

Diseño Digital Avanzado

Unidad 3 - Mapeo de Arq. Dedicadas

Dr. Ariel L. Pola

ariel.pola@mi.unc.edu.ar

September 28, 2024

Tabla de Contenidos

1. Contenidos Temáticos
2. Sistemas Discretos de Tiempo Real
3. Sistema de Hardware Digital Síncrono
4. Redes de Procesamiento tipo Kahn
5. Métodos para Representar Sistemas DSP
6. Medidas de Rendimiento
7. Arquitecturas Dedicadas
8. Introducción a FPGA

Contenidos Temáticos



Presentación del Curso

Contenidos Temáticos

Unidad 3 Mapeo de Arquitecturas Dedicadas

- Sistemas discretos de tiempo real.
- Sistemas síncronos.
- Redes de procesamiento tipo Kahn para modelar aplicaciones de streaming.
- Métodos para representar Sistemas DSP.
- Diagramas de bloque.
- Gráficos de flujo de señal.
- Diagramas de flujo de datos.
- Single, multi-rate y homogeneous SDFGs.
- Gráficos de control de flujo.
- Maquinas de estado finitas.
- Medidas de rendimiento: Periodo de iteración, periodo de muestro y velocidad de transmisión, latencia, disipación de potencia.
- Arquitecturas dedicadas.

Sistemas Discretos de Tiempo Real



Sistemas Discretos de Tiempo Real

Concepto

- Un sistema de tiempo real discreto está limitado por la velocidad de muestreo de la señal de entrada y la cantidad de muestras necesarias para producir muestras de salida a una velocidad específica.
- En un receptor de comunicación digital, la señal de entrada en tiempo real puede ser voz modulada, datos o vídeo y la salida es la señal demodulada respectiva.
- La señal analógica se convierte en una señal de tiempo discreto utilizando un conversor analógico a digital (*Analog-to-Digital Converter - ADC*).
- En muchos diseños esta señal discreta en tiempo real se procesa en bloques de tamaño fijo.
- El tiempo que se tarda en adquirir un bloque de datos y el tiempo necesario para procesar este representan una dura restricción en el diseño.
- El diseño debe ser lo suficientemente rápido para completar su procesamiento antes de que el próximo bloque de datos esté listo para su turno de procesamiento.

Sistemas Discretos de Tiempo Real

Concepto

- El tamaño del bloque también es importante en muchas aplicaciones, ya que provoca un retraso inherente.
 - Un bloque grande aumenta los requerimientos de retardo y memoria.
 - Un bloque más pequeño aumenta la sobrecarga de procesamiento.
- En muchas aplicaciones el tamaño mínimo del bloque está restringido por el algoritmo seleccionado.
- En un transmisor de comunicación, una señal en tiempo real es digitalizada y luego procesada por GPPs, ASICs o FPGAs, o cualquier combinación de estos.
- Sistemas de procesamiento
 - **Tasa Única (Single-Rate)**: El número de muestras por segundo a la entrada y salida del sistema es el mismo, y el número de muestras por segundo no cambia cuando las muestras se mueven de un bloque a otro para su procesamiento.
 - **Multi-Tasa (Multi-Rate)**: Los datos se procesan a diferentes velocidades en bloques diferentes, en donde para cada bloque se especifica el número de muestras por segundo y dependiendo de si el sistema es un transmisor o un receptor, el número de muestras por segundo puede aumentar o disminuir respectivamente para el procesamiento posterior.

Sistema de Hardware Digital Síncrono



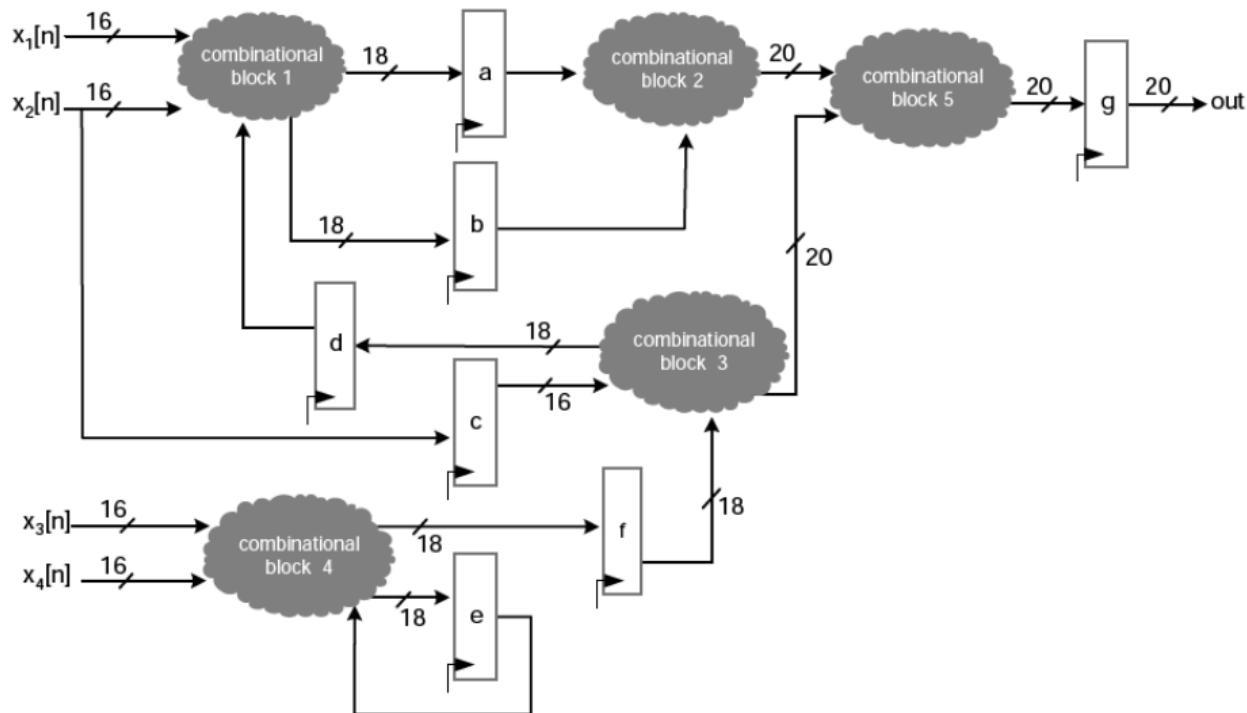
Sistema de Hardware Digital Síncrono

Camino Crítico

- En muchos casos, las aplicaciones de procesamiento de señales se mapean en lógica digital síncrona.
- **Síncrono** significa que todos los cambios en la lógica son controlados por un reloj de circuito.
- La lógica digital síncrona suele estar diseñada en el nivel de transferencia de registros (*Register Transfer Level - RTL*).
- En este nivel el diseño consiste en nubes combinacionales que ejecutan cálculos en el algoritmo y registros síncronos que almacenan valores intermedios y realizan retrasos algorítmicos de la aplicación.
- La nube combinatoria consiste en compuertas lógicas, donde la entrada a la nube son señales discretas almacenadas en registros.
- Al pasar por la lógica combinatoria estas señales experimentan diferentes retrasos en sus respectivos caminos, en donde el camino más largo o lento define el camino crítico (*critical path*).
- El camino crítico limita la frecuencia de reloj más rápida posible de un diseño.

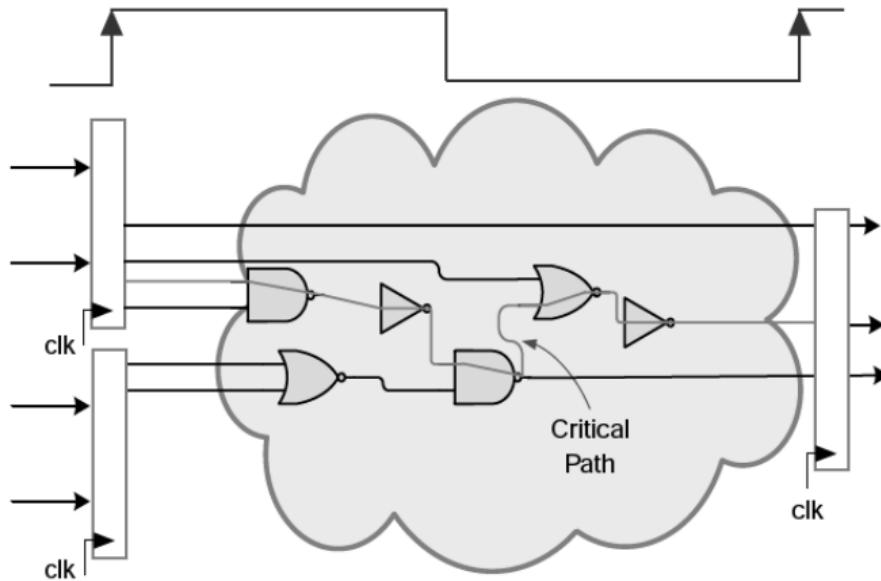
Sistema de Hardware Digital Síncrono

Ejemplo



Sistema de Hardware Digital Síncrono

Camino Crítico



Redes de Procesamiento tipo Kahn



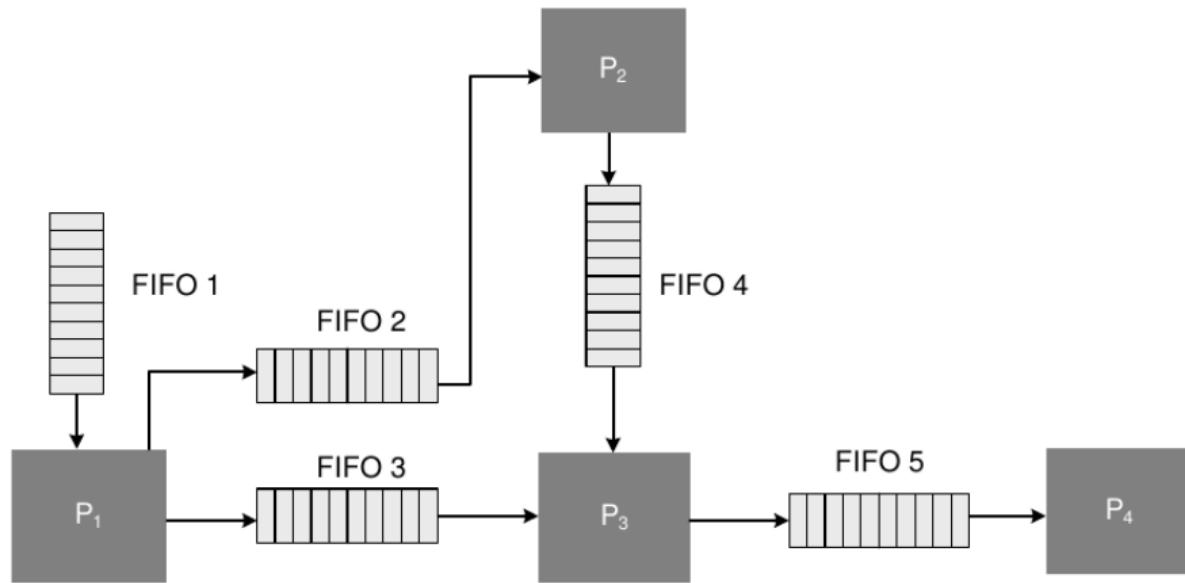
Redes de Procesamiento tipo Kahn

Qué es una red KPN?

- Un sistema que implementa una aplicación de flujo continuo es mejor representarla como componentes que funcionan de forma autónoma tomando entradas de FIFOs y generando salida en FIFOs.
- Las redes de procesamiento tipo Khan (*Kahn Process Network - KPN*) proporcionan un método formal para estudiar como se comporta un sistema que considera sus entradas y salidas interconectadas a FIFOs y su posterior mapeo en el diseño digital.
- El KPN es un conjunto de procesos autónomos que se ejecutan simultáneamente y que se comunican entre sí de una manera punto a punto sobre FIFOs, donde la sincronización en la red se logra mediante una operación de lectura de bloqueo y todas las escrituras a las FIFO son no bloqueantes.
- Un proceso espera en un modo de lectura de bloqueo para las FIFOs en cada uno de sus enlaces entrantes para obtener un número predefinido de muestras.
- Todos los nodos de la red se ejecutan después de que sus FIFOs de entrada asociadas hayan adquirido suficientes datos, donde dicha ejecución de un nodo se llama disparo (*firing*), y las muestras se llaman *tokens*.

Redes de Procesamiento tipo Kahn

Ejemplo



Redes de Procesamiento tipo Kahn

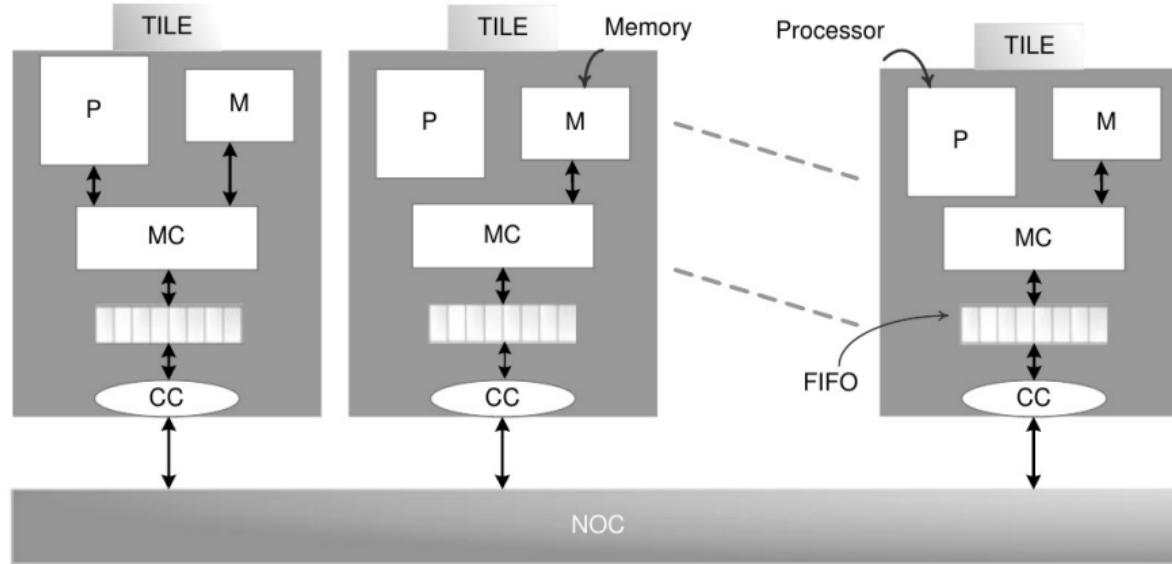
Limitaciones

- *La lectura de datos requiere estricta adhesión a FIFO, que obliga a las lecturas a seguir un orden secuencial desde el primer valor escrito en el buffer hasta el último.* Varios algoritmos de procesamiento de señal no siguen esta secuenciación estricta, un ejemplo es un algoritmo FFT de decimación en el tiempo que lee datos en orden de direccionamiento bit-reverse.
- *Asume que una vez que se lee un valor de la FIFO, se elimina.* En muchos algoritmos de procesamiento de señales, los datos se utilizan varias veces. Por ejemplo, un algoritmo de convolución simple requiere múltiples iteraciones del algoritmo para leer los mismos datos una y otra vez.
- *Supone que todos los valores serán leídos,* mientras que en muchos algoritmos puede haber algunos valores que no requieren ninguna lectura y los datos se lean escasamente.

Redes de Procesamiento tipo Kahn

Modificaciones

- Una forma de solucionar las limitaciones anteriores es usar la memoria local M en el nodo del procesador D para mantener una copia de sus datos FIFO de entrada.



Métodos para Representar Sistemas DSP



Métodos para Representar Sistemas DSP

Descripciones ejecutables basadas en lenguaje

- Los métodos basados en el lenguaje se utilizan para el desarrollo de software, y se utilizan lenguajes de alto nivel para codificar algoritmos.
- Los idiomas son interpretativos o ejecutables (Ejemplo Matlab).
- Para algoritmos computacionalmente intensivos, el diseñador prefiere escribir el código en C/C++.
- Como en estos idiomas el código se compila para la ejecución, el ejecutable funciona mucho más rápidamente que su simulación equivalente de MATLAB.

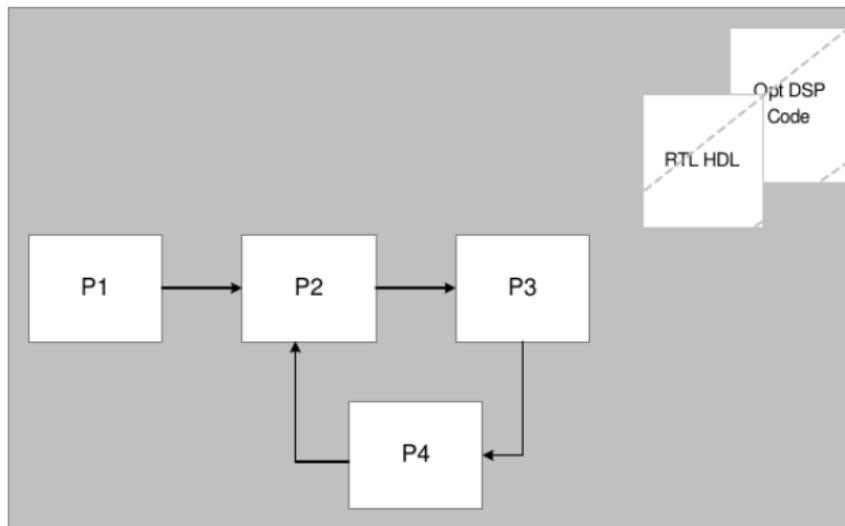
Métodos para Representar Sistemas DSP

Especificación dirigida por gráficos o diagrama de flujo

- Los métodos gráficos son especialmente convenientes para el mapeo de HW y la comprensión del funcionamiento y el flujo del algoritmo.
- Una representación gráfica es el método de elección para desarrollar hardware optimizado, generación de código y síntesis.
- Algunas herramientas que usan esta metodología son Simulink de Mathworks, Advanced Design Systems (ADS) de Agilent, Signal Processing Worksystem (SPW) de Cadence, Cocentric System Studio de Synopsys, LabVIEW de National Instruments y DSP Station de Mentor Graphics.
- También soportan el flujo de diseño jerárquico.
- Estos métodos también hacen hincapié en el diseño de arquitectura basada en componentes.
- Los componentes pueden estar parametrizados para ser reutilizados en una serie de instancias de diseño.
- Cada componente puede describirse adicionalmente en diferentes niveles de abstracción.

Métodos para Representar Sistemas DSP

Ejemplo

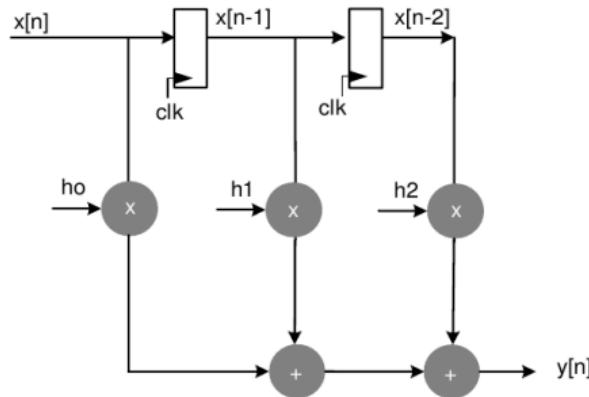


Representación gráfica de un algoritmo de procesamiento de señal donde el nodo P3 se describe en RTL y es optimizado para FPGA o un DSP particular.

Métodos para Representar Sistemas DSP

Diagrama en Bloque

- Un diagrama de bloques es un método gráfico muy simple que consiste en bloques funcionales conectados con bordes dirigidos.
- Un borde conectado representa el flujo de datos del bloque de origen al bloque de destino.

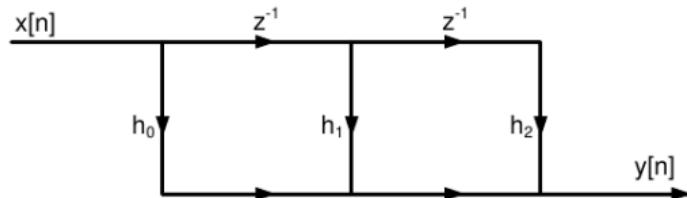


Filtro FIR de tres coeficientes ($y[n] = h_0x[n] + h_1x[n - 1] + h_2x[n - 2]$).

Métodos para Representar Sistemas DSP

Gráfico de Flujo de Señal (SFG)

- Es una versión simplificada de un diagrama de bloques.
- El funcionamiento de la multiplicación con una constante y los retrasos están representados por aristas, mientras que los nodos representan operaciones de suma, resta y de entrada y salida (I/O).
- SFGs se utilizan principalmente en la descripción de los algoritmos DSP.

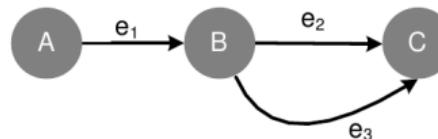


Filtro FIR de tres coeficientes ($y[n] = h_0x[n] + h_1x[n - 1] + h_2x[n - 2]$).

Métodos para Representar Sistemas DSP

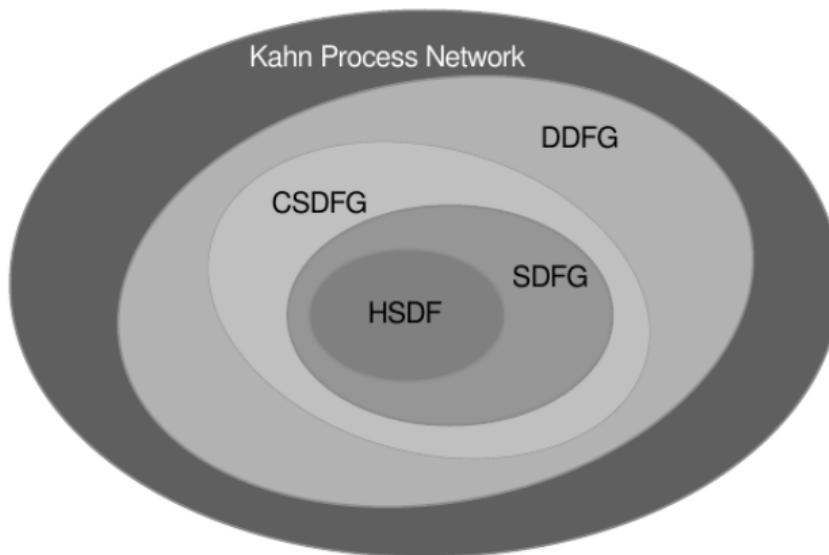
Gráfico de flujo de datos (DFG)

- Se describe un algoritmo de procesamiento de señales mediante un grafo dirigido $G = \langle V, E \rangle$, donde un nodo $v \in V$ representa una unidad computacional o, en un diseño jerárquico. Un borde dirigido $e \in E$ desde un nodo fuente a un nodo destino representa un buffer FIFO o simplemente precedencia de ejecución. También se utiliza para representar los retrasos algorítmicos introducidos en los datos mientras se mueve desde el nodo de origen al nodo de destino.
- Es de especial interés para los diseñadores de hardware, ya que captura la propiedad impulsada por datos de un algoritmo DSP.
- Además, expone la concurrencia oculta entre diferentes partes del algoritmo. Un DFG puede utilizarse para representar algoritmos DSP síncronos, asíncronos y de tasa múltiple.
- Motiva al diseñador a pensar en términos de componentes, mejorando la reutilización de hardware.
- También ayuda en la optimización, testing y verificación a nivel de módulos.



Métodos para Representar Sistemas DSP

Representaciones Gráficas



Kahn Process Network - Dynamic DFG (DDFG) - Cycle Static DFG (CSDFG) - Synchronous DFG (SDFG) - Homogeneous SDFG (HSDFG).

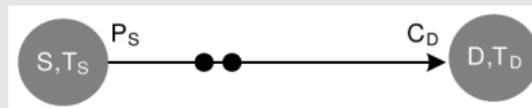
Métodos para Representar Sistemas DSP

Gráfico de flujo de datos (DFG)

Síncrono DFG

- Un ejemplo de una aplicación de transmisión es un sistema de procesamiento multimedia que consiste en procesos o tareas en las que cada tarea opera sobre un número predefinido de muestras y luego produce un número fijo de valores de salida.
- Estas tareas se ejecutan periódicamente en un orden definido.
- Aquí se conocen a priori el número de tokens consumidos por un nodo en cada uno de sus bordes y como resultado de su disparo el número de tokens que produce en sus bordes de salida.

Detalle



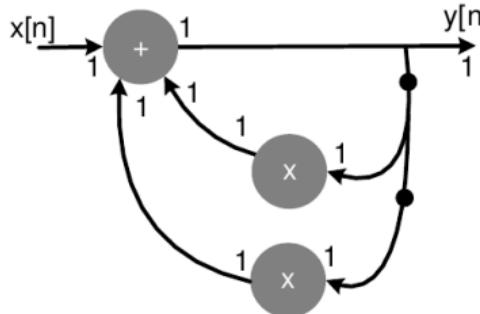
- **S:** Fuente
- **D:** Destino
- T_S / T_D : Tiempo de ejecución
- P_S / C_D : Tasa de producción y consumo de los datos
- ●: Retardo

Métodos para Representar Sistemas DSP

Gráfico de flujo de datos (DFG)

Filtro IIR como un Síncrono DFG

- Ecuación en diferencia
 $y[n] = x[n] + a_1y[n - 1] + a_2y[n - 2]$
- El gráfico consiste en dos nodos para multiplicaciones y un nodo para la adición, tomando cada uno un token como entrada y produciendo un token en la salida.
- Los caminos de realimentación desde la salida a los dos multiplicadores cada uno requieren un retraso.
- Estos retrasos se muestran con puntos negros en los respectivos bordes.



Métodos para Representar Sistemas DSP

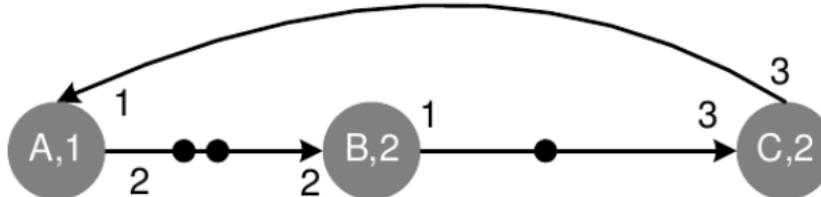
Gráfico de flujo de datos (DFG)

Disparo Auto-temporizado (Self-timed Firing)

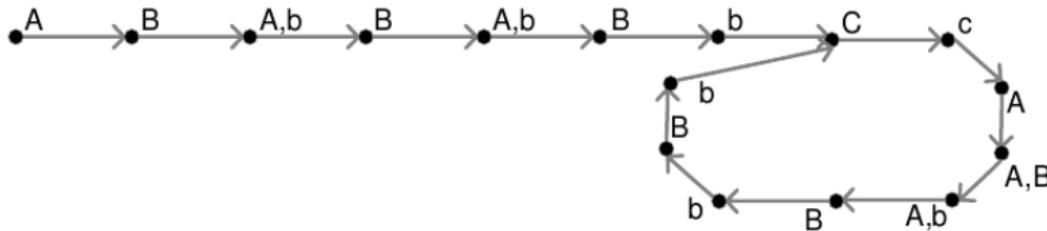
- En un disparo auto-temporizado, un nodo se activa tan pronto como se obtiene el número necesario de tokens en sus bordes entrantes.
- Para implementar SDFG en hardware dedicado, se puede usar la ejecución auto-temporizada de nodos o un vector de repetición se puede calcular primero usando un conjunto equilibrado de ecuaciones.
- Este vector calcula disparos múltiples de cada nodo para hacer el SDFG consistente.
- Una implementación SDFG auto-temporizada generalmente da como resultado una fase transitoria, y después de eso la secuencia de disparo se repite en una fase periódica.

Métodos para Representar Sistemas DSP

Gráfico de flujo de datos (DFG)



(a)



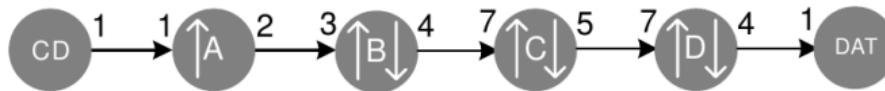
(b)

Métodos para Representar Sistemas DSP

Gráfico de flujo de datos (DFG)

Simple y multi Tasa (*Single-rate and Multy-rate*) SDFGs

- En un SDFG de tasa única, la tasa de consumo r_c es la misma que la tasa de producción r_p .
- En un SDFG multi tasa, estas tasas no son iguales y uno es un múltiplo racional del otro.
- Un sistema es de tasa múltiple de decimación cuando $r_c > r_p$.
- Un sistema es de interpolación cuando $r_p < r_c$.



Métodos para Representar Sistemas DSP

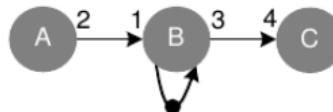
Gráfico de flujo de datos (DFG)

Homogeneous SDFGs

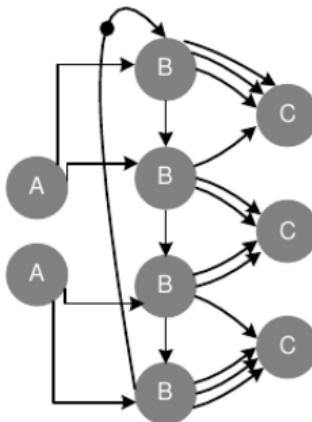
- Es un caso especial de un gráfico de tasa única en el que cada nodo produce un token o valor de datos en todos sus bordes salientes.
- Extendiendo esta cuenta a bordes entrantes, cada nodo también consume un valor de datos de todos sus bordes entrantes.
- Cualquier SDFG consistente se puede convertir en HSDFG.
- La conversión da una medida exacta del rendimiento, aunque la conversión puede dar lugar a un aumento exponencial en el número de nodos y por lo tanto puede ser muy complejo para la interpretación y la implementación.
- La forma más sencilla de convertir un SDFG coherente a HSDFG es encontrar primero el vector de repetición y luego hacer copias de cada nodo tal como se da en el vector de repetición y dibujar apropiadamente los bordes de los nodos de origen a los nodos de destino de acuerdo con el SDFG original.

Métodos para Representar Sistemas DSP

Homogeneous SDFG - Ejemplo



(a)



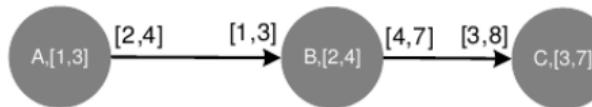
(b)

Métodos para Representar Sistemas DSP

Gráfico de flujo de datos (DFG)

Cyclo-static DFG

- En un DFG ciclo-estático, el número de tokens consumidos por cada nodo, aunque varía de iteración a iteración, exhibe periodicidad y repite el patrón después de un número fijo de iteraciones.
- Adecuado para modelar varias aplicaciones de procesamiento de señal, ya que proporciona flexibilidad de diferentes tasas de producción y consumo de cada nodo siempre que el patrón se repita después de un número finito de iteraciones.
- Esta representación también funciona bien para diseños en los que una secuencia periódica de funciones está asignada en el mismo bloque HW.



Métodos para Representar Sistemas DSP

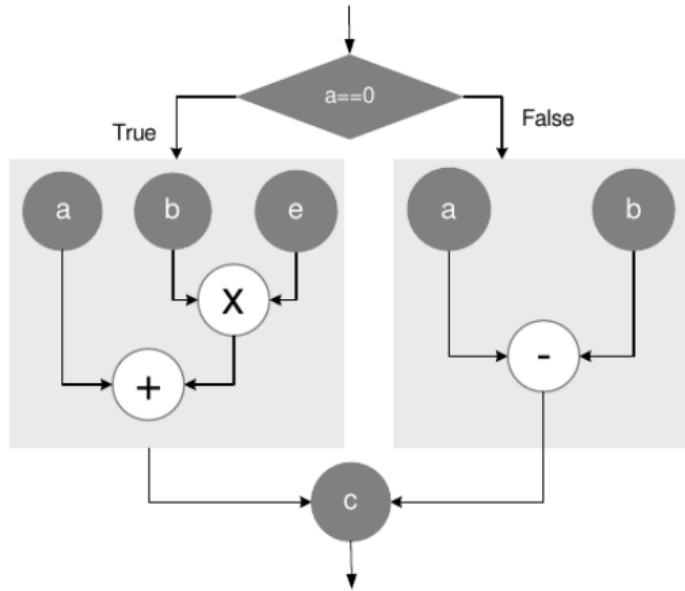
Gráfico de flujo de datos (DFG)

Gráficos de control de flujo

- Es adecuado para procesar un algoritmo de control.
- Estos algoritmos se encuentran generalmente en la implementación de protocolos de comunicación o diseños de controladores para rutas de datos.
- Combina funcionalidad controlada por datos y control específica de un algoritmo.
- Cada nodo del DFG representa una operación matemática, y cada borde representa la precedencia o una dependencia de datos entre las operaciones.
- Un CDFG puede cambiar el número de tokens producidos y consumidos por los nodos en diferentes ajustes de entrada.
- Un DFG con tasas variables de producción y consumo se denomina gráfico de flujo de datos dinámico (DDFG).

Métodos para Representar Sistemas DSP

Gráfico de flujo de datos (DFG)



Métodos para Representar Sistemas DSP

Gráfico de flujo de datos (DFG)

Máquina de estado finita (FSM)

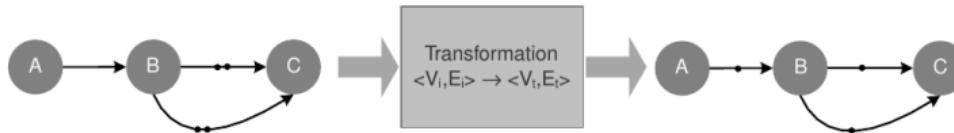
- Se utiliza para proporcionar señales de control a un camino de datos de procesamiento de señales para ejecutar un cálculo con una selección de operandos de un conjunto de operandos.
- Un FSM genérico asume que un sistema está en uno de un número finito de estados.
- Un FSM tiene un estado actual y, basado en un evento interno o externo, calcula el siguiente estado.
- De esta manera, una vez que el FSM transita de un estado a otro, el datapath mantiene la implementación de diferentes porciones del algoritmo.
- Además de implementar un planificador, el FSM también funciona bien para implementar protocolos donde se sigue un conjunto de procedimientos y sincronización entre varios componentes del sistema, como con el arbitraje de buses compartidos.

Métodos para Representar Sistemas DSP

Gráfico de flujo de datos (DFG)

Transformaciones en un gráfico de flujo de datos

- Las transformaciones matemáticas convierten un DFG en un DFG más apropiado para la implementación de hardware.
- Estas transformaciones cambian la implementación del algoritmo de tal manera que el algoritmo transformado satisface mejor las metas específicas del diseño.
- De un conjunto de objetivos de diseño, el diseñador puede querer minimizar el retardo de la ruta crítica o el número de registros.
- Retiming, plegado, despliegue y look-ahead son algunas de las transformaciones utilizadas comúnmente.



Medidas de Rendimiento



Medidas de Rendimiento

Clasificación

■ Período de iteración

- Para un sistema de procesamiento de señal de tasa única, una iteración del algoritmo adquiere una muestra de un convertidor A/D y realiza un conjunto de operaciones para producir una muestra de salida correspondiente.
- El tiempo que tarda el sistema en calcular todas las operaciones en una iteración de un algoritmo se denomina período de iteración y se mide en unidades de tiempo o en número de ciclos.
- Para un sistema digital genérico, la relación entre la frecuencia de muestreo f_s y la frecuencia de reloj de circuito f_c es importante.
- Cuando éstos son iguales, el período de iteración es determinado por el camino crítico.
- En los diseños donde $f_c > f_s$, el período de iteración se mide en términos del número de ciclos de reloj requeridos para calcular una muestra de salida.

Medidas de Rendimiento

Clasificación

■ Período de muestreo y Throughput

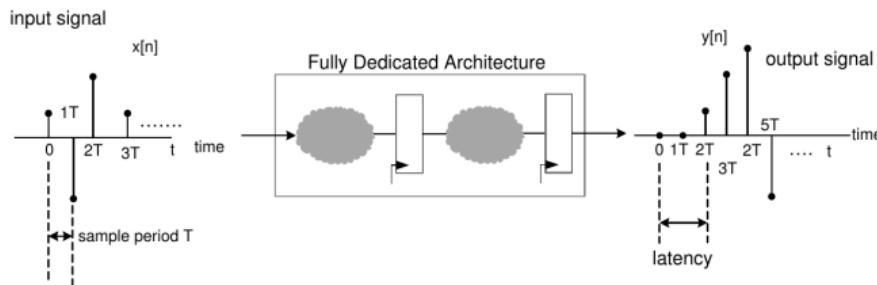
- El periodo de muestreo T_s se define como el tiempo promedio entre dos muestras de datos sucesivas.
- El periodo especifica el número de muestras por segundo de cualquier señal. El requisito de frecuencia o de frecuencia de muestreo ($f_s = 1/T_s$) es específico de una aplicación y, posteriormente, obliga al diseñador a producir hardware que puede procesar los datos que se introducen en el sistema a esta tasa.
- A menudo, esta restricción requiere que el diseñador minimice los retrasos en la ruta crítica.
- Pueden reducirse utilizando unidades de cálculo más optimizadas o añadiendo retrasos en la lógica
- Los retrasos de pipelining añaden latencia en el diseño.
- En los diseños donde $f_s < f_c$, el diseñador digital explora vías de intercambio de recursos para la reutilización óptima de bloques computacionales.

Medidas de Rendimiento

Clasificación

■ Latencia

- La latencia se define como el retardo de tiempo para que el algoritmo produzca una salida $y[n]$ en respuesta a una entrada $x[n]$.
- En muchas aplicaciones los datos se adquieren en un buffer y después se introducen para su procesamiento, introduciendo latencia.
- Además de los retrasos algorítmicos, los registros de pipelining son la fuente principal de latencia en una sistema.
- Generalmente hay una relación inversa entre el camino crítico y la latencia.
- Con el fin de reducir la ruta crítica, se añaden registros de pipeline que dan lugar a un aumento de la latencia del diseño.



Medidas de Rendimiento

Clasificación

■ Potencia

- La disipación de potencia estática se debe a la corriente de fuga en la lógica digital.
- La disipación de potencia dinámica se debe a toda la actividad de conmutación, la cual constituye la mayor parte de la disipación de potencia en un diseño.
- La disipación de potencia dinámica es específica del diseño, mientras que la disipación de potencia estática depende de la tecnología.
- En un FPGA, la disipación de potencia estática se debe a la corriente de fuga a través de diodos de polarización inversa.
- En el mismo FPGA, el uso de la energía dinámica depende de la frecuencia de reloj, la tensión de alimentación, la actividad de conmutación y la utilización de recursos.
- En el nivel de transferencia de registro (RTL), el diseñador puede determinar las partes del diseño que no están realizando cálculos útiles y se pueden apagar para ahorrar energía.
- Se utiliza la técnica *gated clock* para detener selectivamente el reloj en áreas que no están realizando cálculos en el ciclo actual.

Arquitecturas Dedicadas



El espacio de diseño

- Para diseñar una lógica óptima para un problema dado, el diseñador explora el espacio de diseño donde hay varias opciones disponibles para mapear algoritmos en tiempo real en hardware.
- La frecuencia de reloj máxima alcanzable del circuito f_c y la tasa de muestreo requerida de la entrada de datos al sistema f_s juegan un papel crucial en la determinación y selección de la mejor opción.
- Los dispositivos digitales como FPGAs y ASICs pueden ejecutar la lógica a velocidades de reloj en el rango de 30 MHz a 800 MHz aproximadamente.
- En mucho diseño la arquitectura esta definida pero es posible mejorar el área aplicando otras técnicas en las operaciones de adición.
- Lo mismo se aplica a los multiplicadores y shifters. Después de un mapeo uno a uno del DFG a la arquitectura, el diseño se evalúa para satisfacer la tasa de datos de entrada del sistema.
- En ocasiones, en las arquitecturas óptimas para las operaciones básicas, el diseño sintetizado no cumple los requisitos de tiempo.
- El diseñador necesita emplear transformaciones matemáticas apropiadas o puede agregar registros de pipeline para obtener una mejor sincronización.

Pipelining

- La asignación de un gráfico de flujo de datos (DFG) transformado a una arquitectura totalmente dedicada es trivial, ya que cada operación se asigna a su operador de hardware coincidente.
- En muchos casos, esta asignación puede consistir en rutas con lógica combinatoria que viola las restricciones de tiempo.
- Por lo tanto, es imperativo romper estas nubes combinacionales con registros.
- Para las rutas de feedforward, el diseñador también tiene la opción de agregar registros de pipeline en el camino principal.
- Mantener la coherencia de los datos en el gráfico es un problema crítico en el pipelining.
- El diseñador debe cerciorarse de que el datapath trazado de cualquier entrada primaria a cualquier salida primaria pasa con el mismo número de registros de la pipeline.
- En contraste, no existe una forma sencilla de añadir registros de pipeline en los caminos realimentados (feedback).

Arquitecturas Dedicadas

Ejemplo

- Ecuación en diferencia:

$$d[n] = a[n] + b[n] \quad (1)$$

$$out[n] = (d[n-1] - c[n]) * e[n] \quad (2)$$

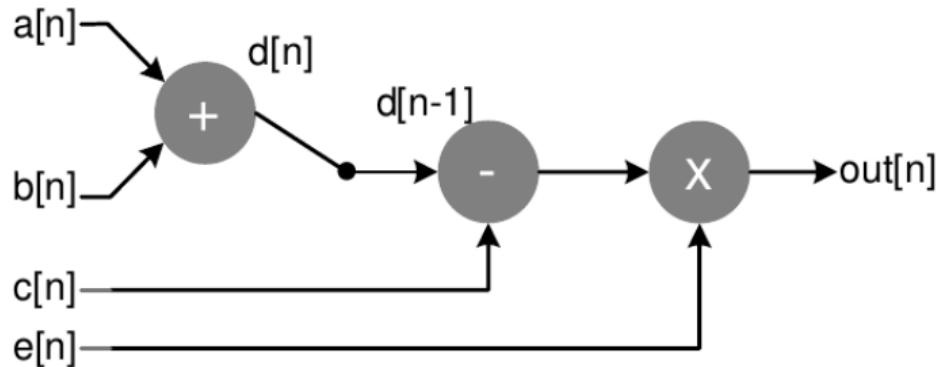


Diagrama de flujo de datos.

Arquitecturas Dedicadas

Ejemplo - Pipelinig

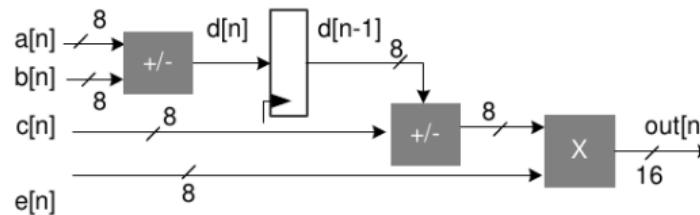
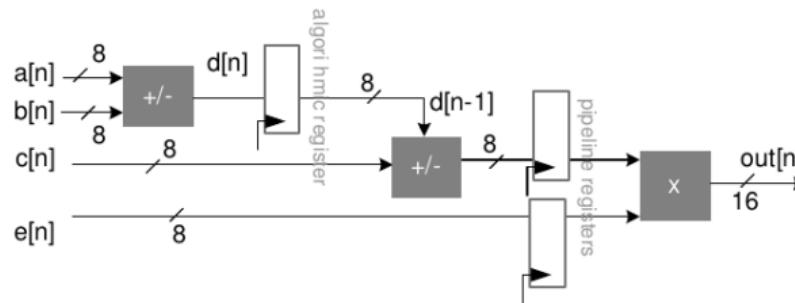


Diagrama en Bloque.



Pipelining

Arquitecturas Dedicadas

Selección de Bloques Diseño Básicos

- Partiendo de un gráfico de flujo de datos que representa un algoritmo de procesamiento de señales se debe elegir los bloques básicos que menos recursos utilicen en el diseño.
- Se busca diseñar una arquitectura óptima con un área mínima y la mejor sincronización mientras mapea el DFG a una arquitectura totalmente dedicada.

Basic building blocks	Relative timing	Relative area
Adder 1 (A1)	T	2.0A
Adder 2 (A2)	1.3T	1.7A
Adder 3 (A3)	1.8T	1.3A
Multiplier 1 (M1)	1.5T	2.5A
Multiplier 2 (M2)	2.0T	2.0A
Multiplier 3 (M3)	2.4T	1.7A

Arquitecturas Dedicadas

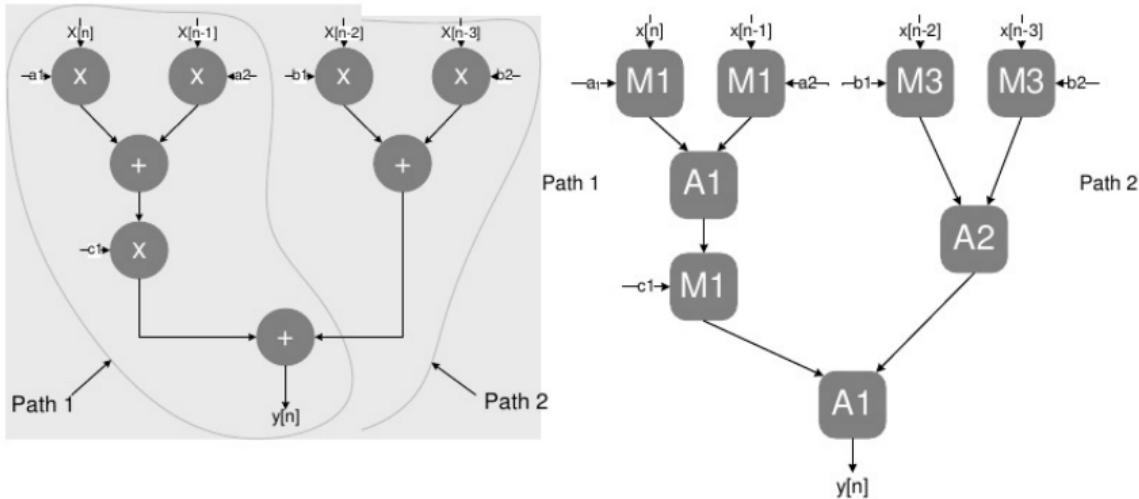


Diagrama de Flujo de Datos y Arquitectura Totalmente Dedicada (FDA).

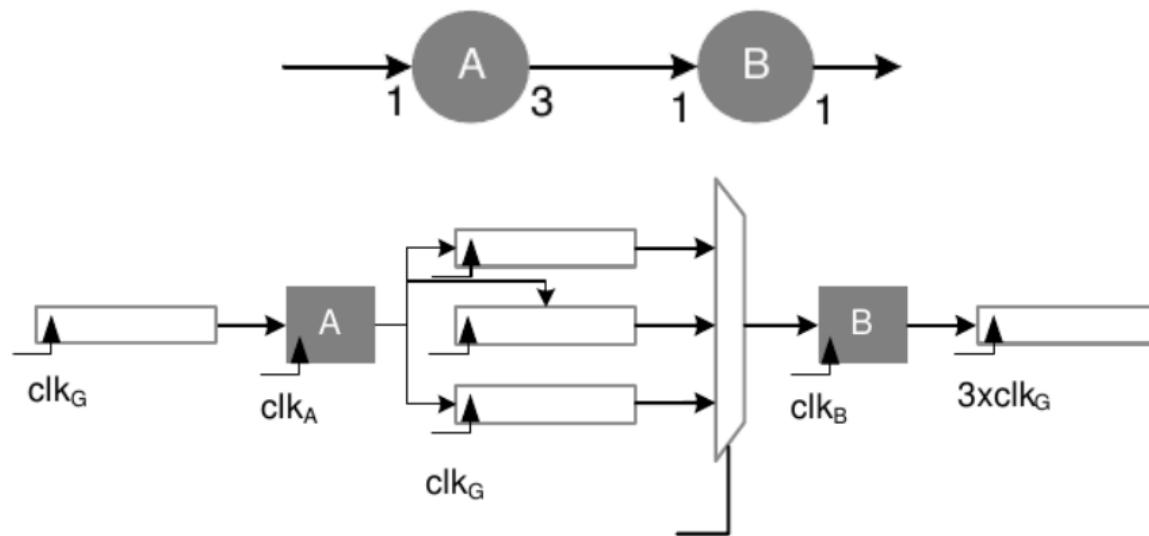
DFG a Hardware

- Un gráfico de flujo de datos proporciona una representación visual completa a los algoritmos de procesamiento de señales.
- Cada nodo se caracteriza por las tasas de producción y consumo en cada puerto.
- En muchos casos, existen varias opciones para un componente representado como un nodo en el DFG en la biblioteca de diseño.
- Estos componentes intercambian el área con el tiempo de ejecución o el número de ciclos necesarios para procesar los datos de entrada.

Mapeo de hardware de un sistema Multi-Rate

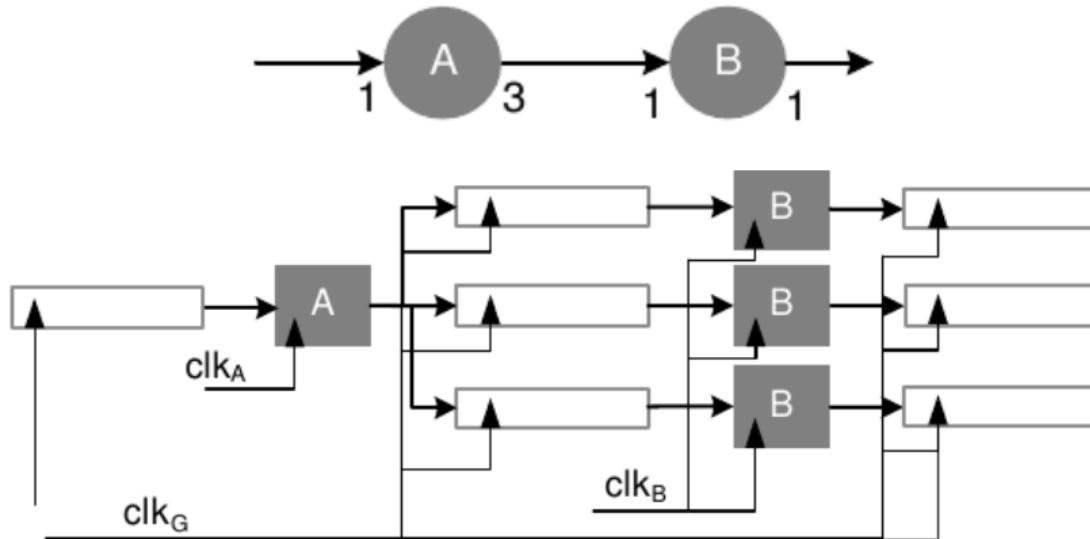
- Cada etapa se considera con sus índices de producción y consumo de muestras.
- Para una ventaja que requiere una producción de tasa múltiple para un consumo de tasa única, es posible una implementación paralela o secuencial.
- Una implementación paralela invoca cada nodo de destino varias veces, mientras que un ajuste secuencial almacena los datos en registros y lo introduce secuencialmente en un único nodo de destino.

Arquitecturas Dedicadas



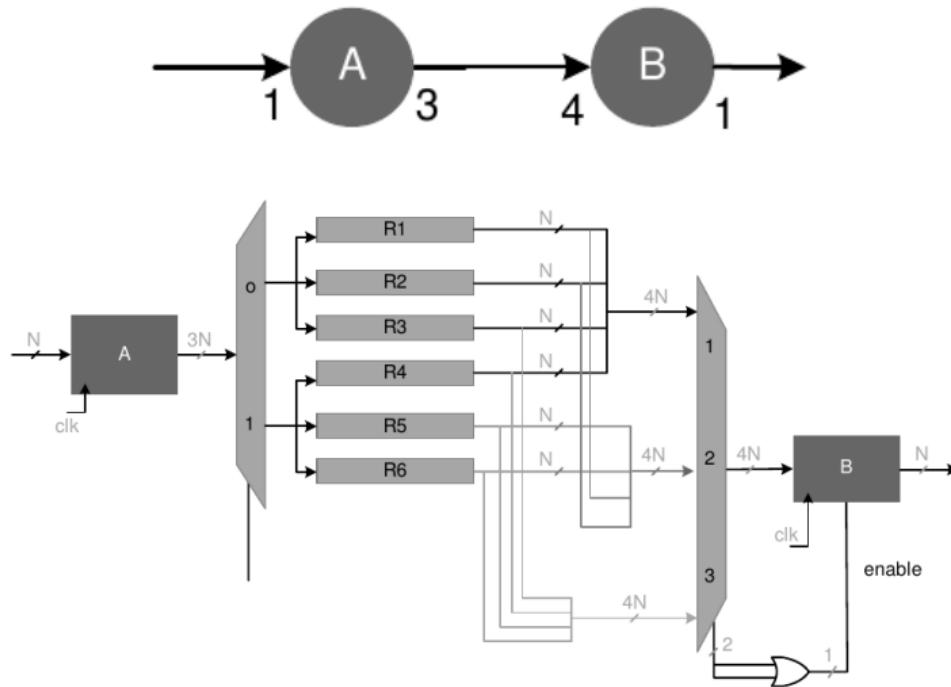
Multi-Rate en formato Secuencial.

Arquitecturas Dedicadas



Multi-Rate en formato Paralelo.

Arquitecturas Dedicadas



Multi-Rate con tasa de producción menor a la tasa de consumo.

Introducción a FPGA



Introducción a FPGA

ASIC vs. FPGA



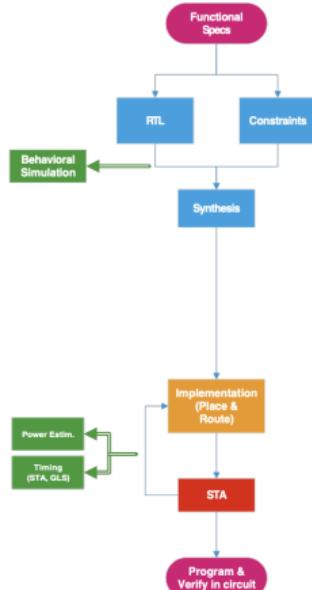
- Un **ASIC** es un circuito integrado creado específicamente para resolver una aplicación de cálculo precisa
 - Construido para un cómputo específico
 - Ciclo de planificación complejo y largo
 - Costos iniciales altos, después costos por unidades bajos
 - Ocupan poco espacio y disipan poca potencia
 - Frecuencias de funcionamiento altas
 - No se pueden modificar

- **FPGA** es un circuito integrado cuya funcionalidades son programables vía software
 - Programable por el diseñador
 - Ciclo de planificación simple y rápido
 - Costos iniciales bajos, después costos por unidades altos
 - Ocupan espacio y disipan más potencia
 - Frecuencias de funcionamiento bajas
 - Es posible modificar algo en cualquier momento

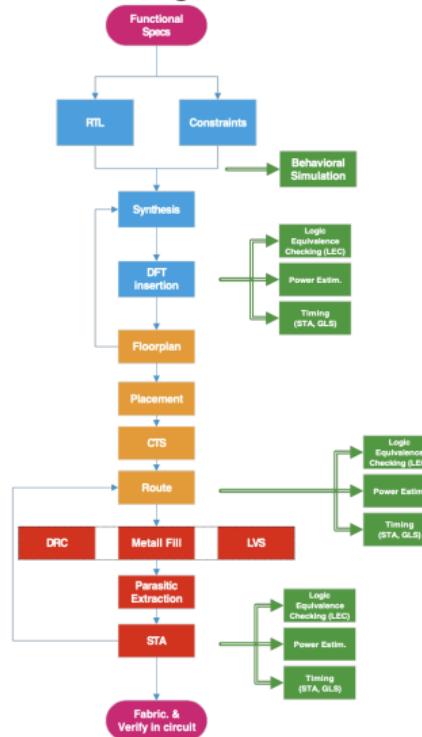
Introducción a FPGA

ASIC vs. FPGA

FPGA Design Flow

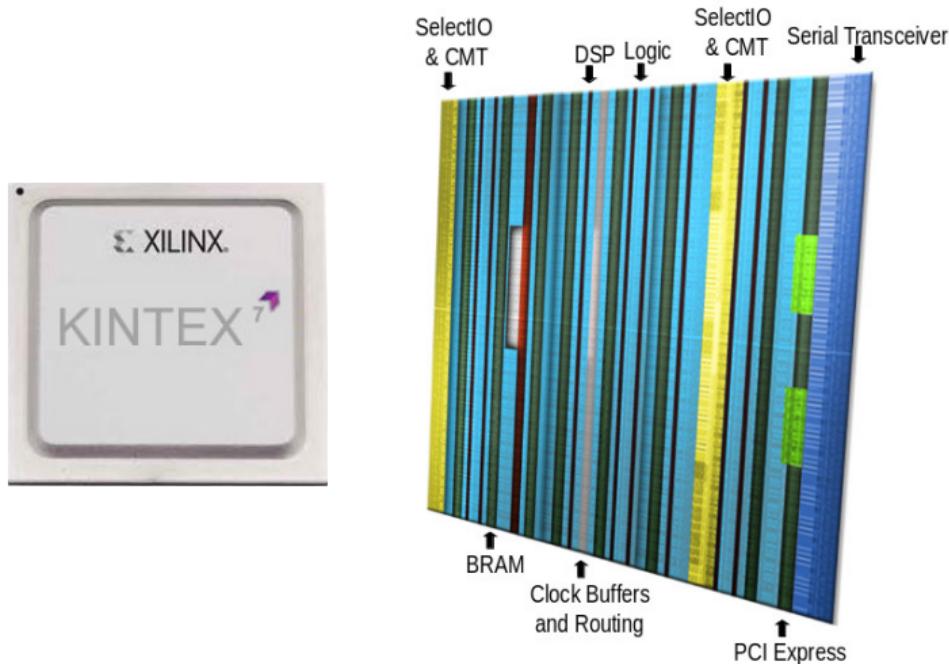


ASIC Design Flow



Introducción a FPGA

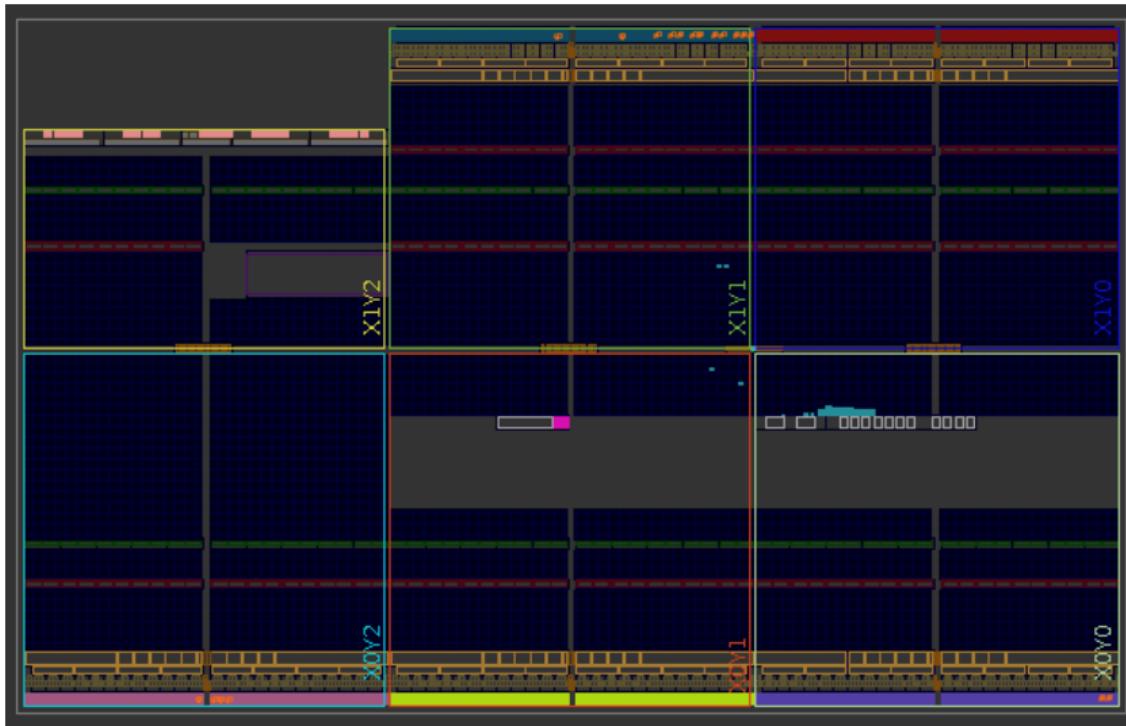
Field Programmable Gate Array



Esquema simplificado de una FPGA (fuente obtenida de “11_7-Series Architecture Overview”).

Introducción a FPGA

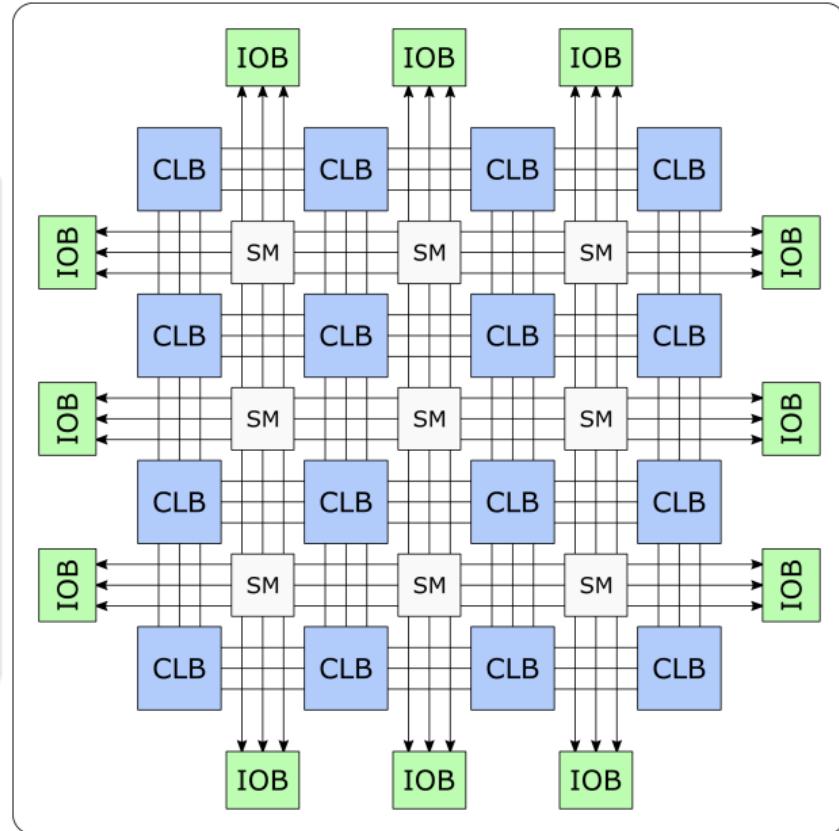
Regiones de Clock



Introducción a FPGA

Arquitectura de una FPGA

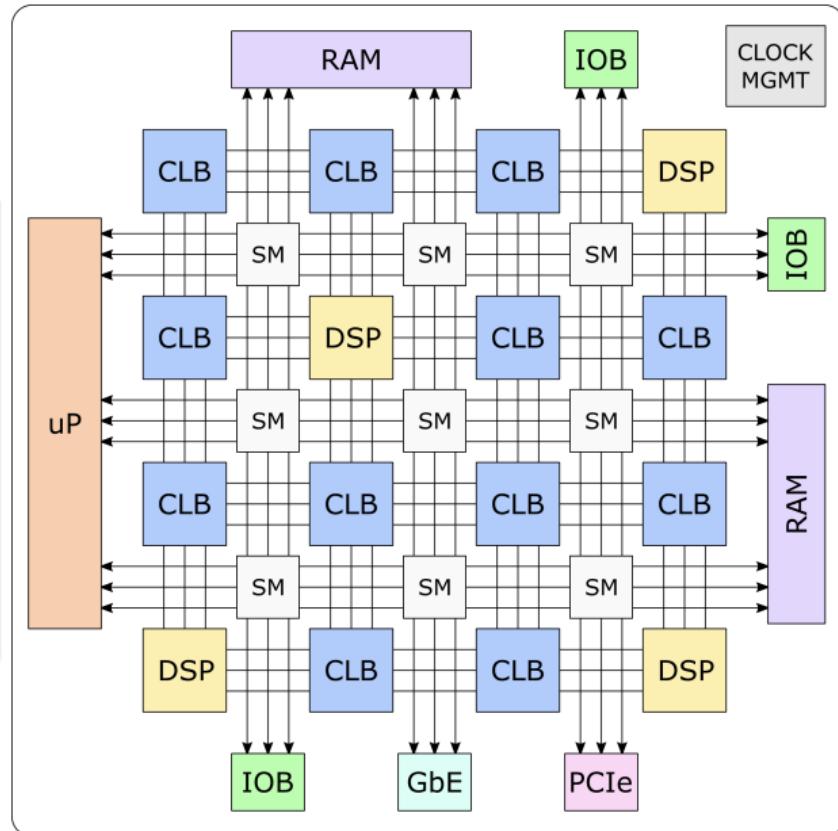
- Una FPGA se compone de tres componentes fundamentales:
 - **CLB (Configurable Logic Block):** Permite implementar diferentes funciones lógicas. Llamado LAB (Logic Array Block) en las FGPA de Altera.
 - **IOB (Input Output Block):** Interfaz entre pines físicos y líneas de conexión internas.
 - **SM (Switch Matrix):** Permite programar las conexiones internas.



Introducción a FPGA

Arquitectura de una FPGA

- Normalmente se integran otros bloques de propósitos especiales, como ser:
 - Microprocesadores
 - Bloques DSP
 - Interfaz Gigabit Ethernet
 - Interfaz PCI Express
 - Manejo/generación de clocks
 - Bloques de memoria RAM (block RAM)
 - Gigabit transceivers
 -



Introducción a FPGA

Arquitectura de una CLB

- Una CLB se divide en dos o mas “Slices”, y su vez cada Slice se compone de dos o más “Logic Cells” (LC).
 - La razón de este diseño jerárquico es que las interconexiones internas de estas jerarquías alcanzan velocidades mucho más altas que las externas debido a la distribución lograda dentro del chip.
 - Se busca tener un equilibrio adecuado entre performance y configurabilidad.
 - En Altera un LC tiene su equivalente denominado “Logic Element” (LE).
 - La cantidad de CLBs, Slices y LCs y su distribución interna depende fundamentalmente de la familia de FPGAs del fabricante.

Introducción a FPGA

Arquitectura de una CLB

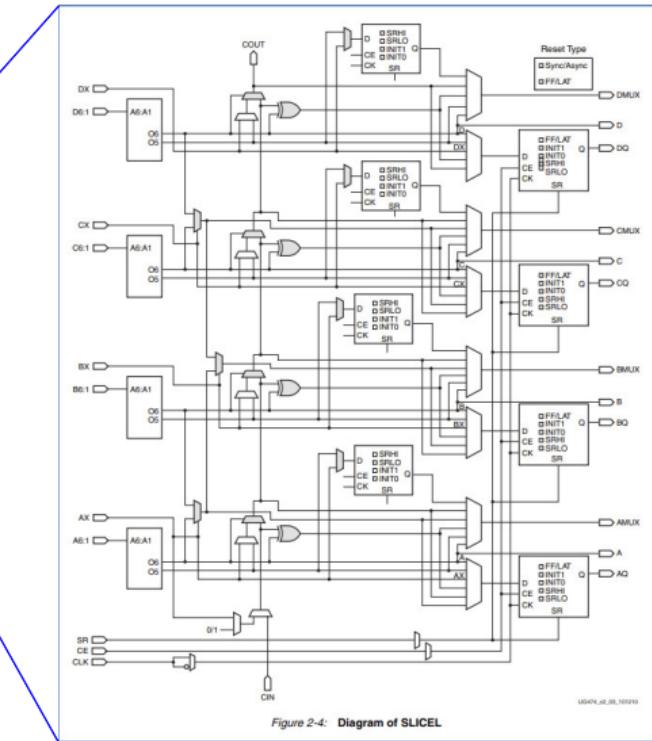
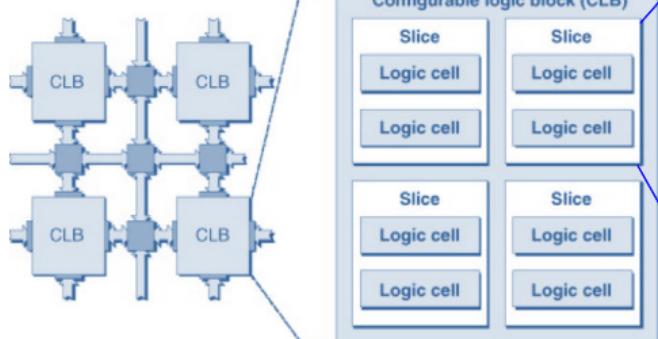


Figure 2-4: Diagram of SLICEL

Introducción a FPGA

Arquitectura de una CLB

ARTIX[®]

KINTEX[®]

VIRTEX[®]

ZYNQ[®]

Maximum Capability	Lowest Power and Cost	Industry's Best Price/Performance	Industry's Highest System Performance	Extensible Processing Platform
Logic Cells	20K – 355K	70K – 480K	285K – 2,000K	30K – 350K
Block RAM	12 Mb	34 Mb	65 Mb	240KB – 2180KB
DSP Slices	40 – 700	240 – 1,920	700 – 3,960	80 – 900
Peak DSP Perf.	504 GMACS	2,450 GMACs	5,053 GMACS	1080 GMACS
Transceivers	4	32	88	16
Transceiver Performance	3.75Gbps	6.6Gbps and 12.5Gbps	12.5Gbps, 13.1Gbps and 28Gbps	6.6Gbps and 12.5Gbps
Memory Performance	1066Mbps	1866Mbps	1866Mbps	1333Mbps
I/O Pins	450	500	1,200	372
I/O Voltages	3.3V and below	3.3V and below 1.8V and below	3.3V and below 1.8V and below	3.3V and below 1.8V and below

Introducción a FPGA

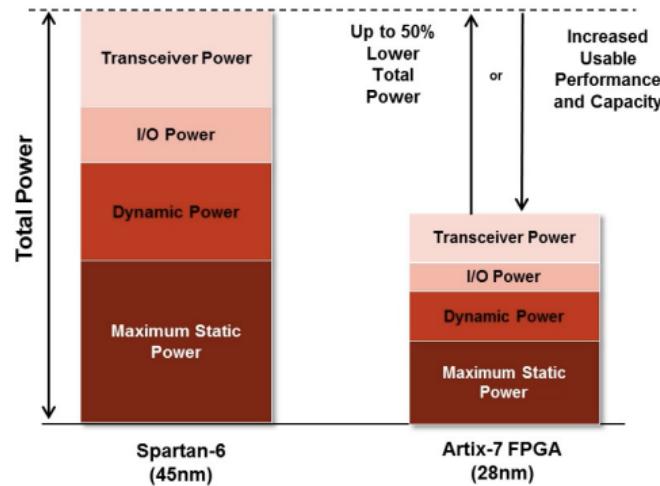
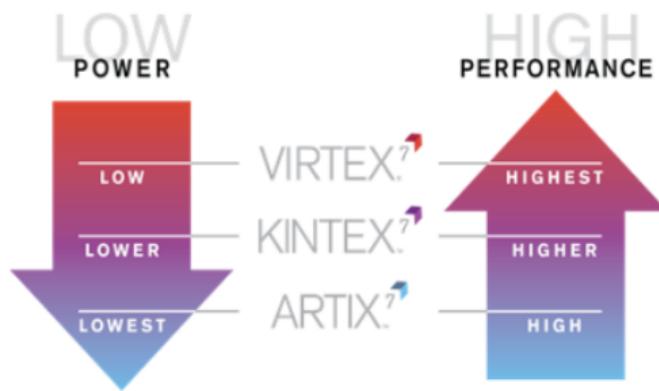
Arquitectura de una CLB

Table 8: Virtex-7 FPGA Feature Summary

Device ⁽¹⁾	Logic Cells	Configurable Logic Blocks (CLBs)		DSP Slices ⁽³⁾	Block RAM Blocks ⁽⁴⁾			CMTs ⁽⁵⁾	PCIe ⁽⁶⁾	GTX	GTH	GTZ	XADC Blocks	Total I/O Banks ⁽⁷⁾	Max User I/O ⁽⁸⁾	SLRs ⁽⁹⁾
		Slices ⁽²⁾	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)									
XC7V585T	582,720	91,050	6,938	1,260	1,590	795	28,620	18	3	36	0	0	1	17	850	N/A
XC7V2000T	1,954,560	305,400	21,550	2,160	2,584	1,292	46,512	24	4	36	0	0	1	24	1,200	4
XC7VX330T	326,400	51,000	4,388	1,120	1,500	750	27,000	14	2	0	28	0	1	14	700	N/A
XC7VX415T	412,160	64,400	6,525	2,160	1,760	880	31,680	12	2	0	48	0	1	12	600	N/A
XC7VX485T	485,760	75,900	8,175	2,800	2,060	1,030	37,080	14	4	56	0	0	1	14	700	N/A
XC7VX550T	554,240	86,600	8,725	2,880	2,360	1,180	42,480	20	2	0	80	0	1	16	600	N/A
XC7VX690T	693,120	108,300	10,888	3,600	2,940	1,470	52,920	20	3	0	80	0	1	20	1,000	N/A
XC7VX980T	979,200	153,000	13,838	3,600	3,000	1,500	54,000	18	3	0	72	0	1	18	900	N/A
XC7VX1140T	1,139,200	178,000	17,700	3,360	3,760	1,880	67,680	24	4	0	96	0	1	22	1,100	4
XC7VH580T	580,480	90,700	8,850	1,680	1,880	940	33,840	12	2	0	48	8	1	12	600	2

Introducción a FPGA

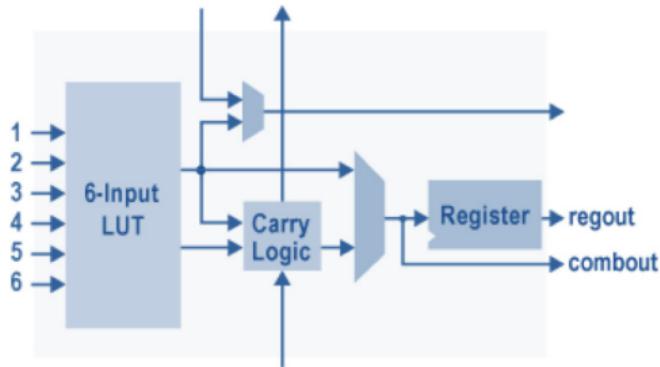
Arquitectura de una CLB



Introducción a FPGA

Arquitectura de una LC

- Una Logic Cell (LC) se compone de los siguientes elementos:
 - LUT (lookup table) de N entradas
 - Lógica de Carry
 - Multiplexores
 - Flip Flops

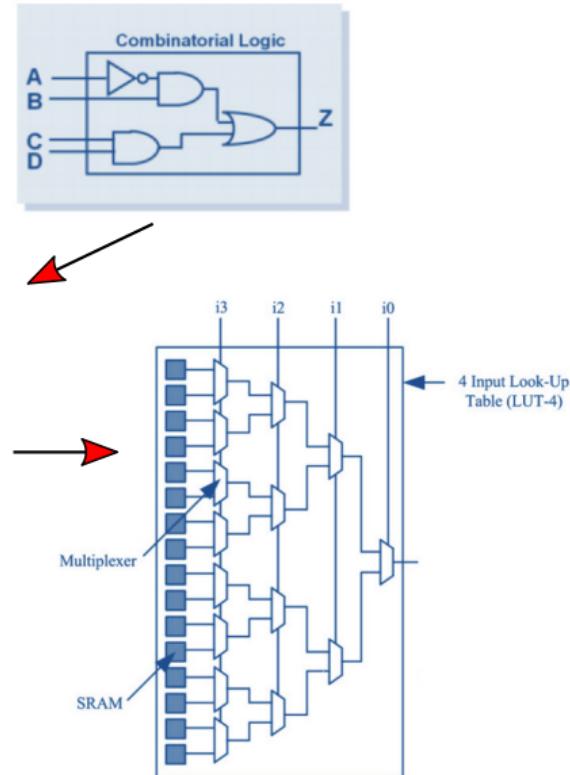


Introducción a FPGA

Lookup Tables (LUTs)

- Las LUTs permiten implementar operaciones lógicas de forma programable.
- Las LUTs empleadas por Xilinx y Altera se basan en SRAMs, es decir, que son volátiles.
- Una LUT de k entradas puede mapear 2^k combinaciones de salida.
- La interconexión de LUTs entre slices permite aumentar el orden de la entrada.
- Además de implementar lógica combinacional, los fabricantes otorgan la posibilidad de emplear las LUTs para inferir pequeños bloques de memoria RAM (distributed RAM) o como Shift Register (SR), conectando en cadena las distintas celdas de SRAM.

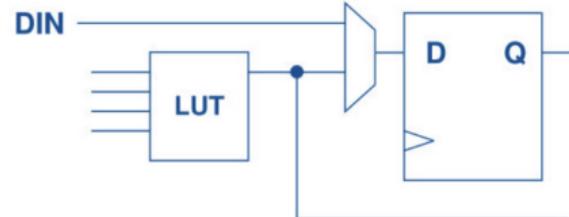
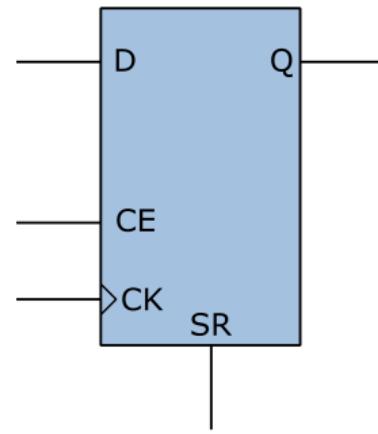
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
.	.	.	.	
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



Introducción a FPGA

Flip Flops (FF)

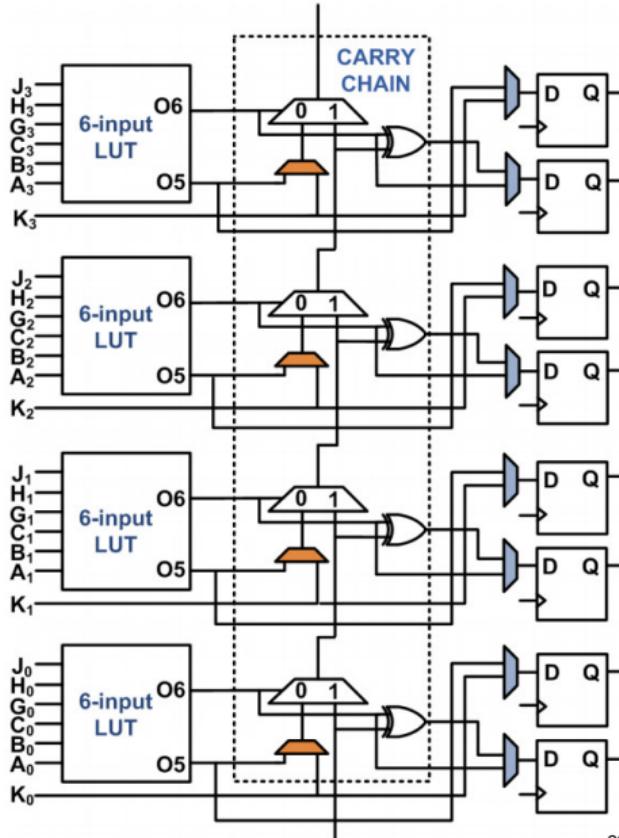
- Los flip flops integrados en las LC son del tipo D.
- Poseen una entrada de clock (CK) con polaridad programable.
- Poseen una entrada de clock enable (CE).
- Poseen una entrada de reset (SR) que puede configurarse como síncrona o asíncrona.
- Dentro de las LC pueden conectarse a la salida de la LUT o bien con entrada directa, dejando la LUT libre para otra funcionalidad.



Introducción a FPGA

Carry Chain

- La lógica de carry permite interconectar las diferentes LCs a través de los slices y CLBs para propagar el carry con alta velocidad.
- Empleado en operaciones aritméticas (sumadores, restadores, comparadores, etc.)



Introducción a FPGA

Block RAM (BRAM)

- Son bloques dedicados dentro de la FPGA para inferir memorias RAM de gran tamaño (de forma más eficiente que las RAM distribuidas). Algunos usos comunes son:
 - Almacenamiento de grandes Look-up tables (ej. funciones trigonométricas o matemáticas).
 - Almacenamiento de datos provenientes de dispositivos externos (ej. ADC).
 - Memoria RAM de soft-cores.
 - FIFOs de cruce de dominio de clocks.
- Normalmente vienen agrupadas en columnas dentro de la FPGA, para poder interconectarlas eficientemente.
- Pueden ser configuradas para diferentes funcionalidades:
 - Single port RAM
 - Dual port RAM o FIFO
- El fabricante provee una GUI para generalas o bien un template en HDL para ser instanciado por el diseñador.

Introducción a FPGA

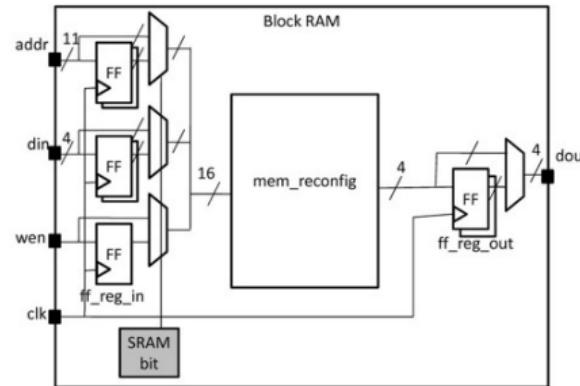
Block RAM (BRAM)

Single port BRAM

Port A

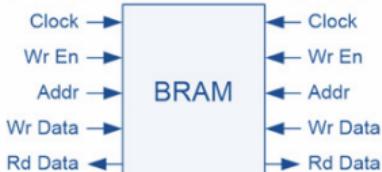


Port B



Dual port BRAM

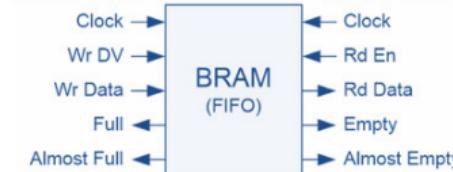
Port A



Port B

FIFO BRAM

Write Side



Read Side

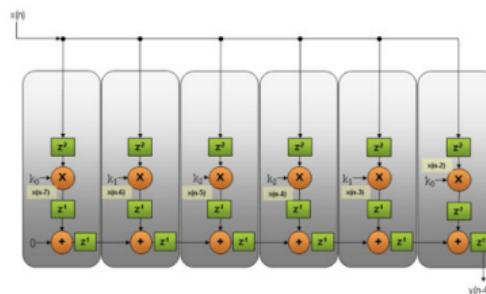
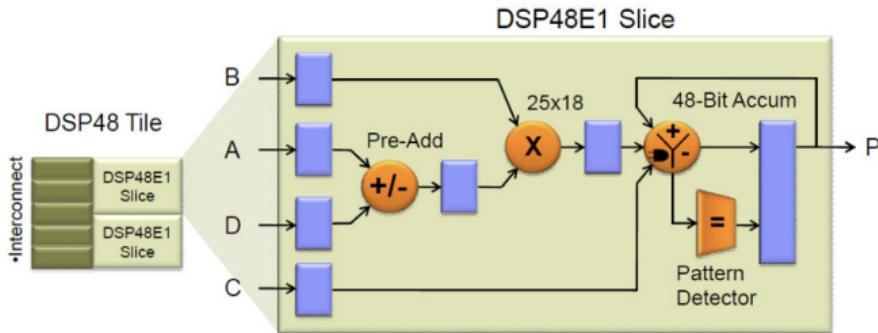
Introducción a FPGA

DSP Blocks

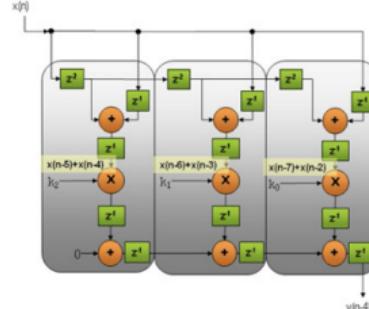
- Bloques destinados a operaciones aritméticas: sumadores, comparadores, multiplicadores, acumuladores.
- En Xilinx, un bloque DSP48 se constituye de dos slices cuyas operaciones pueden interconectarse.
- Cada DSP slice contiene:
 - Un pre-sumador de 25 bits
 - Un multiplicador de 25x18 bits (combinando dos slices se puede llegar a 50x36 bits)
 - Un acumulador de 48 bits (combinando dos slices se puede llegar a 96 bits)
 - Un detector de patrón (comparador)
 - Operaciones lógicas adicionales, lógica de redondeo simétrico, redondeo no simétrico y saturación, etc.
 - Flip flops con lógica de bypass

Introducción a FPGA

DSP Blocks



Filtro FIR (forma transpuesta) usando DSP slices



Filtro FIR con coeficientes simétricos (aprovechando pre-adder)

Introducción a FPGA

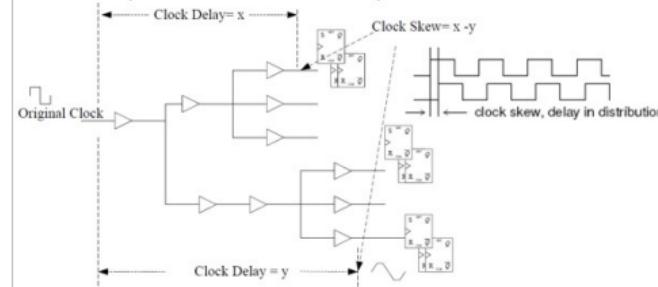
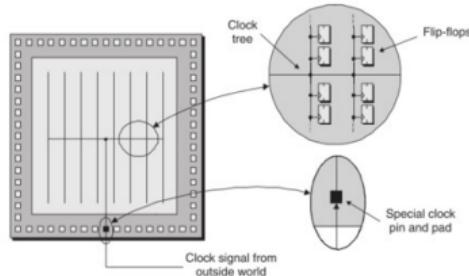
Digital Clock Manager (DCM)

- El clock se routea por caminos especiales dentro de la FPGA para reducir el “clock skew”.
Se forman árboles de clock.
- Un Clock Manager es un bloque dedicado que permiten recibir una señal de clock externa y generar una serie de clocks internos con diferentes características.
- Se basan en PLLs (pase-locked loops) o DLLs (digital delay-locked loops), de acuerdo a requerimientos de precisión, estabilidad y sensibilidad al ruido.
- Integran divisores y multiplicadores de clock y lógica de monitoreo para corrección automática de skew en los diferentes árboles.

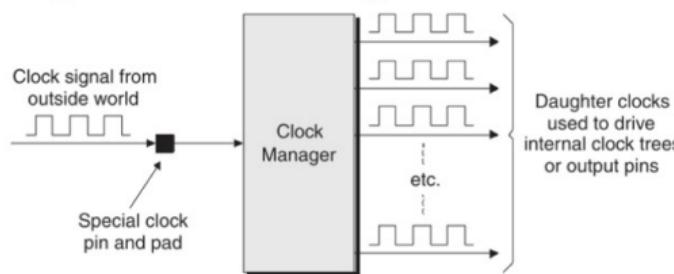
Introducción a FPGA

Digital Clock Manager (DCM)

Árbol de Clock

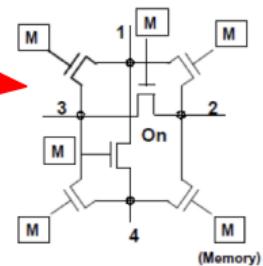
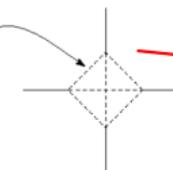
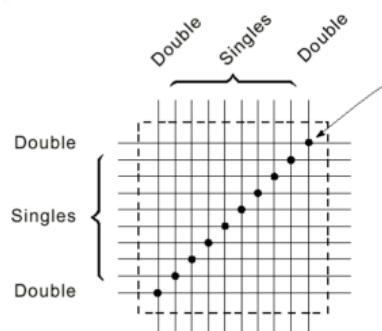
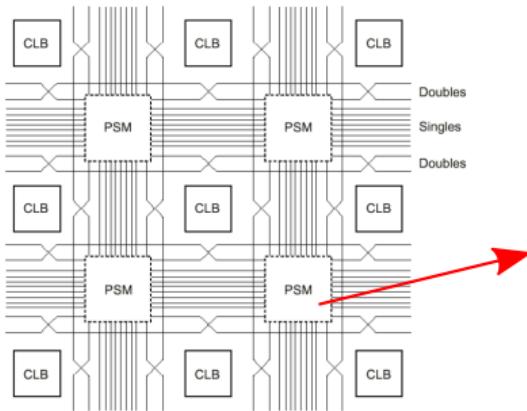


Digital Clock Manager



Introducción a FPGA

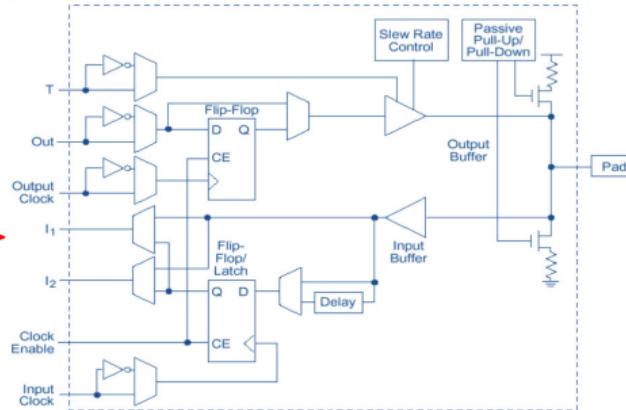
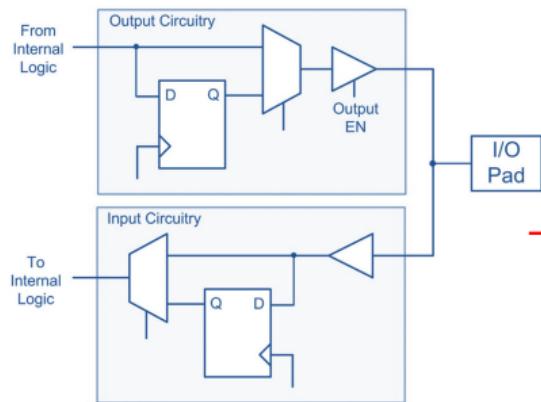
Red de interconexión. Switch Matrix



El control de los transistores que realizan el conexionado se maneja por los distintos bits de la SRAM de configuración.

Introducción a FPGA

Input-Output Blocks (IOB)

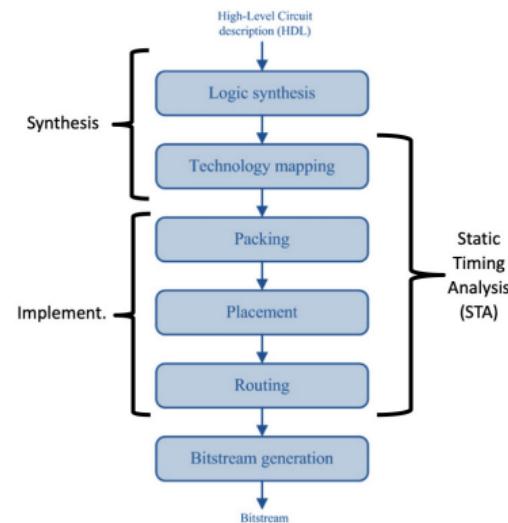


- Los bloques IOB permiten configurar cada puerto como entrada o salida.
- Además permiten conectar o no Flip Flops para soportar entradas/salidas síncronas o asíncronas.
- Permiten agregar resistores de pull up/down y mapeo a distintas tensiones según la tecnología de los periféricos a conectar.

Introducción a FPGA

Síntesis (e Implementación)

- **Síntesis lógica:** en primer lugar se convierte el HDL en un conjunto de compuertas y Flip Flops interconectados. Se aplican optimizaciones booleanas.
- **Mapeo:** Las operaciones del netlist se agrupan y mapean a los bloques lógicos provistos en la FPGA (LUTs, FFs, comparadores, etc.)
- **Clustering/Packing:** los bloques lógicos se agrupan de la forma más eficiente posible dentro de las jerarquías de la FGPA (slices, CLBs, DSPs)
- **Placement:** Se posicionan físicamente los bloques lógicos de acuerdo a diferentes estrategias (minimización de recursos o cableado, maximización de velocidad, minimización de consumo de potencia, etc.)
- **Routeo:** Se interconectan todos los bloques lógicos (datos y clocks) con algoritmos iterativos que permiten reducir la congestión y cumplir con las constraints de timing.
- **Generación del Bitstream:** Se genera el binario que definirá los valores de las SRAM (Xilinx o Altera) de configuración dentro de la FPGA.



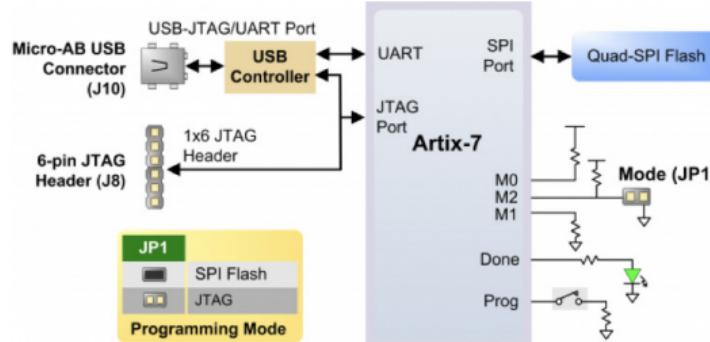
Introducción a FPGA

Configuración de la FPGA

- Existen actualmente tres tecnologías:

- SRAM (Xilinx, Altera)
- Antifuse (Actel)
- EEPROM/FLASH

- Las placas de desarrollo de Xilinx/Altera normalmente permiten configurar la SRAM de la FPGA desde el JTAG o bien desde una memoria Flash incluida en la misma placa.



Introducción a FPGA

Configuración de la FPGA

Feature	SRAM	Antifuse	E2PROM/FLASH
Technology node	State-of-the-art	One or more generations behind	One or more generations behind
Reprogrammable	Yes (in system)	No	Yes (in-system or offline)
Reprogramming speed (inc. erasing)	Fast	—	3x slower than SRAM
Volatile (must be programmed on power-up)	Yes	No	No (but can be if required)
Requires external configuration file	Yes	No	No
Good for prototyping	Yes (very good)	No	Yes (reasonable)
Instant-on	No	Yes	Yes
IP Security	Acceptable (especially when using bitstream encryption)	Very good	Very good
Size of configuration cell	Large (six transistors)	Very small	Medium-small (two transistors)
Power consumption	Medium	Low	Medium

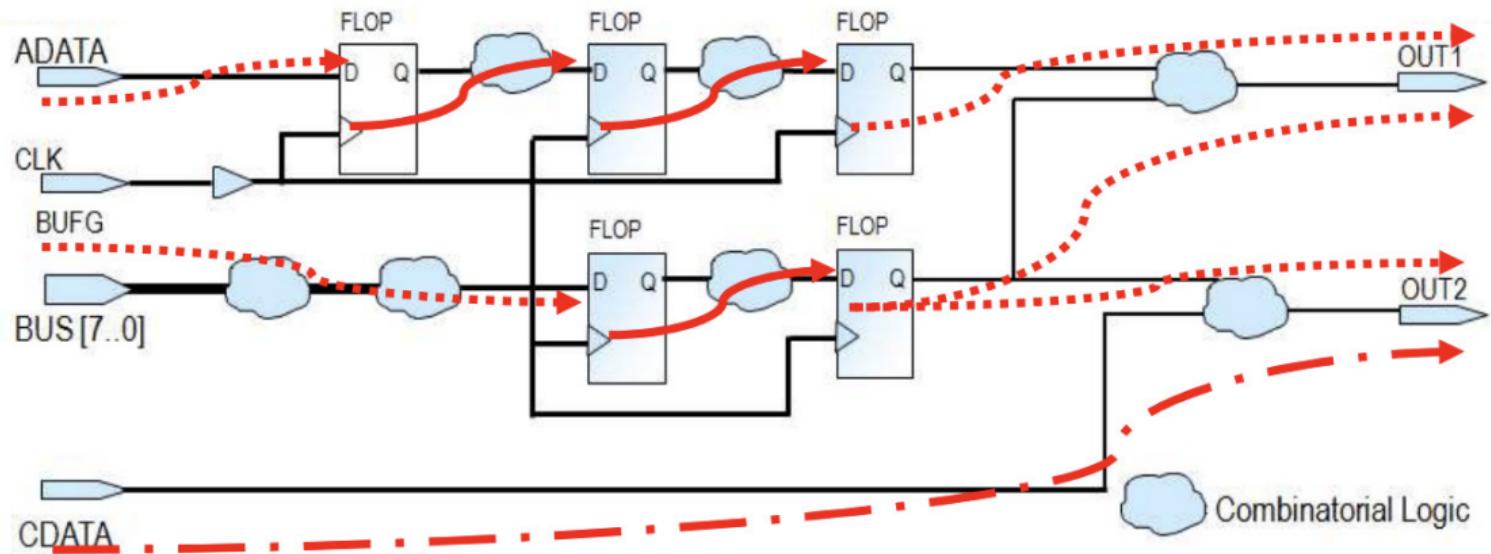
Introducción a FPGA

Timing Analysis

- El Análisis Estático de Timing (STA) es una técnica que se basa en calcular todos los retardos de entrada-salida y entre caminos internos asociados a registros para detectar violaciones de timing (setup, hold, etc.) bajo ciertas condiciones determinadas.
- Se aplica en distintas partes del flujo de síntesis: inicialmente con estimaciones basadas en cantidad de elementos lógicos y conexiones, y luego del place & route usando la información exacta de los retardos mediante la extracción de parásitos del diseño.
- Es un proceso mucho más rápido y robusto que la verificación basada en simulaciones usando vectores (GLS).

Introducción a FPGA

Timing Analysis

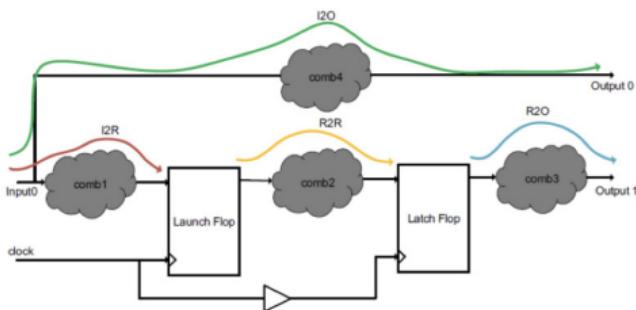
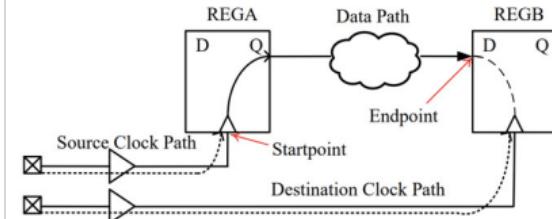


Introducción a FPGA

Timing Analysis

■ Pasos de STA:

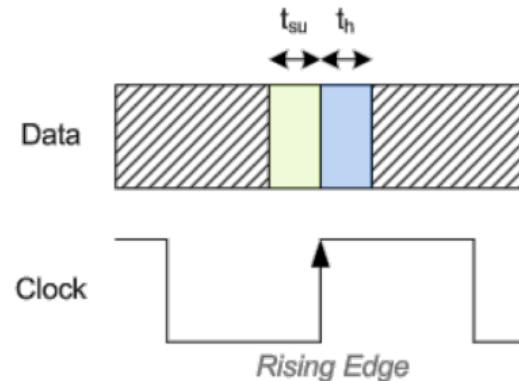
1. El diseño se divide en distintos caminos de timing (timing paths). Tipos:
 - Entrada a salida(I2O)
 - Entrada a registro(I2R)
 - Registro a registro(R2R)
 - Registro a salida(R2O)
2. Se calcula el retardo de propagación de todas las señales a lo largo de cada camino
3. Se chequean violaciones en las constraints de timing dentro del diseño y en las interfaces de entrada/ salida.



Introducción a FPGA

Timing Analysis

- Tiempo de Setup(t_{su}): El tiempo mínimo antes del cual el dato debe estar estable para ser capturado adecuadamente en el flanco activo de clock.
- Tiempo de Hold(t_h): El tiempo mínimo durante el cual el dato debe permanecer estable luego del flanco activo de clock para ser capturado adecuadamente.
- Error de timing: Violaciones del tiempo de setup o hold.
- Consecuencias de errores de timing:
 - Incertidumbre en el tiempo de propagación de los datos
 - Perdida de datos
 - Mestaestabilidad



Introducción a FPGA

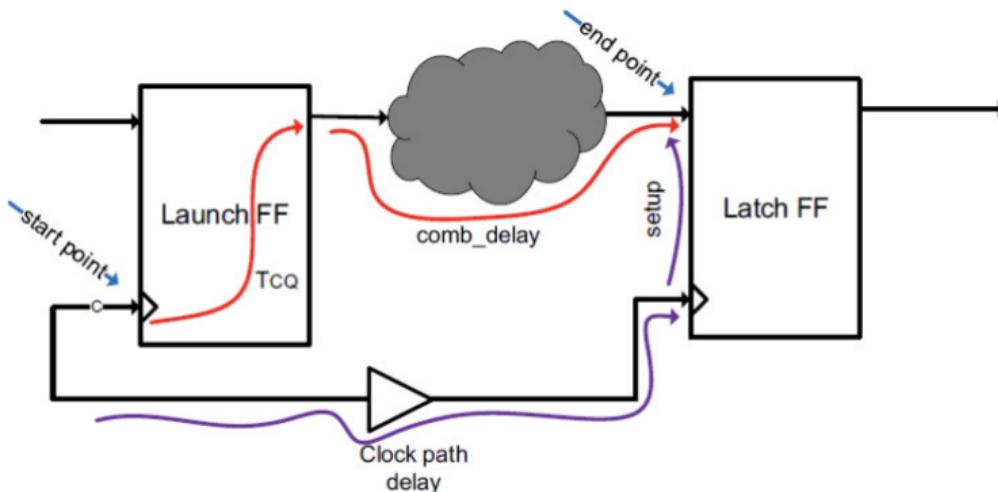
Timing Analysis

- Condición para cumplir tiempo de setup:

$$t_{CQ} + \text{comb_delay}(\text{max}) + t_{su}(\text{setup_time_of_latch_ff}) < \text{clock_path_delay}(\text{skew}) + T(\text{clk_period})$$

$$t_{CQ} + t_{\text{comb}} + t_{su} < t_{\text{skew}} + T_{\text{clk}}$$

$$\text{Setup_slack} = t_{\text{skew}} + T_{\text{clk}} - (t_{CQ} + t_{\text{comb}} + t_{su})$$



Introducción a FPGA

Timing Analysis

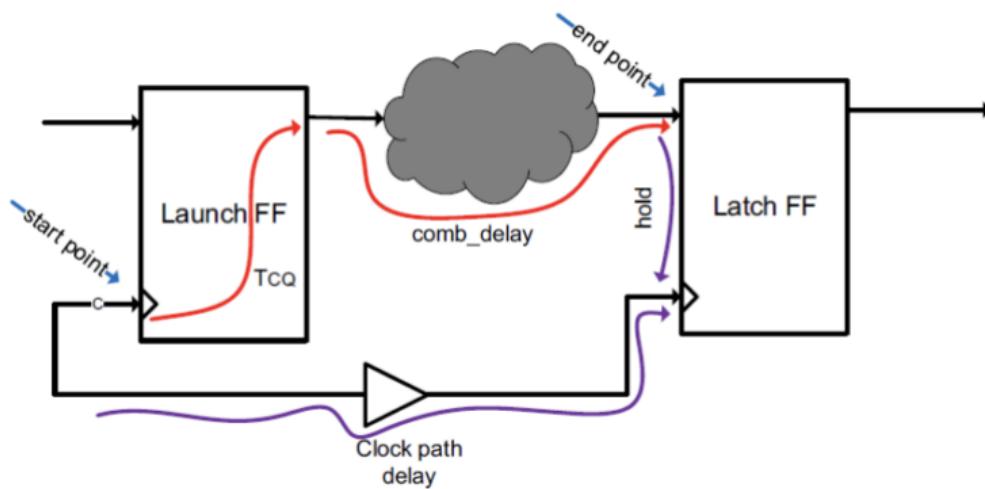
- Condición para cumplir tiempo de hold:

$$t_{CQ} + \text{comb_delay(min)} > \text{clock_path_delay(skew)} + t_h(\text{hold_time_of_latch}_ff)$$

$$t_{CQ} + t_{\text{comb}} > t_{\text{skew}} + t_h$$

$$\text{Hold_slack} = t_{CQ} + t_{\text{comb}} - (t_{\text{skew}} + t_h)$$

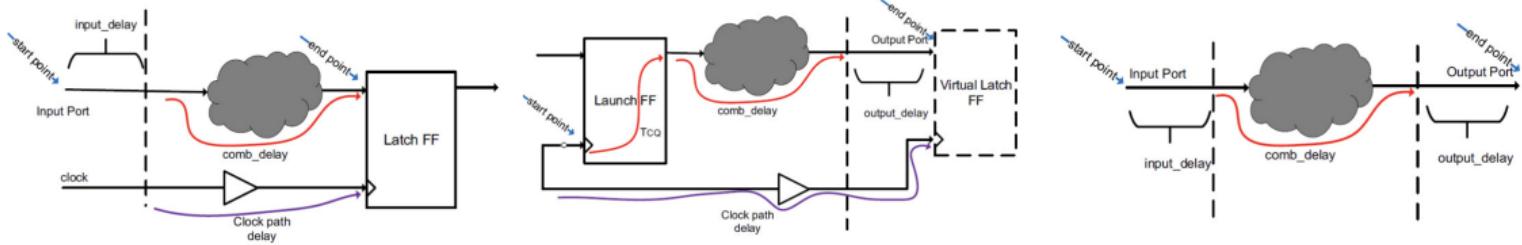
- La condición de hold no depende del período de clock.



Introducción a FPGA

Timing Analysis

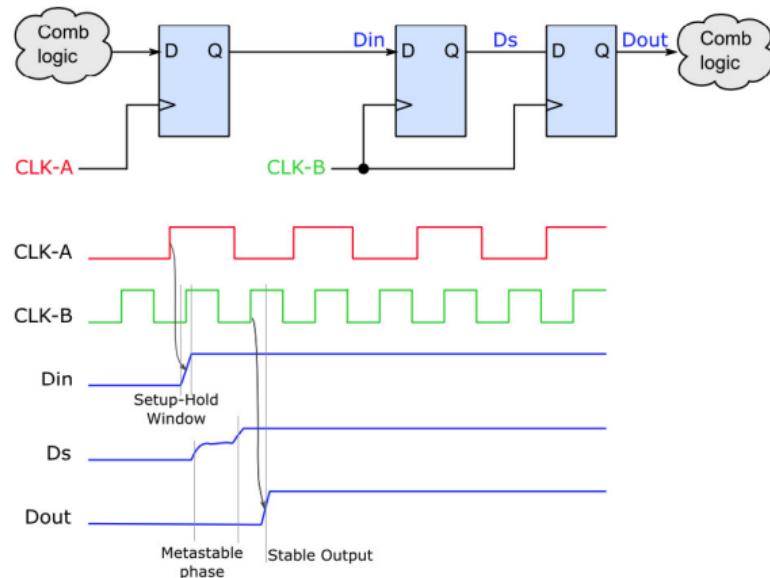
- Para los casos de caminos que inician en puertos de entrada o finalizan en puertos de salida se emplean las constraints para indicar un input_delay u output_delay a tener en cuenta en el análisis de setup y hold.



Introducción a FPGA

Metaestabilidad

- Es un estado en el cual la salida de un Flip Flop adopta un valor desconocido o metaestable.
- Puede suceder al violar las condiciones de setup o hold (violar timing).
- La permanencia en el estado metaestable no es determinística y depende del proceso tecnológico y de las condiciones de funcionamiento.
- El caso típico se da cuando hay cruces de dominio de clocks. En tal caso, se utilizan circuitos sincronizadores (double-flop syncs) para evitar que la metaestabilidad se propague a través del diseño.



Introducción a FPGA

Arreglando problemas de timing

- Además de realizar optimizaciones en la arquitectura del diseño o modificar parámetros del mismo como ser el paralelismo o frecuencia de operación, se pueden aplicar las siguientes técnicas para mejorar el timing de un diseño:
 - **Pipelining:** agregar registros para dividir caminos críticos en el diseño (aumentando la latencia)



Introducción a FPGA

Arreglando problemas de timing

- **Retiming:** reubicación de los registros del diseño para reducir caminos críticos.



Introducción a FPGA

Arreglando problemas de timing

- **Cloning:** replicación de registros para reducir fanout y por ende mejorar timing.

