S8;
S8, X, X, X      => S9;
S9, X, X, X      => S10;
S10, X, X, 1     => S11;
S10, X, X, 0     => S4;
S11, X, X, X     => S12;
S12, X, X, X     => S13;
END TABLE;
END;
```