

Diseño Digital Avanzado

Presentación

Dr. Ariel L. Pola

ariel.pola@mi.unc.edu.ar

August 10, 2024

Tabla de Contenidos

1. Presentación del Curso

2. Introducción

3. Estrategias de Diseño

4. Flujo de Diseño

- Desarrollo de Algoritmos
- Diseño a Nivel de Sistema, V&T
- Desarrollo de Software y Testeo
- Desarrollo de Hardware y Testeo

Presentación del Curso



Presentación del Curso

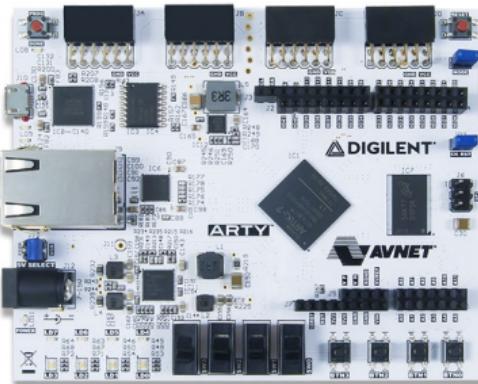
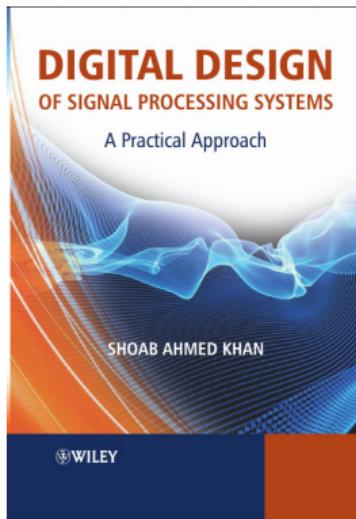
Objetivos

- Capacitar al alumno en conceptos avanzados de diseño digital en FPGA y ASIC, de manera de aplicarlos sobre un sistema digital de procesamiento de señales.
- Desarrollar competencia tales como la utilización de un variado espectro de técnicas de diseño y mapeo de algoritmos sobre FPGAs/ASICs usando un lenguaje de descripción de hardware (*HDL*). Como así también se interiorizará y podrá verificar los distintos circuitos utilizando técnicas de chequeo que le darán criterios para determinar la viabilidad de un proyecto dado ya sea en términos de velocidad, área, verificación y potencia.
- El dictado del curso se orienta a proveer al alumno de la capacidad de diseñar los sistemas de procesamiento de señal, brindándole las herramientas necesarias para que pueda seleccionar las arquitecturas que cumplan con los requerimientos del sistema final.

Presentación del Curso

Cursado

- Entrega de ejercicios y trabajos de laboratorio.
- Proyecto Final
 - Diseño e implementación en FPGA de un proyecto propuesto por el alumno o docente.



Arty A7-35T features Xilinx XC7A35TICSG324-1L

Presentación del Curso

Herramientas

- Python 3
- Vivado 2023.1
- Vitis 2023.1
- Verilog

Presentación del Curso

Aula Virtual

Classroom > CursoDDA 2023

Inicio Trabajo de clase Personas Calificaciones Personalizar

Clases impartidas Clases archivadas Ajustes

CursoDDA 2023

Código de clase u5en3tv

Anuncia algo a tu clase

Ariel Luis Pola ha publicado nuevo material: Vitis/Vivado

Próximas entregas

No tienes ninguna tarea para esta semana

Ver todo

2 comentarios de la clase

Presentación del Curso

Fechas

Clases	Sabado 9:00 a 13:00	Lunes 17:00 a 21:00	Lugar	Unidades	P - L	Horas
1	10/08/2024	12/08/2024	UNC	I		4
2	17/08/2024	19/08/2024	UNC	I	Práctico	4
3	24/08/2024	26/08/2024	UNC	I - II	Lab	2 - 2
4	31/08/2024	02/09/2024	UNC	II	Práctico	4
5	07/09/2024	09/09/2024	UNC	II	Lab	4
6	14/09/2024	16/09/2024	UNC	II		4
7	21/09/2024	23/09/2024	UNC/FF	II - III	Lab	1 - 3
8	28/09/2024	30/09/2024	UNC/FF	III	Práctico	4
9	05/10/2024	07/10/2024	UNC/FF	III - IV	Lab	3 - 1
10	12/10/2024	14/10/2024	UNC	IV		4
11	19/10/2024	21/10/2024	UNC	IV	Práctico	4
12	26/10/2024	28/10/2024	UNC	IV		4
13	02/11/2024	04/11/2024	UNC	IV		4
14	09/11/2024	11/11/2024	UNC/FF	IV - V	Lab	3 - 1
15	16/11/2024	18/11/2024	UNC/FF	V	Práctico	4
16	23/11/2024	25/11/2024	UNC/FF	V	Lab	4
17	30/11/2024	02/12/2024	UNC	V - VI		1 - 3
18	A Definir	A Definir	UNC/FF	VI	Lab	4
19	A Definir	A Definir	UNC/FF	TF	Lab	3

Feriado, se reprograma la clase según audiencia

Presentación del Curso

Contenidos Temáticos

Unidad 1

Introducción a Verilog como lenguaje de diseño de Hardware

- Introducción.
- Historia.
- Síntesis Lógica.
- Módulos.
- Partición del diseño y diseño jerárquico.
- Valores lógicos y tipos de datos.
- Los cuatro niveles de abstracción (switch, gate, dataflow y behavioral).
- Tareas y funciones.
- Aritmética signada.
- Verificación en diseño de hardware.
- Cobertura de código.

Presentación del Curso

Contenidos Temáticos

Unidad 2 Modelo de Implementación de Sistemas

- Flujo de Diseño en Sistemas.
- Principios.
- Requerimientos y especificaciones de sistema.
- Guías de codificación para descripciones comportamentales a alto nivel.
- Comparación entre arquitecturas de punto fijo y punto flotante.
- Repaso representaciones numéricas.
- Representación en complemento a 2.
- Formatos de punto flotante y punto fijo.
- Conversión de números de punto flotante a punto fijo.
- Operaciones en ambos sistemas.
- Saturación y Overflow.
- Redondeo y truncamiento.
- Soporte de Python para punto fijo.
- Formas de Filtros Digitales.
- Cuantización de los coeficientes de filtros IIR y FIR.
- Generación de vectores para verificación de código RTL.

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.

Presentación del Curso

Contenidos Temáticos

Unidad 4 Bloques Básicos de Diseño en FPGA y VLSI

- Procesadores embebidos y unidades aritméticas en FPGAs.
- Instanciación de los mismos.
- Mapeo óptimo para una tecnología dada.
- Bloques básicos de diseño en ASICs: Sumadores básicos. Half y Full adder. Ripple carry adder.
- Sumadores rápidos: Carry Look-ahead Adder, Hybrid Ripple Carry and Carry Look-ahead Adder, Binary Carry Look-ahead Adder, Carry Skip Adder, Conditional Sum Adder, Carry Select Adder, Hybrid Adders.
- División con Barrel Shifters.
- Sumadores Carry Save (CSA) y Compresores.
- Multiplicadores paralelos.
- Generación de productos parciales.
- Reducción de productos parciales.
- Multiplicadores seccionados.
- Optimización de Compresores.
- Contadores de uno o múltiples columnas.
- Multiplicadores signados en complemento a dos.
- Eliminación de extensión de signo.

Presentación del Curso

Contenidos Temáticos

Unidad 4 Bloques Básicos de Diseño en FPGA y VLSI

- Propiedad de cadena.
- Multiplicador modificado de Booth.
- Árboles de compresión para sumas multi-operando.
- Algoritmos para transformar CSA.
- Multiplicación por constantes.
- Representación canónica de dígito signado.
- Arquitecturas dedicadas de filtros FIR en forma directa, transpuesta e híbrida.
- Aritmética distribuida.

Presentación del Curso

Contenidos Temáticos

Unidad 5

Transformaciones para Alta Velocidad, Bajo Consumo y Optimización de Área

- Pipelining and Retiming.
- Conceptos.
- Métodos: de corte o por el teorema de la transferencia de retardos.
- Aplicación a sistemas con y sin realimentación.
- Retiming en herramientas de síntesis.
- Minimización del numero de registros y camino crítico.
- Descomposición de Shannon.
- C-Slow Retiming.
- Plegado y desplegado (Unfolding y Folding) de arquitecturas.
- Consideraciones sobre la velocidad de muestreo.
- Técnicas para maximizar uso de arboles de compresión, uso efectivo de recursos de FPGAs.
- Técnicas de plegado aplicadas a estructuras regulares: filtros y FFTs.

Presentación del Curso

Contenidos Temáticos

Unidad 6 Diseño de Máquinas de Estado

- Ejemplos de arquitecturas multiplexadas en tiempo.
- Arquitecturas seriales a nivel de bit y palabra.
- Arquitecturas secuenciales.
- Máquinas de estado finitas.
- Codificación de los estados.
- Máquinas de Moore y Mealy.
- Guías para la codificación de las máquinas de estado.
- C-Slow Retiming.
- Distintas representaciones gráficas de las mismas.
- Ejemplos de máquinas de estado.
- Diseño para verificación.
- Metodología.
- Métricas de cobertura.
- Métodos de reducción de potencia.

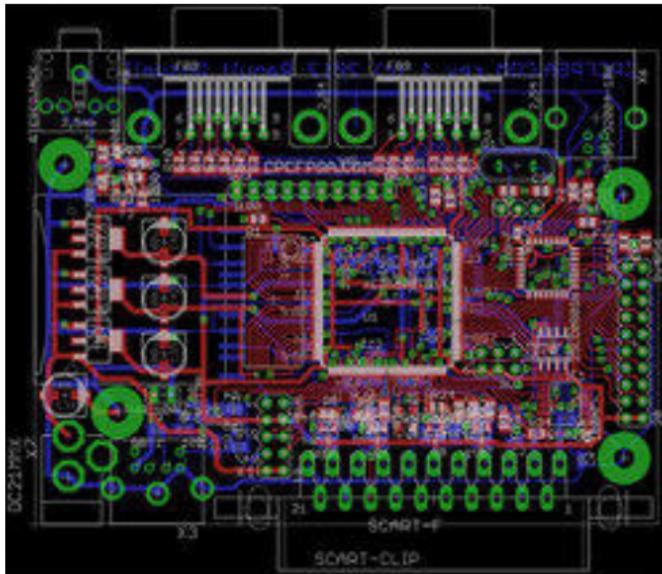
Introducción



Introducción

Qué es el Diseño Digital?

- El diseño digital es el proceso de diseño que utiliza componentes lógicos digitales para una aplicación específica.



Introducción

- La aparición del transistor en 1947 en los Laboratorios BELL, permitió un incremento vertiginoso en los avances de los sistemas electrónicos, logrando el diseño de circuitos integrados (*Integrated Circuit - IC*) extremadamente compactos.
- Desarrollo de circuitos de gran escala de integración (*Very Large Scale Integration - VLSI*) jugando un papel fundamental sobre los sistemas con los que convivimos hoy en día.
- Innumerables aplicaciones en donde los ingenieros y científicos han podido probar su capacidad de invención produciendo nuevas áreas de desarrollo.
- Un factor común que posee la mayoría de las aplicaciones es el procesamiento de señales en tiempo real, el cual exige aún más el ingenio de los investigadores.



Introducción

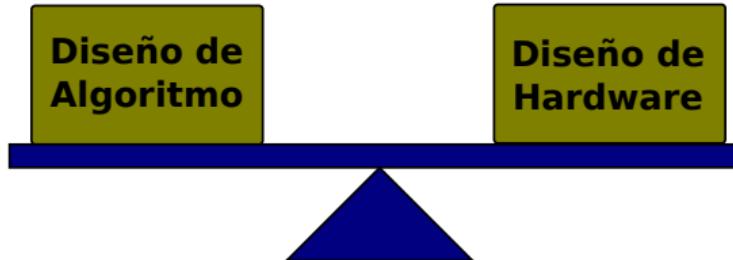


- ADAS
- Automated Driving
- In Vehicle
- Electrification
- Networking
- Video Imaging
- Database
- Data Analytics
- Avionics & Milcom
- Satcom & Space
- Radar
- Video Processing
- Routers & Switches
- Encoders & Decoders
- Video Conferencing
- HPC
- Telco Acceleration
- Wireless
- Telecommunications
- Network Security
- Emulation & Prototyping
- Robotics
- Computer Vision
- Video Surveillance

Introducción

Diseñadores

- El diseño de los algoritmos para estas aplicaciones y su implementación en VLSI siguen un proceso de evolución paralela cuyos caminos se ven influenciados por:
 - **Los diseñadores de algoritmos:** enfatizan la mejora del rendimiento.
 - **Los diseñadores de hardware:** buscan flexibilidad en la implementación de acuerdo a la tecnología empleada.
- Estos objetivos complementarios resultan en un largo proceso iterativo entre ambos equipos de diseño para poder converger a una arquitectura implementable y que cumpla con los requisitos de desempeño.



Objetivos Críticos

- **Área** ocupada por compuertas lógicas (*logic gates*) e interconexiones
- El retardo del **camino crítico** (critical path) de la ruta más larga a través de la lógica.
- Disipación de **potencia** de las compuertas lógicas
- El grado de **capacidad de prueba** (testability) del circuito, medida en términos del porcentaje de las fallas cubiertas por un conjunto específico de vectores de pruebas para un modelo de falla apropiado.

Equilibrio Óptimo

- **El arte del diseño digital es encontrar el equilibrio entre estos cuatro pilares**

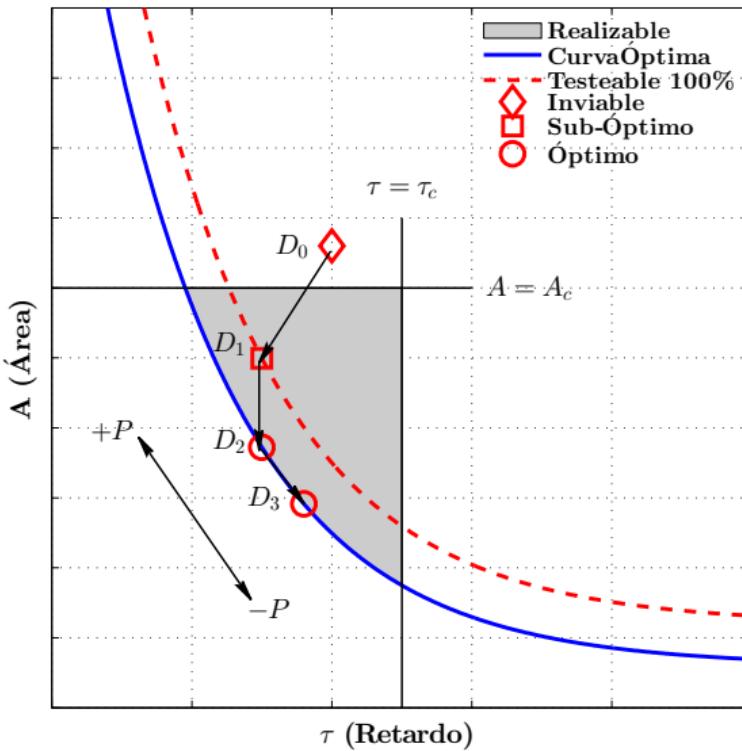
El Arte del Diseño Digital

Ejemplo

- Se definen los límites de área (A_c) y retardos ($delay - \tau_c$) máximos según los requerimientos, lo cual demarca un área o región.
- Cualquier punto dentro de esta zona genera un circuito realizable.
- Normalmente, el diseño comienza como un diseño no realizable (D_0) fuera de los límites establecidos, que está muy alejado de la curva óptima (línea continua).
- El diseño se mejora rediseñando ciertos parámetros para mejorar tanto área como retardo (D_1 es realizable pero sub-óptimo, D_2 y D_3 son realizables y óptimos).
- Moviéndonos hacia la izquierda por la curva óptima (mayor área y menor retardo) incrementamos la potencia, ya que se asocia con un incremento en la complejidad.
- En dirección contraria reducimos la potencia, ya que lo asociamos con una reducción en la velocidad del sistema.
- Cada diseño tiene su propia curva de capacidad de prueba, generalmente más alta que la curva óptima del diseño (línea de trazo).
- Diseños sobre esta curva es probable que tengan cierto grado de redundancia en los circuitos implementados.

El Arte del Diseño Digital

Relación entre Área, Retardo, Potencia y Capacidad de Prueba



El Arte del Diseño Digital

- Todo lo mencionado, es generalmente cierto cuando los requerimientos de velocidad, potencia y área se encuentran dentro de las capacidades de la tecnología.
- Esto permite desacoplar o minimizar la interacción a nivel de sistema y el diseño VLSI y concentrarnos en optimizar los aspectos críticos del diseño.
- Si los requerimientos sobrepasan los límites permitidos por la tecnología, es necesario evaluar nuevamente los algoritmos haciendo que el ciclo de diseño vuelva a comenzar.
- En síntesis, esto nos lleva a que los diseñadores de algoritmos tengan que abordar los problemas teniendo en cuenta aspectos de implementación críticos que pueden definir la eficiencia y desempeño del diseño final del sistema.



Estrategias de Diseño



Estrategias de Diseño

Qué elegir?

- A nivel del sistema, el diseñador tiene un espectro de opciones de diseño muy amplio.
- En la etapa de diseño del sistema el diseñador divide el algoritmo según la tecnología y el destino.

↑ Flexibilidad - ↓ Potencia

■ GPP

- Flexibilidad de programación.
- Baja complejidad de cálculo.
- Menor consumo.

■ DSP

- Flexibilidad de programación.
- Algoritmos de cálculos complejos no estructurados.

↑ Flexibilidad - ↑ Potencia

■ FPGA

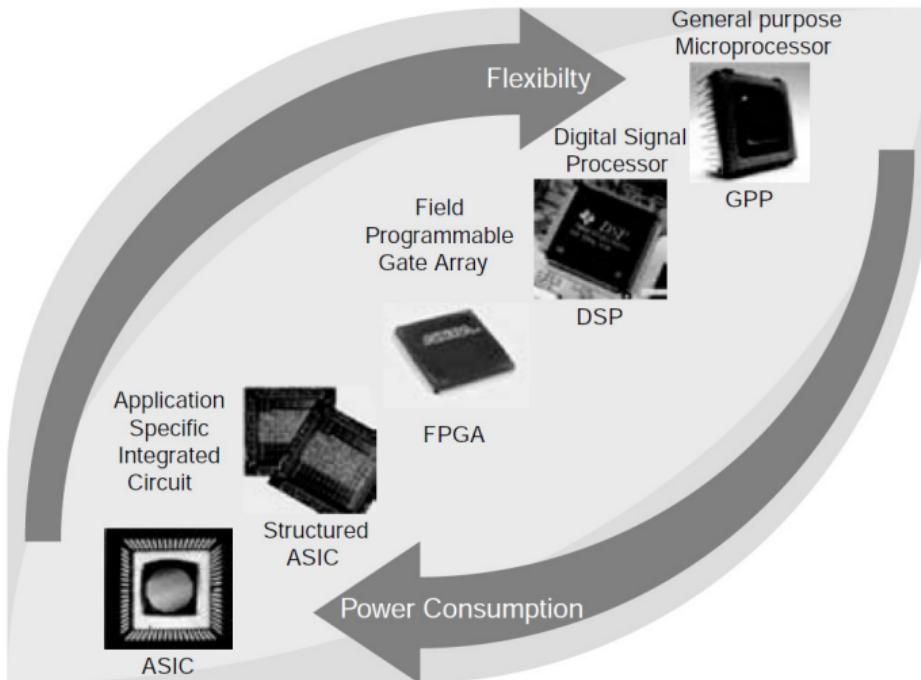
- Algoritmos de cálculos complejos estructurados.
- Flexibilidad en el diseño de aplicaciones.
- La programabilidad es más compleja.

↓ Flexibilidad - ↑ Potencia

■ ASIC

- El desempeño es el enfoque principal.
- No presenta flexibilidad en la programación.
- Mayor consumo de potencia.

Estrategias de Diseño



Relación de flexibilidad y potencia entre los dispositivos.

Estrategias de Diseño

Dispositivos



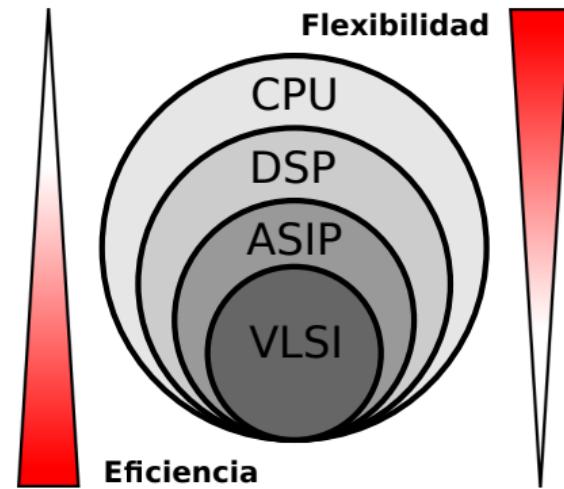
- Un **ASIC** es un circuito integrado creado específicamente para resolver una aplicación de cálculo precisa
 - Construido para un cómputo específico
 - Ciclo de planificación complejo y largo
 - Costos iniciales altos, después costos por unidades bajos
 - Ocupan poco espacio y disipan poca potencia
 - 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

Estrategias de Diseño

Flexibilidad y Eficiencia

- La parte del código con cálculos complejos de la aplicación se asigna en GPPs.
- Los algoritmos de procesamiento de señales no estructurados se asignan al DSPs.
- Los algoritmos estructurados se asignan en FPGAs.
- Los algoritmos estándar se implementan en ASICs.



Relación de flexibilidad y eficiencia.

Estrategias de Diseño

Decisiones de Diseño y Complejidad

- Las decisiones a nivel conceptual son menos complejas pero tienen un impacto importante en el diseño final.
- La complejidad del diseño crece a medida que avanzamos en el ciclo del diseño.

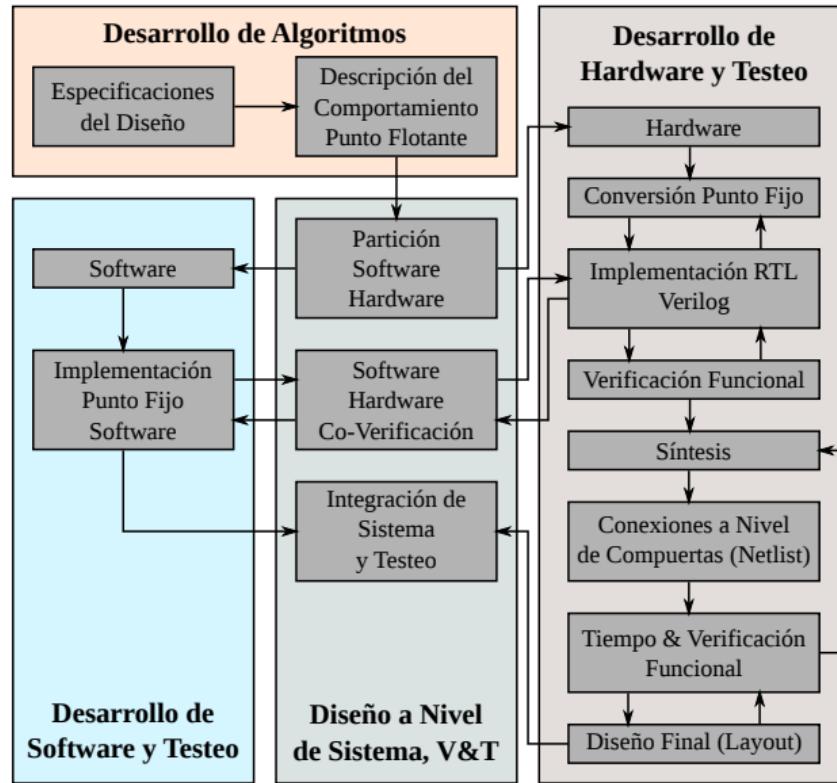


Relación entre impacto del diseño y complejidad.

Flujo de Diseño

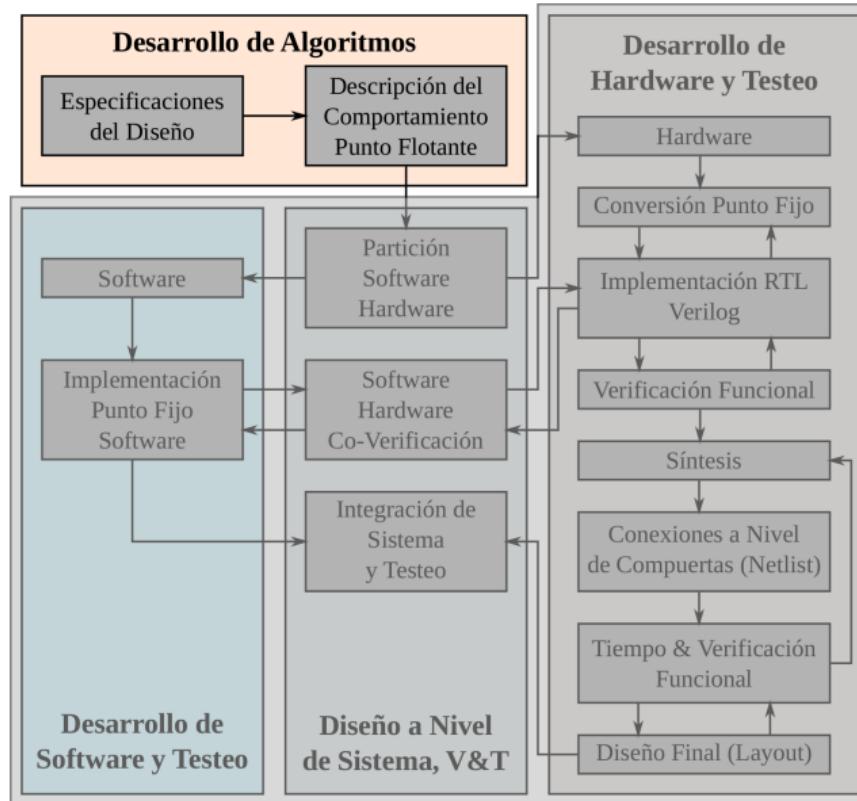
Flujo de Diseño Digital

Clasificación



Flujo de Diseño Digital

Desarrollo de Algoritmos



Flujo de Diseño Digital

Desarrollo de Algoritmos

Especificaciones del Diseño

- El primer paso en el flujo de diseño consiste en captar los requerimientos y especificaciones del sistema.
- Requerimientos & Especificaciones
 - **Requerimientos:** *Características que se desea que un sistema posea.*
 - Máxima tasa de datos (*data rate*) soportada en bits por segundo (*bits per second - bps*) en el transmisor.
 - La tasa de error de bit (*Bit Error Rate - BER*) permitida.
 - El ancho de banda del canal.
 - Las frecuencias de portadora e intermedias.
 - Límite de potencia.
 - Las interfaces.
 - **Especificaciones:** *Como las aplicaciones y componentes del sistema hacen para cumplir con los requerimientos.*
 - Velocidad de muestreo.
 - Una medida cuantitativa de rendimiento en presencia de ruido.
- Estas definiciones acotan al diseñador a un grupo de opciones de diseño y algoritmos.

Flujo de Diseño Digital

Desarrollo de Algoritmos

Ejemplo de Requerimientos para Aplicación UHF

- Output power: 2W
- Spurious emission: < 60dB
- Harmonic suppression: > 55dB
- Frequency stability: 2 ppm or better
- Reliability: > 10.000 hours MTBF minimum - < 30 minutes MTTR

Ejemplo de Especificaciones Tx y Rx

- Frequency range BW: 420MHz to 512MHz
- Data rate: Up to 512 kbps multi channel non line of sight
- Channel: Multi path with 15 ms delay spread and 220 km/h relative speed between transmitter and receiver
- Modulation: OFDM supporting BPSK, QPSK and QAM
- FEC: Turbo codes, convolution, Reed Solomon
- Frequency hopping: > 600 hops/s, frequency hopping on full hopping band

Flujo de Diseño Digital

Desarrollo de Algoritmos

Descripción del Comportamiento

- Los requisitos relacionados con el diseño digital se envían a desarrolladores de algoritmos.
- Los desarrolladores de algoritmos toman estos requisitos y explorar las técnicas de comunicación digitales que pueden cumplir con las especificaciones indicadas.
- Estos algoritmos son generalmente codificados en las herramientas de modelado de comportamiento utilizando diversas funciones que ayudan al diseñador.
- Tienen la finalidad de analizar el comportamiento y el desempeño del algoritmo que se esta diseñando.
- Se utiliza el criterio de operaciones en punto flotante.

Flujo de Diseño Digital

Desarrollo de Algoritmos

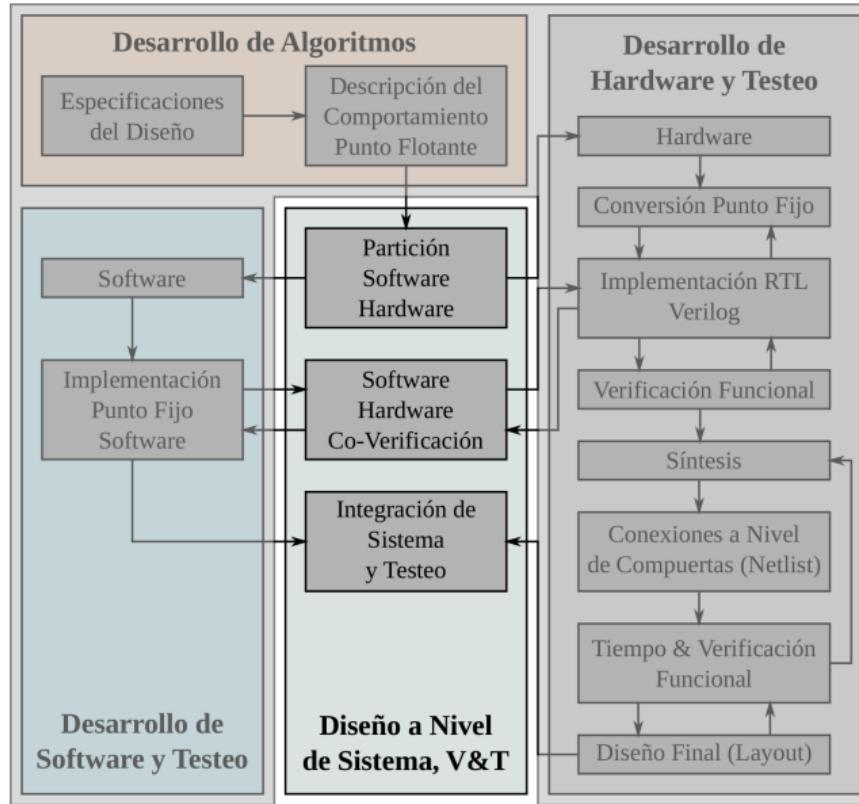
Descripción del Comportamiento

- **MATLAB** (*MAT*rix *LAB*oratory, *laboratorio de matrices*): es una herramienta de software matemático
- **Python**: es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.
- **C/C++**: C es un lenguaje para programadores en el sentido de que proporciona una gran flexibilidad de programación y una muy baja comprobación de incorrecciones, de forma que el lenguaje deja bajo la responsabilidad del programador acciones que otros lenguajes realizan por si mismos.
- **Definición de parámetros tales como ancho de banda de los filtros y número de coeficientes, factor de sobre-muestreo, paralelismos, frecuencias de relojes, etc.**
- Partición del diseño en Software y Hardware.



Flujo de Diseño Digital

Diseño a Nivel de Sistema, V&T

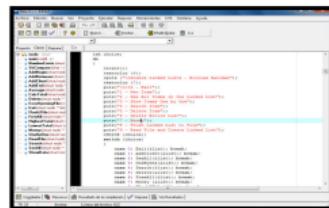


Flujo de Diseño Digital

Diseño a Nivel de Sistema, V&T

Partición en Software y Hardware

- Un paso crítico es la partición de la aplicación modelada en la etapa anterior en **hardware y software**.
- La división es impulsada por factores como el rendimiento, costo, la disipación de energía, tiempo en el mercado, etc.
- Aunque una implementación de software es más fácil de codificar e implementar, en muchas instancias de diseño puede que no sea eficiente.
- El software también se lo sub-divide dependiendo el dispositivo de destino, el cual puede ser implementado en un micro-procesador embebido o en un DSP.



SOFTWARE



HARDWARE

Flujo de Diseño Digital

Diseño a Nivel de Sistema, V&T

SW y HW Co-Verificación

- En el nivel más básico, la co-verificación (*verificación conjunta*) de SW/HW [?] significa que el software del sistema embebido se ejecuta correctamente en el hardware del sistema embebido.
- Ejecutar el software en el hardware para asegurarse de que no hay errores de hardware antes de que el diseño se mande a la fabricación.
- El objetivo se puede lograr usando muchas formas diferentes que se diferencian principalmente por la representación del hardware, el motor de ejecución utilizado y cómo se modela el microprocesador.
- La herramienta de co-verificación, debe proporcionar al menos una depuración de software usando un depurador de código fuente y una depuración de hardware usando formas de onda.
- La co-verificación se suele denominar prototipado virtual (*virtual prototyping*), ya que la simulación del diseño de hardware se comporta como el hardware real, pero a menudo se ejecuta como un programa de software en una PC.
- Ejecutar el software en cualquier representación del hardware que no sea el producto final, (chip o FPGA) califica como co-verificación.

Flujo de Diseño Digital

Diseño a Nivel de Sistema, V&T

SW y HW Co-Verificación

- La co-verificación proporciona dos beneficios principales:
 - Permite que el software que depende del hardware sea probado y depurado antes de que un prototipo esté disponible.
 - También proporciona un estímulo de prueba adicional para el diseño de hardware. Este estímulo adicional es útil para aumentar el banco de prueba (*testbenches*) desarrollado por ingenieros de hardware, ya que es el verdadero estímulo que se producirá en el producto final.
- Estos beneficios de co-verificación abordan el problema de integración de hardware y software y se traducen en:
 - Un cronograma de proyecto más corto.
 - Un proyecto de menor costo.
 - Un producto de mayor calidad.

Flujo de Diseño Digital

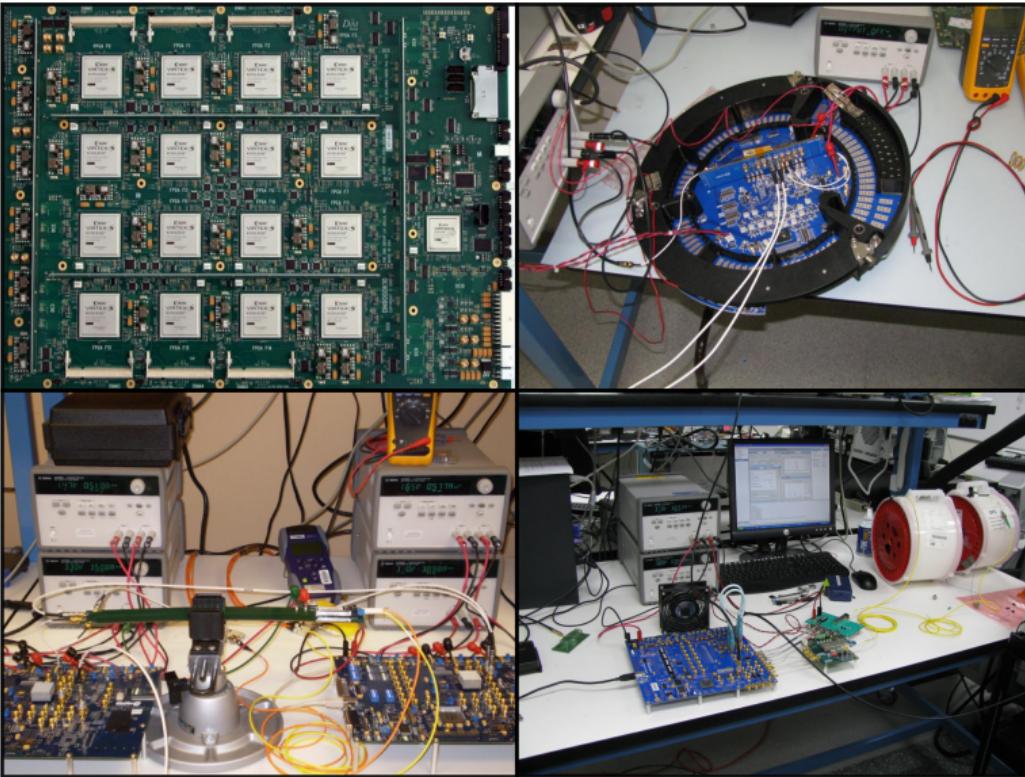
Diseño a Nivel de Sistema, V&T

Integración de Sistema y Testeo

- Completa la etapa final del desarrollo.
- Consiste en la elaboración de una placa de circuito impreso (Printed Circuit Board - PCB) que integra todas las interfaces (PCI, Ethernet, USB) y módulos complementarios (DSPs, FPGAs, ASICs) del producto final.
- El primer desafío es la verificación de cada módulos instanciado y sus inter-conexiones.
- En muchos casos se emplea la interfaz gráfica de usuario (*Graphic User Interface - GUI*) facilitando la revisión inicial de los componentes.
- Se verifica que cumpla con todos los requerimientos y especificaciones del diseño, sometiendo la aplicación a pruebas reales de trabajo (variaciones de temperatura, vibraciones, etc).

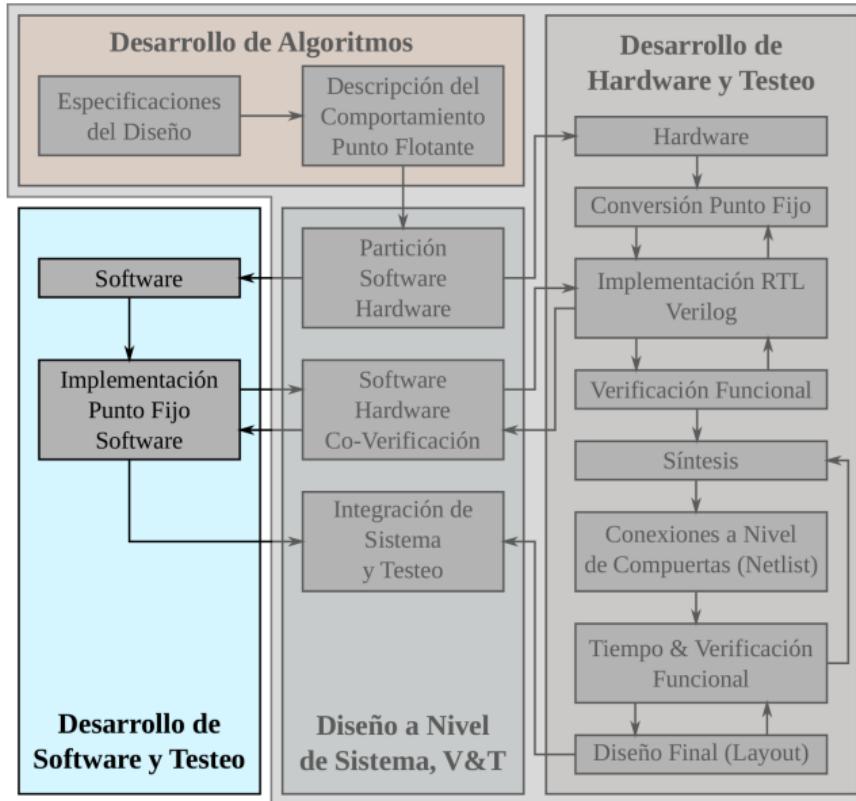
Flujo de Diseño Digital

Diseño a Nivel de Sistema, V&T



Flujo de Diseño Digital

Desarrollo de Software y Testeo



Flujo de Diseño Digital

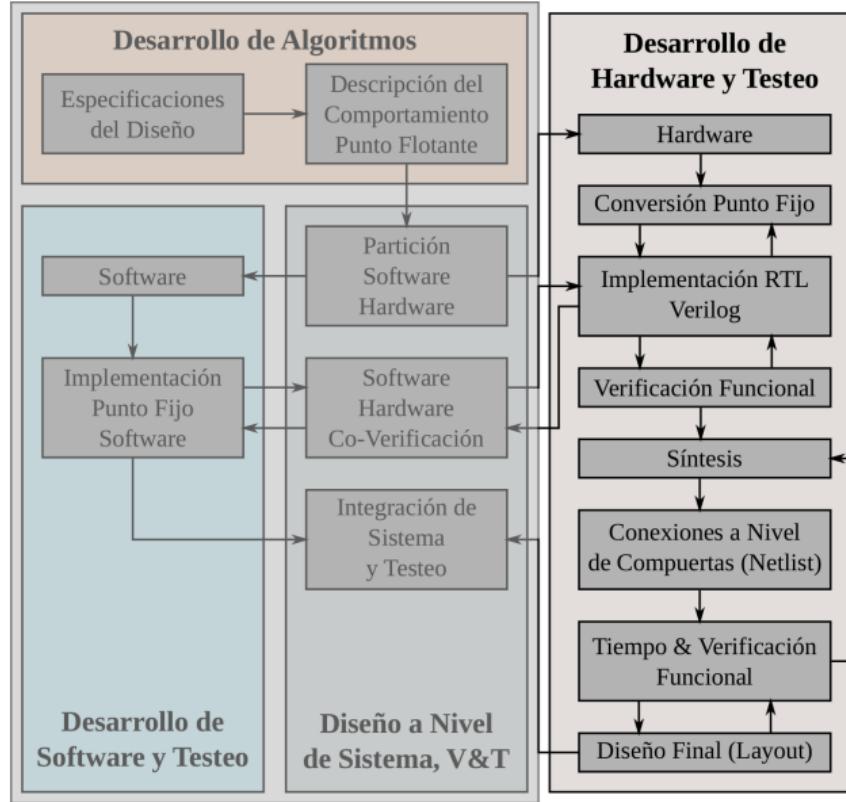
Desarrollo de Software y Testeo

Implementación

- El diseñador toma el código de punto flotante que está diseñado en el modelo inicial y lo convierte en formato de punto fijo (“char” (8 bits), “short” (16 bits) o “int”(32 bits)).
- La elección del tipo de formato utilizado en cada variable no tiene un impacto significativo en el diseño del sistema comparado con la etapa de desarrollo de Hardware.
- Se emplean herramientas de software para la compilación dependiendo de la aplicación final.
- La utilización de emuladores del entorno de trabajo permiten la verificación del desarrollo. Estos emuladores se ejecutan en una PC permitiendo generar el entorno de trabajo real del diseño final.

Flujo de Diseño Digital

Desarrollo de Hardware y Testeo

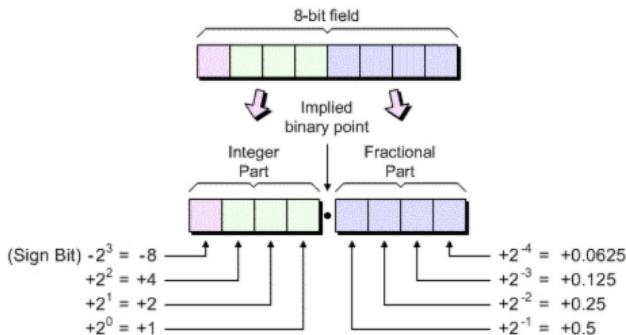
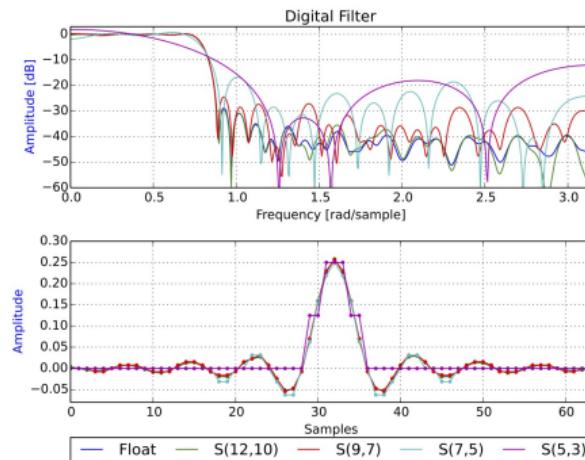


Flujo de Diseño Digital

Desarrollo de Hardware y Testeo

Conversión en Punto Fijo

- Conversión a punto fijo con mayor rigurosidad en el mapeo de hardware, ya que una mala elección de la resolución implica un incremento directo en el hardware.
- Verificación de desempeño para evaluar la degradación que produce la cuantización de las variables.
- El simulador en punto fijo es diseñado para modelar el comportamiento del hardware final. Es decir, incluye referencias de relojes, registros y el modelado del comportamiento en paralelo de la arquitectura que se implementará.

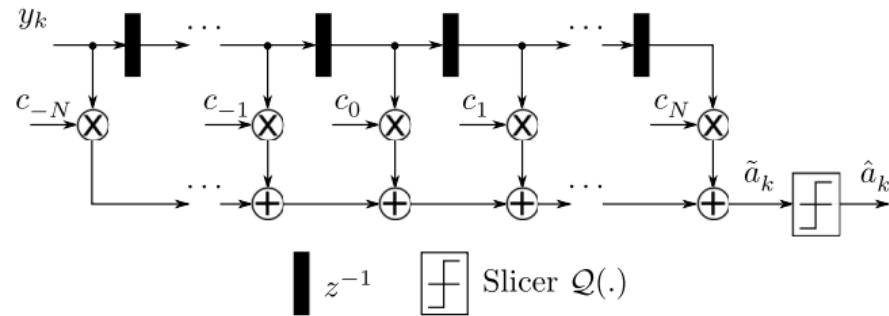


Flujo de Diseño Digital

Codificación RTL

Verilog

- Codificación en lenguaje de descripción de hardware (*Hardware Description Language - HDL*) (por ejemplo, Verilog o VHDL).
- El diseño de nivel superior destaca la partición del sistema en sus diversos componentes.
- Cada componente se define adicionalmente en el nivel de transferencia de registro (*Register Transfer Level - RTL*).
- Este es un nivel de abstracción donde el diseñador digital especifica todos los registros y elabora cómo los datos fluirán a través de estos registros.

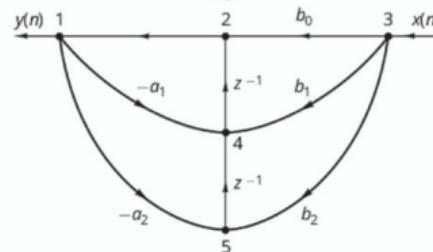
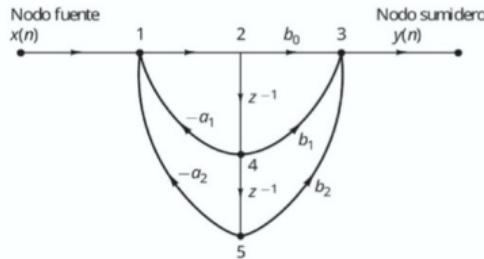
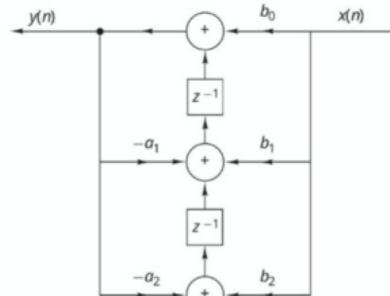
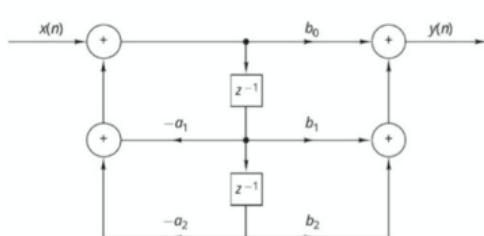


Flujo de Diseño Digital

Técnicas de Implementación

Definición de arquitecturas

- IIR: Forma directa I, forma directa II, forma directa II transpuesta y cascada.

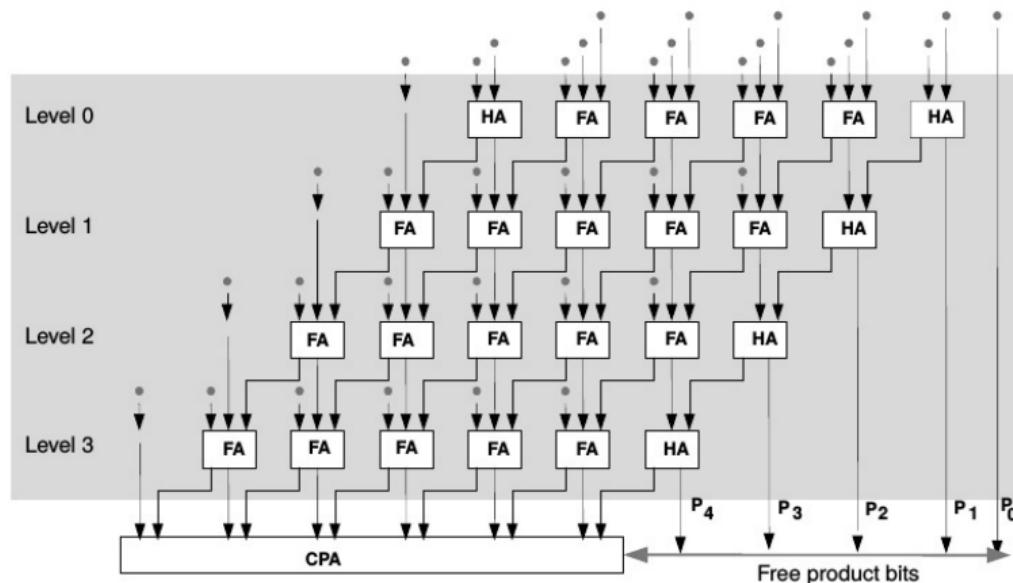


Flujo de Diseño Digital

Técnicas de implementación

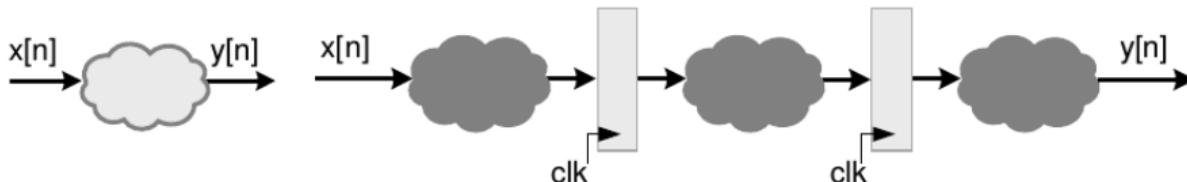
Definición de bloques básicos

- Sumadores: Ripple Carry, Carry Look-ahead y Carry Save Adder.
- Multiplicadores: Binarios y Booth Modificado.
- Optimizaciones: Eliminación de extensión de signo y reducción de productos parciales.

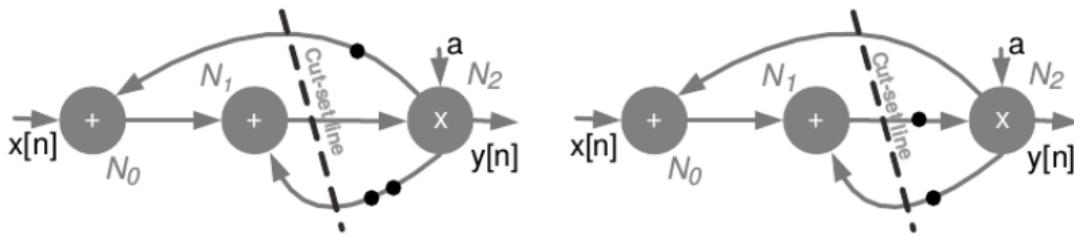


Flujo de Diseño Digital

Técnicas de implementación



Pipelining. Incluir registros entre la lógica combinacional.



Retiming. Mover los registros con el objetivo de reducir complejidad.

Flujo de Diseño Digital

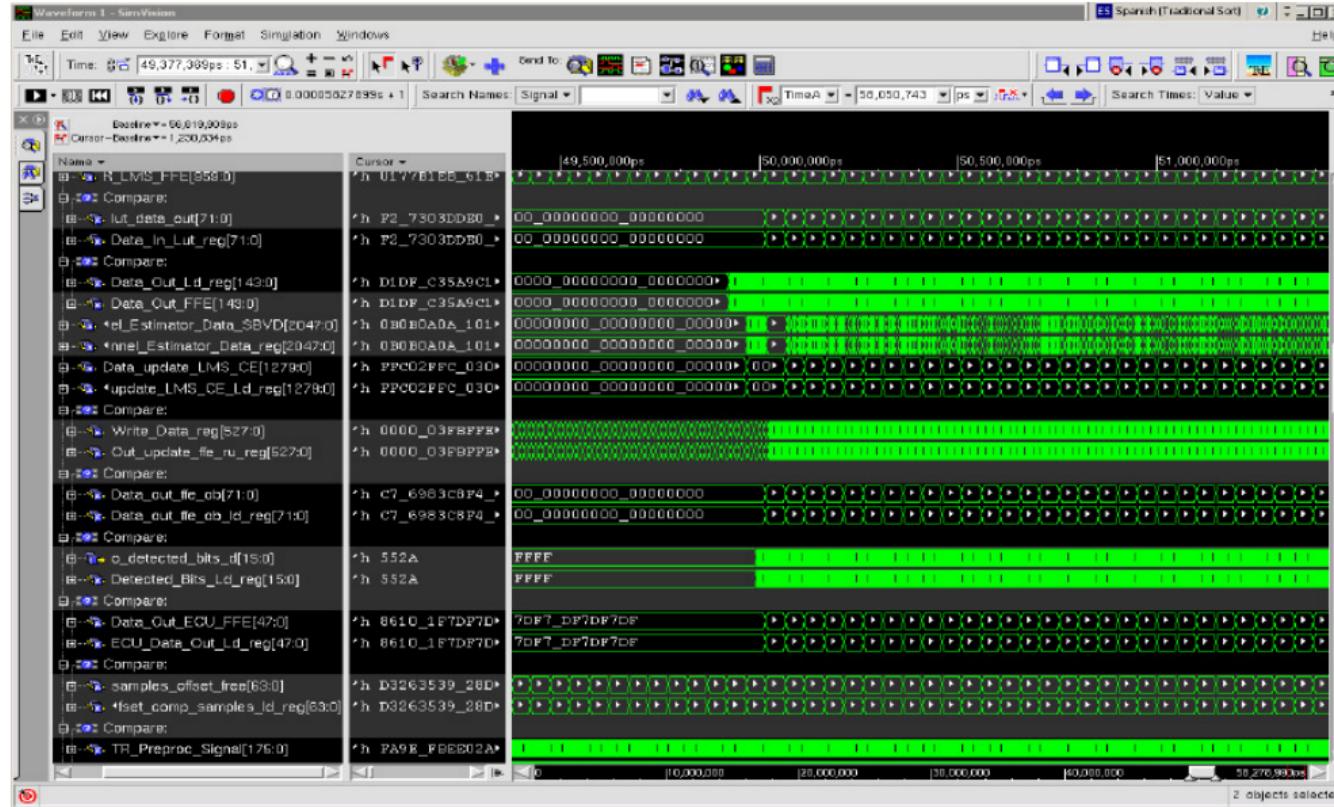
Desarrollo de Hardware y Testeo

Verificación Funcional

- Etapa en la cual se verifica el comportamiento del diseño RTL.
- Técnicas de Verificación
 - **Apareamiento de vectores (*vector matching - VM*):** Consiste en la generación de vectores de datos empleando el simulador de punto fijo que modela el hardware.
 - **Vectores de estímulos:** Se los emplea como señal de entrada al módulo diseño en Verilog.
 - **Vectores de salida:** Se los emplea para comparar el comportamiento del simulador con el módulo diseñado en Verilog.
 - **Lenguaje de Assertions:** Destina a especificar el comportamiento esperado del diseño.
 - **Cobertura de código (*coverage*):** Da una medida cuantitativa de cuanto esta verificado del funcionamiento del módulo.

Flujo de Diseño Digital

Vector Matching

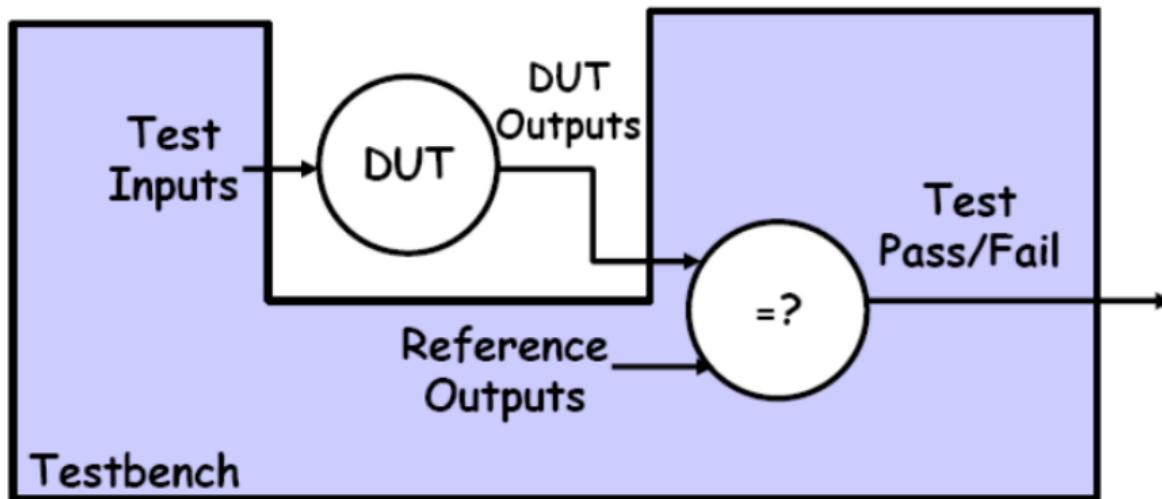


Flujo de Diseño Digital

TestBench

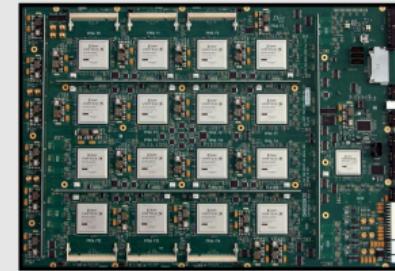
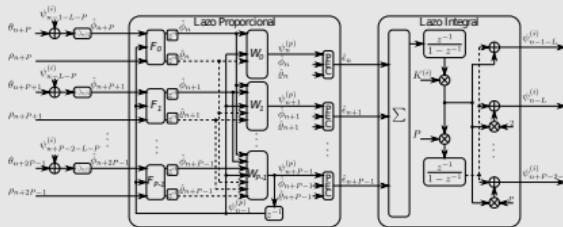
Unit Test

- Es necesario estimular la unidad bajo prueba (Design Under Test- DUT) con entradas de prueba, y comparar las salidas con los resultados esperados.
- Se usan otros programas, escritos en lenguajes de propósito general como C++, fuera del testbench escrito en Verilog o VHDL para generar y chequear entradas y salidas.

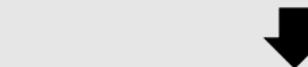
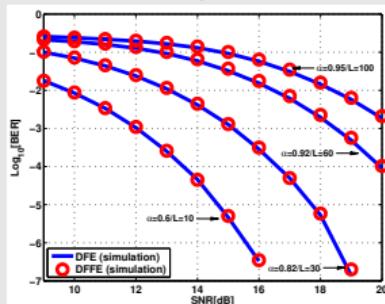


Flujo de Diseño Digital

Verificación Funcional



Emulación de Sistemas



Evaluación de Desempeño

Flujo de Diseño Digital

Síntesis

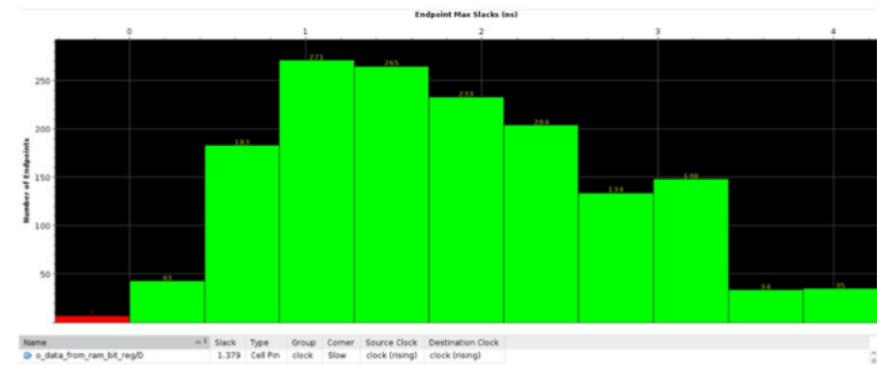
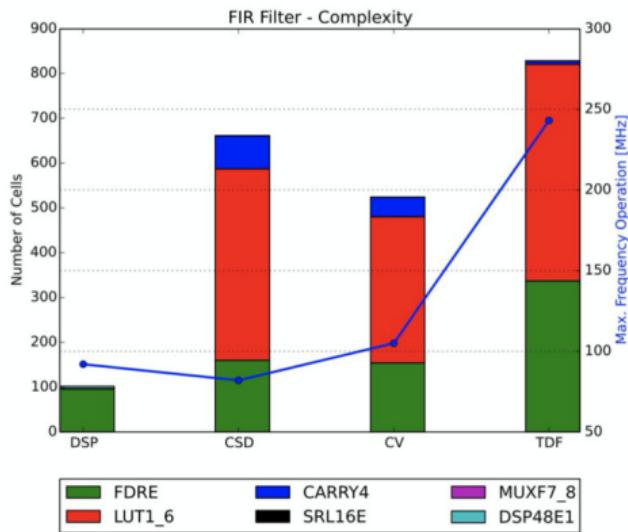
Diseño de Constraints

- El diseño de los archivos de constraints es uno de los más críticos e importantes ítems en el diseño de un SoC.
- El proceso de síntesis toma el RTL y lo traduce en conexiones a nivel de puerta (*netlist*).
- El usuario especifica restricciones de diseño y la tecnología de destino en forma de una biblioteca de celdas estándar (*cell library*).
- Requiere un conocimiento del **diseño circuital, su estructura y funcionalidad**, para poder obtener el mejor compromiso entre velocidad, área y consumo.
- En general usar constraints muy **ajustados** generan diseños con mayor consumo.
- Los constraints permiten **guiar** las herramientas EDA en las diferentes fases de optimización, estos nos permiten evaluar el diseño y verificar si cumple con nuestras especificaciones en términos de velocidad, área y consumo.
 - **Constraints de timing:** Período de clocks, incertidumbres, IO delays, false paths, multicycle paths, etc.
 - **Constraints físicos:** Dimensiones del floorplan, regiones, coordenadas de puertos, power grid, restricciones de placement, etc.
 - **Constraints Generales:** Transiciones máximas, capacidades máximas, fanout máximo, máxima carga a la salida del bloque, máxima área, máxima potencia.

Flujo de Diseño Digital

Síntesis - Verificación de Timing

- Se sintetiza el código, donde la salida de este proceso es el número de celdas y timing del mismo. Para esto se determinan constraints de síntesis (freq. de trabajo, potencia, área, etc)



Flujo de Diseño Digital

Desarrollo de Hardware y Testeo

Diseño Final (Layout)

- Es la última etapa del flujo digital.
- El *layout* es la representación de un circuito integrado en términos de formas geométricas planas que corresponden a los patrones de capas de metal, óxido o semiconductor que integran los componentes del circuito integrado.
- En aplicaciones que utilizan FPGA, la herramienta de síntesis también genera la disposición final de los componentes.

Flujo de Diseño Digital

Implementación

