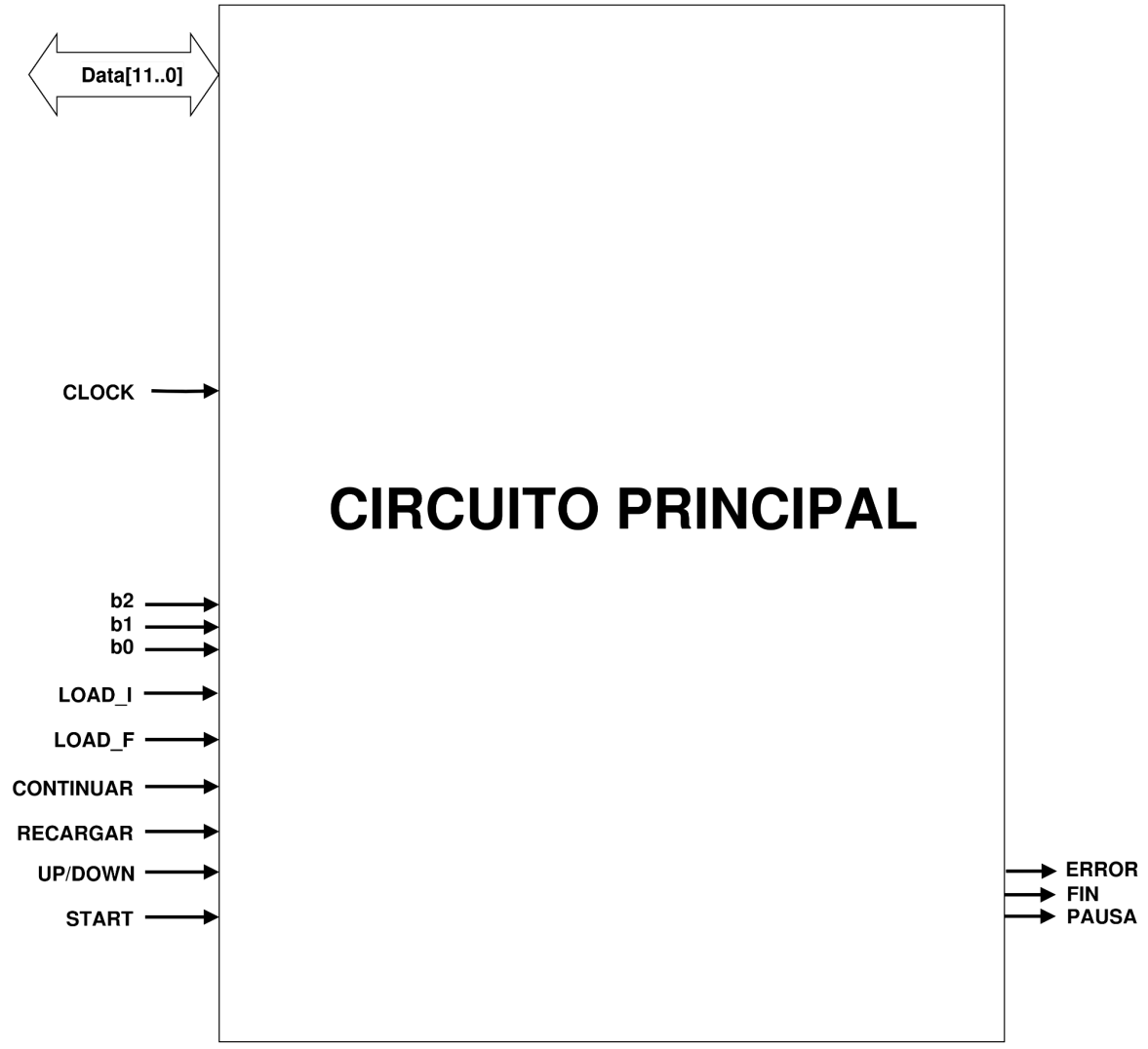
Trabajo Integrador - Introducción al Diseño Lógico

Participantes:
- Federico Goncalves - 03866/5
- Joaquín Guzmán - 03751/4
- Tomás Gamarra - 03852/8

Descripción general del sistema

A continuación se ve un esquema del circuito principal que representa nuestro contador programable.



Entradas

-**CLOCK:** Señal de reloj para poder sincronizar el circuito y poder llevar a cabo la cuenta, cambios de estado, almacenamiento en FF´s, etc.
-**DATA:** Este bus se utilizará para ingresar los valores de inicio de cuenta (**VI**) y de fin de cuenta (**VF**) que el usuario elija para llevar la cuenta. Además, se utilizara para mostrar el valor de la cuenta actual (**VC**).
-**b2b1b0:** 3 líneas que representan los bits que componen al incremento/decremento que se aplicará al valor inicial (**VI**) de la cuenta.
-**LOAD_I:** Habilita la carga del valor inicial (**VI**) a través del bus bidireccional **DATA**.
-**LOAD_F:** Habilita la carga del valor final (**VF**) de la cuenta a través del bus bidireccional **DATA**.
-**CONTINUAR:** Indica si se debe seguir o no con la cuenta una vez que haya alcanzado o sobrepasado el **VF**.
-**RECARGAR:** Al activarse esta señal es posible utilizar las señales **LOAD_I** y **LOAD_F** para volver a cargar el **VI** y **VF**.
-**UP/DOWN:** En ‘0’ indica que se debe incrementar la cuenta, mientras que en ‘1’ indica que se debe decrementar la cuenta.
-**START:** Esta señal hace que el contador **intente** llevar a cabo la cuenta en base a **VI, VF y b2b1b0**. Hacemos énfasis en la palabra **intente** ya que podría suceder que haya un error en cuanto a los valores inicializados para la cuenta y su incremento/decremento asociado.

Salidas

-**ERROR:** Se activa cuando no hay correlación entre **VI, VF** y **UP/DOWN.** Podría suceder que el usuario quisiera hacer una cuenta la cual no tenga sentido matemático (Ej. Ir de **VI**=2 a **VF**=10 **restando**)
-**FIN:** Se activa cuando la cuenta ha llegado a su fin, ya sea por alcanzar el **VF** o por superarlo.
-**PAUSA:** Se activa cuando el usuario coloca **b2b1b0 = 000** indicando que no se quiere seguir con la cuenta.
-**DATA:** Se utilizara este bus para mostrar los distintos valores que surjan de hacer la cuenta que desee el usuario.

Aclaraciones sobre interpretación del problema para nuestra solución

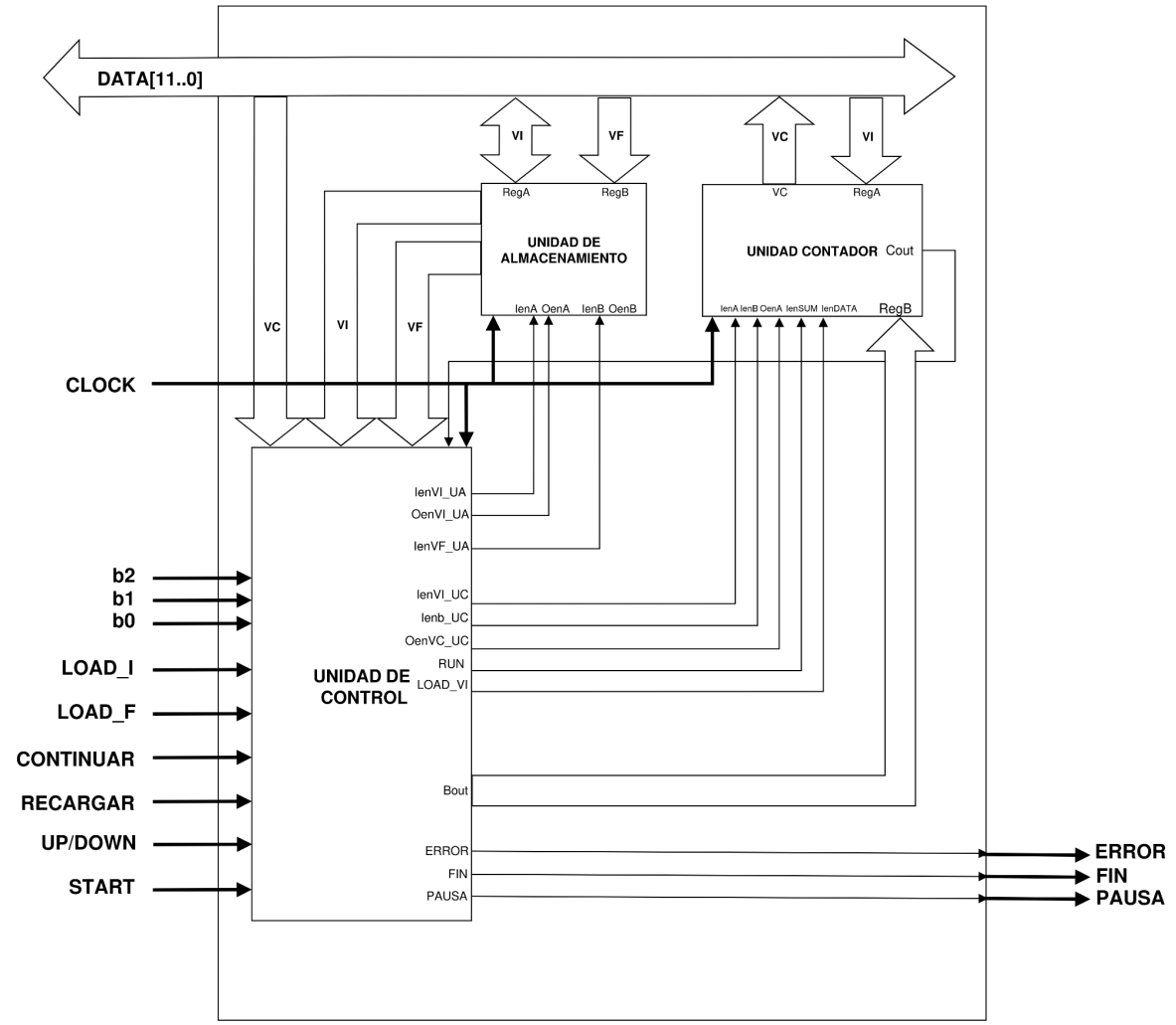
1. Entendimos que **b2b1b0** podrá cambiar durante el transcurso de una cuenta.
2. Trabajaremos con **VI** y **VF** siendo solo valores positivos para la cuenta.
3. La elección sobre incrementar o decrementar la cuenta (en base a la señal **UP/DOWN**) se elige en el momento de la carga de los valores **VI** y **VF**. Guardaremos este valor en un registro de 1 bit, modificable hasta que se ponga en alto la señal de

START. Una vez que comience la cuenta ya no será posible cambiar ese valor hasta que se vuelva al estado de carga mediante la señal de **RECARGAR**.

Diagrama de bloques

Podemos representar al circuito principal como el trabajo en conjunto de múltiples módulos, se vería de la siguiente forma:

CIRCUITO PRINCIPAL



En este diagrama, las tareas para lograr la funcionalidad deseada se reparten a cada unidad de la siguiente forma:

Unidad de Almacenamiento

- **Almacenar VI y VF:** Leerá del bus **DATA** los valores inicial y final del contador para guardarlos respectivamente en los registros A y B (regA y regB). El momento en el que guarde los valores vistos en el bus será dictado por la **Unidad de Control** mediante las señales de control **lenVI_UA** y **lenVF_UA**. También podrá mostrar en el bus **DATA** los valores de sus registros según la **Unidad de Control** le indique mediante las señales de control **OenVI_UA** y **OenVF_UA**. Para nuestro caso, no es necesario mostrar el valor en el registro B, por lo que la señal **OenVF_UA** quedará desconectada.

Unidad Contador

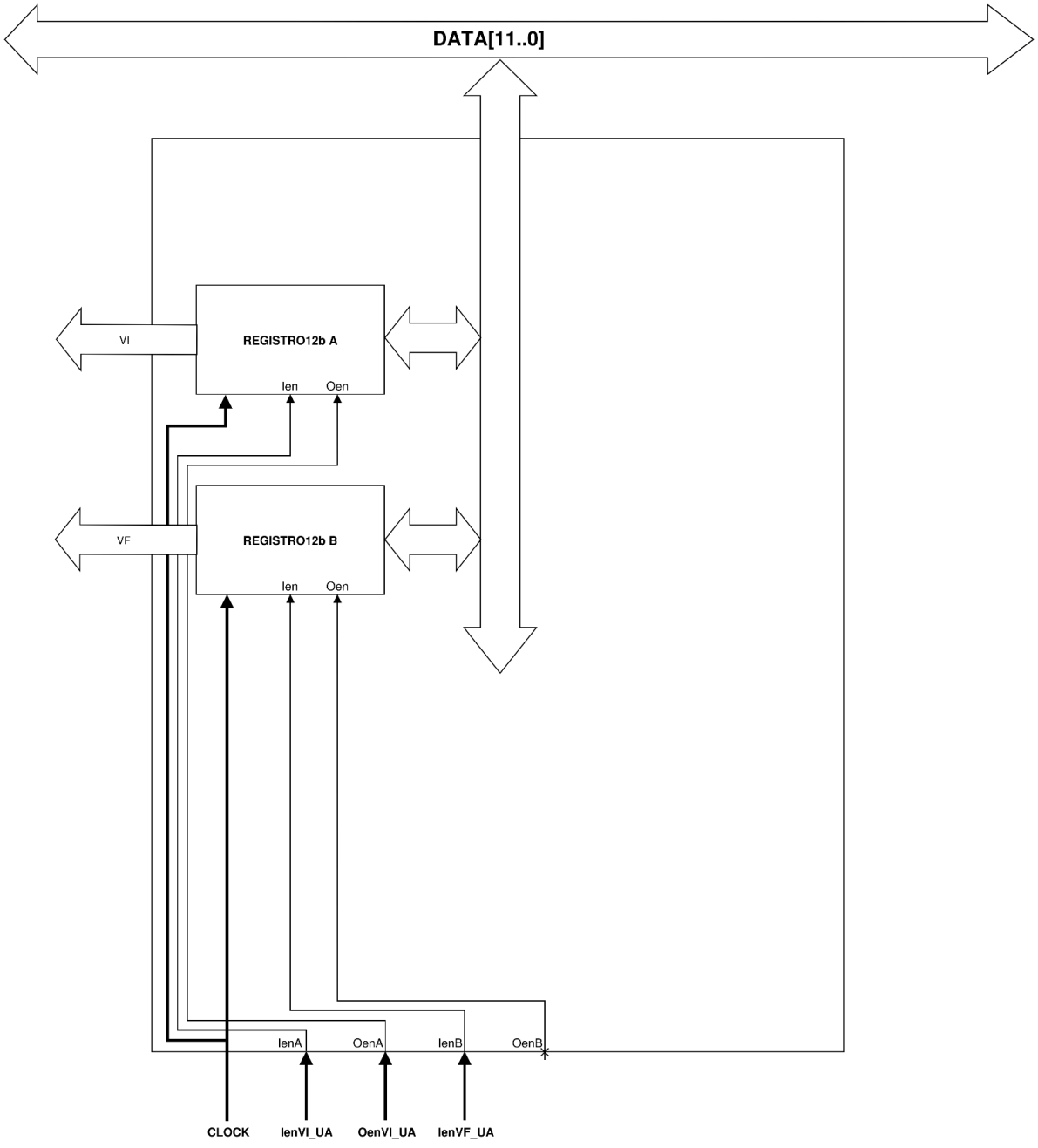
- **Suma A con B:** Se le podrá cargar un valor inicial al contador **(VI)** desde el bus **DATA** al registro A (que también será el registro acumulador) y un valor a sumar **(B)** a por cada ciclo reloj en el registro B. Cuando el contador esté andando, podrá elegirse si mostrar o no el valor del registro A en el bus DATA, y si detenerlo o no. Todas estas funcionalidades se lograrań mediante la combinación de sus señales de control **lenVI_UC, lenb_UC, OenVC_UC, RUN** y **LOAD_VI.**

Unidad de Control

- **Controlar a la Unidad de Almacenamiento:** Dependiendo de la combinación de señales de entrada CONTINUAR, RECARGAR y START, decide si se colocara en el bus **DATA** los valores **VI** o **VF** y dicta a la **UA** que los almacene.
- **Controlar a la Unidad Contador:** Dependiendo de la combinación de señales de entrada CONTINUAR, RECARGAR y START, decide si es el momento de iniciar el contador o frenarlo.
- **Determinar estados:** Se encargará en base a datos como los valores de b2b1b0, VC, VI y VF y en base a señales como LOAD_I, LOAD_F, CONTINUAR, RECARGAR, UP/DOWN y START de determinar qué señal enviar a cada unidad del circuito.

Ahora veremos cómo están conformados cada unidad por dentro también en forma de diagrama de bloque junto con una descripción de ellos:

UNIDAD DE ALMACENAMIENTO

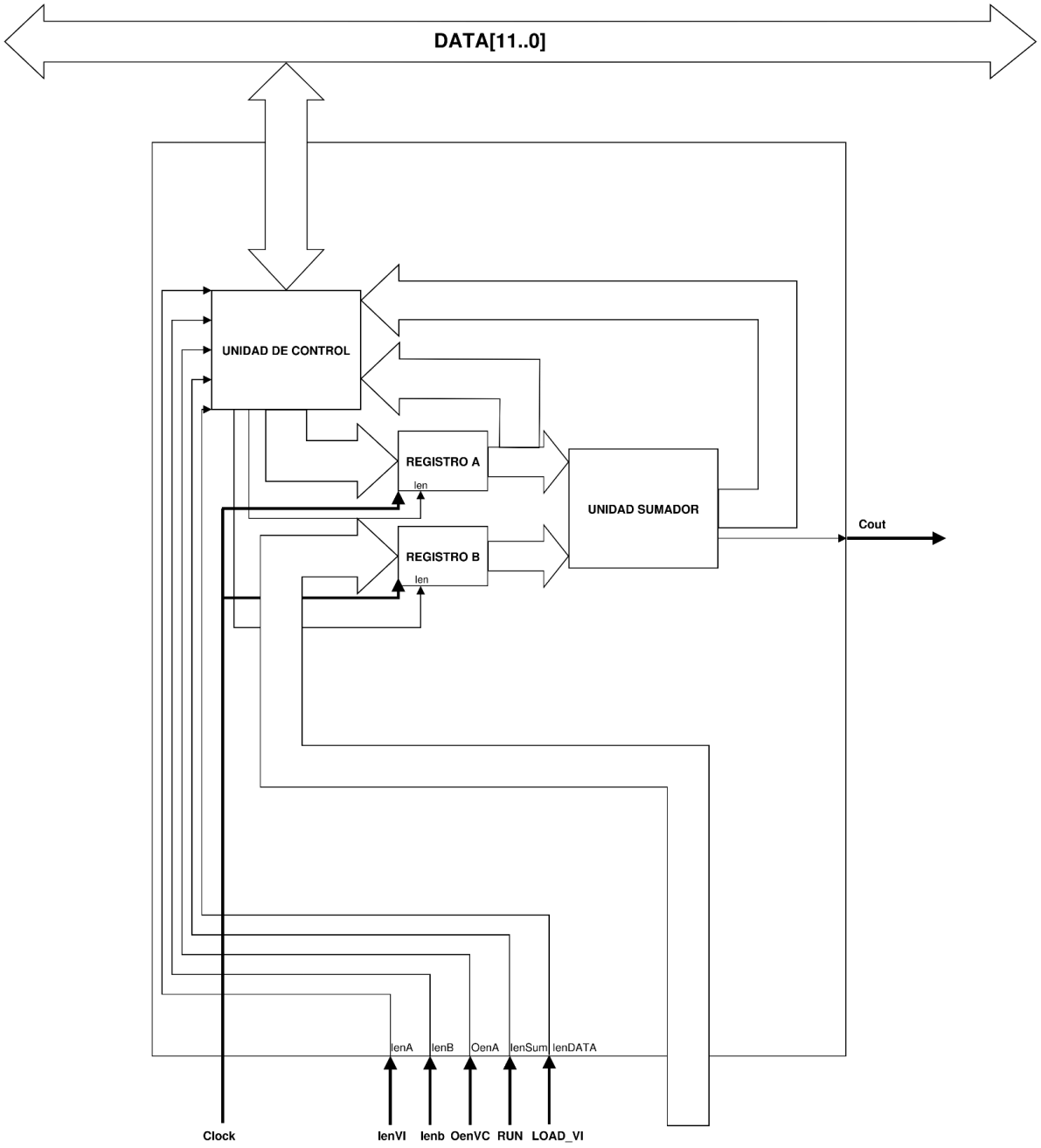


El diagrama de la unidad de almacenamiento necesaria para este circuito es más bien simple, y un desarrollo más profundo será ya realizado directamente en la implementación del circuito en Quartus.

- **Registro A:** Si la señal **lenA** está activa, lee y guarda los datos del bus **DATA.** Si la señal **OenA** está activa, muestra el contenido del registro en el bus **DATA.**

- **Registro B:** Si la señal **lenB** está activa, lee y guarda los datos del bus **DATA.** Si la señal **OenB** está activa, muestra el contenido del registro en el bus **DATA.**

UNIDAD CONTADOR

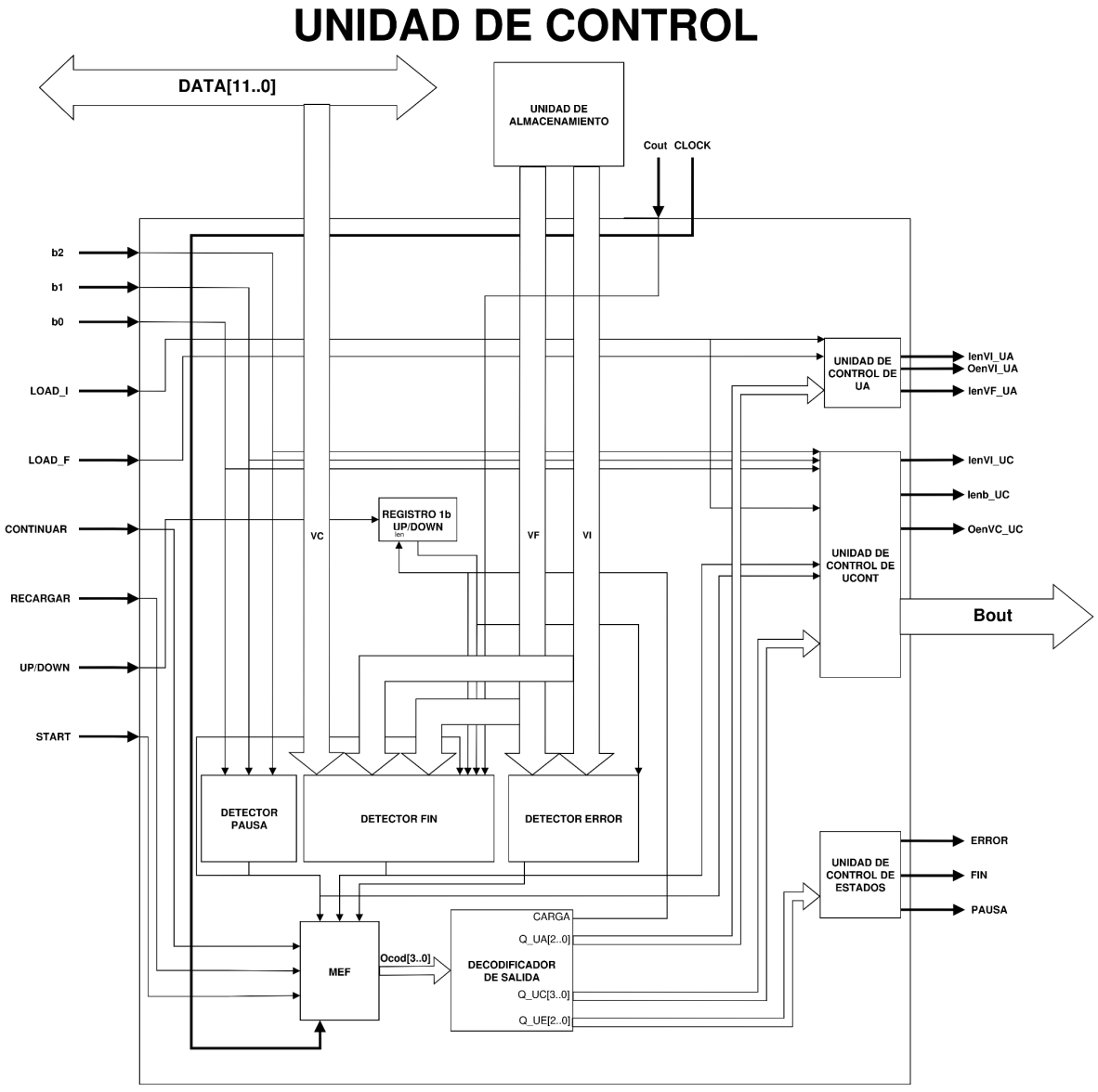


Vemos que a diferencia de la **Unidad de Almacenamiento,** la **Unidad Contador** es algo más compleja y cuenta con una **Unidad de Control** propia.

- **Unidad de Control:** Permitirá el paso de **VI** del bus **DATA** cuando la **Unidad de Control (UC)** se lo indique mediante las señales **lenVI** (para permitir que registro A se escriba) y **LOAD_VI** (para permitir que el dato puesto en el bus **DATA** pase a registro A). Por otro lado en el registro B se cargará el valor a sumar a **VI**, la **UC**

controlará esto mediante la señal **lenb** para permitir la escritura del registro B y el bus **B** por donde se pondrá el valor a sumar a cada ciclo de reloj. Cuando la **Unidad Contador** se encuentre en funcionamiento se deberá mantener **RUN** en alto junto con **lenA** para permitir que el registro A se escriba con los datos del sumador. Para además, mostrar el valor del registro A (acumulador) en el bus **DATA** se tendrá que mantener **OenVC** en alto.

- **Sumador:** Esta unidad permite sumar 2 números de 12 bits. Además de la suma provee la salida del carry de la suma llamado **Cout (Carry out).**



La **Unidad de Control** del **Circuito Principal** será la de mayor complejidad ya que está se ha de encargar de, según las señales de entrada, transicionar de estado en la **MEF** y en base al estado actual mandar ciertas señales de control a cada unidad del **Circuito Principal.** De esta forma, se logra el funcionamiento deseado y, en caso de querer modificarlo, se puede cambiar la lógica de la **MEF.**

- **Registro 1b UP/DOWN:** Almacena el modo en el que se contará desde **VI** hacia **VF.**
- **Detector Pausa:** Manda una señal a la **MEF** en el caso de que se verifique la condición de pausa **b2b1b0** = 000.
- **Detector Fin:** Si no se está en el estado de **CARGA** manda una señal a la **MEF** en el caso de que se verifique la condición de fin dependiendo **VI, VF y UP_DOWN**. Para cuando se está en el estado de **CARGA** cambia su lógica y únicamente manda la señal de **FIN** a la **MEF** si **VI=VF**.
- **Detector Error:** Manda una señal a la **MEF** en el caso de que se verifique la condición de error **VI** $>$ **VF** $\wedge$ **UP_DOWN=0** o **VI** $<$ **VF** $\wedge$ **UP_DOWN=1**.
- **MEF:** En base a las entradas **CONTINUAR**, **RECARGAR**, **START**, **PAUSA**, **FIN** y **ERROR** (siendo estas últimas 3 entradas provenientes de los detectores) decide a qué estado transicionar (estados los cuáles serán definidos más adelante).
- **Decodificador:** En base a las salidas de la **MEF** decide qué señal enviar a cada unidad de control dedicada (**UC de UA**, **UC de UCont** y **UC de Estados**).
- **Unidad de Control de UA:** En base a las entradas de control del decodificador puede: permitir el paso de las señales **LOAD_I** y **LOAD_F**, mostrar en el bus **DATA** el valor inicial (**VI**) o no hacer nada.
- **Unidad de Control de UCont:** En base a las entradas de control del decodificador puede: Habilitar la salida del contador (**VC**) al bus DATA, cargar **VI** en **regA**, cargar **Bout** en **regB** y permitir o no que se actualice la cuenta.
- **Unidad de control de Estados:** En base a las entradas de control del decodificador (las cuales hacen referencia al estado en el que se encuentra la MEF) se encarga de proporcionar la salida de **ERROR**, **FIN** o **PAUSA**.

Finalmente, con el trabajo conjunto de todas estas unidades, la **Unidad de Control** actúa en consecuencia al estado en el que el sistema se encuentra actualmente .

Descripción e implementación de los bloques

A continuación se desarrollará la lógica de los distintos bloques que necesitamos para implementar las distintas unidades (**UControl, UAlmacenamiento, UContador**).

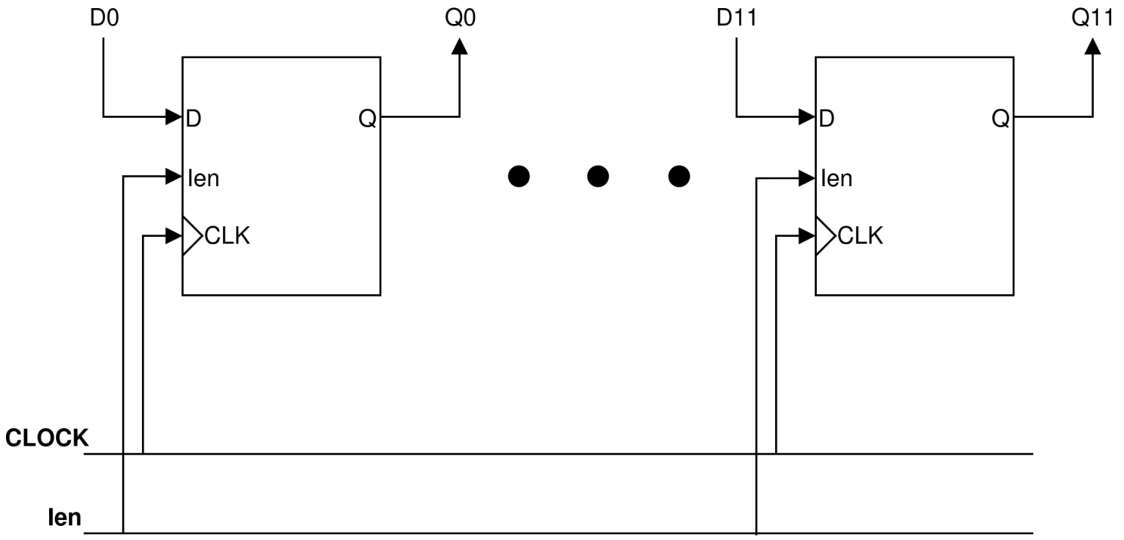
Generales

Aquí se detallarán ciertos bloques que se utilizaran en distintas unidades o para diferentes bloques independientes.

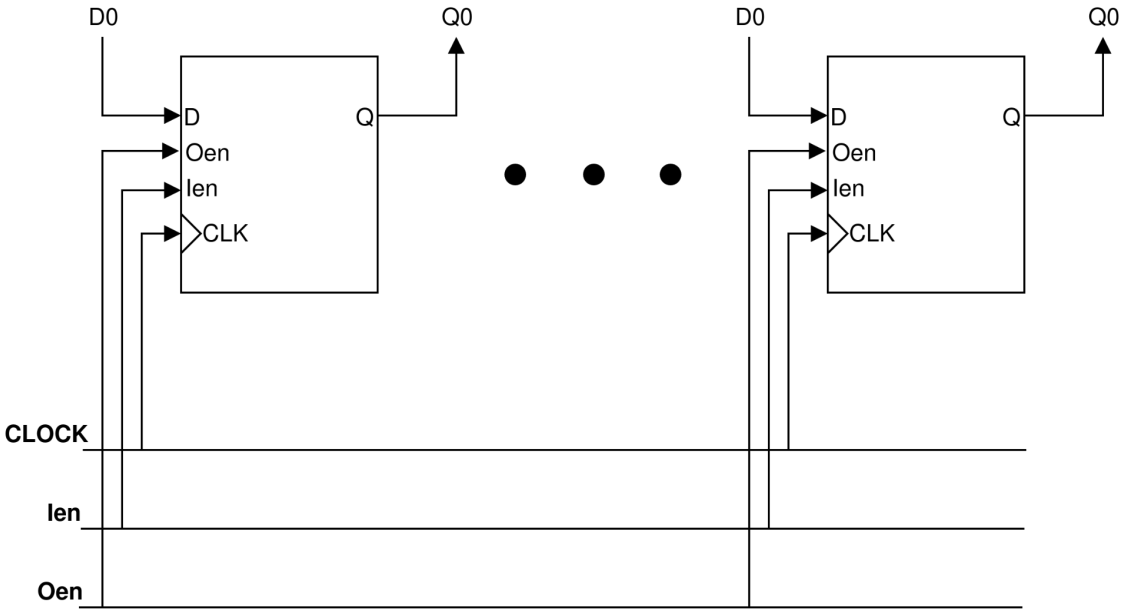
Registro 12 bits

Debido a que trabajaremos con números binarios de 12 bits, necesitaremos almacenar estos mismos en varias etapas del circuito para poder lograr la funcionalidad deseada. Es por esto que se diseñaron 2 tipos de registros de 12 bits de uso general, uno con señal Input enable y otro con Input enable y Output enable. En particular, se eligió un modelo de registro de 12 bits de entradas y salidas en paralelo (PIPO).

Registro 12 bits con len



Registro 12 bits con len y Oen



Comparador de 12 bits

Se encargará de comparar 2 valores de 12 bits cada uno (**A y B**) y obtendremos a su salida señales que indican el resultado de la comparación (**si A=B, A>B o A<B**).

Entradas

-**A:** Valor a comparar
-**B:** Valor a comparar

Salidas

-**AmayB:** Se activa cuando A>B.
-**AeqB:** Se activa cuando A=B.
-**AmenB:** Se activa cuando A<B.

Para su implementación nos basaremos en un comparador de 1 bit. El cual **además** de las entradas y salidas detalladas anteriormente, posee las siguientes entradas:

-**lAmayB:** Se activa cuando A>B.
-**lAeqB:** Se activa cuando A=B.
-**lAmenB:** Se activa cuando A<B.

Las entradas que comienzan con '**l**' **pasan a ser la salida del comparador cuando estamos en caso de igualdad**. Esto quiere decir que cuando **A sea igual a B** obtendremos en las salidas (**AmayB, AeqB**, **AmenB**) los valores que tengan las entradas con '**l**' delante (**lAmayB, lAeqB, lAmenB)** . Estas entradas son necesarias para luego poder conectar estos comparadores en cascada y formar un comparador de mayor cantidad de bits.

De esta forma la tabla de verdad para el comparador de 1 bit sin tener en cuenta las entradas para el caso de igualdad sería:

A	B	(A>B)	(A=B)	(A<B)
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$$(A > B) = A\overline{B}$$
$$(A = B) = \overline{A \oplus B}$$

$$(A < B) = \overline{A}B$$

Ahora hay que agregar la lógica de que si **A=B**, tendremos en la salida lo que indiquen las entradas con **I** delante.

$$(A > B) = A\overline{B} + ((\overline{A \oplus B})I(A > B))$$
$$(A = B) = (\overline{A \oplus B})(I(A = B))$$
$$(A < B) = (\overline{A}B) + ((\overline{A \oplus B})I(A < B))$$

De esta manera ya tenemos un comparador de 1 bit el cual además en caso de igualdad puede decidir la comparación en base a sus entradas con '**I**' delante. Esto permite conectarlos en cascada haciendo que las salidas de los comparadores de los bits menos significativos se conecten a las entradas del comparador siguiente (las que deciden en caso de igualdad). Al tener todo conectado de esta forma el comparador del MSB será el que decida cuál será la salida del comparador de 12 bits, ya que en caso de igualdad se decidirá mediante lo que obtengamos de los demás comparadores.

Sumador completo de 12 bits

Sumará 2 números de 12 bits más un Carry in. Se implementará en Quartus.

Unidad de Almacenamiento

Registros A y B

Se utilizarán los registros de 12 bits detallados en "Generales".

Unidad Contador

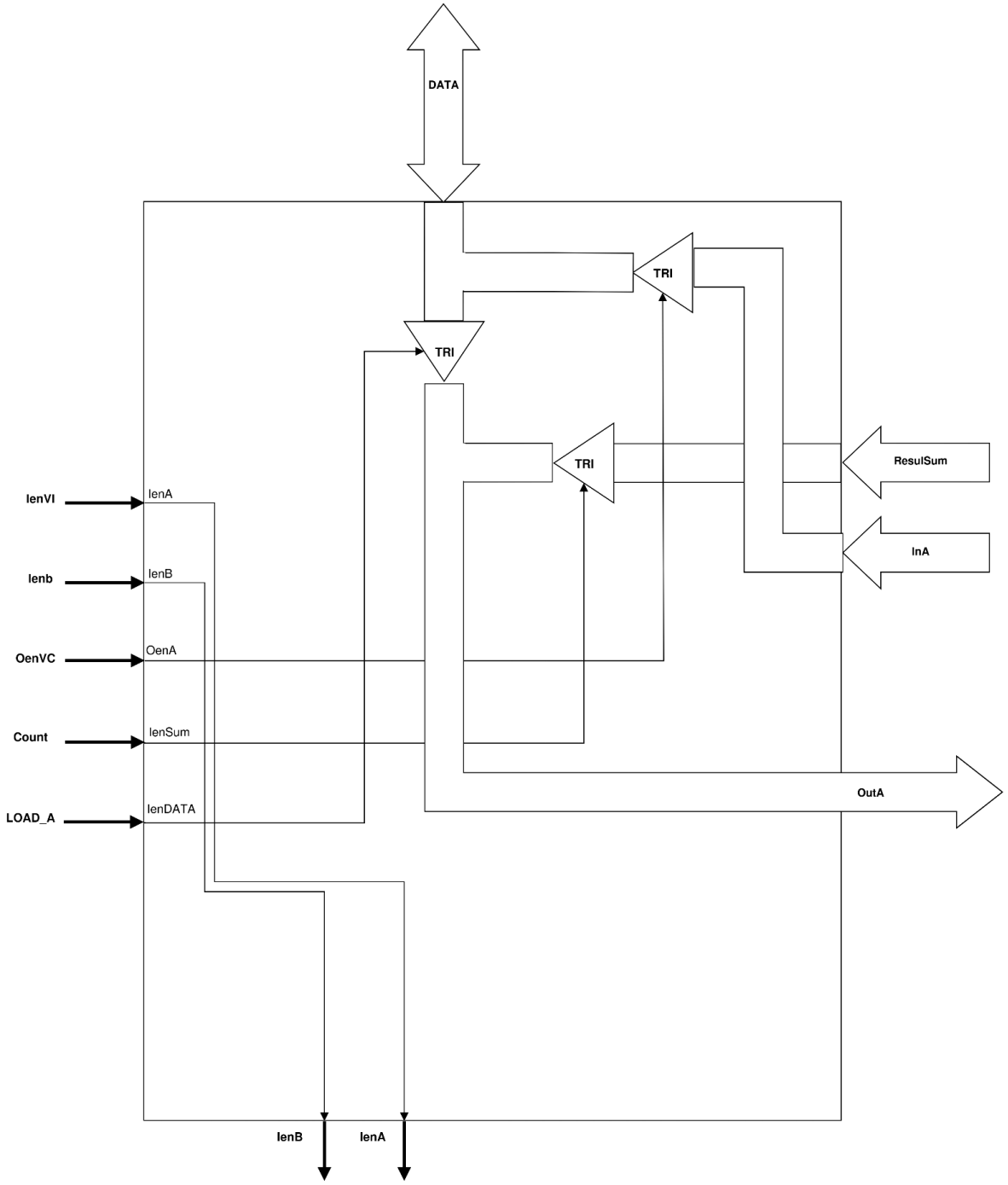
Unidad de control (dentro de Unidad Contador)

Entradas

-lenA
-lenB
-OenA
-lenSum
-lenDATA
-ResulSum (Bus entrada)
-InA (Bus entrada con datos de registro A)
-DATA (bus **DATA** bidireccional)

Salidas

-DATA (bus **DATA** bidireccional)
-lenA
-lenB



Las entradas **lenSum** y **lenDATA** permiten a la **Unidad de Control** (principal) seleccionar qué valor será el que le entre al registro A, si el del bus **SUM** (con el resultado del sumador) o si el del bus **DATA** (con el valor inicial a cargar). La entrada **OenA** permite que se muestre el valor del registro A (acumulador) en el bus **DATA** mediante el bus **InA**, logrando de esta forma mostrar un estado estable del contador. Por otro lado, las entradas l**enA** y **lenB** no tendrán lógica asociada (aunque sí es importante que pasen primero por la unidad de control por si en un futuro quisiera condicionarse la activación de estas señales) y pasarán directamente a sus registros respectivos.

Registros A y B

Se utilizarán los registros de 12 bits (no Tristate) detallados en “Generales”.

Unidad Sumador

Se utilizará la unidad sumadora de 12 bits nombrada en “Generales”, implementada directamente en Quartus.

Unidad de Control

Registro UP/DOWN

Este será un registro para almacenar 1 bit, el cual permitirá que ingresen valores (Input enable activado) únicamente durante el estado de CARGA. Ya que el usuario solo podrá modificar este valor durante ese estado. Una vez que comience la cuenta queda almacenado en este registro el valor seleccionado.

Detector de Pausa

El contador entrará en pausa cuando **b2b1b0 = 000.**

Entradas

-**b2**
-**b1**
-**b0**

Salida

-**F**

Por el enunciado sabemos que se tendrá que detener el contador cuando el valor del sumando **b2b1b0 = 000** por lo que la salida del Detector de Pausa valdrá ‘**1**’ únicamente cuando intervengan las 3 entradas negadas, de esta forma la función que describa al detector será :

$$F = \overline{b_2}\,\overline{b_1}\,\overline{b_0}$$

Detector Error

Este detector se encargará de chequear que haya un sentido entre **VF, VI** y la elección de incrementar o decrementar la cuenta (**UP/DOWN)**. En caso que no tenga sentido obtendremos un ‘1’ a la salida.

Entradas

-**VF**
-**VI**
-**UP/DOWN**

Salida

-**F**

El detector contiene dentro un comparador de 12 bits que nos proveerá las salidas **VF>VI, VF=VI, VF<VI.** Gracias a ellas y la señal **UP/DOWN** obtendremos la lógica de la función.

La tabla de verdad que describa la lógica del detector de errores es la siguiente:

VF>VI	VF=VI	VF<VI	UP/DOWN	F
0	0	0	X	X
0	0	1	0	1
0	0	1	1	0
0	1	0	X	0
0	1	1	X	X
1	0	0	0	0
1	0	0	1	1
1	0	1	X	X
1	1	0	X	X
1	1	1	X	X

Cabe aclarar que las salidas de los estados que no tienen sentido (como que sea mayor e igual al mismo tiempo) las tomaremos como “don't care” o X.
Renombramos las entradas de la siguiente forma para realizar k-map:
(VF > VI): A
(VF = VI): B

(VF < VI): C
UP/DOWN: D

Ahora armando el k-map obtenemos:

F	$\overline{C} \cdot \overline{D}$	$\overline{C} \cdot D$	$C \cdot D$	$C \cdot \overline{D}$
$\overline{A} \cdot \overline{B}$	X	X	0	1
$\overline{A} \cdot B$	0	0	X	X
$A \cdot B$	X	X	X	X
$A \cdot \overline{B}$	0	1	X	X

$$F = AD + C\overline{D}$$

Como se ve, la salida del comparador que nos dice si las entradas son iguales (**Entrada B** o **VF=VI**) no tiene una implicancia en la lógica del circuito, lo cual hace sentido ya que para saber si hay un error solo me interesan los casos donde VF o VI es mayor o menor que el otro. Esto es debido a que en el caso donde VF=VI no importa si la cuenta es ascendente o descendente. Se podría haber excluido la señal A=B (o entrada B) de la tabla de verdad.

Detector Fin

Siguiendo la misma lógica que para el detector de errores utilizaremos las salidas del comparador de 12 bits (al colocarle en su entrada **VC** y **VF)** en conjunto con la entrada de **UP/DOWN** para determinar si la cuenta ha llegado a su fin. Para los casos límite necesitaremos saber el valor del carry out que genere el sumador (**Cout**).
Caso 1: Overflow. En estos casos **Cout = 1**.
Caso 2: Underflow. En estos casos **Cout = 0**.

Entradas

-**VF.** Valor final de la cuenta
-VI. Valor inicial de la cuenta
-VC. Valor de la cuenta
-**UP/DOWN.** Señal que indica si estamos en modo incremento/decremento.
-**Cout.** Carry out del sumador
-CARGA. Señal que nos indica si estamos en estado de CARGA.
-PAUSADET. Salida del detector de pausa

Salida

-**F**

Como se puede ver, además de las entradas mencionadas se tiene la entrada **CARGA**. Esta entrada la utilizaremos para saber cuando estamos en estado de **CARGA**, ya que para ese caso deberemos únicamente constatar si **VI=VF**. Esto que hacemos es debido a que si en estado de carga quisiéramos comparar el **VC** con **VF** obtendremos algo indeterminado, ya que en este estado todavía no hay un valor asignado a **VC**, por lo que compararemos **VI con VF** para saber únicamente si son iguales y así informar que la cuenta ha llegado a su fin.

La tabla de verdad que describa la lógica del detector de fin para cuando **no estamos en estado de carga (CARGA=0)** es la siguiente:

VF>VC	VF=VC	VF<VC	UP/DOWN	Cout	F
0	**0**	**0**	**X**	**X**	**X**
0	**0**	**1**	**0**	**X**	**1**
0	**0**	**1**	**1**	**0**	**1**
0	**0**	**1**	**1**	**1**	**0**
0	**1**	**0**	**X**	**X**	**1**
0	**1**	**1**	**X**	**X**	**X**
1	**0**	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**1**	**1**
1	**0**	**0**	**1**	**X**	**1**
1	**0**	**1**	**X**	**X**	**X**
1	**1**	**0**	**X**	**X**	**X**
1	**1**	**1**	**X**	**X**	**X**

Cabe aclarar que tome las salidas de los estados que no tienen sentido (como que sea mayor e igual al mismo tiempo) como **“don't care”** o **X**.

Con el siguiente renombramiento para las salidas del comparador de 12 bits y las demás entradas:
(VF > VC): A
(VF = VC): B
(VF < VC): C
UP/DOWN: D
Cout: E

Ahora armando los k-maps obtenemos :

Para E=0

$F1$	$\overline{C} \cdot \overline{D}$	$\overline{C} \cdot D$	$C \cdot D$	$C \cdot \overline{D}$
$\overline{A} \cdot \overline{B}$	X	X	1	1
$\overline{A} \cdot B$	1	1	X	X
$A \cdot B$	X	X	X	X
$A \cdot \overline{B}$	0	1	X	X

$$F1 = \overline{A} + D$$

Para E=1

$F2$	$\overline{C} \cdot \overline{D}$	$\overline{C} \cdot D$	$C \cdot D$	$C \cdot \overline{D}$
$\overline{A} \cdot \overline{B}$	X	X	0	1
$\overline{A} \cdot B$	1	1	X	X
$A \cdot B$	X	X	X	X
$A \cdot \overline{B}$	1	1	X	X

$$F2 = \overline{C} + \overline{D} = \overline{CD}$$

Ahora obtenemos F de la siguiente forma :

$$F = \overline{E}F1 + EF2 = \overline{E}\,(\overline{A} + D) + E(\overline{CD})$$

Pero esta F no está completa ya que estábamos en el caso de que CARGA=0. Así que ahora agregaremos la lógica para cuando CARGA=1. Esto lo haremos comparando **VF con VI** y utilizando la salida del comparador AeqB a la cual llamaremos G, asi quedaria:

$$F = \overline{CARGA}(\overline{E}\,(\overline{A} + D) + E(\overline{CD})) + CARGA \,.\, G$$

Como se puede ver se multiplicó a lo obtenido para el caso de CARGA=0 por $\overline{CARGA}$ y además se agregó el caso donde estemos en estado de CARGA y VF=VI.

Por último, es necesario que la salida del detector de fin sea válida únicamente cuando no estamos en estado de **PAUSA**. Esto es debido que cuando **b2b1b0 = 000** el **Cout** del sumador da 0, obteniéndose en el detector de fin una señal como si hubiese terminado la cuenta. Para solucionar esto nos aseguramos que la señal de fin sea únicamente válida en los casos donde no estamos en pausa. Asi la salida del detector de fin sera finalmente:

$$F = \overline{PAUSADET}(\overline{CARGA}(\overline{E}\,(\overline{A} + D) + E(\overline{CD})) + CARGA\,.\,G)$$

Máquina de estados (MEF)

Utilizaremos una MEF de tipo Moore.

Tabla de codificación de estados

Estados	Codificación
Carga	000
Error	001
Detenido	011
Pausa	010
Contando	110
Reinicio	111
Fin	101
No Alcanzable	100

Diagrama de la MEF

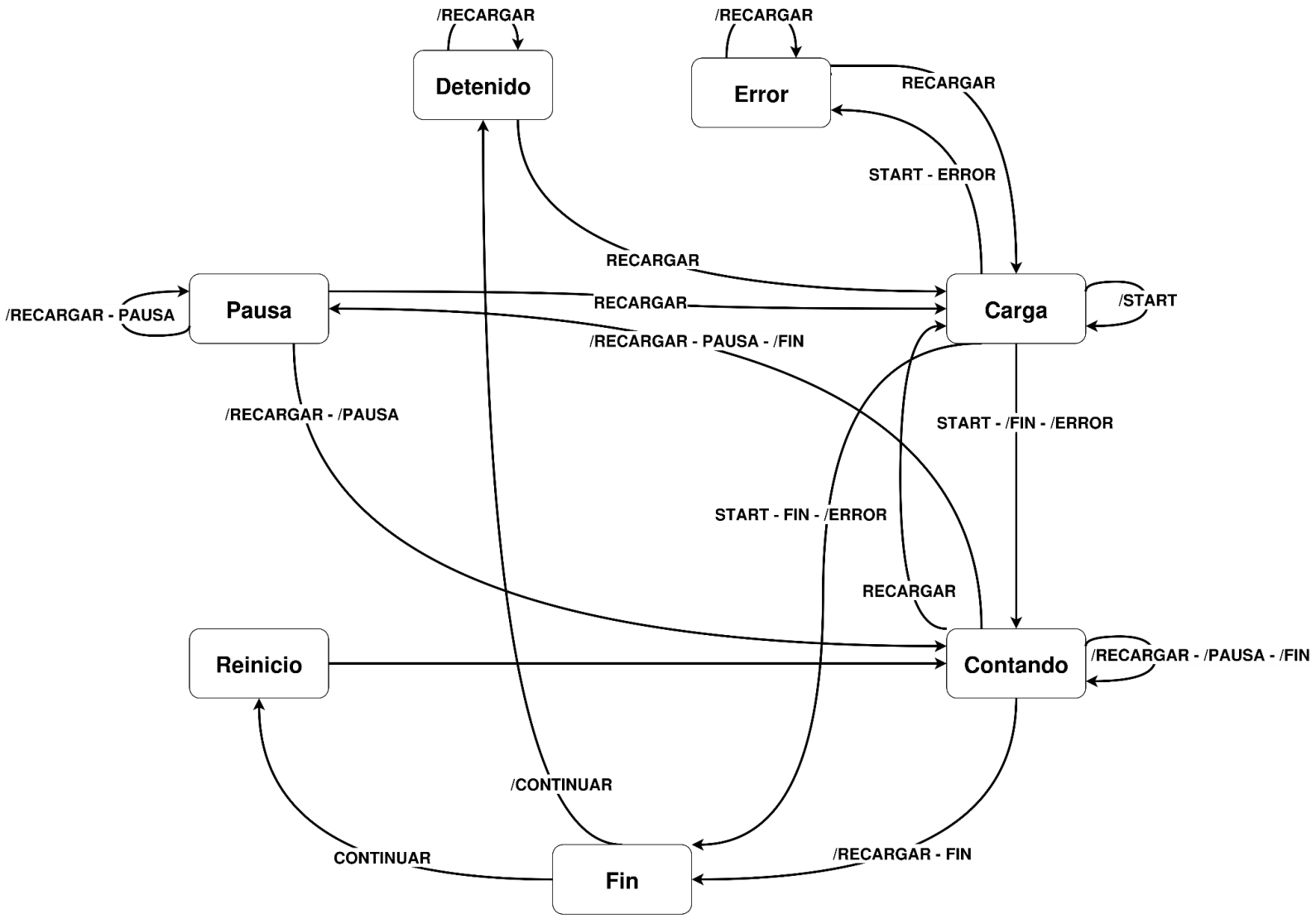

/RECARGAR
/RECARGAR
Detenido
Error
RECARGAR
START - ERROR
RECARGAR
Pausa
RECARGAR
Carga
/START
/RECARGAR - PAUSA
/RECARGAR - PAUSA - /FIN
/RECARGAR - /PAUSA
START - /FIN - /ERROR
START - FIN - /ERROR
RECARGAR
Reinicio
Contando
/RECARGAR - /PAUSA - /FIN
/CONTINUAR
CONTINUAR
/RECARGAR - FIN
Fin

Tabla de transiciones de estado

Se hará teniendo en cuenta únicamente las transiciones relevantes

Estados	Cod	Entradas						Estados	Cod	FF D
	$Q_2Q_1Q_0$	START	RECARGAR	CONTINUAR	PAUSA	FIN	ERROR		$Q_2Q_1Q_0$	$D_2D_1D_0$
Carga	000	0	X	X	X	X	X	Carga	000	000
Carga	000	1	X	X	X	X	1	Error	001	001
Carga	000	1	X	X	X	0	0	Contando	110	110
Carga	000	1	X	X	X	1	0	Fin	101	101
Error	001	X	0	X	X	X	X	Error	001	001
Error	001	X	1	X	X	X	X	Carga	000	000
Detenido	011	X	0	X	X	X	X	Detenido	011	011
Detenido	011	X	1	X	X	X	X	Carga	000	000
Pausa	010	X	0	X	1	X	X	Pausa	010	010
Pausa	010	X	0	X	0	X	X	Contando	110	110
Pausa	010	X	1	X	X	X	X	Carga	000	000
Contando	110	X	0	X	0	0	X	Contando	110	110
Contando	110	X	1	X	X	X	X	Carga	000	000
Contando	110	X	0	X	1	0	X	Pausa	010	010
Contando	110	X	0	X	X	1	X	Fin	101	101
Reinicio	111	X	X	X	X	X	X	Contando	110	110
Fin	101	X	X	1	X	X	X	Reinicio	111	111
Fin	101	X	X	0	X	X	X	Detenido	011	011

Obtención de funciones lógicas (No se buscará la eficiencia en las expresiones a hallar, pues se dará prioridad a facilitar la comprensión de cada término de la función lógica y en la implementación en Quartus):

D2

$$D2carga = \overline{Q2}.\overline{Q1}.\overline{Q0}.START.\overline{ERROR}$$
$$D2error = 0$$
$$D2detenido = 0$$
$$D2pausa = \overline{Q2}.Q1.\overline{Q0}.\overline{RECARGAR}.\overline{PAUSA}$$
$$D2contando = Q2.Q1.\overline{Q0}.(\overline{RECARGAR}.FIN + \overline{RECARGAR}.\overline{PAUSA}.\overline{FIN})$$
$$D2reinicio = Q2.Q1.Q0$$
$$D2fin = Q2.\overline{Q1}.Q0.CONTINUAR$$

Queda la función lógica D2:

$$D2 = \overline{Q2}.\overline{Q1}.\overline{Q0}.START.\overline{ERROR} + \overline{Q2}.Q1.\overline{Q0}.\overline{RECARGAR}.\overline{PAUSA} + Q2.Q1.\overline{Q0}.(\overline{RECARGAR}.FIN + \overline{RECARGAR}.\overline{PAUSA}.\overline{FIN}) + Q2.Q1.Q0 + Q2.\overline{Q1}.Q0.CONTINUAR$$

D1

$$D1carga = \overline{Q2}.\overline{Q1}.\overline{Q0}.START.\overline{FIN}.\overline{ERROR}$$
$$D1error = 0$$
$$D1detenido = \overline{Q2}.Q1.Q0.\overline{RECARGAR}$$
$$D1pausa = \overline{Q2}.Q1.\overline{Q0}.\overline{RECARGAR}$$
$$D1contando = Q2.Q1.\overline{Q0}.\overline{RECARGAR}.\overline{FIN}$$
$$D1reinicio = Q2.Q1.Q0$$
$$D1fin = Q2.\overline{Q1}.Q0$$

Queda la función lógica D1:

$$D1 = \overline{Q2}.\overline{Q1}.\overline{Q0}.START.\overline{FIN}.\overline{ERROR} + \overline{Q2}.Q1.Q0.\overline{RECARGAR} + \overline{Q2}.Q1.\overline{Q0}.\overline{RECARGAR} + Q2.Q1.\overline{Q0}.\overline{RECARGAR}.(PAUSA + \overline{FIN}) + Q2.Q1.Q0 + Q2.\overline{Q1}.Q0$$

D0

$$D0carga = \overline{Q2}.\overline{Q1}.\overline{Q0}.START.(ERROR + FIN)$$
$$D0error = \overline{Q2}.\overline{Q1}.Q0.\overline{RECARGAR}$$
$$D0detenido = \overline{Q2}.Q1.Q0.\overline{RECARGAR}$$
$$D0pausa = 0$$
$$D0contando = Q2.Q1.\overline{Q0}.\overline{RECARGAR}.FIN$$
$$D0reinicio = 0$$
$$D0fin = Q2.\overline{Q1}.Q0$$

Queda la función lógica D0:

$$D0 = \overline{Q2}.\overline{Q1}.\overline{Q0}.START.(ERROR + FIN) + \overline{Q2}.\overline{Q1}.Q0.\overline{RECARGAR} + \overline{Q2}.Q1.Q0.\overline{RECARGAR} + Q2.Q1.\overline{Q0}.FIN + Q2.\overline{Q1}.Q0$$

Decodificador de salida

Se encargará de tomar la salida de la MEF y convertirla en señales de control dependiendo del estado en el que se encuentre la máquina. Estas señales de control serán para cada una de las sub-unidades de control que se encargan de darle órdenes al contador, a la unidad de almacenamiento y de brindar la salida del circuito principal.

Entradas

-**Q2**
-**Q1**
-**Q0**

Estos 3 bits representan a **Q=Q2Q1Q0** el cual es la salida de la MEF.

Salidas

-**Q2_UC**
-**Q1_UC**
-**Q0_UC**
3 bits de control para la unidad de control del contador.
-**Q1_UE**
-**Q0_UE**
2 bits de control para la unidad de control de estados.
-**Q1_UA**
-**Q0_UA**
2 bits de control para la unidad de control de la unidad de almacenamiento.
-**len**
Señal de input enable para el registro de 1 bit que utilizamos para almacenar el valor de UP/DOWN. Solo se activa en estado de carga.

Armamos la tabla de verdad dependiendo la salida de la MEF y la codificación que utilizamos para cada sub-unidad de control:

Q2	Q1	Q0	Q2_UC	Q1_UC	Q0_UC	Q1_UE	Q0_UE	Q1_UA	Q0_UA	len
0	0	0	0	0	1	1	1	0	0	1
0	0	1	1	0	0	1	0	1	0	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	1	0	1	1	1	0	0
1	0	0	X	X	X	X	X	X	X	0
1	0	1	0	1	0	0	1	1	0	0
1	1	0	0	0	0	1	1	1	0	0
1	1	1	0	1	1	1	1	0	1	0

Armando los respectivos k-maps obtenemos:

$Q2\_UC$	$\overline{Q1} \cdot \overline{Q0}$	$\overline{Q1} \cdot Q0$	$Q1 \cdot Q0$	$Q1 \cdot \overline{Q0}$
$\overline{Q2}$	0	1	0	0
$Q2$	X	0	0	0

$$Q2\_UC = \overline{Q2}\,\overline{Q1}\,Q0$$

$Q1\_UC$	$\overline{Q1} \cdot \overline{Q0}$	$\overline{Q1} \cdot Q0$	$Q1 \cdot Q0$	$Q1 \cdot \overline{Q0}$
$\overline{Q2}$	0	0	1	0
$Q2$	X	1	1	0

$$Q1\_UC = Q2\,Q0 + \overline{Q2}\,Q1\,Q0$$

$Q0\_UC$	$\overline{Q1} \cdot \overline{Q0}$	$\overline{Q1} \cdot Q0$	$Q1 \cdot Q0$	$Q1 \cdot \overline{Q0}$
$\overline{Q2}$	1	0	0	0

$Q2$	X	0	1	0

$$Q0\_UC = \overline{Q1}\,\overline{Q0} + Q2Q1Q0$$

$Q1\_UE$	$\overline{Q1} \cdot \overline{Q0}$	$\overline{Q1} \cdot Q0$	$Q1 \cdot Q0$	$Q1 \cdot \overline{Q0}$
$\overline{Q2}$	1	1	1	0
$Q2$	X	0	1	1

$$Q1\_UE = \overline{Q2}\,\overline{Q1} + Q1Q0 + Q2Q1 = \overline{Q2 \oplus Q1} + Q1Q0$$

$Q0\_UE$	$\overline{Q1} \cdot \overline{Q0}$	$\overline{Q1} \cdot Q0$	$Q1 \cdot Q0$	$Q1 \cdot \overline{Q0}$
$\overline{Q2}$	1	0	1	0
$Q2$	X	1	1	1

$$Q0\_UE = Q2 + \overline{Q1}\,\overline{Q0} + Q1Q0 = Q2 + \overline{Q1 \oplus Q0}$$

$Q1\_UA$	$\overline{Q1} \cdot \overline{Q0}$	$\overline{Q1} \cdot Q0$	$Q1 \cdot Q0$	$Q1 \cdot \overline{Q0}$
$\overline{Q2}$	0	1	1	1
$Q2$	X	1	0	1

$$Q1\_UA = \overline{Q1}\,Q0 + Q1\overline{Q0} + \overline{Q2}\,Q0 = (Q1 \oplus Q0) + \overline{Q2}\,Q0$$

$Q0\_UA$	$\overline{Q1} \cdot \overline{Q0}$	$\overline{Q1} \cdot Q0$	$Q1 \cdot Q0$	$Q1 \cdot \overline{Q0}$
$\overline{Q2}$	0	0	0	0
$Q2$	X	0	1	0

$$Q0\_UA = Q2Q1Q0$$

Y por último el **Ien,** el cual solo se activa en el estado de carga:

$$Ien = \overline{Q2}\,\overline{Q1}\,\overline{Q0}$$

Unidad de control de UA

Este bloque se encargará de dictar a la **UA (Unidad de almacenamiento)** el comportamiento que debe seguir.
Para este bloque vamos a necesitar las entradas **LOAD_I** y **LOAD_F** además de 2 bits de control (**Q=Q1Q0**) los cuales provienen del decodificador de salida. Estos bits indican que estado debe tomar la unidad de almacenamiento dependiendo de lo que se busque hacer con ella. Existen **3 variantes** de estados deseados en la unidad de almacenamiento:

-Q=00. Queremos que la unidad de almacenamiento permita escribir en sus registros de **VI y VF**. Por lo que habilitaremos el input para estos registros (**lenVI_UA e lenVF_UA**) que se cargarán cada uno dependiendo de la señal **LOAD_I** o **LOAD_F**, básicamente en este estado dejaremos pasar los valores que tomen las señales de **LOAD** hacia los inputs enables correspondientes.

-Q=01. Buscamos que la unidad de almacenamiento coloque en el bus **DATA** el **VI** del contador, para cuando se desea CONTINUAR con la cuenta. En este estado activaremos **OenVI_UA**

-Q=10. Para el último caso queremos que no entre ni salga ningún valor desde/hacía los registros de la unidad de almacenamiento. Básicamente deshabilitaríamos tanto el input como el output.

Entradas

-**LOAD_I**
-**LOAD_F**
-**Q1**
-**Q0**

Salidas

-**lenVI_UA**
-**lenVF_UA**
-**OenVI_UA**

Obtenemos así la siguiente tabla de verdad para la unidad de control de almacenamiento: (Cuando colocamos LOAD_I y LOAD_F en las filas nos referimos a un valor que tome la señal, ese valor lo usamos para replicarlo en la salida).

LOAD_I	LOAD_F	Q1	Q0	lenVI_UA	lenVF_UA	OenVI_UA
LOAD_I	LOAD_F	0	0	LOAD_I	LOAD_F	0
X	X	0	1	0	0	1
X	X	1	0	0	0	0

X	**X**	**1**	**1**	**0**	**0**	**0**

Obtenemos así:

$$IenVI\_UA = LOAD\_I\,\overline{Q1}\,\overline{Q0}$$
$$IenVF\_UA = LOAD\_F\,\overline{Q1}\,\overline{Q0}$$
$$OenVI\_UA = \overline{Q1}\,Q0$$

Unidad de control de estados

Este bloque es el encargado de brindar las salidas del circuito principal (**ERROR,FIN y PAUSA**).
Esta unidad recibe una señal de control de 2 bits del decodificador de salida. Esta entrada se interpretará para los diferentes escenarios y de esa manera poder generar las diferentes señales de salida para cada caso. Los posibles casos definiendo a los 2 bits que recibe como **Q=Q1Q0** son:
-Q = 00. Este caso indicará que estamos en estado de PAUSA.
-Q = 01. Este caso indicará que estamos en estado de FIN.
-Q = 10. Este caso indicará que estamos en estado de ERROR.
-Q = 11. Este caso indicará que no hay que activar ninguna de las señales de salida de la unidad.

Entrada

-**Q1**
-Q0
2 bits de control recibidos desde el decodificador de salida.

Salidas

-**PAUSA**
-**FIN**
-**ERROR**

De esta manera quedan definidas las salidas en base a los 2 bits de control de la siguiente manera:

$$PAUSA = \overline{Q1}\,\overline{Q0}$$
$$FIN = \overline{Q1}\,Q0$$
$$ERROR = Q1\overline{Q0}$$

Unidad de control de UC

Este bloque será el encargado de determinar lo que tiene que hacer la UC (Unidad de Contador) dependiendo de lo que se le indique.

Entradas

-**Q2**
-**Q1**
-**Q0**
Estos bits representan a **Q=Q2Q1Q0** el cual es la salida del decodificador de salida para esta unidad. Indican el estado en el que debe estar el contador.
-**b2**
-**b1**
-**b0**
Estos bits forman a **b=b2b1b0** el cual es el valor a sumar o restar a la cuenta.
-**UP/DOWN**
Indica si la cuenta es en modo de incremento o decremento de **b**.
-**LOAD_I**
Señal que se utiliza para cargar el **VI** en el regA del contador.
-**FINDET.** Es la salida del detector de fin.
-**PAUSADET.** Es la salida del detector de pausa.

Salidas

-lenVI_UC. Input enable del regA del contador.
-lenb_UC. Input enable del regB del contador.
-OenVC_UC. Output enable que permite la salida del VC hacia el bus **DATA**.
-Bout. Bus de 12 bits que se colocarán en el regB del contador.
-**RUN.** Permite que la salida del sumador se coloque a la entrada del regA (el cual se utiliza para guardar el **VC**).
-**LOAD_VI.** Permite que se coloque en la entrada del regA lo que se encuentre en el bus **DATA**.

Los diferentes casos que deseamos en la unidad de contador son:

-Caso de Q = 000. Este caso se dará tanto para el estado de **CONTANDO** como para el de **PAUSA**. Ya que para ambos casos lo que haremos será dejar pasar el valor de **b2b1b0** pero teniendo en cuenta que enviaremos 12 bits para el regB del sumador. Por lo que tendremos que obtener los 9 faltantes, cuyo valor dependerá del valor que tenga **UP/DOWN**.
Para el caso de decrementar la cuenta (**UP/DOWN = 1**) necesitaremos hacer la conversión a **CA2 de b2b1b0** y además **completar con 1`s** los bits faltantes (extensión de signo).
Para el caso de incrementar la cuenta (**UP/DOWN = 0**) necesitaremos **completar con 0`s** los bits faltantes.
Ambas situaciones las lograremos haciendo lo siguiente:
Para obtener la salida **Bout**, primero se extiende el valor de entrada de 3 bits b2b1b0 a 12 bits, completando con ceros los 9 bits más significativos. Luego, cada uno de los 12 bits

resultantes se conecta a un **multiplexor de 2 entradas** (MUX2) junto con su versión negada. La selección de cada MUX está controlada por la señal **UP/DOWN**. De esta manera, si **UP/DOWN = 0**, la salida de los MUX será el **valor original** (caso de suma), y si **UP/DOWN = 1**, la salida será el **valor negado** (primer paso para obtener el complemento a 2). Finalmente, este valor de 12 bits se conecta a un sumador junto con un carry-in igual a el valor que tenga **UP/DOWN**. Esto permite obtener directamente el valor extendido en complemento a 2 (cuando **UP/DOWN** = 1) o sin modificar (cuando **UP/DOWN** = 0).
Por último para este caso las salidas serán las siguientes:

lenVI_UC = 1.
lenb_UC = 1
OenVC_UC = 1
RUN = 1
LOAD_VI = 0
B = 000000000b2b1b0 o CA2 (000000000b2b1b0) dependiendo la señal UP/DOWN.

-Caso de Q = 001. Este caso se dará para el estado de **CARGA**. Lo que hacemos es inicializar el **regA** con el **VI** colocado en **DATA** (siempre y cuando esté activado LOAD_I). Tendremos que activar el input enable del regA y desactivar el output enable del VC.

lenVI_UC = LOAD_I
lenb_UC = 1
OenVC_UC = 0
RUN = 0
LOAD_VI = LOAD_I
B = 000000000b2b1b0 o CA2 (000000000b2b1b0) dependiendo la señal UP/DOWN.

-**Caso de Q = 010.** Este caso se dará para el estado de **DETENIDO y FIN**. Para este caso buscamos que el contador no siga sumando/restando por lo que desactivaremos **lenVI_UC** pero mantendremos activado **OenVC_UC** para tener en el bus **DATA** el último valor que se contabilizo.

lenVI_UC = 0
lenb_UC = 1
OenVC_UC = 1
RUN = 0
LOAD_VI = 0
B = 000000000b2b1b0 o CA2 (000000000b2b1b0) dependiendo la señal UP/DOWN.

-**Caso de Q = 011.** Este caso se dará para el estado de **REINICIO**. Para este caso buscamos que se pueda cargar el **VI** dentro del regA del contador y deshabilitar el **OenVC_UC.**

lenVI_UC = 1
lenb_UC = 0
OenVC_UC = 0
RUN = 0
LOAD_VI = 1

B = 000000000b2b1b0 o CA2 (000000000b2b1b0) dependiendo la señal UP/DOWN.

-**Caso de Q = 100.** Este caso se dará para el estado de **ERROR**. Para este caso buscamos que esté todo deshabilitado.

IenVI_UC = 0
Ienb_UC = 0
OenVC_UC = 0
RUN = 0
LOAD_VI = 0
B = 000000000b2b1b0 o CA2 (000000000b2b1b0) dependiendo la señal UP/DOWN.

Para la salida **Bout** ya se explicó la lógica a seguir, la cual será siempre la misma sin importar el caso. Ahora obtendremos la lógica para las demás salidas:

Para el caso de **IenVI_UC** deberemos tener en cuenta las entradas **FINDET** y **PAUSADET**. Esto se debe a que la **MEF** tarda un ciclo extra en poder ver la señal de **FIN/PAUSA** que emiten los detectores (la cual es usada para dejar de contar), provocando así que se desactive de forma tardía el IenVI_UC y es debido a esto que la cuenta llega hasta un valor extra del deseado. Para solucionar esto agarramos directamente las salidas de los detectores de fin y pausa y las recibimos en esta unidad para poder cortar de forma instantánea la cuenta.

$$IenVI\_UC = \overline{Q2}\,\overline{Q1}\,\overline{Q0}\,.\,\overline{FINDET}.\,\overline{PAUSADET} + \overline{Q2}\,\overline{Q1}\,Q0\,LOAD\_I + \overline{Q2}\,Q1\,Q0$$

$$Ienb\_UC = \overline{Q2}\,\overline{Q1}\,\overline{Q0} + \overline{Q2}\,Q1\overline{Q0} + \overline{Q2}\,\overline{Q1}\,Q0 = \overline{Q2}\,\overline{Q0} + \overline{Q2}\,\overline{Q1}\,Q0$$

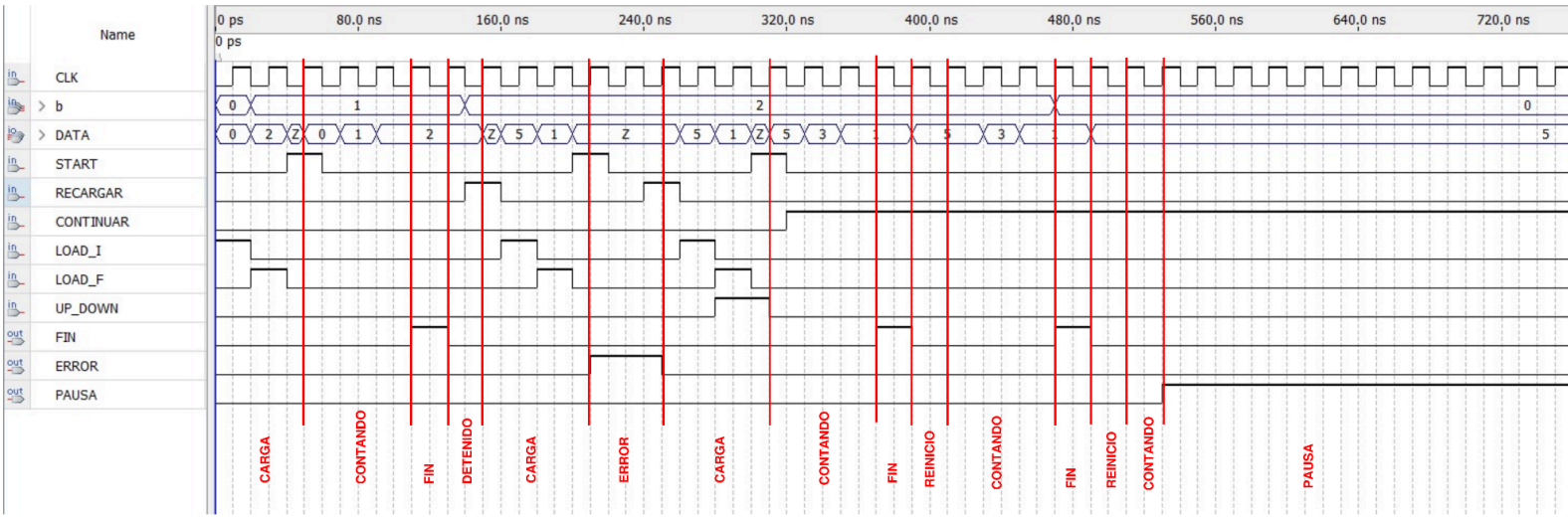
$$OenVC\_UC = \overline{Q2}\,\overline{Q1}\,\overline{Q0} + \overline{Q2}\,Q1\overline{Q0} = \overline{Q2}\,\overline{Q0}$$

$$RUN = \overline{Q2}\,\overline{Q1}\,\overline{Q0}$$

$$LOAD\_VI = \overline{Q2}\,\overline{Q1}\,Q0\,LOAD\_I + \overline{Q2}\,Q1\,Q0$$

Testeo de implementación en Quartus

A continuación dejamos una imágen de test de funcionamiento del circuito implementado en Quartus. El ejemplo hace una cuenta tanto incrementando como decrementando y pasa por todos los estados posibles.


Name
CLK
b
DATA
START
RECARGAR
CONTINUAR
LOAD_I
LOAD_F
UP_DOWN
FIN
ERROR
PAUSA
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
CARGA
CONTANDO
FIN
DETENIDO
CARGA
ERROR
CARGA
CONTANDO
FIN
REINICIO
CONTANDO
FIN
REINICIO
CONTANDO
PAUSA