Técnicas de

Programación Hardware:

CAD para FPGAs y CPLDs

Clase 6: Soft de PPR y análisis de diseños

Por: Nelson Acosta & Daniel Simonelli

UNICEN - Tandil - 1999

Etapas de Diseño (1)

- Diseño Inicial
- Traducción al formato XNF
- Partición
- Place & Route
- Cálculo de performance

- Verificación del diseño

Etapas de Diseño (2)

Diseño Inicial:

- Captura esquemática *(Editores CAD, Foundation, Viewlogic, Cadence, Spice, ...)*
- HDL *(VHDL, Verilog, Handel-C, Hardware-C, ABEL, Ecuaciones Booleanas, ...)*
- Sistemas de generación automática para grupos de aplicaciones *(multiplicadores, ALU, máquinas de estado, procesadores a medida, controladores a medida, ...)*

Etapas de Diseño (3)

Traducción al formato **XNF**:

- Proceso automático en el M1.
- XNF es un formato de NET LIST *(lista de conexiones)*.
- XNF implementa: (a) *Componentes de la librería BÁSICA*. (b) *MACROS utilizando componentes básicos*.
- Permite diseños jerárquicos.
- Los componentes BÁSICOS son representados por *ecuaciones Booleanas*.

Etapas de Diseño (4)

Formato **XNF**:

- **Encabezado** *(descripción del módulo, FPGA)*
- **Lista de componentes**
  - Comando SYM
  - Descripción de pines del componente *(como parámetros por nombre)*
  - Conexión entre componentes *(por nombre de la señal o cable)*
- **Lista de pines** *(de entrada/salida, de alimentación, relojes, ...)*

Etapas de Diseño (5)

Traducción a **XNF**:

- **Esquemas** *(traducción directa, ya que se diseña por componentes, y se traduce a componentes).*
- **Generación Automática** *(directa: se puede generar XNF, o indirecta: se genera un esquema o HDL).*
- **HDL** *(directa: diseños por componentes, donde se puede generar XNF, o indirecta: diseños por RTL o FUNCIONAL, donde se debe traducir un algoritmo de alto nivel a un sistema por componentes que lo implemente).*

Etapas de Diseño (6)

Partición

- De la lista de componentes BÁSICOS, genera ecuaciones Booleanas de *n* variables.
- Representa todas las ecuaciones Booleanas en ecuaciones que maneje la FPGA *(para la XC4000e, son ecuaciones de 4, 5, 8, 9 variables)*.

Etapas de Diseño (7)

Place:

- Ubicación de todas las ecuaciones en los CLBs de la FPGA.
- Considera restricciones de:
  - Directas de ubicación *(un componente se debe ubicar a partir de un determinado CLB, horizontal o vertical)*.
  - Indirectas de pines de i/o *(cuando un PIN debe ser conectado a una pata fija de la FPGA)*.

Etapas de Diseño (8)

Route:

- Conexión de líneas entre CLBs.
- Conexión de líneas entre los CLBs y los PADs de la FPGA.
- Tarea **muy costosa**, utiliza algoritmos no exhaustivos de búsqueda de caminos.
- Es '*casi*' imposible obtener la mejor conexión.

Etapas de Diseño (9)

Route:

Tipo de **restricciones** permitidas:

- Requerimientos de tiempo estándar *(se toman los cables en cualquier orden).*
- Requerimientos de tiempo por CABLE *(se toman primero los cables con restricciones temporales).*
- Requerimientos de tiempo para todo el DISEÑO *(se repite hasta alcanzar los requerimientos).*

Etapas de Diseño (10)

Place & Route:

- El diseño debe ENTRAR completamente en la FPGA:
  - Lógica *(representada por la cantidad de CLBs necesarios para representar el diseño)*.
  - Conexiones *(todas las variables necesarias deben tener su orígen y destino conectados)*.
  - Pads de entrada y salida.

- En esta etapa se define la calidad del diseño.

Etapas de Diseño (11)

Place & Route:

- Herramientas que repiten el **P&R** hasta que:
  - entre en la FPGA.
  - alcance las restricciones definidas por el usuario.
- Permite crear software de alto nivel para que con un PLACE se repitan tantos ROUTE como se deseen.

Etapas de Diseño (12)

Cálculo de performance del diseño:

- Realiza el cálculo de:
  - Demora de todas las conexiones internas.
  - Demora de todos los pines de la FPGA.
  - Frecuencia máxima de operación.
  - SKEW de las pistas de reloj.

- **RESTRICCIÓN**: No se puede hacer esta operación hasta que no se haya terminado el P&R. ..... ***WHY ?***

Etapas de Diseño (13)

Verificación del diseño:

- **Simulación temporal**: se utilizan las demoras de las señales calculadas en el P&R. Se obtiene la onda de todas las señales (internas y externas). Se puede ejecutar por comandos o por archivo.
- **Ejecución en la FPGA**: se realiza la configuración de la placa con la FPGA, debiendo analizarse las ondas de las señales con un osciloscopio. Se deben generar los impulsos y leer las señales de salida (por ejemplo usando una PC).

Principios del diseño (1)

- Herramientas automáticas, se puede trabajar sin conocer la arquitectura.
- Es deseable conocer la arquitectura para que el software produzca diseños más efectivos.
- Las herramientas de *Xilinx* permiten analizar el diseño a **muy bajo nivel**.
- Esta sección describe: *técnicas para hacer más efectivo el uso de las FPGAs basadas en SRAM.*

Principios del diseño (2)

- Lógica duplicada. Técnica de diseño.
- Máquinas de estado *"ONE-HOT"*. Técnica de diseño.
- Habilitación de reloj. Técnica de diseño.
- Consideraciones de performance **? ?**

Principios del diseño (3)

Lógica duplicada (a):

- TABLAS implementan **cualquier función** de sus entradas.
- **AREA** de tabla con función de 2 entradas **IGUAL** a una función de 4 entradas.
- **VENTAJA**: ***duplicar*** *lógica en caminos críticos*.

Principios del diseño (4)

Lógica duplicada (b):

- Lógica duplicada para reducir el número :
  - total de tablas en el resultado
  - de tablas a lo largo del camino desde las entradas hasta las salidas.
- La **lógica** resultante es entonces más **DENSA** y más **RÁPIDA**.

Principios del diseño (4)

Máquinas de estado “ONE-HOT” (a):

- ASIC el uso de *flip-flop* es caro.
- FPGA tienen lógica seguida de *flip-flop*.
- Muchos *flip-flop* fomenta la codificación de máquinas de estado *one-hot*.
- En *one-hot*, cada estado es representado por un *flip-flop* separado.

Principios del diseño (4)

Máquinas de estado “ONE-HOT” (b):

- **Por ejemplo**, una máquina de 16 estados:
  - “*one-hot*” requiere 16 *flip-flop*.
  - “*totalmente codificada*” requiere sólo 4 *flip-flop*.

Principios del diseño (5)

Máquinas de estado “ONE-HOT” (c):

- La codificación *one-hot* simplifica el cálculo del próximo estado, porque los estados están ya totalmente codificados.
- El estado decodificado puede ser implementado con la ecuación de transición al próximo estado en una tabla en un nivel de lógica.

Principios del diseño (6)

Máquinas de estado “ONE-HOT” (d):

- Una codificación de estados complicada necesita 2 niveles de lógica para: decodificar el estado y unir (AND) el estado con la ecuación de transición.
- La codificación *one-hot* ahorra un nivel de lógica.
- Las transiciones de estado tienden a ser localizadas, al igual que el *ruteo* de los bits de estado.
- Interconexiones localizadas generan retrasos de reducidos.

Principios del diseño (7)

Habilitación de reloj (a):

- Problemas para sincronizar puertas por reloj, dado a que las condiciones de *race* en la puerta no pueden ser resueltas antes del *ruteo* porque el retraso preciso requiere el conocimiento de los caminos de *ruteo* de todas las señales.
- Xilinx evita la necesidad de puertas sincronizadas por medio de una señal de habilitación de reloj sobre todos los *flip-flop*.

Principios del diseño (8)

Habilitación de reloj (b):

- Más que relojes separados, el diseño puede incluir un reloj simple, usando su señal como habilitación de reloj sobre el *flip-flop*.
- Este diseño elimina potenciales *glitches* y permite al software poner un reloj simple sobre las interconexiones globales de reloj lo que minimiza el *skew* del reloj.

Principios del diseño (9)

Consideraciones de Performance (a):

- Las FPGAs disponen de muchos *flip-flop*, sistemas *pipelined* tiene un costo bajo.
- Diseños de alto desempeño: segmentados en etapas de funciones simples con un *flip-flop*.
- Con esta técnica se ha logrado que el xc3000 corra a frecuencias de hasta 100Mhz (el 75% de la frecuencia de respuesta de un *flip-flop*).

Principios del diseño (10)

Consideraciones de Performance (b):

- Es importante el diseño de alto nivel para permitir que la FPGA implemente efectivamente la lógica.
- La arquitectura del xc4000 puede implementar un contador usando el soporte aritmético de alta velocidad, pero sólo si es diseñado como un contador (no si es diseñado con lógica que implementa un contador).

Metodología de diseño interactivo (1)

- La capacidad de re - programación permite a los diseñadores **experimentar** con la tecnología de implementación.
- Los diseñadores pueden tomar ventaja de esto, implementando el diseño por pedazos.

Metodología de diseño interactivo (2)

Una técnica efectiva de diseño es la que:

- identifica e implementa las partes con desempeño crítico primero,
- las fija y recién entonces,
- procede a implementar el resto del circuito.

Fin