TRABAJO FIN DE GRADO EN FÍSICA

TECNOLOGÍAS ACTUALES PARA DISPOSITIVOS LÓGICOS PROGRAMABLES

Junio 2013

Itsaso Artetxe Querejeta

Dirigido por

Inés del Campo Hagelstrom

ÍNDICE

INTRODUCCIÓN Y OBJETIVOS

En la primera mitad de los años noventa, la industria de la electrónica experimentó un enorme impulso en la demanda de ordenadores, móviles y dispositivos de comunicaciones. La competitividad de los fabricantes, en dispositivos con mayor funcionalidad, mayor rendimiento, menor coste, menor consumo de energía y dimensiones más pequeñas, es la responsable del gran desarrollo de la microelectrónica. Los avances en microelectrónica han permitido la miniaturización de los componentes para obtener mayores beneficios de los circuitos integrados y ampliar las posibilidades de aplicación [7].

La evolución en el desarrollo de los circuitos integrados ha dado lugar a una tecnología que perfecciona el diseño de los circuitos integrados de aplicación específica (ASICs) logrando dispositivos muy potentes que ocupan un mínimo espacio. Sin embargo, el elevado coste y los largos tiempos de desarrollo de estos dispositivos dificulta que los fabricantes sigan desarrollando nuevas tecnologías. Con el paso de los años se ha ido perfeccionando una innovadora propuesta, que sugiere la utilización de celdas programables insertadas en un circuito integrado. En base a esta idea, surgió la familia de dispositivos lógicos programables (PLDs). Actualmente, se pueden distinguir dos grandes grupos de PLDs: CPLD (*Complex Programmable Logic Device*) y FPGA (*Field Programmable Gate Array*) [7].

El objetivo principal de este trabajo es analizar las tecnologías asociadas a las dos grandes familias de dispositivos de lógica programable, los CPLDs y los FPGAs, con especial atención a la tecnología de programación y borrado empleada en cada dispositivo, así como su arquitectura interna. Otros de los objetivos del trabajo son conocer los entornos de desarrollo que los fabricantes ponen a disposición de los diseñadores y familiarizarse con las técnicas de diseño de circuitos integrados digitales actuales. Además, dichas herramientas permitirán la implementación de un circuito digital siguiendo los diferentes procesos que intervienen en el diseño: captura del esquema, simulación, síntesis, programación del dispositivo y verificación en tiempo real.

CAPÍTULO 1

CIRCUITOS INTEGRADOS

Este capítulo es una introducción a los circuitos integrados digitales. Se pretende dar a conocer las consideraciones básicas a tener en cuenta para el diseño de circuitos integrados digitales. Se establecerá una clasificación de los circuitos integrados dependiendo de su metodología de diseño y fabricación.

1.1 Introducción

Un circuito integrado, CI, es aquel que está formado por gran número de transistores en un único dispositivo (chip) semiconductor.

Los primeros circuitos integrados comerciales fueron los TTL (*transistor-transistor logic*) basados en transistores bipolares. Después, los circuitos integrados empezaron a utilizar tecnología NMOS y PMOS. Actualmente la tecnología mas utilizada es la CMOS (*complementary metal-oxide semiconductor*) basada en transistores de canal N y canal P en el mismo dispositivo. En 1965, Gordon Moore predijo correctamente que el número de transistores en un circuito integrado se duplica aproximadamente cada 18 meses [1]. La elevada densidad de transistores en circuitos integrados conlleva diversos aspectos. Por un lado, los aspectos positivos son la reducción de costes, la disminución del area de silicio, el aumento de la velocidad y la posibilidad de diseñar sistemas de gran complejidad. Por otro lado, los aspectos negativos se centran en el aumento del consumo.

1.2 Consumo

En todas las aplicaciones de circuitos digitales hay que tener en cuenta la potencia consumida. La potencia eléctrica consumida causa el calentamiento en los componentes del circuito. Este calentamiento provoca fallos en el circuito. Por tanto a la hora de diseñar un circuito se deberá tener en cuenta la potencia disipada [2].

Hay dos aspectos a tener en cuenta en el consumo de potencia. Por una parte, el consumo de potencia estática debida a que los transistores no son aislantes perfectos. Siempre se va a tener este tipo de perdida independientemente del tipo de circuito. Podemos controlarlo empleando componentes que tengan menores pérdidas o desconectando aquellas partes del circuito que durante ciertos periodos de tiempo no son usadas. Por otra parte, tenemos el consumo de potencia dinámica que se debe a la carga y descarga de las capacidades parásitas y las asociadas a las patillas de entrada/salida de los CIs cuando las señales cambian de niveles lógicos. Este tipo de consumo dependerá de la frecuencia del

cambio de señales entre niveles lógicos. Podemos controlarlo reduciendo el número y la frecuencia de las señales [3].

1.3 Áreas y empaquetamiento

Uno de los factores que un diseñador de circuitos debe tener en cuenta el coste del producto final es el área del circuito. En una oblea de silicio se fabrican múltiples CIs [4]. Luego se separan para poderlos encapsular individualmente. Desafortunadamente el proceso de fabricación no es perfecto y se producen defectos en la superficie de la oblea. Por tanto habrá circuitos integrados que no funcionen correctamente.

A la hora de crear una lámina tendremos que calcular una óptima relación entre el área y el número de circuitos integrados no dañados para que el coste sea menor.

En cuanto al empaquetamiento existe una gran amplitud de posibilidades. Para escoger la mejor hay que tener en cuenta ciertas especificaciones, por ejemplo: la potencia disipada, el número de pines necesarios o el rango de temperaturas.

1.4 Metodología del diseño

Una buena metodología permite reducir fallos y el coste final. La metodología del diseño hace referencia al proceso sistemático del diseño, la tecnología utilizada y su verificación [2].

El desarrollo de la microelectrónica ha permitido la aparición de diferentes tipos de circuitos integrados digitales [3]. Se pueden clasificar los circuitos integrados dependiendo de su metodología de diseño y fabricación. Existen tres grandes grupos: circuitos integrados estándar, circuitos integrados de aplicación específica, (ASICs, *Application Specific Integrated Circuit)* y dispositivos de lógica programable, (PLDs).

Los circuitos integrados estándar poseen características lógicas y eléctricas perfectamente definidas. Este tipo de circuitos presenta la ventaja de su reducido coste y su gran fiabilidad debido a que se fabrican en grandes series. Sin embargo, para poder realizar circuitos un tanto complejos se necesitarán un gran número de circuitos de este tipo. Además es muy común que pueda haber copias no autorizadas.

Los ASICs son circuitos que se diseñan para realizar una función específica en una aplicación concreta. Existen varios tipos de ASICs, siendo los más utilizados los de tecnología full-custom y semi-custom.

En los circuitos full-custom, el diseñador tiene control total en el diseño de los dispositivos. Son los más rápidos y eficientes, pero son los que requieren más tiempo de diseño y tienen mayores costes.

Otros son los semi-custom, dan cierta libertad de diseño, pudiendo, por ejemplo, especificar las conexiones que deben ser realizadas entre los transistores. Dentro de este grupo pueden distinguirse dos tipos: Gate Arrays y Standard Cells. En la tecnología Gate Arrays el fabricante pone a disposición del diseñador múltiples transistores tanto de canal n como de canal p. El diseñador deberá especificar las conexiones de los transistores. Las conexiones se realizan en las últimas fases del proceso de fabricación, diseñando las capas de metalizaciones. En los Standard Cells el fabricante pone a disposición del diseñador una serie de librerías de componentes digitales básicos, denominados celdas. El diseñador especifica las celdas que requiere, así como las conexiones entre ellas [9].

Por último, en los dispositivos de lógica programable el fabricante dispone de una gran cantidad de circuitos básicos en un sustrato de silicio. Estos circuitos básicos pueden utilizarse para diferentes funciones cambiando sus conexiones internas, lo cual puede hacer el diseñador. Los primeros dispositivos de lógica programable fueron los SPLDs (*Simple Programmable Logis Device*). Su estructura interna está formada por un conjunto de matrices de puertas AND y puertas OR. El avance en el desarrollo de estos dispositivos ha dado lugar a dos grupos: CPLD (*Complex Programmable Logic Device*) y FPGA (*Field Programmable Gate Array*), Figura-1.

Los CPLDs y FPGAs contienen múltiples copias de elementos lógicos (LE) o celdas. Los elementos lógicos se distribuyen en filas y columnas en el chip. Para llevar a cabo funciones más complejas los elementos lógicos pueden conectarse mediante una red de interconexión programable.

Los CPLDs tienden a tener tiempos de propagación más rápidos y más predecibles mientras que las FPGAs ofrecen una mayor densidad de integración. Aparte de estas diferencias, la diferencia fundamental entre las FPGAs y los CPLDs es su arquitectura y la tecnología de programación. Estos aspectos se analizarán en profundidad en los siguientes capítulos.

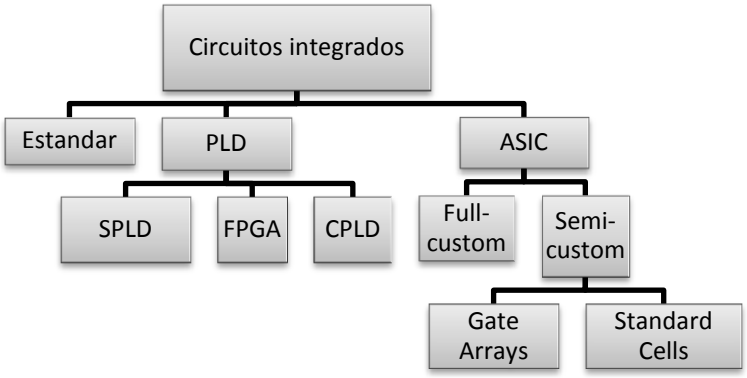


Figura-1: Clasificación de los circuitos integrados.

En la Tabla-1 se puede ver una comparativa de los distintos grupos de circuitos integrados.

Tabla-1: Comparativa de los circuitos integrados

	Estándar	ASICs	Programable
Coste del desarrollo	Bajo	Alto	Medio
Tiempo de diseño	Bajo	Alto	Medio
Espacio ocupado	Alto	Bajo	Medio
Facilidad de rediseño	Nulo	Bajo	Alto
Facilidad de copias no autorizadas	Alto	Nulo	Bajo

CAPÍTULO 2

DISPOSITIVO CPLD

En este capítulo se presentan las tecnologías de programación que se desarrollan en los CPLDs. Primero se analizarán las tecnologías de los SPLDs, debido a que los SPLDs junto con elementos de memoria dan lugar a los CPLDs .Luego se estudiaran las tecnologías EPROM y EEPROM. Por último, se dará una visión general de la arquitectura típica de un CPLD.

2.1 Tecnologías propias de dispositivos CPLDs

2.1.1 Antecedentes

La memoria PROM (Programmable Read Only Memory) fue inventada en 1956 por Wen Tsing Chow, trabajando para la American Bosch Arma Corporation en Garden City, Nueva York [5]. Aunque en un principio la PROM fue creada como una memoria también puede ser utilizada para construir circuitos lógicos programables.

Este tipo de dispositivos SPLDs (Simple Programable Logic Devices) consta de dos matrices de puertas lógicas (Figura-2): una AND, fija, y otra OR, programable.

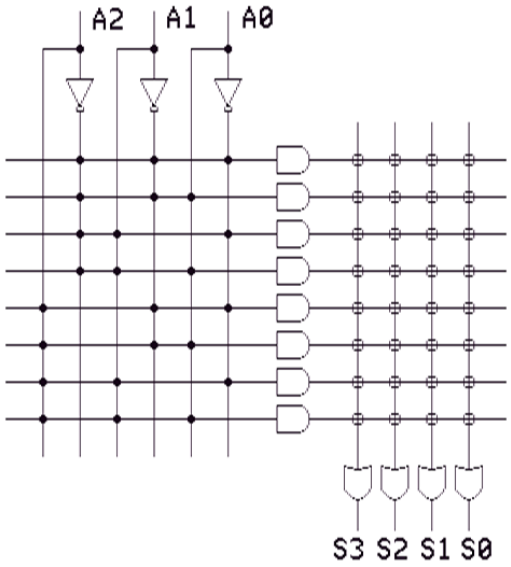


Figura-2: Matriz AND fija - OR programable.

"•" conexión fija.

"o" conexión programable.

Las interconexiones programables están formadas por fusibles (NiCr o Titanio-Tungsteno) y diodos unidos en serie como observamos en la Figura-3.

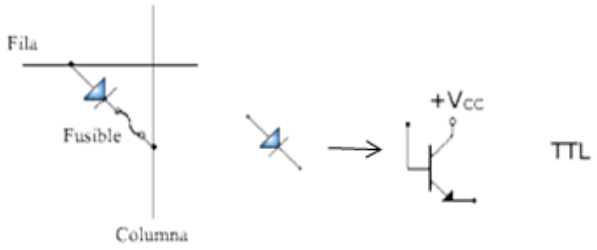


Figura-3: Elementos programables de tipo fusible.

La programación [6] consiste en destruir los fusibles en los puntos en los que se desea almacenar un cero. Esto se consigue haciendo circular una corriente suficientemente alta capaz de fundir el fusible.

Uno de los principales inconvenientes del PROM, aparte de que no son reprogramables, es que son inviables cuando se contemplan grandes números de entradas puesto que por cada variable que se anexe, el arreglo de fusibles se duplica. Con el fin de solucionar el problema planteado por la PROM se crearon las matrices PLA y PAL [7]. La estructura PLA tiene las dos matrices de puertas lógicas, AND y OR, programables. Sin embargo, la programación con las matrices PLA puede ser bastante complicada. Las matrices PAL, en cambio, son más fáciles de programar debido a que solo tienen una matriz programable, la matriz de puertas lógicas AND. Por la facilidad de programación y el reducido coste de la estructura PAL, se ha convertido en la más utilizada como matriz lógica básica de los circuitos programables. La estructura PAL combinada con elementos de memoria da lugar a los CPLDs.

2.1.2 Tecnología EPROM

i) Descripción de la celda EPROM

Los dispositivos de tecnología EPROM (*Erasable Programmable Read Only Memory*) son dispositivos no volátiles reprogramables [6], [7]. La celda de almacenamiento EPROM consiste en un único transistor NMOS con una puerta flotante, también conocido como FAMOS (Floating-gate Avalanche-injection MOS). En la Figura-4 puede verse el símbolo del elemento programable. La puerta flotante es una capa de polisilicio rodeada de óxido, aislante, que se coloca entre la puerta de control y las capas de difusión del sustrato. En la Figura-5 puede verse la estructura del transistor FAMOS.

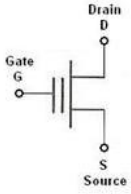


Figura-4: Elemento de matriz EPROM.

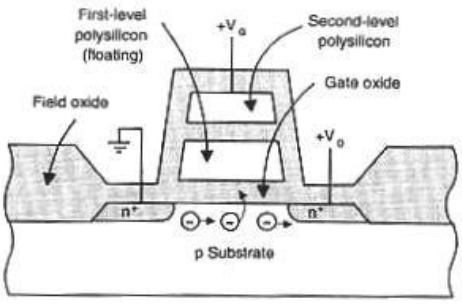


Figura-5: Sección lateral de una celda EPROM.

ii) Proceso de programación de una celda EPROM

Al igual que en los NMOS (o PMOS) de enriquecimiento, en los FAMOS hay que crear el canal, para ello hay que aplicar una tensión en la puerta, $\mathrm{V_{GS}}$, mayor que la tensión umbral. La tensión de la puerta crea un campo eléctrico que repele los huecos en la región del sustrato tipo p, situada justo debajo del óxido, dando lugar a una zona de deplexión. Si se sigue incrementando V_{GS} el potencial electrostático en la zona de debajo del óxido alcanza un valor crítico invirtiéndose la población de los portadores. La tensión $\mathrm{V_{GS}}$ a la que se produce esta situación es la tensión umbral. La concentración de electrones bajo el óxido de la puerta crea un canal que conecta el drenador y la fuente.

Si se aumenta aún más $\mathrm{V_{GS}}$ (20V aproximadamente) se aplica una tensión entre la fuente y el drenador que provoca la aceleración de los electrones de la fuente al drenador. En la Figura-6 a) pueden verse las tensiones de programación que son siempre mayores que las de operación normal. Cuando estos electrones llegan al drenador tienen una gran energía cinética y son conocidos como hot electrons. La energía que tienen es suficiente para sobrepasar la barrera de potencial y quedar inyectados en la puerta. En la Figura- 6 b) puede verse el estado del transistor cuando se eliminan las tensiones.

Afortunadamente, el proceso de carga de la puerta flotante esta autolimitada. La carga negativa que se acumula en la puerta flotante reduce la intensidad el campo eléctrico en el óxido hasta el punto en el que se convierte incapaz de acelerar más hot electrons.

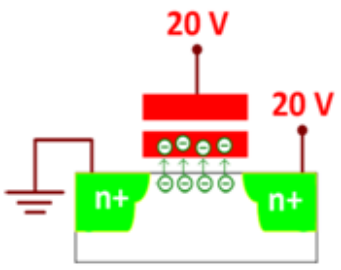


Figura-6 a) Inyección por avalancha. Se aplica una tensión y los electrones saltan a la puerta flotante.

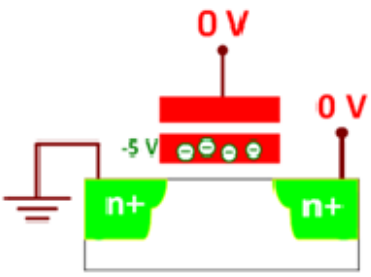


Figura-6 b) Se retira la tensión y los electrones quedan "atrapados".

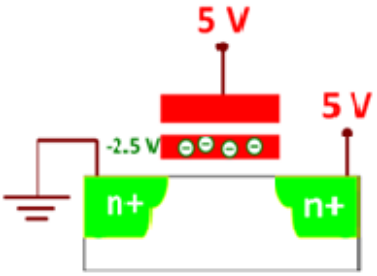


Figura-6 c) La carga negativa altera la tensión umbral del transistor.

Vamos ahora a analizar efecto de la carga negativa de la puerta flotante sobre el funcionamiento del transistor, Figura-6 c). La carga negativa atrapada en la puerta flotante hará que los electrones de la superficie del sustrato sean repelidos. Esto implica que para formar un canal la tensión positiva que tiene que ser aplicada a la puerta tendrá que ser mayor que la requerida cuando la puerta flotante no está cargada. En otras palabras, se incrementa la tensión umbral del transistor, lo que provoca que quede polarizado en corte. El transistor ya no conduce y el bit de memoria queda programado con un 0.

En definitiva, la programación de una celda EPROM altera la tensión umbral requerida para hacer conducir el transistor. El dato almacenado en una celda EPROM está representado por una tensión umbral requerida para hacer conducir el transistor. Si se quiere saber cómo está programada una celda aplicamos una tensión intermedia en la puerta del transistor como puede verse en la Figura-7. Si el transistor conduce, se interpreta como un 1, no programado y si conduce, es un 0 programado.

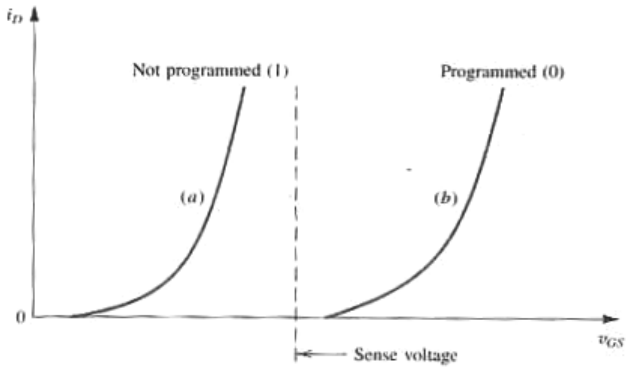


Figura-7: Se observan los cambios en i_D-v_{Gs} característico de los transistores con puerta flotante debido a la programación. Para una tensión V_{GS}(Sense voltage), el transistor no programado, conduce, está en el estado "1", en cambio, el programado no conduce, "0".

iii) Proceso de borrado de una celda EPROM

Los electrones pueden ser borrados por exposición a la luz ultravioleta (λ~2532Å), por ello los EPROM suelen venir encapsulados con una ventana de cuarzo. Los fotones son absorbidos por los electrones. Esto coloca a los electrones en un estado de alta energía, lo que les permite superar la tensión umbral (4.3eV) de la interfase de polisilicio-oxido. Los electrones son capaces de alcanzar la puerta de control o el sustrato. Se reduce la carga negativa de la puerta flotante volviendo a la tensión umbral inicial. El tiempo de borrado suelen ser 20-30 minutos [8]. El principal inconveniente de estas celdas es que utilizan un mecanismo diferente para el borrado y la escritura, y por tanto, son tecnológicamente más difíciles de optimizar. Además no pueden ser borradas en el sistema (normalmente una tarjeta de circuito impreso, PCB) ya que deben ser extraidas para aplicar la radiación UV. Para evitar estas desventajas se desarrolla la tecnología EEPROM, que se estudiará en el siguiente apartado *2.1.3 Tecnología EEPROM* .

iv) Esquema de la matriz de celdas programables

En la Figura-8 se puede ver la matriz programable de un dispositivo CPLD típico con elemento programable de tecnología EPROM.

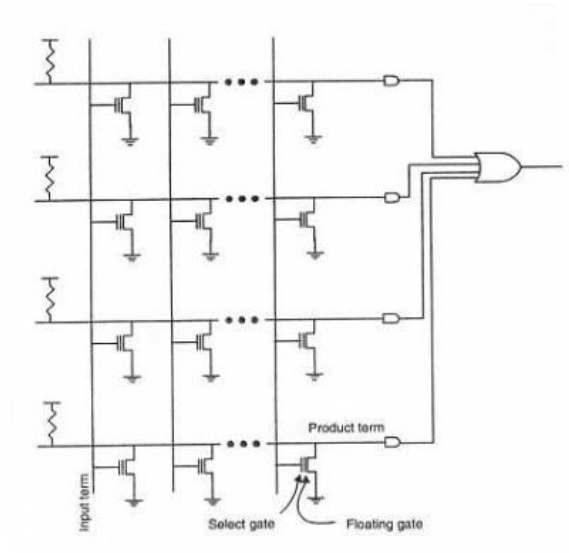


Figura-8: Matriz EPROM conectada a una puerta AND.

Empresas como Atmel[12] y Dinies [14] comercializan dispositivos EPROM.

2.1.3 Tecnología EEPROM

Una celda de EEPROM (*Electically Erasable Programmable Read Only Memory*)se puede realizar mediante dos estructuras:

- FLOTOX (*Floating Gate Tunnel Oxide*) [9]
- Celdas de polisilicio texturizado (*Textured Poly Cell*) [10]

La transferencia de electrones se realiza mediante el mecanismo de túnel de Fowler-Nordheim, que se explicará posteriormente. Empresas como ATMEL [11], RHOM semiconductor [12] o Microchip [14] comercializan CPLDs EEPROM.

2.1.3.1 Estructura FLOTOX

i) Descripción de la celda FLOTOX

La celda básica consta de dos transistores: un MOSFET y un transistor de puerta flotante con una fina capa de óxido sobre la zona del drenador (FLOTOX). En la Figura-9, puede observarse el símbolo de la celda básica.

Figura-9: Elemento programable EEPROM compuesto por un transistor NMOS y un FLOTOX

ii) Proceso de programación de una celda FLOTOX

La programación, introducción de electrones en la puerta flotante, se consigue aplicando una tensión elevada en la puerta de control, Figura-10. El drenador está conectado a tierra, los electrones se mueven del drenador a la puerta flotante, donde permanecen indefinidamente si no se aplica otro fuerte campo eléctrico inverso.

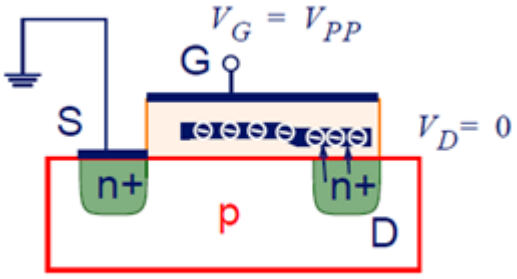


Figura-10: Representación del proceso de programación en el transistor FLOTOX .

iii) Proceso de borrado de una celda FLOTOX

El borrado, retirada de electrones de la puerta flotante, se consigue poniendo a tierra la puerta de control y a una tensión elevada el drenador como puede verse en la Figura-11.

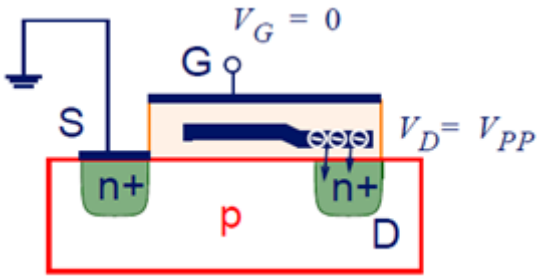


Figura-11: Representación del proceso de borrado en el transistor FLOTOX.

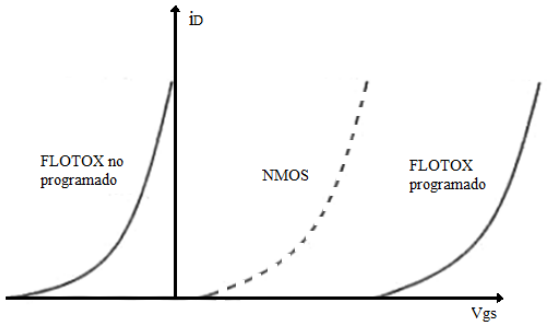


Figura-12: Funciones características del transistor FLOTOX (programado y sin programar) y del transistor NMOS.

La presencia del transistor NMOS se debe al difícil control del voltaje umbral. Se recomienda seguir el razonamiento observando la Figura-12. Se debe tener en cuenta que la tensión umbral de un transistor FLOTOX sin programar es negativa. Este hecho se debe a que en el proceso de borrado, se crean cargas positivas en la puerta flotante y por tanto, para cualquier tensión positiva el transistor conducirá y no se puede distinguir entre un "0" ($V_{GS} \approx 5V$) y un "1" ($V_{GS} \approx 10V$). Para poder distinguir los estados se añade un transistor NMOS que determina el estado lógico. Cuando el transistor FLOTOX ha sido programado la tensión umbral se ha desplazado significativamente a la parte positiva, por tanto, independientemente del estado de lógico de NMOS, el transistor FLOTOX no conducirá y el estado será "0".

iv) Esquema de la matriz de celdas programables

En la Figura-13 se puede ver la configuración de la matriz programable de un dispositivo EEPROM con un elemento programable basado en dos transistores: NMOS y FLOTOX .

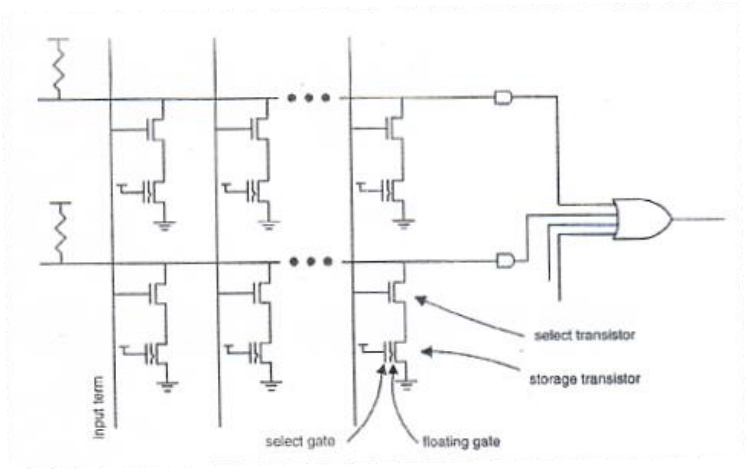


Figura-13: matriz AND de celda EEPROM. La celda EEPROM consta de dos transistores: NMOS y FLOTOX.

2.1.3.2 Celdas de polisilicio texturizado (texture poly cell)

i) Descripción de la celda de polisicio texturizado

Los dispositivos EEPROM pueden tener como celda básica distintos tipos de transistores. El más común es el explicado anteriormente, FLOTOX. A continuación se presentará otro tipo de celda con el fin de dar cuenta de las distintas opciones.

La celda de polisilicio texturizado consiste en tres capas de polisilicio solapadas y separadas por oxido aislante. Se pretende relacionar las tres capas con la estructura de la celda FLOTOX, (Figura-14):

- El polisilicio 1 y 2 equivalen al transistor de la línea de programación de los FLOTOX. El 1 hace las veces del drenador y el 2 de la puerta flotante.
- El polisilicio 3 equivale al transistor de selección.

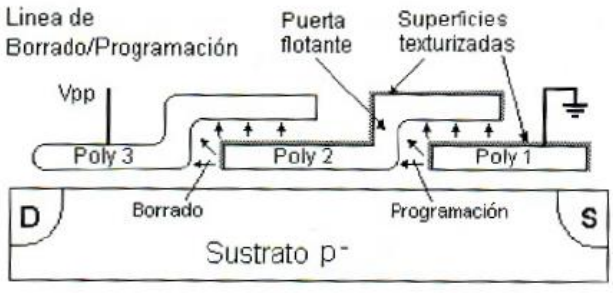


Figura-14: Celda de polisiclicio texturizado.

ii) Proceso de programación y borrado de una celda de polisilicio texturizado

La programación tiene lugar mediante el movimiento de electrones por efecto túnel Fowler-Nordheim desde el polisilicio1 (“drenador”) hacia el polisilicio2 (“puerta flotante”) y el

borrado se consigue por el movimiento de electrones del mismo modo desde el polisilicio2 hacia el polisilicio3.

2.1.3.3 Efecto Fowler-Nordhein

Las dos estructuras anteriores se basan en la inyección de electrones de baja energía hacia o desde la puerta flotante por el efecto túnel Fowler-Nordhein [16] [17]. Fowler y Nordheim demostraron que en presencia de un campo eléctrico elevado, las bandas de energía se distorsionaban. Esas distorsiones, Figura-16 y Figura-17, pueden interpretarse como barreras de potencial triangulares, y por tanto se aumenta la posibilidad de que los electrones pasen a través SiO2.

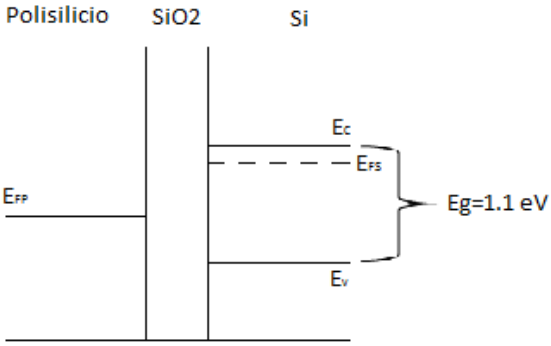


Figura-15: Estructura de bandas

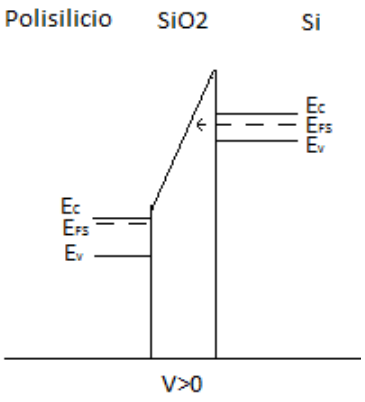


Figura-16: Estructura de bandas habiendo aplicado
Una tensión positiva. Representa un estado programado.

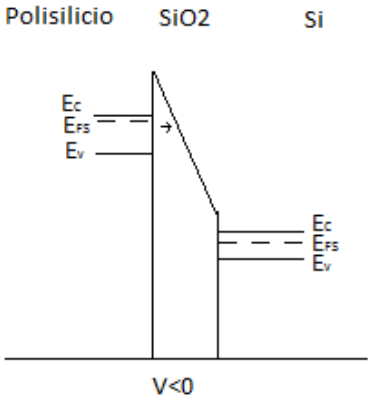


Figura-17: Estructura de bandas habiendo aplicado una tensión negativa.

.

2.2 Arquitectura típica de un CPLD

Las tecnologías de programación y borrado EPROM y EEPROM presentadas anteriormente se utilizan como elementos programables en diferentes familias de CPLDs. En la Figura-18 puede verse la arquitectura genérica de un CPLD [7]. Las celdas EPROM o EEPROM se encuentran en los bloques lógicos y en el bloque PI (*Programmable Interconnection*).

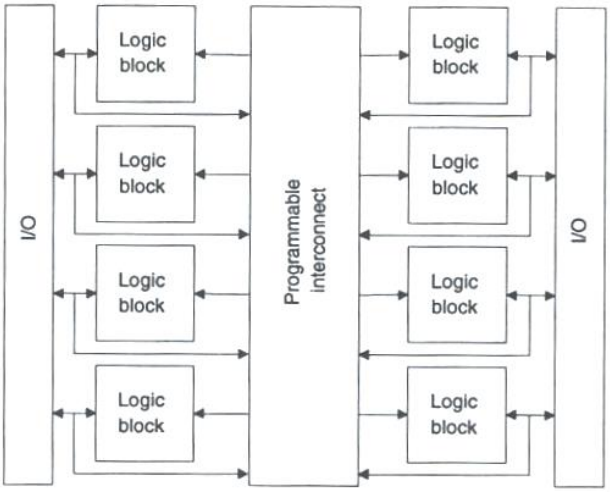


Figura-18: Arquitectura típica de un CPLD.

En la Figura-19 puede verse la estructura de un bloque lógico genérico. Consta de una estructura similar a la estructura lógica PAL, presentada en el apartado *2.1.1 Antecedentes.*

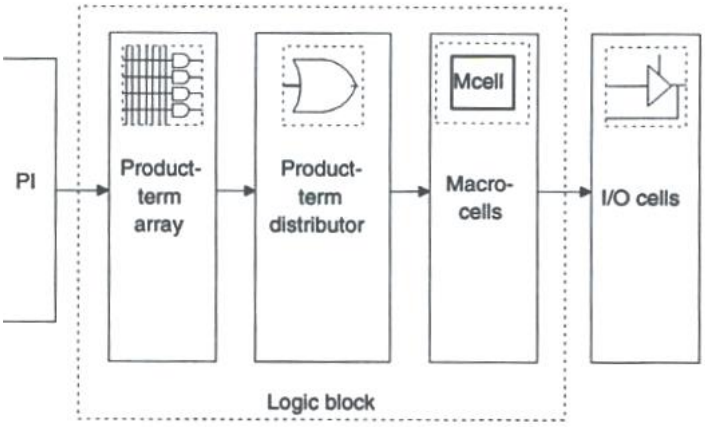


Figura-19: Bloque lógico genérico de un CPLD.

CAPÍTULO 3

DISPOSITIVO FPGA

En este capítulo se presentan las tecnologías de programación que se desarrollan en la FPGAs: antifusibles y SRAM. Además, se dará una visión general de la arquitectura típica de una FPGA actual.

3.1 Tecnologías propias de dispositivos FPGAs

Existen fundamentalmente dos tecnologías de programación en los FPGAs que son la tecnología antifusible y la tecnología SRAM.

3.1.1 Tecnología antifusible

Los antifusibles están compuestos por dos capas conductoras separas por un aislante de alta impedancia [20]. Al aplicar una tensión elevada entre las dos zonas conductoras el aislante se convierte en un conductor, creando así un "link" o zona de paso. De esta forma se reduce la impedancia. Esta situación es irreversible y por tanto no son reprogramables. Actualmente las empresas que comercializan estos dispositivos son Actel [17] y Quicklogic [18].

Estructuras:

i) Metal-metal: Las dos capas conductoras son metales mientras que el aislante es silicio amorfo como puede verse en la Figura-20, que posee una resistencia del orden de 1GΩ [19]. Al aplicar una tensión elevada entre los elementos de los metales se logra una conexión que posee una impedancia del orden de 50Ω.

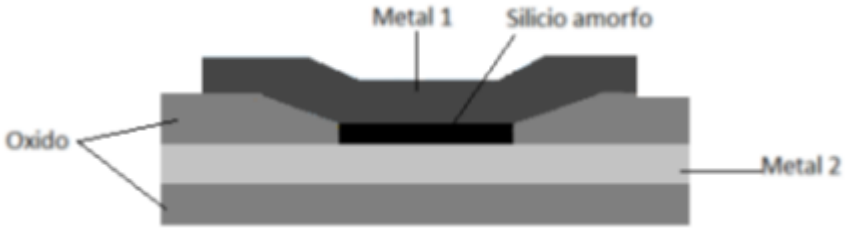


Figura-20: Sección lateral de un anti fusible metal-metal.

ii) ONO (*oxide-nitride-oxide*): Una capa conductora es de polisilicio y la otra es una capa de difusión n+, están separadas por un dieléctrico. El dieléctrico puede ser una capa de óxido fina o bien una multicapa (*oxide-nitride-oxide*). El elemento programable ONO aparece representado en la Figura-21. La resistencia de la conexión al programar el elemento varía entre 200 y 500Ω.

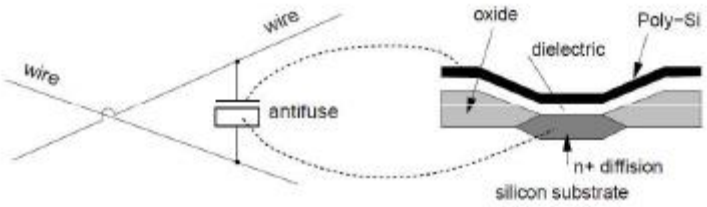


Figura-21: Símbolo electrónico y sección lateral de un antifusible ONO.

Como se observa en la Figura-22, una vez aplicada una elevada tensión la resistencia disminuye considerablemente. Si ahora se le aplica una tensión inferior, circulará corriente por el mismo y se interpretara como un cero. Si no se hubiese aplicado una tensión elevada, la resistencia sería muy alta y no circularía corriente, se interpretaría como estado lógico 1.

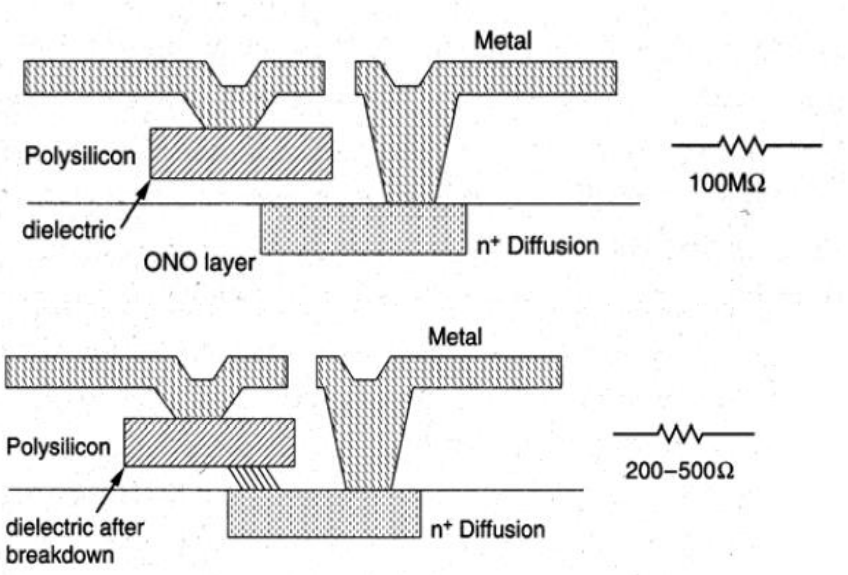


Figura-22: Antifusible ONO, antes y después de aplicar una alta tensión.

3.1.2 Tecnología SRAM

i) Descripción de la celda SRAM

La celda básica de la tecnología SRAM se compone cuatro transistores, formando un biestable [7], [20]. Un biestable es un circuito que tiene un bit de memoria, puede estar en uno de los dos estados lógicos. Además se utilizan otros dos transistores adicionales, como puede verse en la Figura-23, para controlar el acceso al biestable durante las operaciones de lectura y escritura.

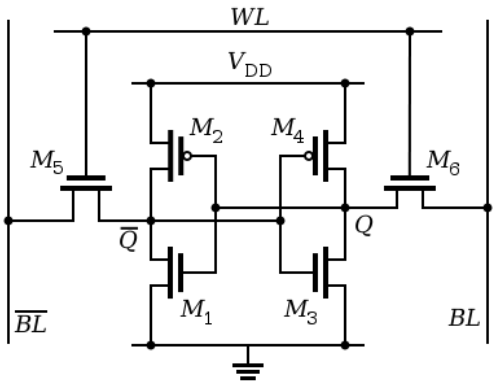


Figura-23: Celda de la memoria SRAM compuesta por seis transistores**.**

WL es la línea de selección y $\overline{BL}$ y BL son las líneas de datos. Los transistores M1-M2 y M3-M4 forman dos inversores (Figura-24 y 25) y se añaden otros dos (M5 y M6) para habilitar la lectura y escritura.

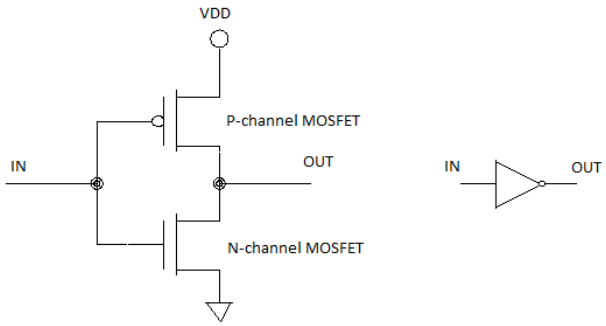


Figura-24: Inversor.

El circuito equivalente con inversores pude observarse en la Figura-26.

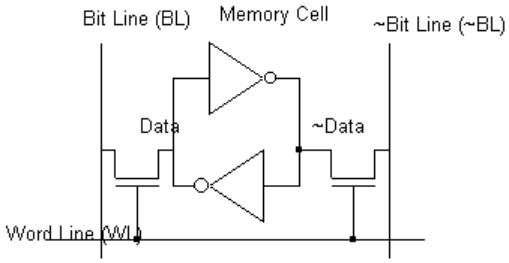


Figura-25: Circuito equivalente con inversores.

ii) Selección, lectura y escritura (programación) de la celda

Selección de celda: La selección de celda se producirá mediante una tensión V_{dd}en WL, provocando que los transistores M5 y M6 conduzcan y que por lo tanto se pueda leer o escribir en la celda.

Lectura: Tras seleccionar la celda con WL, hará que M5 y M6 conduzcan y por lo tanto los valores de los puntos Q y $\bar{Q}$ aparezcan en las líneas BL y $\overline{BL}$ respectivamente.

Escritura: Se empieza seleccionando la celda mediante WL e introduciendo el dato que queremos meter por BL. Por ejemplo supongamos que se quiere introducir un "1", para ello se activa WL e introduce un "1" por BL y un "0" por $\overline{BL}$. Un "1" en BL provoca que la puerta de M3 se polarice con 0V, es decir, está en corte y la VDS=VDD. Dicha tensión se aplica a la puerta de M1 y por lo tanto VGS1=VDD. Esto provoca que M1 conduzca y por lo tanto VDS1=0V, reforzando y memorizando el nivel inicialmente introducido de 0V. Una vez almacenado un bit su valor se mantiene en el interior del lazo de realimentación, mientras no se cambie externamente.

Por último, en la Figura-26 se puede ver la matriz de 4x4 celdas.

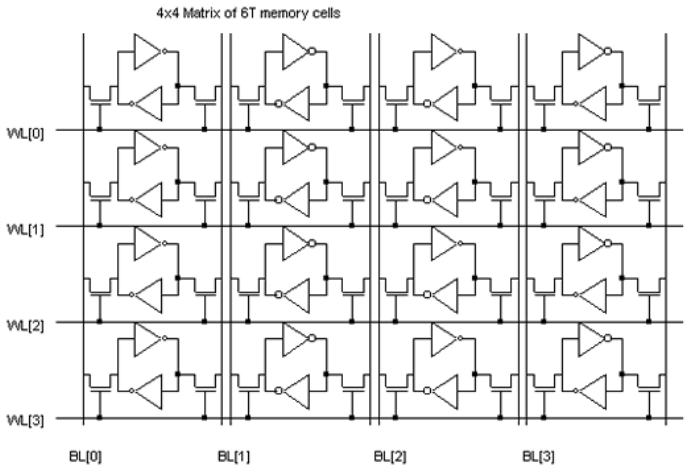


Figura-26: Matriz de celdas de tecnología SRAM.

iii) Ventajas y desventajas de la tecnología SRAM

Las principales ventajas del uso de tecnología SRAM en FPGAs son:

- El proceso de fabricación estándar permite reducir los costes.

- Son reprogramables un número ilimitado de veces, incluso cuando el FPGA está soldado en un circuito impreso PCB.

- El uso de celdas SRAM no requiere el procesamiento de circuitos integrados especiales más allá del estándar CMOS. Como resultado, los FPGAs basados en SRAM pueden utilizar la última tecnología CMOS disponible y, así, se benefician de la mayor integración, las velocidades más altas y el menor consumo de potencia dinámica.

Por otro lado, hay ciertas desventajas:

- El contenido de estos bloques se pierde cuando se deja de suministrar la energía, son volátiles. Frente a esto, es necesario el uso especial de memoria externa, no volátil, por ejemplo una memoria EPROM o FLASH.

- El tamaño, ya que una celda SRAM necesita 5 o 6 transistores y el elemento programable para interconectar las señales requiere al menos un transistor.

- Grandes retardos de ruteo. El ruteo es el camino de conexión entre los CLB.

3.2 Arquitectura típica de una FPGA

La arquitectura de un FPGA varía de un fabricante a otro y también dependiendo de la tecnología que se use. De forma genérica [7], [20] y como puede verse en la Figura-27, una FPGA está formada por bloques lógicos configurables (CLB), que se comunican entre ellos y con los terminales de entrada/salida (E/S) por medio de canales de comunicación verticales y horizontales.

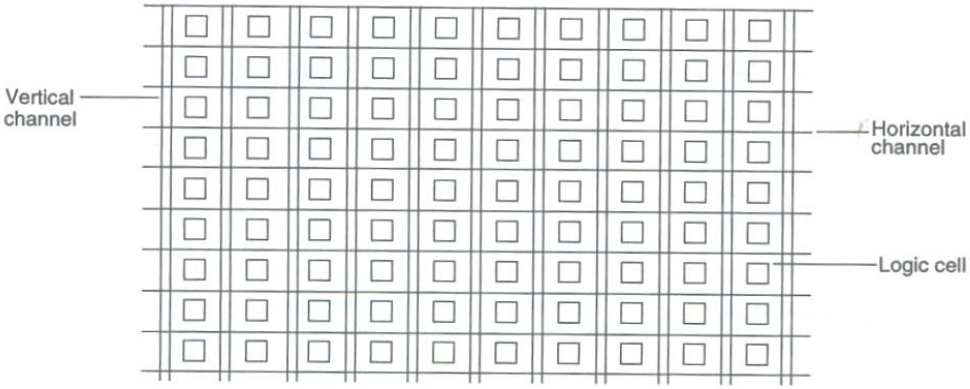


Figura-27: Arquitectura genérica de una FPGA.

Los bloques principales de la arquitectura de un FPGA son bloques lógicos programables CLB. Estos bloques son la parte lógica mayoritaria dentro de un FPGA.

Se ha seleccionado un dispositivo Spartan/Virtex del fabricante Xilinx, uno de los más utilizados por los diseñados. Su estructura se detalla en la Figura-28, donde se puede ver que un CLB está constituido por 4 Slices (SLICEL y SLCEM), red de conexión formada por Cin y Cout (carry in, carry out), conexiones a la matriz de conexiones (Switch Matrix) que proveen acceso a las rutas de conexiones generales y conexiones locales entre las SLICE del mismo CLB y CLBs vecinos.

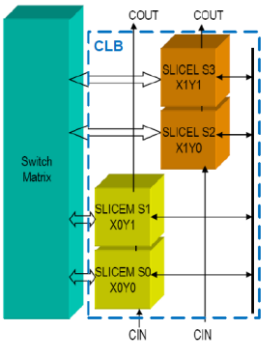


Figura-28: Esquema de un CLB y su conexión a la matriz de conexión.

En la Figura-29 puede verse la estructura interna de un SLICE. Cada SLICE está formado por tablas look-up (LUTs), lógica de acarreo y elementos de memoria.

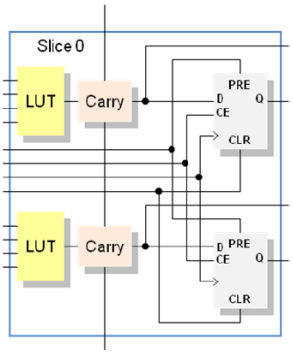
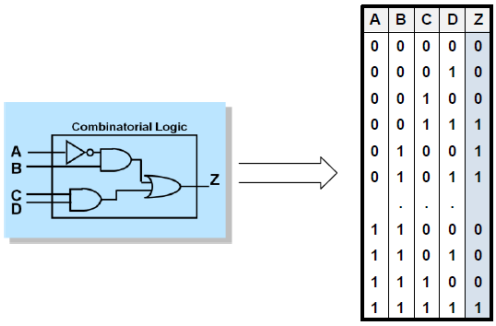


Figura-29: Vista simplificada de un SLICE de un FPGA Spartan/ FPGAVirtex. Consta de LUTs, flip-flops y lógica de acarreo.

i) Look-up Table (LUT)

En una FPGA toda la lógica combinacional se implementa utilizando tablas de búsqueda o LUT. La función lógica se almacena en una tabla de verdad de 16x1 (para las LUTs de 4 entradas). La Figura-30 detalla la similitud entre una tabla de verdad y una LUT. La columna de valores Z, son los valores que realmente se almacenan en la LUT de 16x1.



A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
	.	.	.	
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Figura-30: Implementación de una función combinacional en una LUT.

ii) Lógica de Acarreo (Carry)

El CLB tiene lógica dedicada exclusivamente para el acarreo de la suma aritmética con el objeto de mejorar el rendimiento de sumadores, contadores, comparadores y funciones lógicas relacionadas.

iii) Elementos de Almacenamiento. Flip-Flops

Los dispositivos incluyen además bloques de Entrada/Salida (Input/Output Block, IOB) y en muchas familias actuales se pueden encontrar también bloques de memorias y multiplexores embebidos.

Por último, en las FPGAs el cálculo de los retardos es complicado, pues son varios los aspectos a tener en cuenta: numerosas celdas básicas, interconexiones programables o bloques de entradas/salida se utilizan para cada señal. Así como las conexiones entre todos los bloques. Las herramientas de desarrollo para FPGAs suelen incluir un analizador de tiempos, que sirve para calcular tiempos de set-up, de clock-to-output, y sobre todo dar una estimación de la frecuencia máxima de operación.

3.3 Fabricantes

Los fabricantes más importantes son Xilinx [19] y Altera [21]. Otras empresas como ATMEL [11] y Lucent Technologies [22] también comercializan FPGAs basados en tecnología SRAM.

La empresa Xilinx ofrece al mercado dos tipos de FPGAs y que tienen la arquitectura analizada en el apartado anterior: una de bajo costo, performance media y otra de alto costo, performance alta-muy alta. La primera se denomina la familia de los FPGA Spartan, y la segunda los FPGA Virtex. A su vez, y debido a la gran competitividad del mercado, estas familias se van renovando cada dos o tres años. Tal es así que, por ejemplo, la familia Spartan fue evolucionando como Spartan, Spartan 2, Spartan 3 y la recientemente lanzada Spartan 6. Del mismo modo para la Virtex, comenzando con Virtex, Virtex-E, Virtex 2, Virtex 2Pro, Virtex 4, Virtex 5 y la reciente Virtex 6.

CAPÍTULO 4

EJEMPLO DE DISEÑO: GENERACIÓN DE SEÑALES DE VIDEO VGA MEDIANTE PLDS

En este capítulo se explicará uno de los posibles usos de los dispositivos lógicos programables, concretamente la generación de señales de video VGA. Además, Se analizarán la tarjeta de desarrollo y el dispositivo empleados. Por último, se analizaran los recursos empleados en el proceso de implementación.

4.1 Introducción

Para comprender como es posible generar señales de video con PLDs, es necesario ver en primer lugar cuales son las señales que forman parte de una señal de video VGA (*Video Graphics Array*) [25]. Una señal VGA consta de tres señales activas. Dos de estas señales son el sincronismo horizontal y el sincronismo vertical. Las otras tres señales se usan para controlar el color, se denominan RGB.

Cuando se definió el formato VGA el elemento fundamental de un monitor era el tubo de rayos catódicos (CTR). El tubo de rayos catódicos es una tecnología que permite visualizar imágenes mediante un haz de rayos catódicos constante dirigido contra una pantalla de vidrio recubierta de distintos tipo de fósforo, uno por cada uno de los colores RGB. El formato VGA sigue siendo compatible con los monitores actuales.

Para generar imágenes el haz de electrones debe barrer el área visible de la pantalla en una secuencia de líneas horizontales. La desviación del haz de electrones a las distintas posiciones de la zona visible del CTR se consigue mediante la aplicación de campos magnéticos o electrostáticos. Teniendo en cuenta que el ojo humano puede detectar frecuencias de refresco inferiores a 30 Hz la señal de video debe barrer la pantalla completa 60 veces por segundo para proporcionar una imagen adecuada. Este parámetro se denomina frecuencia de refresco. Con una frecuencia de refresco de 60Hz, sabiendo que la pantalla tiene 640x480 pixels, se puede decir que cada pixel permanece activo 40 ns aproximadamente. El proceso de refresco comienza en la parte superior izquierda y activa un pixel a la vez, de izquierda a derecha. Al finalizar la primera fila, el contador de filas se incrementa en 1 y el contador de columnas vuelve a cero. Cuando se ha barrido toda el área visible, el proceso de refresco vuelve a comenzar. Para llevar a cabo este proceso se utilizan las señales de sincronismo [21]. La señal RGB permite determinar el color de cada pixel, controlando la intensidad del haz de electrones.

La implementación se lleva a cabo en el chip EPF10K20 de la familia FLEX10K [24] que se encuentra en una placa de desarrollo UP1 del fabricante ALTERA. Se empleará también el entorno de desarrollo MAX+PLUS II [23]. Este entorno es compatible tanto con la captura de esquemáticos como con el lenguaje de descripción de hardware VHDL.

4.2 Tarjeta de desarrollo

La tarjeta UP1 de Altera, Figura-31, incorpora dos dispositivos programables de las dos grandes familias estudiadas en los capítulos 2 y 3: FLEX10K (FPGA) y MAX7000 (CPLD). Además dispone de una serie de elementos de entrada y salida como pulsadores, microinterruptores, leds y visualizadores de 7 segmentos que pueden ser utilizados para generar las entradas y salidas del diseño realizado [23].

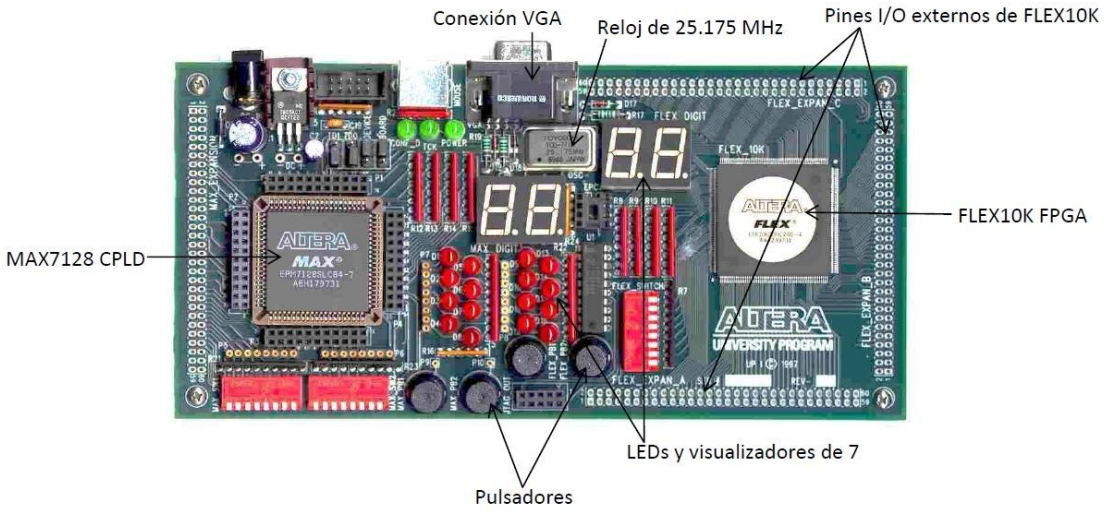


Figura-31: Tarjeta UP1 con los elementos más importantes señalados.

4.3 Dispositivo

El dispositivo empleado es de tipo FPGA y pertenece a la familia FLEX10K de ALTERA [24]. La arquitectura de la familia FLEX (Ver Figura-32) consiste en matrices lógicas y bloques de memoria (embedded array block) con una red de interconexiones horizontales y verticales en todo el dispositivo (matriz FastTrack).

El dispositivo se configura mediante la carga de la memoria de acceso aleatorio estática interna (SRAM), como se ha explicado en el Capítulo 3. En ese capítulo también se observó como la configuración se pierde cada vez que se desconecta la alimentación. En los sistemas actuales, se emplea una pequeña PROM de bajo coste para cargar automáticamente la información de la programación cuando se enciende el dispositivo.

Se puede ver en la Figura-32 un esquema del dispositivo completo. Existen múltiples bloques SRAM, llamados EAB (*Embedded Array Blocks*), cada uno de 2048 bits, cuyo formato puede ser definido individualmente como 256 palabras de 8 bits, 512 palabras de 4 bits, 1024 palabras de 2 bits, o 2048 palabras de 1 bit. Además, tenemos los elementos de entradas y salidas (IOE).

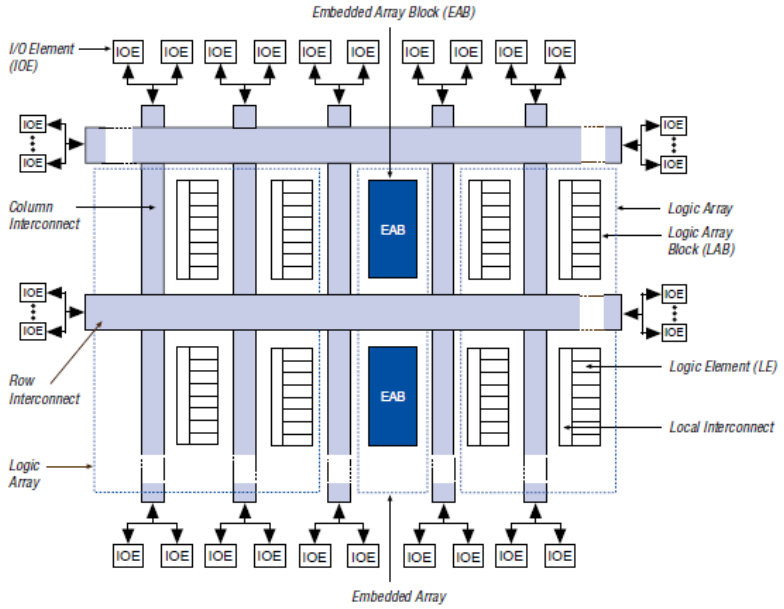


Figura-32: Diagrama de bloques de la arquitectura FLEX10K. Cada grupo de 8 LE se combina en un LAB, Los LABs se organizan en filas y columnas. Cada fila contiene un único EAB. Los LABs y EABs están interconectados por la matriz Fast Track. Las IOEs se encuentran al final de cada fila y columna del Fast Track.

La Figura-33 muestra el LAB (*logic array block*).Cada LAB está compuesto por ocho LEs. Esta agrupación facilita la implementación de ciertas funciones.

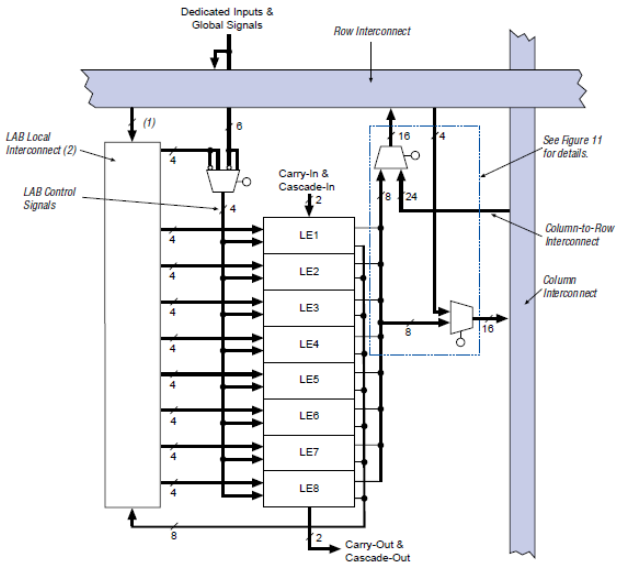


Figura-33: Bloque de matriz lógica LAB.

Por ultimo en la Figura-34 puede verse el elemento lógico que utiliza la familia FLEX.

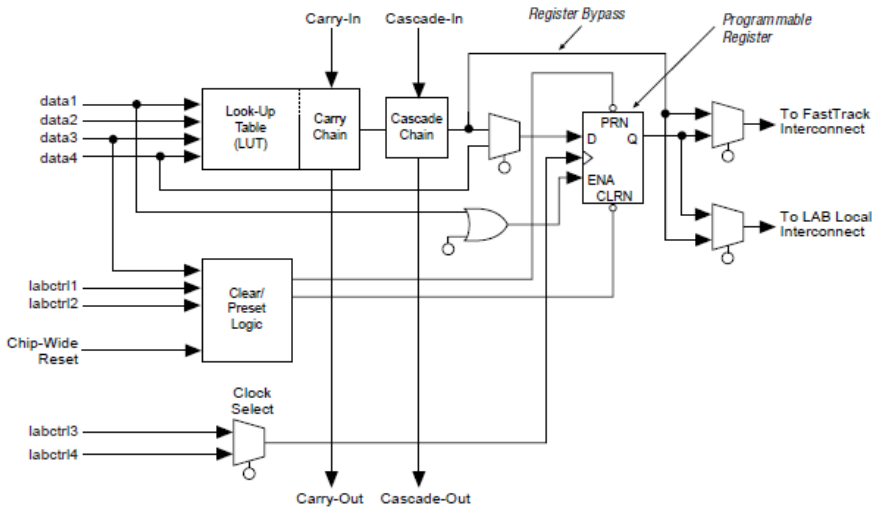


Figura-34: Elemento lógico de FLEX10K. Se analiza en el párrafo siguiente.

El elemento lógico consta de

- LUTs (*lookup table*), que permiten la implementación de funciones lógicas de 4 entradas. La función lógica se almacena en una tabla de verdad de 16x1, implementada con tecnología SRAM.
- Un flip-flop*, Programmable Register,* conectado a la señal de reloj y con dos posibles salidas: LAB y Fast Track. La matriz FastTrack se compone de un conjunto de líneas, en forma de filas y columnas, que atraviesan todo el chip, permitiendo la interconexión entre LEs de distintos LABs.
- Carry and Cascade chains permiten conectar los LE realizando distintas funciones aritméticas y lógicas. Así se facilita el cálculo de ciertas funciones de muchas variables.

4.4. Utilización de PLDs para generar señales de video VGA

Las señales de video pueden generarse utilizando CPLDs o FPGAs [25]. Las señales RGB proporcionadas por estos dispositivos digitales deben ser transformadas a señales analógicas con niveles de tensión en el rango 0.7-1V. Para ello la tarjeta UP1 incorpora un sencillo circuito compuesto por una resistencia y un diodo. Así se consiguen dos niveles para cada señal RGB, en total ocho colores. Las señales de sincronismo son señales digitales, compatibles con el formato VGA, y se generan a partir del subcircuito VGA_SYNC (escrito en lenguaje VHDL).

A continuación, se lleva a cabo, como ejemplo de diseño, la visualización de imágenes desde una memoria RAM/ROM [25]. Como paso previo se generará una señal de ajuste que consiste en la visualización de una serie de barra de colores.

4.4.1 Visualización de barras de colores

El primer paso es la generación de la señal de ajuste. Para ello se emplea el subcircuito VGA_SYNC. Se puede ver en la Figura-35 las entradas: RGB y clock_25MHz, y las salidas: RGB , las señales de sincronismo (*horiz_sync_out* y *vert_sync_out*) y apuntadores (*pixel_row* y *pixel_column*). Los apuntadores indican en todo momento la dirección de un pixel y se emplean para generar el color de entrada del pixel.

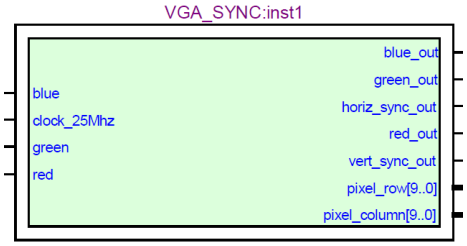


Figura-35: Representación esquemática del módulo VGA_SYNC.

Utilizando el subcircuito VGA_SYNC se han generado secuencias de barras verticales de colores que permiten visualizar los ocho colores (Figura-36) que se producen mediante las señales RGB. Se han utilizado 32 pixels de anchura por cada barra. Con ayuda del entorno de desarrollo se ha generado el "bitstream" que se carga en las celdas SRAM del dispositivo. Una vez configurada la FPGA se visualizaran correctamente las barras de colores de la Figura-36.

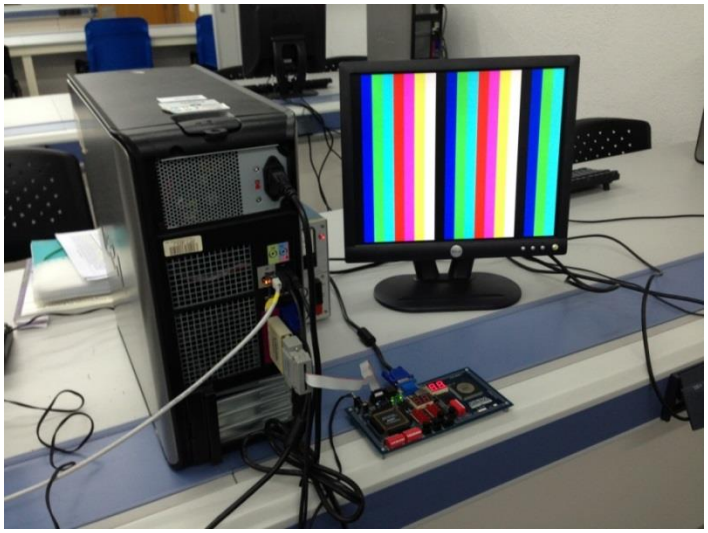


Figura-36: Generación de la señal de ajuste

4.4.2 Visualización de caracteres

Como se ha visto anteriormente, las imágenes de video requieren el refresco constante de pixels. Por este motivo es necesario utilizar registros y memorias RAM o ROM para almacenar las imágenes y acceder a ellas en cada barrido. Dicha memoria se implementará en el mismo PLD que el circuito de sincronismo. Para ello se utilizará el

subcircuito CHAR_ROM. Este subcircuito genera una imagen a partir de fichero de texto donde se definen los caracteres que se quieren visualizar (Figura-37).

```
Depth = 512;
Width = 8;
Address_radix = oct;
Data_radix = bin;
% Character Generator ROM Data %
Content
  Begin
000  : 00111100 ; %    ****     %
001  : 01100110 ; %   **  **    %
002  : 01101110 ; %   ** ***    %
003  : 01101110 ; %   ** ***    %
004  : 01100000 ; %   **        %
005  : 01100010 ; %   **   *    %
006  : 00111100 ; %    ****     %
007  : 00000000 ; %             %

010  : 00011000 ; %     **      %
011  : 00111100 ; %    ****     %
012  : 01100110 ; %   **  **    %
013  : 01111110 ; %   ******    %
014  : 01100110 ; %   **  **    %
015  : 01100110 ; %   **  **    %
016  : 01100110 ; %   **  **    %
017  : 00000000 ; %             %
```

Figura-37: Extracto del fichero “tcgrom.mif”, donde se definen los caracteres que se quieren representar.

Para cargar los patrones en una ROM durante la configuración de un PLD se utilizan ficheros de inicialización (“tcgrom.mif”), en la Figura-37 se ve una parte del fichero. Se ha utilizado una técnica de acceso a los datos basada en dos niveles de memoria. El primer nivel de la ROM contiene la dirección de inicio de cada carácter y el segundo nivel contiene los patrones de bits correspondientes.

A continuación, utilizando los bloques VGA_SYNC y CHAR_ROM se diseña un sistema que emite una secuencia de caracteres blancos sobre fondo rojo (Figura-38). El tamaño de los caracteres es de 16x16 pixels.

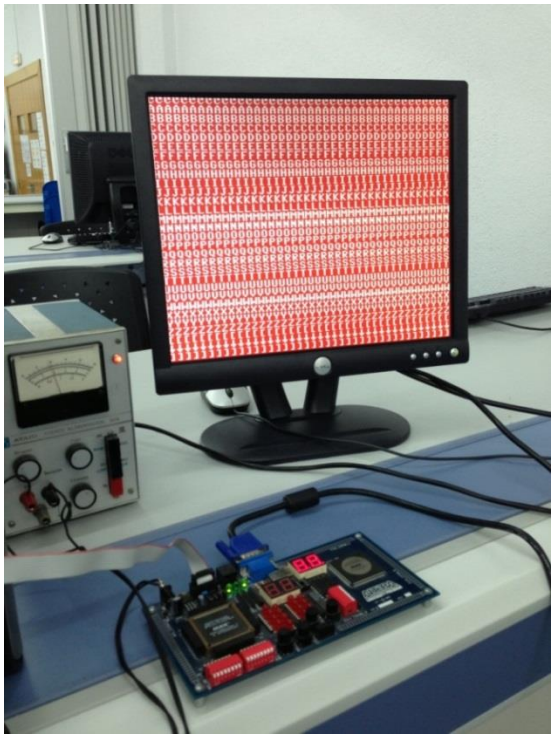


Figura-38: Visualización de caracteres blancos en fondo rojo.

En las Figura-39 se muestra el esquemático del diseño.

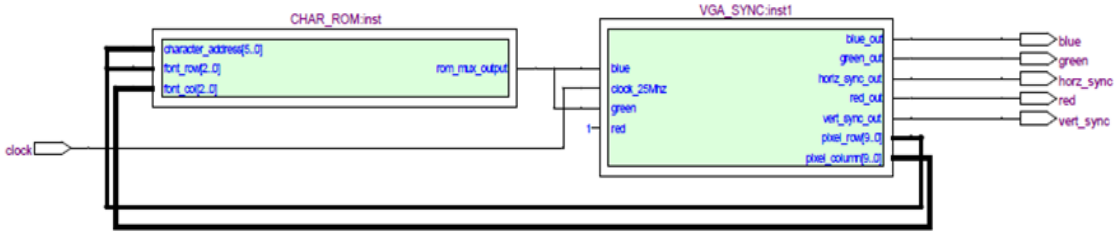


Figura-39: esquemático del diseño de la visualización de caracteres a nivel RTL (*Register Transfer Logic*).

Podemos ver la configuración de CHAR_ROM tanto a nivel tecnológico como a nivel de transferencia de registros (RTL).

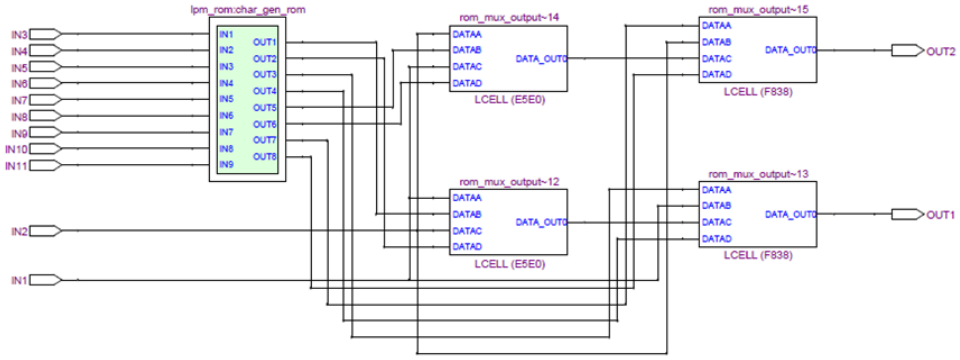


Figura-40: Implementación del módulo CHAR_ROM mediante primitivas de la tecnología.

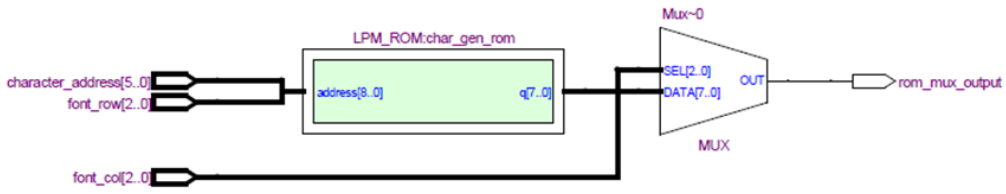


Figura-41: Implementación del módulo a nivel RTL.

En la Figura-40 se ven cuatro LCs (LUTs). La combinación de ellos da lugar a un multiplexor, como se puede ver la Figura-41.

La Figura-41 representa el esquema tecnológico del módulo VGA_SYNC. Se ven múltiples celdas lógicas y dos módulos, un sumador (“lpm_add_sub:add_rtl_1”) y un contador (“lpm_counterv_count_rtl_0”).

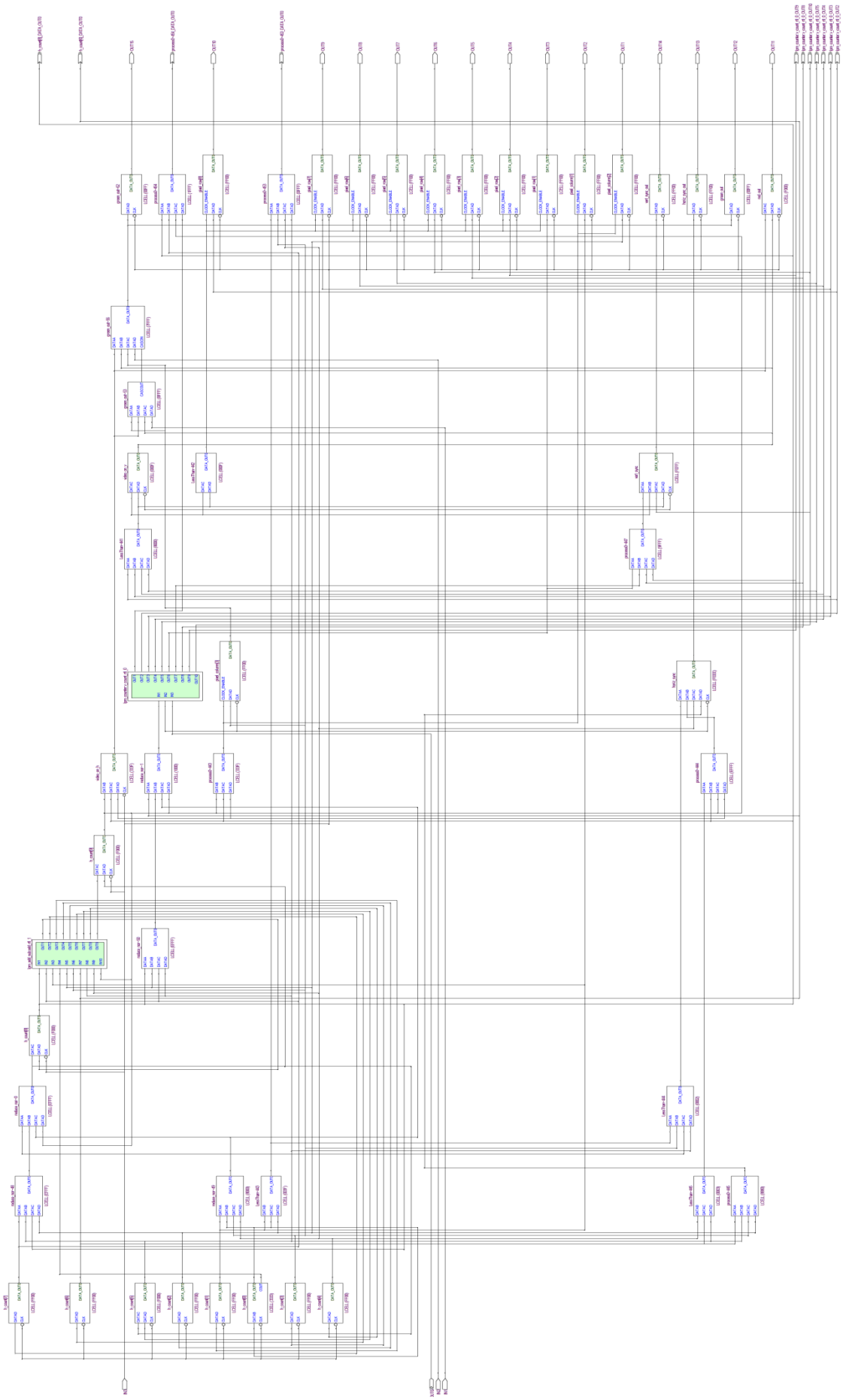


Figura-42: Esquema del módulo VGA_SYNC en función de las primitivas (tecnológico).

A continuación, Figura-43, se muestra la tecnología empleada en el contador, donde puede verse que consta de 10 LC (LUT+RE).

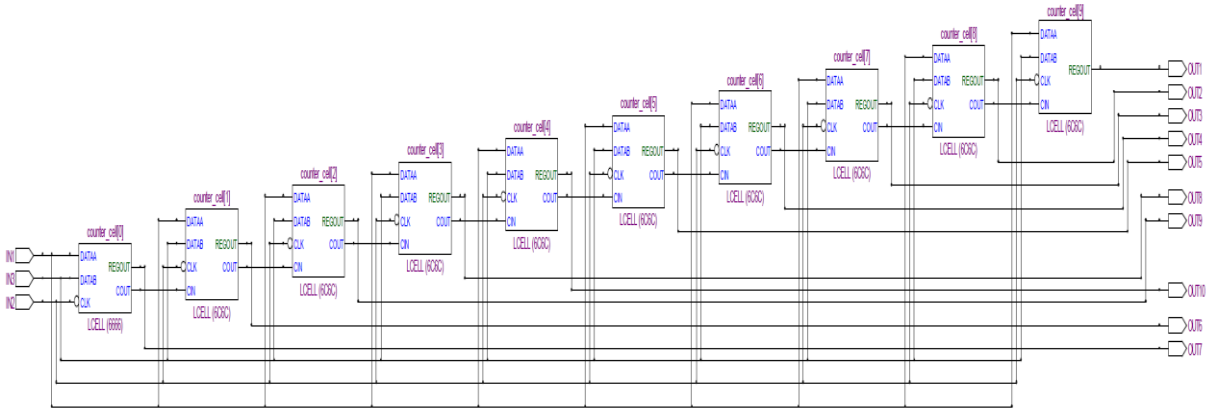


Figura-43: tecnología empleada en el contador.

El sumador, Figura-44, consta de una celda lógica (LUT con acarreo) y otro módulo "a_csnbuffer:result_node".

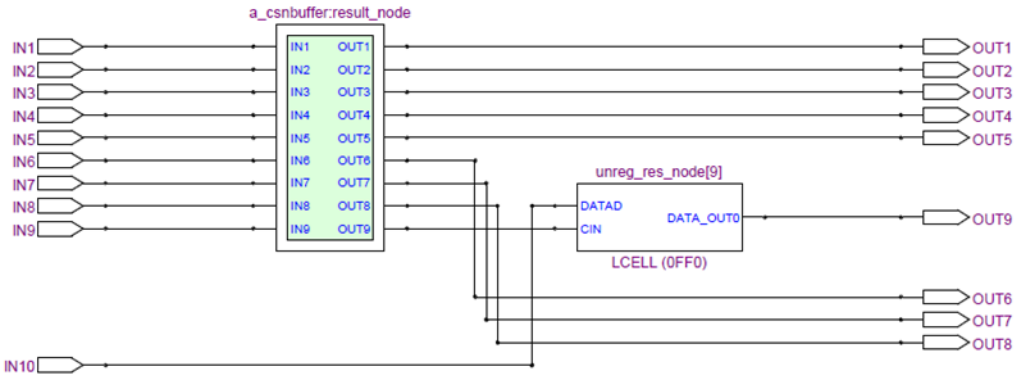


Figura-44: tecnología empleada en el sumador.

Por último, se muestra en la Figura-45, la tecnología empleada en el último bloque, "a_csnbuffer:result_node", que consta de 8 LCs (LUTs con carry).

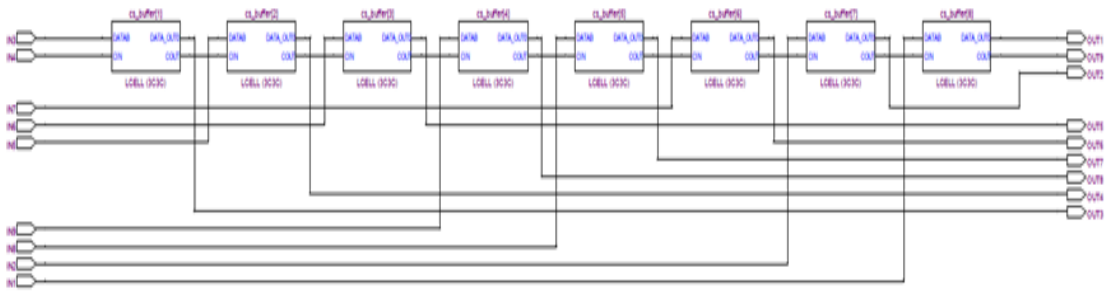


Figura-45: tecnología empleada en “a_csnbuffer:result_node”.

4.4.3 Recursos utilizados y frecuencia de operación

Por último, se muestran una serie de tablas donde se pueden observar todos los recursos empleados.

Tabla-2: Analysis and Synthesis Summary

Quartus II Version	5.0 Build 148 04/26/2005 SJ Full Version
Revision Name	proyecto
Top-level Entity Name	proyecto
Family	FLEX10K
Device	EPF10K20RC240-4
Timing Models	Final
Total logic elements	76 / 1,152 (6 %)
Total pins	6 / 189 (3 %)
Total memory bits	4,096 / 12,288 (33 %)

Tabla-3: Fitter Resource Usage Summary

Resource	Usage
Registers	40 / 1,152 (3 %)
Total LABs	0 / 144 (0 %)
Logic elements in carry chains	20
User inserted logic elements	0
I/O pins	6 / 189 (3 %)
-- Clock pins	1
-- Dedicated input pins	2 / 4 (50 %)
Global signals	1
EABs	2 / 6 (33 %)
Total memory bits	4,096 / 12,288 (33 %)
Total RAM block bits	4,096 / 12,288 (33 %)
Maximum fan-out node	clock
Maximum fan-out	40
Total fan-out	316
Average fan-out	3.51

En la Tabla-2 podemos ver el dispositivo que se ha empleado así como el número total de celdas lógicas y pines. En la Tabla-3 podemos ver que se emplean 2 EABs (Embedded Array Blocks), son memorias RAM de 2048 bits. Como se están usando dos el número total de bits disponibles son 4096 bits, como indica la Tabla-3.

Tabla-4: Analysis & Synthesis Resource. Recursos Lógicos empleados tanto en el diseño en general como en cada módulo.

Compilation Hierarchy Node	Logic Cells	LC Registers	Memory Bits	Pins	LUT-Only LCs	Register-Only LCs	LUT/Register LCs	Carry Chain LCs
|proyecto	75 (1)	39	4096	6	36 (1)	20 (0)	19 (0)	20 (0)
|Char_ROM:inst|	4 (4)	0	4096	0	4 (4)	0 (0)	0 (0)	0 (0)
|lpm_rom:char_gen_rom|	0 (0)	0	4096	0	0 (0)	0 (0)	0 (0)	0 (0)
|altrom:srom|	0 (0)	0	4096	0	0 (0)	0 (0)	0 (0)	0 (0)
|VGA_SYNC:inst1|	70 (51)	39	0	0	31 (22)	20 (20)	19 (9)	20 (1)
|lpm_add_sub:add_rtl_1|	9 (0)	0	0	0	9 (0)	0 (0)	0 (0)	9 (0)
|addcore:adder|	9 (1)	0	0	0	9 (1)	0 (0)	0 (0)	9 (1)
|a_csnbuffer:result_node|	8 (8)	0	0	0	8 (8)	0 (0)	0 (0)	8 (8)
|lpm_counter:v_count_rtl_0|	10 (0)	10	0	0	0 (0)	0 (0)	10 (0)	10 (0)
|alt_counter_f10ke:wysi_counter|	10 (10)	10	0	0	0 (0)	0 (0)	10 (10)	10 (10)

En la Tabla-4 se muestran los recursos lógicos que se han empleado en el diseño y que se han observado en el apartado *4.4.2 Visualización de caracteres.* En la tabla se puede ver que

hay tres tipos de celdas lógicas: LUT, REGISTER y LUT-REGISTER. Además, era de esperar que las dos EABs se encontrasen en CHAR_ROM de acuerdo con lo estudiado en el apartado *4.3 Dispositivo*.

Como se ha comentado en el final del apartado *3.2 Arquitectura típica de una FPGA,* es importante tener en cuenta la frecuencia máxima de operación del dispositivo. Se ha empleado una frecuencia de 25 MHz pero el análisis nos dice que se podría haber empleado una de 28.41 MHz (Tabla-5).

Tabla-5: Dato de frecuencia máxima de operación extraída de Timing Analyzer Summary

Type	Slack	Required Time	Actual Time
Worst-case tco	N/A	None	15.500 ns
Clock Setup: 'clock'	N/A	None	28.41 MHz (period = 35.200 ns)

CONCLUSIONES

Las primeras conclusiones a destacar son las ventajas de usar PLDs frente a ASICs. En los PLDs el usuario lleva a cabo el ciclo completo de desarrollo con herramientas de ayuda que tiene a su disposición, como se ha podido comprobar en la realización del ejemplo de diseño. Además, los fabricantes de PLDs ofrecen la posibilidad, cuando se trata de un gran volumen de mercado, de implementar el diseño en los PLDs convirtiéndolos en CIs no programables. De este modo se reducen considerablemente los costes en comparación con ASIC.

Para continuar, se recuerda que la finalidad de este trabajo es el estudio de las diferentes tecnologías de programación. Por ello, se introduce la Tabla-6, que resume las características principales de cada tecnología.

Tabla-6 Resumen de las características de las diferentes tecnologías de programación

Dispositivo (PLD)	Tecnología del elemento programable	Elemento programable	Símbolo	Método de programación	Reprogramable	Método de borrado	Volátil	Fabricantes
SPLD	Bipolar	Fusible en serie con diodo		Eléctrica en corriente	No	-	No	ATMEL, RHOM semiconductor
CPLD	EPROM	Transistor FAMOS		Eléctrica en tensión	Si	Luz ultravioleta	No	ATMEL, Dinies
CPLD	EEPROM	Transistor FLOTOX		Eléctrica en tensión	Si	Eléctrico	No	Microchip, ATMEL, RHOM semiconductor
FPGA	Antifusible	Antifusible		Eléctrica en tensión	No	-	No	Actel, Quicklogic
FPGA	SRAM	Celda SRAM		Eléctrica en tensión	Si	Eléctrico	Si	Altera, Xilinx

Tras haber analizado todas las tecnologías, se considera preferible el uso de un tipo de dispositivo para cada aplicación. Por una parte, los CPLD por su reducido tamaño y su bajo consumo se emplean a menudo en aplicaciones portátiles junto con pequeñas baterías. Ejemplo de ello es la “ropa inteligente”, es decir, prendas que constan de sensores biológicos (humedad), sensores de movimiento (zapatillas de deportistas) y de vibraciones (pulsaciones), entre otros. La información proveniente de estos sensores se procesa con la ayuda de CPLDs. Cabe destacar que los CPLDs de tecnología EEPROM tienen mayor uso que los EPROM debido al método de borrado.

Por otra parte, el rango de aplicaciones de las FPGAs es muy amplio debido a la gran variedad de familias y tamaños. Esto se ha podido comprobar en el análisis de un dispositivo Spartan/Virtex del fabricante Xilinx (*3.2 Arquitectura típica de una FPGA*) y de un dispositivo de la familia FLEX de Altera (*4.3 Dispositivo*). Distinguiendo el tipo de aplicaciones según la tecnología empleada, la principal aplicación de las FPGAs de tecnología SRAM es el

procesamiento de señales, debido a su alta frecuencia de trabajo. El procesamiento de señales mediante las FPGAs se emplea, por ejemplo, en sistemas de visión artificial (video-vigilancia, robots), en sistemas de imágenes médicas o en codificación y cifrado. Las prestaciones de las FPGAs de tecnología SRAM también permiten su uso en el control industrial. Las utilidades de las FPGAs, tanto por el procesamiento de señal, como por los sistemas de control, hacen que estos dispositivos tengan un papel relevante en el sector del automóvil. Sin embargo, en situaciones donde sea posible el borrado total o parcial de la configuración del dispositivo, puede ser preferible emplear las FPGAs de tecnología antifusible. Por este motivo las FPGAs basadas en antifusibles tienen gran utilidad en los sectores espacial (radioastronomía) y militar (cifrado).

En relación con el ejemplo implementado se ha visto la gran ayuda que supone el uso de los entornos de desarrollo y los leguajes de descripción del hardware para el diseño de circuitos integrados. Además, el entorno de desarrollo permite la visualización esquemática de los recursos empleados. En el diseño se ha usado un porcentaje muy reducido de recursos (Tablas 2 y 3). Teniendo en cuenta que el dispositivo es de los más pequeños de la familia FLEX, se pone de manifiesto la facilidad de integrar gran cantidad de funciones.

Finalmente, en cuanto al futuro de los dispositivos lógicos programables, teniendo en cuenta la rápida evolución de esta tecnología es de esperar que se puedan desarrollar sistemas completos mediante SOPCs (*System on a Programmable Chip Builder*). Los SOPCs son la última evolución de las FPGAs, que permiten integrar junto con la lógica, uno o más microprocesadores (embebidos o configurados por el usuario), bloques de memoria y periféricos de entrada /salida, en un único chip.

BIBLIOGRAFÍA

[1] Gordon Moore, in *Cramming more components onto integrated circuits,* Electronics Magazine Vol. 38, No. 8, (April 19, 1965).

[2] Peter J. Ashenden, *Digital Design:An Embedded Systems Approach Using VHDL* (Morgan Kaufmann, USA, 2008).

[3] Anant Agarwal and Jeffrey H. Lang, *Foundations of Analog and Digital Electronic Circuits* (Morgan Kufmann,Elsevier,San Francisco , 2005), Chap. 11.

[4] Stephen A. Campbell, The Science and Engineering of Microelectronic Fabrication, (Oxford University Press, New York, 2001).

[5] Website: *es.wikipedia.org/wiki/Memoria_PROM*.

[6] Adel S. Sedra and Kenneth C. Smith, *Microelectronic Circuits* (Oxford University Press, New York, 2010).

[7] Kevin Skahill, *VHDL for programmable logic* (Addison-Esley, Massachusetts, 1996).

[8] Ron D. Katznelson and Frohman-Bentchkowsky, in *An Erase Model for FAMOS EPROM Devices,* IEEE, Vol. ED-27, No.9, pp. 1744-1752, (1980).

[9] Jose L. Martín Gonzalez, *Electronica digital*, (Delta publicaciones, Madrid, 2007).

[10] Volnei A. Pedroni, *Digital electronics and design with VHDL* (Elsevier, Amsterdam, 2008).

[11] Website: www.atmel.com

[12] Website: www.rohm.com

[13] Website: www.dinies.com

[14] Website: www.microchip.com

[15] Enrique Mandado Pérez and Yago Mandado Rodríguez, *Sistemas digitales electrónicos* (Marcombo, Barcelona, 2008)

[16] R. H. Fowler and L. Nordheim, in Electron Emission in Intense Electric Fields, Proc. R. Soc. Lond. A 1928, 119 (1928).

[17] Website: www.actel.com

[18] Website: www.quicklogic.com/

[19] Website: www.xilinx.com

[20] Cristian Sisterna, *Field programmable gate arrays(FPGAS),* Universidad Nacional de San Juan. Website: dea.unsj.edu.ar/sisdig2/

[21] Website: www.altera.com

[22] Website: http://www.alcatel-lucent.com/

[23] Altera, *University Program Design Laboratory Package*, User guide, ver. 1.02,(November 1999). Website: http://elm.eeng.dcu.ie/~ee404/UP1Setup/UP1_user_guide.pdf

[24] Altera, *FLEX 10K Embedded Programmable Logic Device Family*, Data Sheet, ver.42, (January, 2003). Website: http://www.altera.com/literature/ds/archives/dsf10k.pdf

[25] James O. Hamblen and Michael D. Furman, *Rapid Prototyping of Digital Systems* (Kluwer Academic, Boston, 2000).