Implementación de insertion sort mediante un diseño de máquina algorítmica

Gabriel Rodríguez Canal

1 Introducción

En el presente documento se detalla el proceso de desarrollo de una máquina algorítmica que implementa el algoritmo de ordenación por inserción (*insertion sort*). La implementación se ha llevado a cabo mediante el software CAD Proteus, con el procedimiento de captura esquemática. En la sección 6 se muestran algunas capturas del proyecto. Se han omitido aquellas partes del mismo que, debido a sus grandes dimensiones, necesitarían de varias capturas para plasmarlas en el documento, por lo que perderían su propósito. Por tanto, se remite al lector al proyecto Proteus adjunto.

La máquina algorítmica se compone de la unidad de control y la unidad de proceso. La unidad de control implementa el autómata de control, que genera señales en función del estado de la máquina que son enviadas a la unidad de proceso para realizar los cálculos necesarios. La unidad de proceso de la máquina diseñada se detalla en la sección 3, mientras que la unidad de control se describe en la sección 4.

2 Algoritmo

A continuación se presenta el pseudocódigo del algoritmo que se ha implementado como máquina algorítmica:

Algorithm 1 Insertion Sort

```
1: for i = 1 to n do
2:     value = arr[i]
3:     j = i
4:     while j > 0 and arr[j − 1] > value do
5:         arr[j] = arr[j − 1]
6:         j = j − 1
7:     end while
8:     arr[j] = value
9: end for
```

Esta forma de describir el algoritmo ofrece una traducción fácil a un lenguaje de programación imperativo como C. Sin embargo, para la implementación como máquina algorítmica es más adecuada una reescritura en función de los recursos de cálculo disponibles:

NOTA: A lo largo del documento se omitirá la inicialización de los registros auxiliares R1-R5 por brevedad, sobreentendiéndose que estos estarán inicializados a 0.

```
N=12, I=1, T=1, V=1, J=0, VALUE=0
WHILE T
        VALUE=ARR[I]
        J=I
        WHILE V
                ARR[J] = ARR[J-1]
                J=J-1
                V=CM(0, J) AND CM(VALUE, ARR[J-1])
ARR[J]=VALUE
I=I+1
T=CM(I, N)
```

Para una mejor comprensión del algoritmo se remite al lector a la referencia [4], en la que podrá, además, encontrar un vídeo explicativo del mismo.

3 Unidad de proceso

La unidad de proceso se encarga de realizar los cálculos pertinentes en función de los valores de las señales Z recibidos desde la unidad de control. En la subsección se detalla el flujo de la unidad de proceso, mientras que en la

subsección 3.2 se dan los esquemas de cálculo que materializan el flujo del programa. En la subsección 3.4 se presenta el diseño final de la unidad de proceso, resultado de los pasos anteriores.

3.1 Planteamiento

ETAPA 1

Asig. Datos
N=12, I=1, T=1, V=1, J=0, VALUE=0
Salto
Etapa 2

ETAPA 2

Si T=1:

Asig. Datos
VALUE=ARR[I]
J=I
Salto
Etapa 3

Si T=0:

FIN

ETAPA 3

Si V=1:

Cálculos
F1=J-1
F2=CM(0,F1)
F3=ARR[F1]
F4=CM(VALUE,F3)
Asig. Datos
ARR[J]=F3
J=F1
V=F2 AND F3
Salto
Etapa 3

Si V=0:

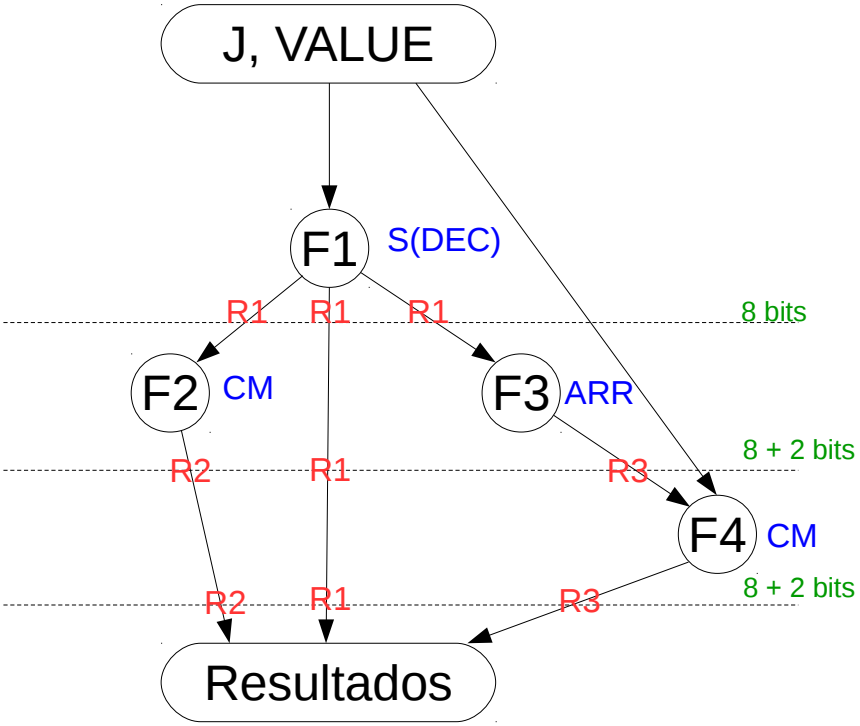
Salto

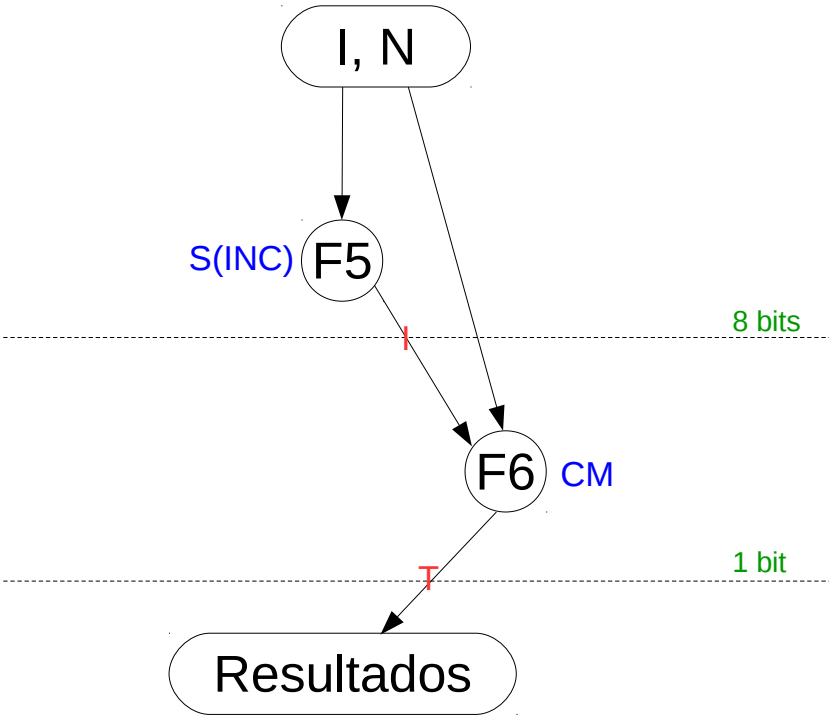
Etapa 4

ETAPA 4

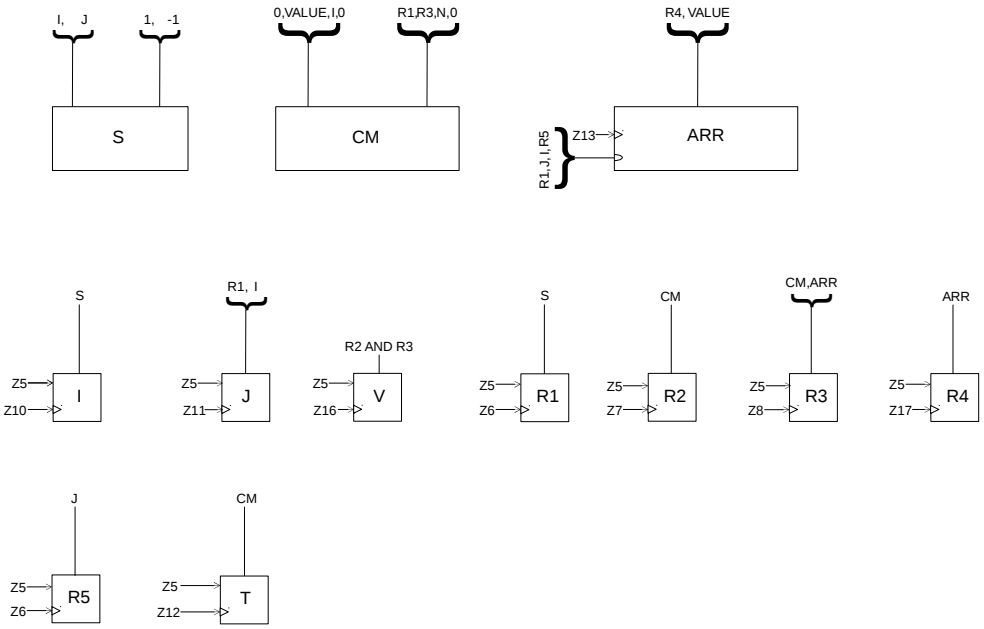
Cálculos
F5=I+1
F6=CM(F5,N)
Asig. Datos
ARR[J]=VALUE
I=F5
T=F6
Salto
Etapa 2

3.2 Esquemas de cálculo

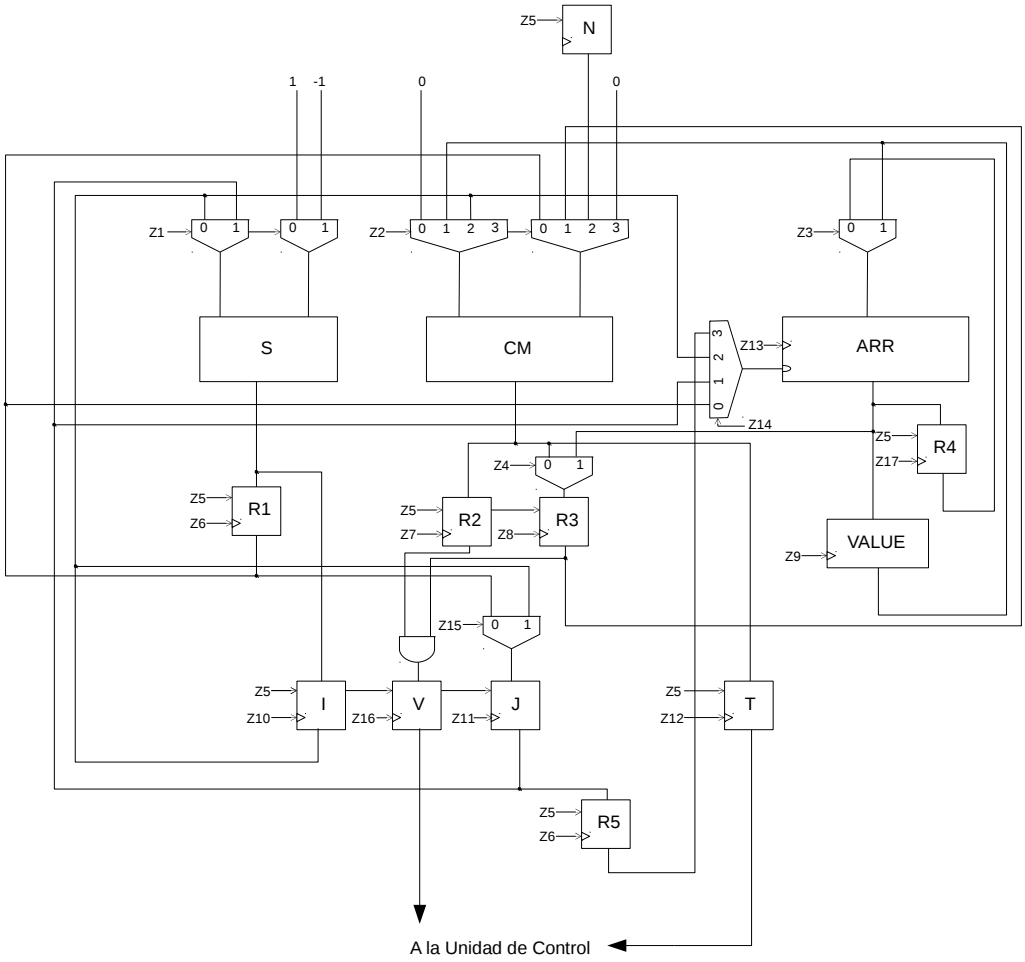



I, N
S(INC)
F5
8 bits
F6
CM
1 bit
Resultados

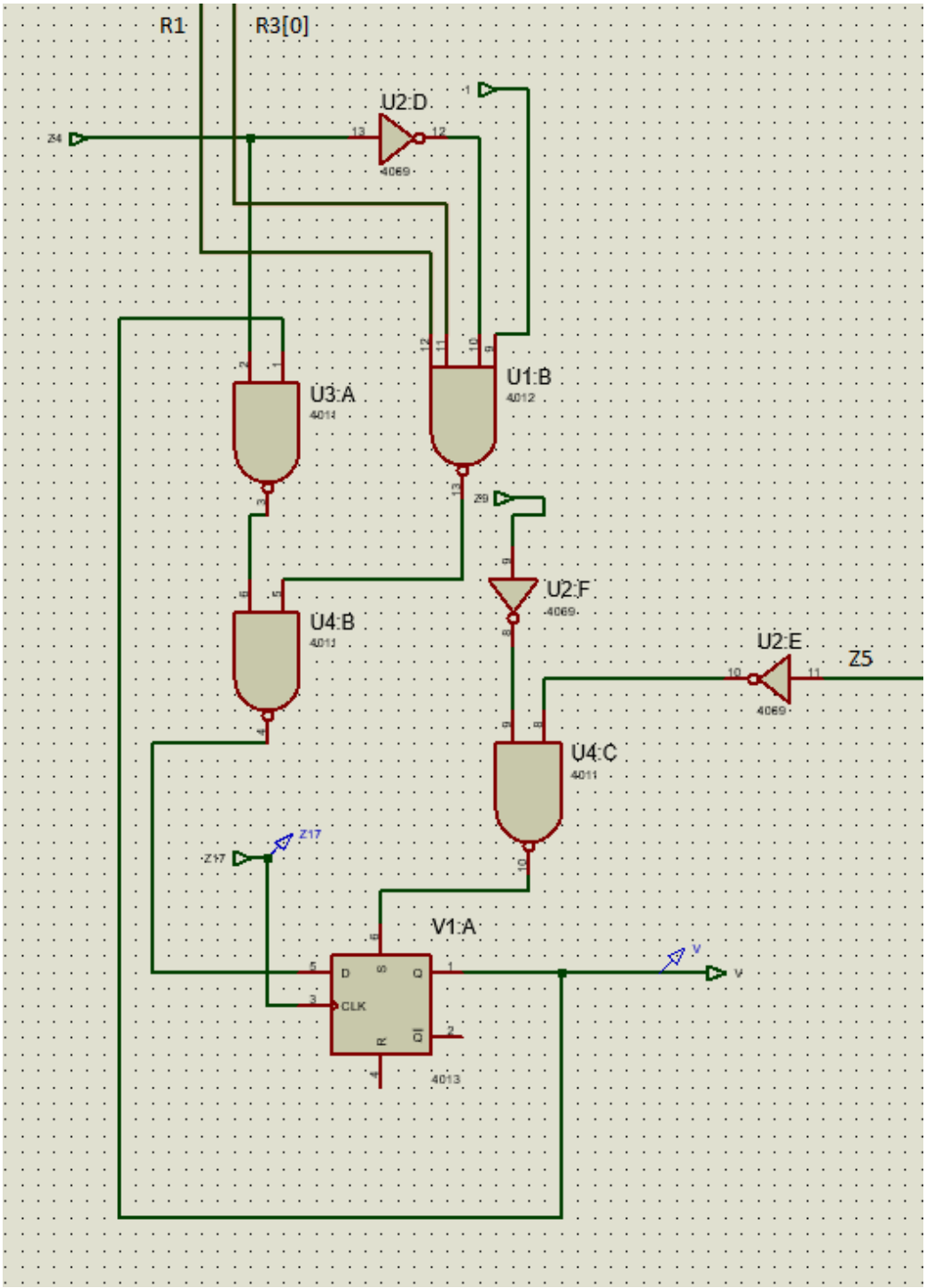
3.3 Asociación de recursos y registros



3.4 Diseño



Se ha hecho una simplificación en el circuito que calcula V. No se trata de R2 AND R3 directamente, sino de un circuito combinacional algo más elaborado (consultar fichero Proteus para mayor detalle):


R1
R3[0]
U2:D
4069
Z4
U3:A
4011
U1:B
4012
Z9
U2:F
4069
U2:E
4069
Z5
U4:B
4011
U4:C
4011
Z17
Z17
V1:A
D
CLK
Q
R
4013
V
V

j	ARR[j] (DEC)	ARR[j] (HEX)
0	23	17
1	8	8
2	1	1
3	9	9
4	6	6
5	45	2D
6	7	7
7	6	6
8	10	A
9	30	1E
10	62	3E
11	24	18

Table 1: Valores iniciales de los registros del recurso ARR

3.5 Recurso ARR

El recuros ARR es un array de 12 registros (N) de 8 bits. Permite la inicialización del mismo mediante su entrada de reloj. También existe una entrada de selección de registro. De este modo, la salida del recurso será la del registro seleccionado mediante esta entrada. Los valores iniciales de los registros son:

4 Unidad de control

La unidad de control se encarga de generar las señales Z que indican a la unidad de proceso qué cálculo ha de efectuar en la etapa actual. Implementa el autómata de control (subsección 4.2), que depende del estado de la máquina (codificado mediante y_2, y_1 e y_0), de la señal de INICIO y de los bits T y V, que proceden de la unidad de proceso. Pese a que se presente el esquema de la unidad de control como un PLD programable, en realidad se ha implementado en Proteus mediante puertas NAND (ver anexo y proyecto adjunto). El programa de control (subsección 4.1) y el autómata de control son dos representaciones equivalentes del flujo seguido por la máquina de estados. En la subsección 4.3 se detallan todas las señales generadas por la unidad de control.

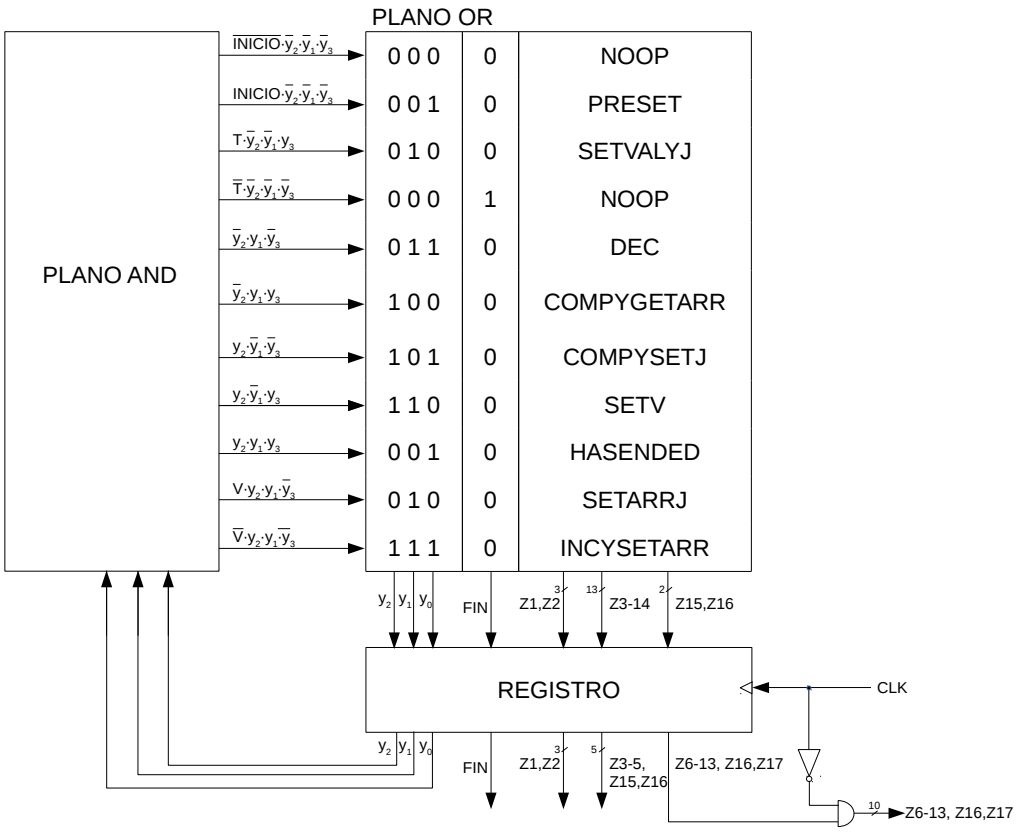


Figure 1: Esquema de la unidad de control

4.1 Programa de control

1:		, [N=12, I=1, T=1, V=1, J=0, VALUE=0]	, 2;
2:	T	, [VALUE=ARR[I], J=I]	, 3;
	$\bar{T}$	, FIN	
3:		, [R1=S(J,-1), R5=J]	, 4;
4:		, [R2=CM(0,J), R3=ARR[R1]]	, 5;
5:		, R3=CM(VALUE,R3)	, 6;
6:		, [V=R2 AND R3, R4=ARR[R1]]	, 7;
7:	V	, [ARR[R5]=R4, J=R1]	, 3;
	$\bar{V}$	, [I=S(I,1), ARR[J]=VALUE]	, 8;
8:		, T=CM(I,N)	, 2;

4.2 Diagrama de flujo (Autómata de control)

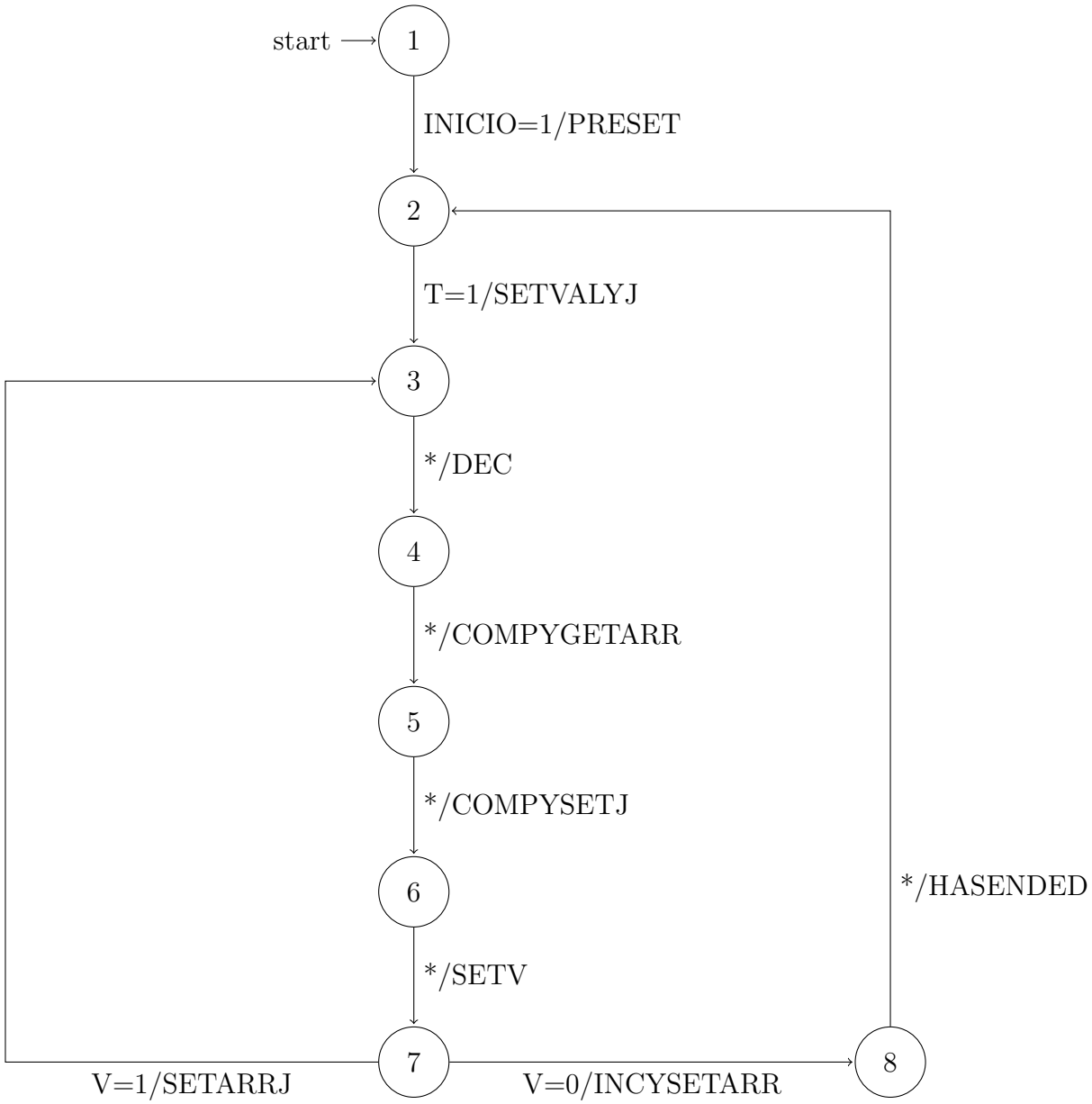


Figure 2: Autómata de control

4.3 Señales

Orden	Operaciones	Z1	Z2
PRESET	N=12, I=1, V=1, J=0, VALUE=0, INIC(ARR)	*	*
INCYSETARR	I=S(I,1), ARR[J]=VALUE	0	*
DEC	R1=S(J-1), R5=J	1	*
COMPYGETARR	R2=CM(0,J), R3=ARR[R1]	*	0
COMPYSETJ	R3=CM(VALUE,R3)	*	1
SETV	V=R2 AND R3, R4=ARR[R1]	*	*
HASENDED	T=CM(I,N)	*	2
SETVALYJ	VALUE=ARR[I], J=I, V=1	*	*
NOOP	No hace nada	*	*
SETARRJ	ARR[R5]=R4, J=R1	*	*

Orden	Z3	Z4	Z5	Z6	Z7	Z8	Z9	Z10	Z11	Z12	Z13
PRESET	*	*	1	0	0	0	0	0	0	0	0
INCYSETARR	1	*	0	0	0	0	0	1	0	0	1
DEC	*	*	0	1	0	0	0	0	0	0	0
COMPYGETARR	0	1	0	0	1	1	0	0	0	0	0
COMPYSETJ	0	0	0	0	0	1	0	0	0	0	0
SETV	0	*	0	0	0	0	0	0	0	0	0
HASENDED	*	*	0	0	0	0	0	0	0	1	0
SETVALYJ	*	*	0	0	0	0	1	0	1	0	0
NOOP	*	*	0	0	0	0	0	0	0	0	0
SETARRJ	0	*	0	0	0	0	0	0	1	0	1

Orden	Z14	Z15	Z16	Z17
PRESET	*	*	0	0
INCYSETARR	1	*	0	0
DEC	*	*	0	0
COMPYGETARR	0	*	0	0
COMPYSETJ	0	0	0	1
SETV	0	*	1	1
HASENDED	*	*	0	0
SETVALYJ	2	1	0	0
NOOP	*	*	0	0
SETARRJ	3	0	1	0

Table 2: Señales de la unidad de control.

5 Comprobación de funcionamiento del circuito

Para comprobar el funcionamiento del circuito se sugiere al lector hacer una simulación con el gráfico digital de la unidad de proceso. Entre otros, se monitorizan las sondas ARR0[0..7], ARR1[0..7], ..., ARR11[0..7], que comienzan la ejecución con los valores de la tabla 1 y terminar ordenados.

6 Anexo

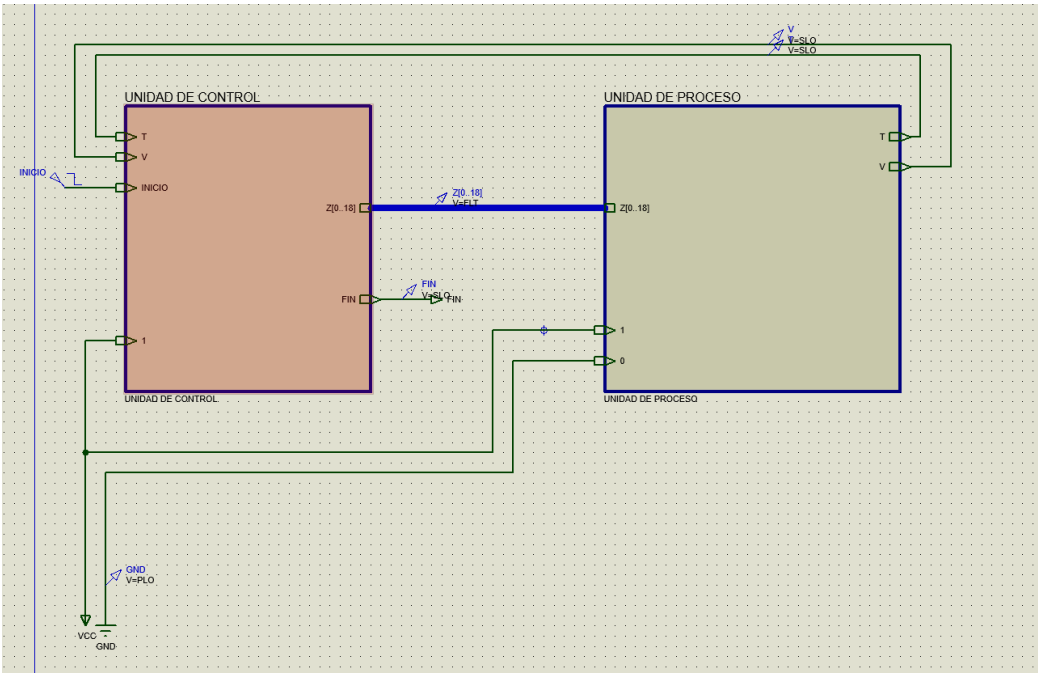


Figure 3: Hoja raíz

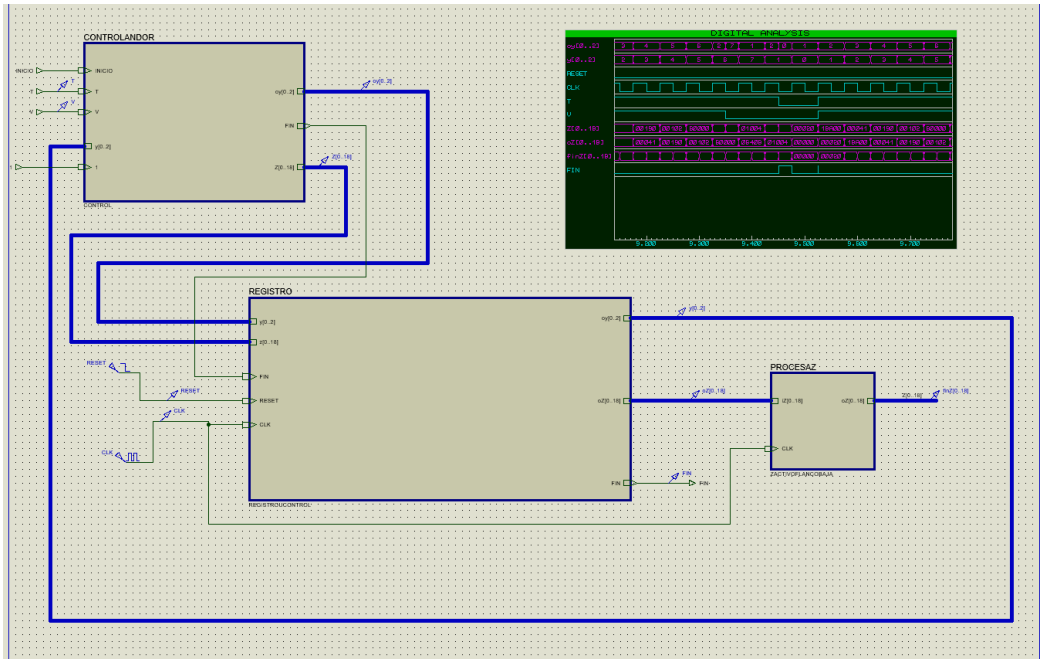


Figure 4: Unidad de control

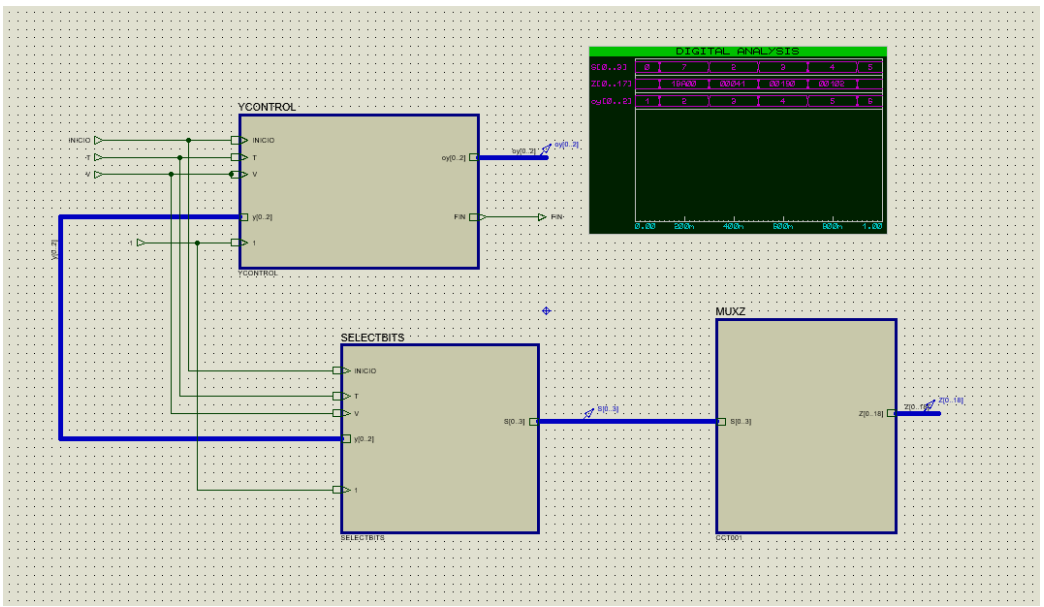


Figure 5: Detalle del bloque CONTROLANDOR

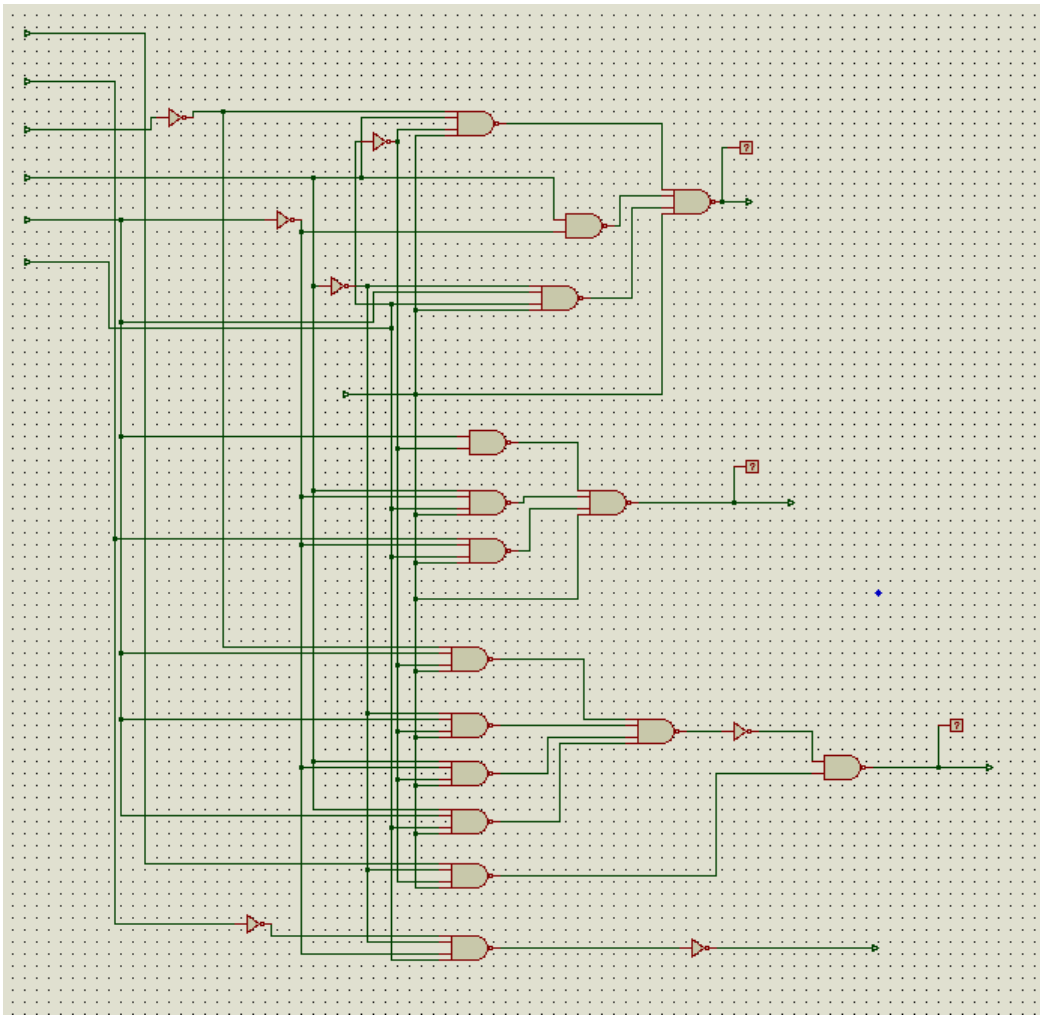


Figure 6: Detalle del bloque YCONTROL

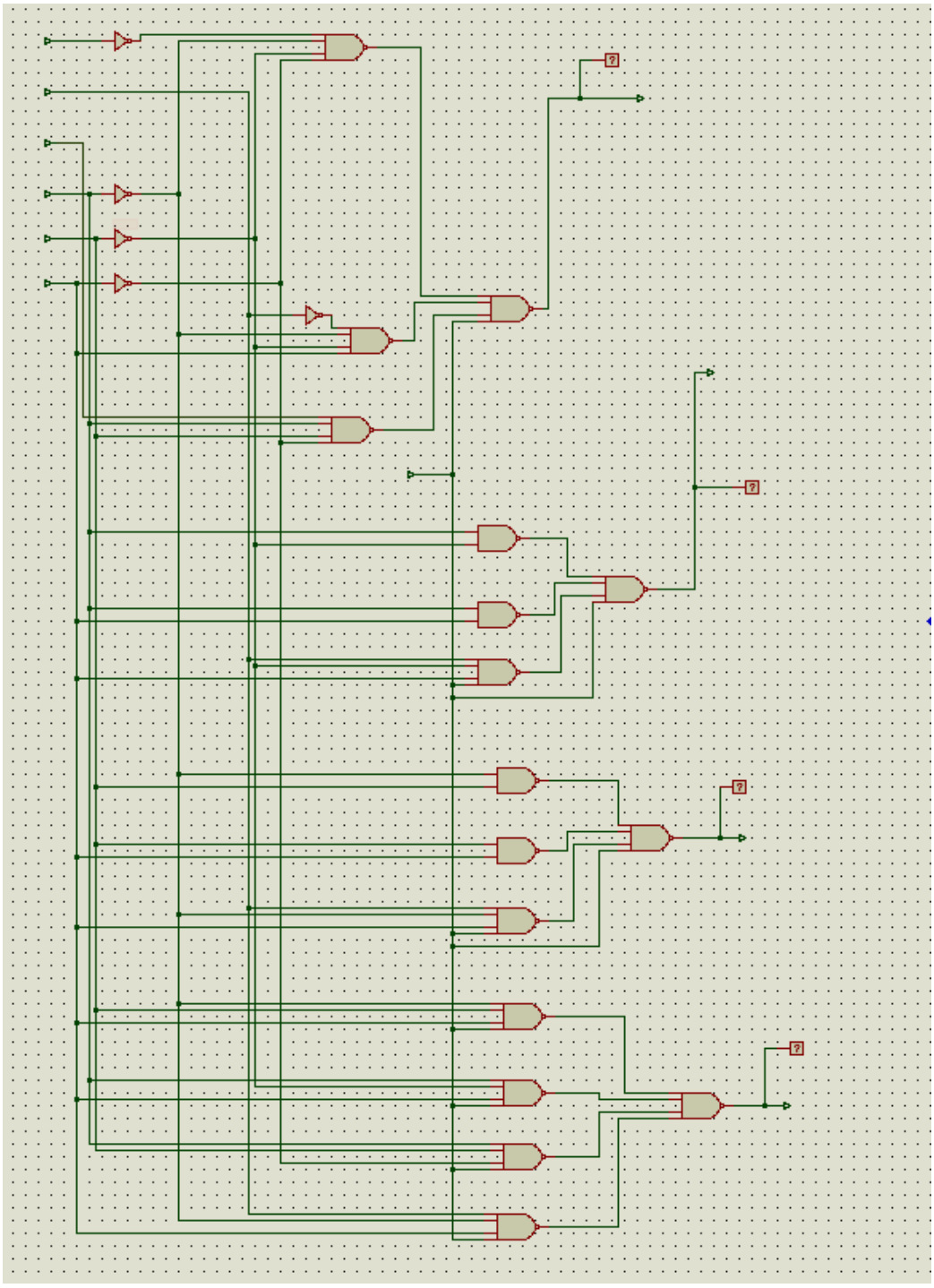


Figure 7: Detalle del bloque SELECTBITS

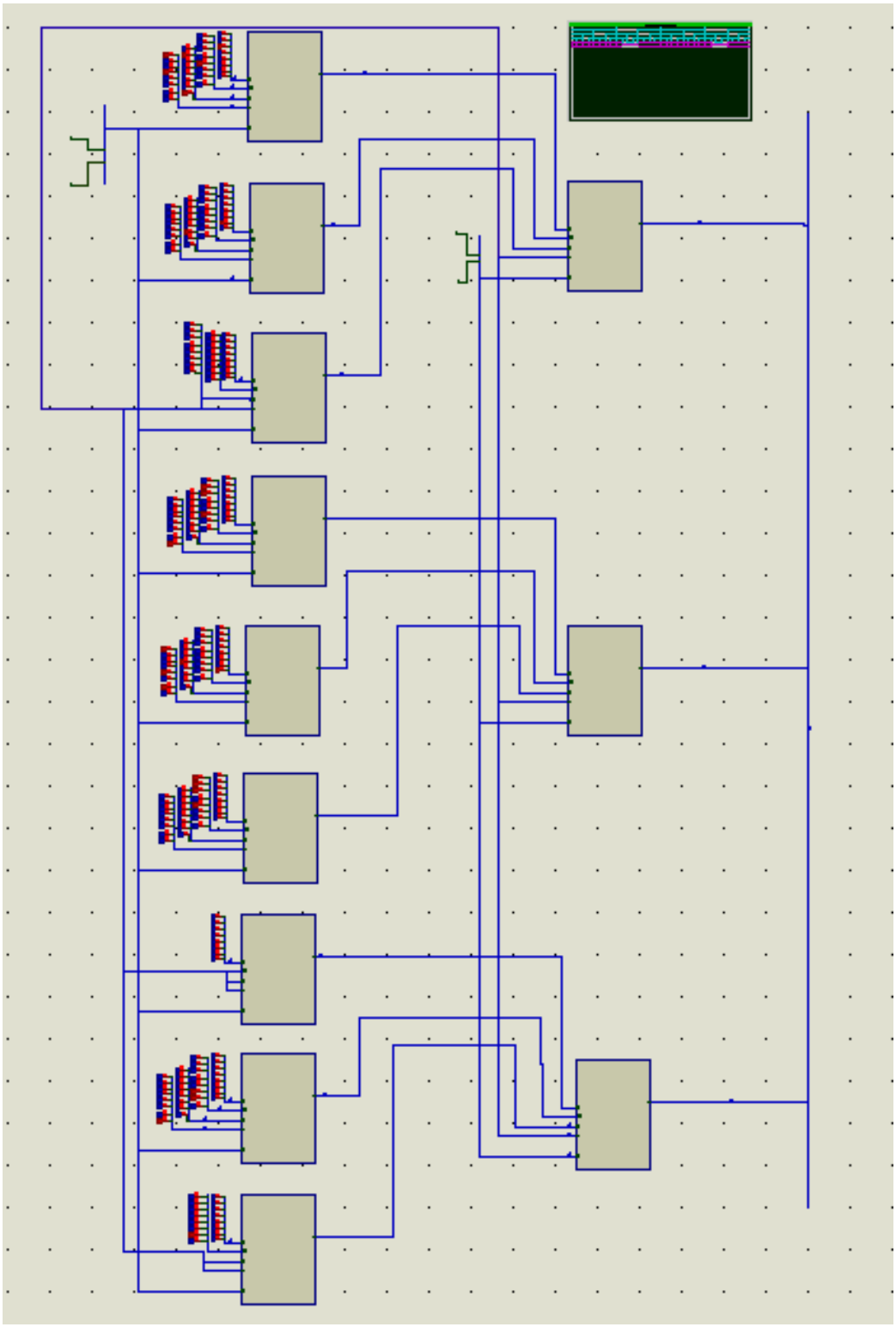


Figure 8: Detalle del bloque MUXZ

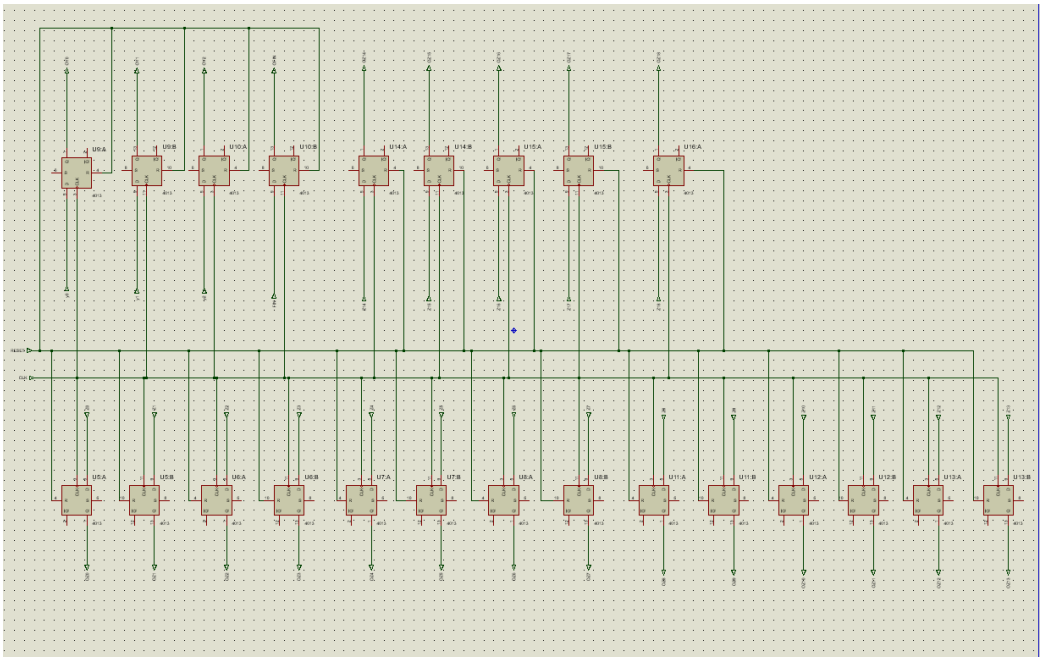


Figure 9: Detalle del bloque REGISTRO

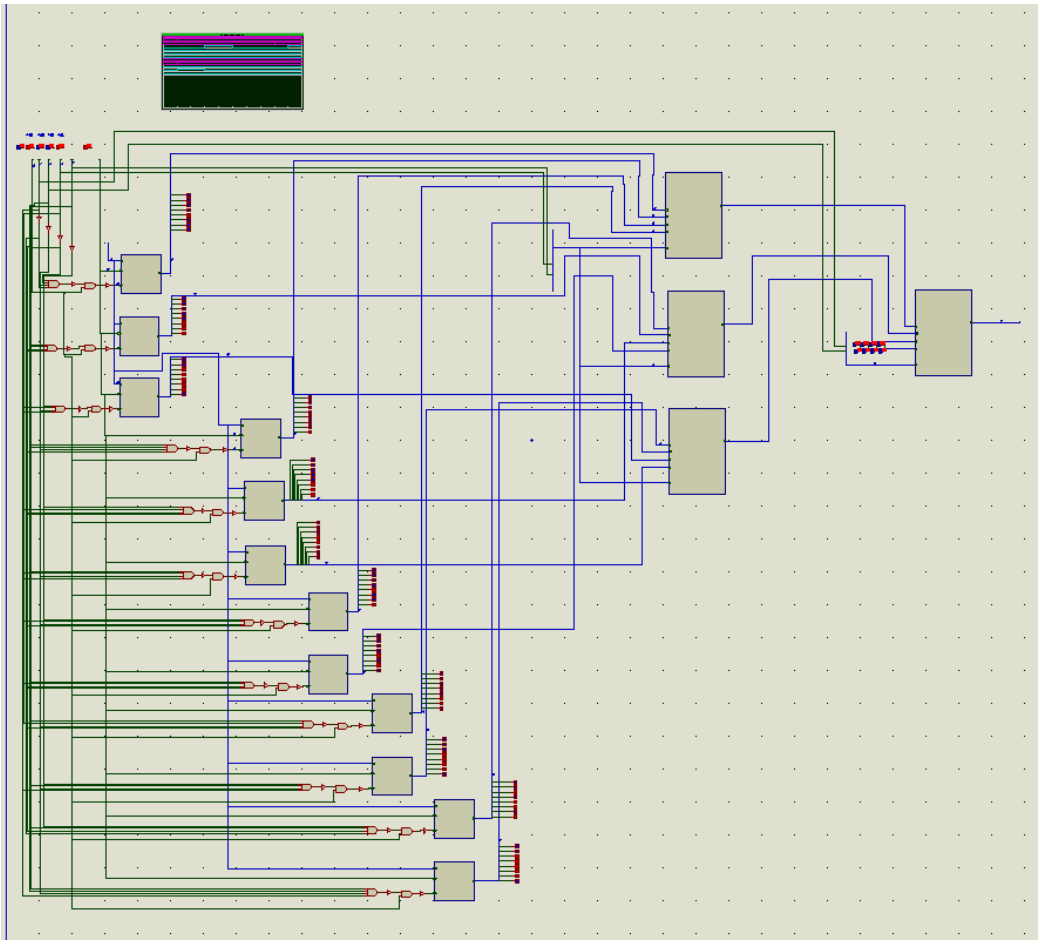


Figure 10: Detalle del recurso ARR

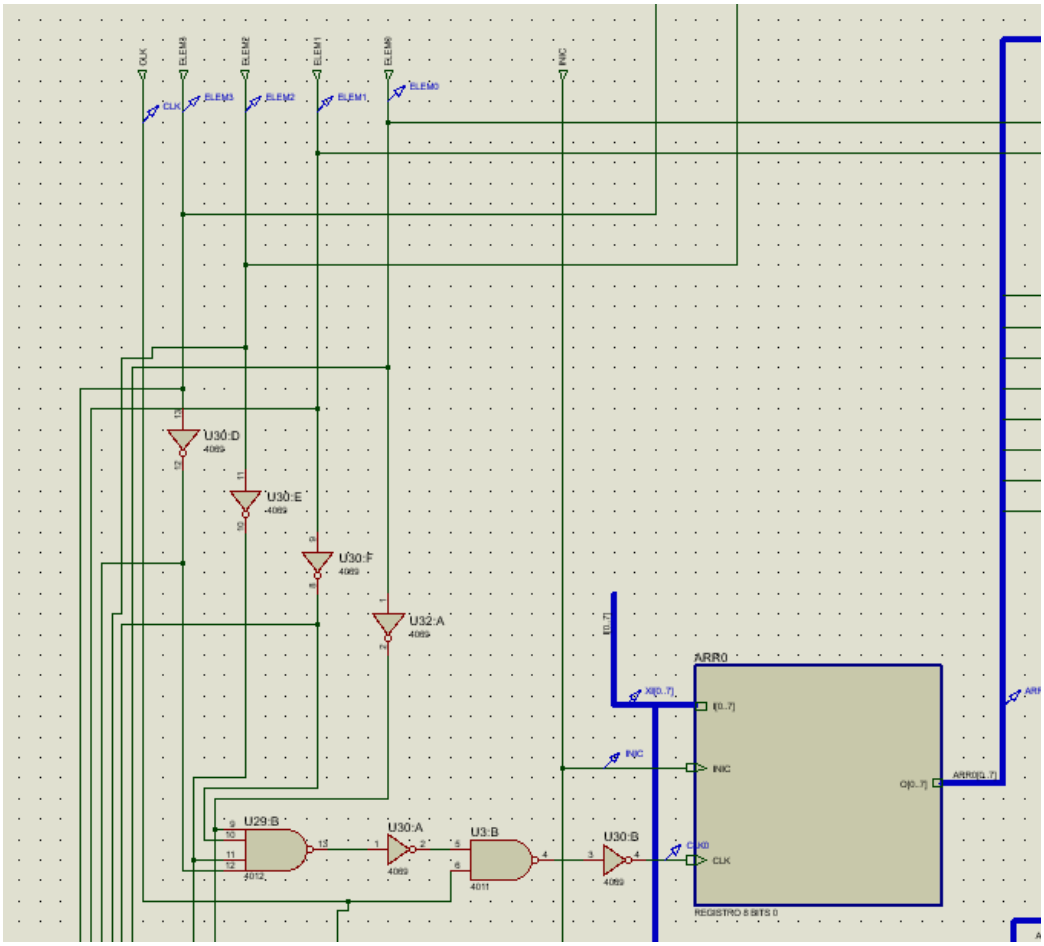


Figure 11: Detalle del combinacional de selección del registro 0 del bloque ARR

7 Referencias

1. Marqués Cuesta, Luis Alberto. "Diseño de Hardware Específico - Tema 5: Esquemas de cálculo". Universidad de Valladolid.

2. Marqués Cuesta, Luis Alberto. "Diseño de Hardware Específico - Tema 6: Máquinas algorítmicas". Universidad de Valladolid.

3. P. Deschamps y J.M. Angulo, Diseño de sistemas digitales, metodología moderna, Paraninfo.

4. Fecha de consulta: 05/05/2019. Insertion sort. Obtenido de https://www.geeksforgeeks.org/insertion-sort/