Fundación FULGOR

Diseño Digital Avanzado

Unidad 5 - Transformaciones para Alta Velocidad, Bajo Consumo y Optimización de Área

Dr. Ariel L. Pola
apola@fundacionfulgor.org.ar
October 11, 2021

Tabla de Contenidos

1. Contenidos Temáticos

2. Pipelining y Retiming

3. Sistemas sin Realimentación

4. Sistemas Realimentados

5. C-Slow Retiming

6. Arquitecturas Folding y Unfolding

Contenidos Temáticos

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.

Pipelining y Retiming

Pipelining y Retiming

Concepto

- Con la llegada de la integración a gran escala (VLSI) se ha reducido el costo de los dispositivos de hardware.
- Esto ha dado más flexibilidad a los diseñadores de sistemas para implementar aplicaciones computacionalmente intensivas en pequeños factores de forma.
- Los diseñadores ahora se centran más en el rendimiento y menos en las densidades, ya que la tecnología está permitiendo más y más compuertas en una sola pieza de silicio.
- Desde la perspectiva HW, pipelining y procesamiento en paralelo son dos métodos que ayudan a lograr un alto throughput.
- Un camino crítico que atraviesa una nube combinatoria en un sistema feedforward puede ser roto por la adición de registros pipeline.
- Un sistema feedforward es aquel en el que la salida actual sólo depende de muestras de entrada actuales y anteriores.
- Pipeline registers sólo agregan latencia.
- Si se añaden L registros, la función de transferencia del sistema se multiplica por z^{-L}.

Pipelining and Retiming

Sistemas con o sin realimentación

- Los sistemas de procesamiento de señal pueden clasificarse como Feedforward o Feedback.
- En los sistemas Feedforward los datos fluyen desde la entrada a la salida sin presencia de lazos de realimentación. (ejemplos: filtros FIR, FFT)
- Las optimizaciones de camino crítico pueden lograrse agregando etapas de pipelining en el diseño de hardware.
- Sistemas de tipo recursivo o realimentados (feedback).
- La ecuación en diferencia relaciona muestras anteriores de la salida con valores actuales de las muestras de entradas y salidas. (Ejemplos: Filtros IIR, sistemas recuperación de sincronismo y frecuencia en receptores de comunicación).
- Debido a que se requieren muestras anteriores de la salida en el procesamiento, agregar registros para mejorar el timing no es una opción implementable de forma directa en el diseño.

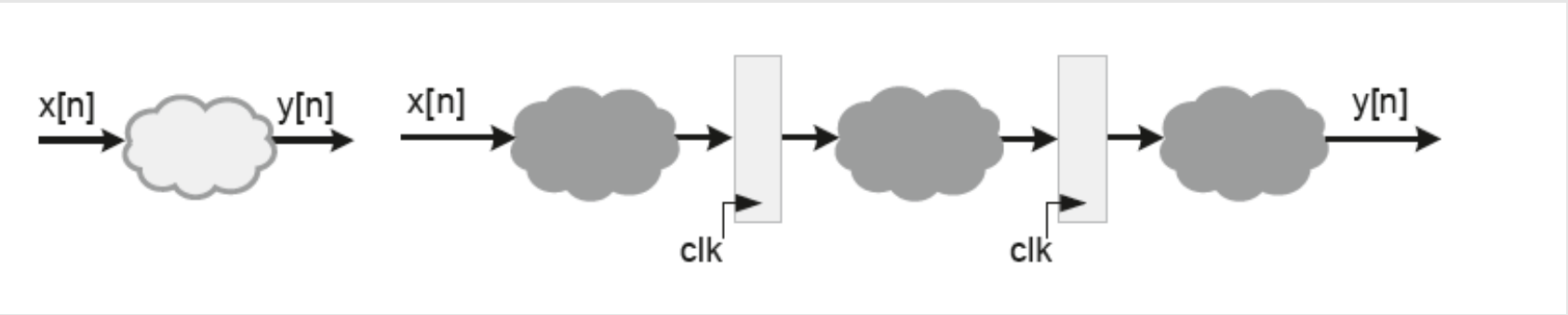
Pipelining and Retiming

Conceptos Básicos

- En un sistema de tipo feedforward, un camino critico a través de una etapa combinacional puede ser "cortado", agregando registros de pipeline. Si se agregan L registros de pipeline la función de transferencia del sistema se multiplica por z^{-L}.
- En los sistemas realimentados la incorporación de retardos modifica la función de transferencia y resulta en el cambio de orden de la ecuación en diferencias.

Ejemplo

Bloque combinacional separado en tres partes con dos etapas de pipelining.



Pipelining and Retiming

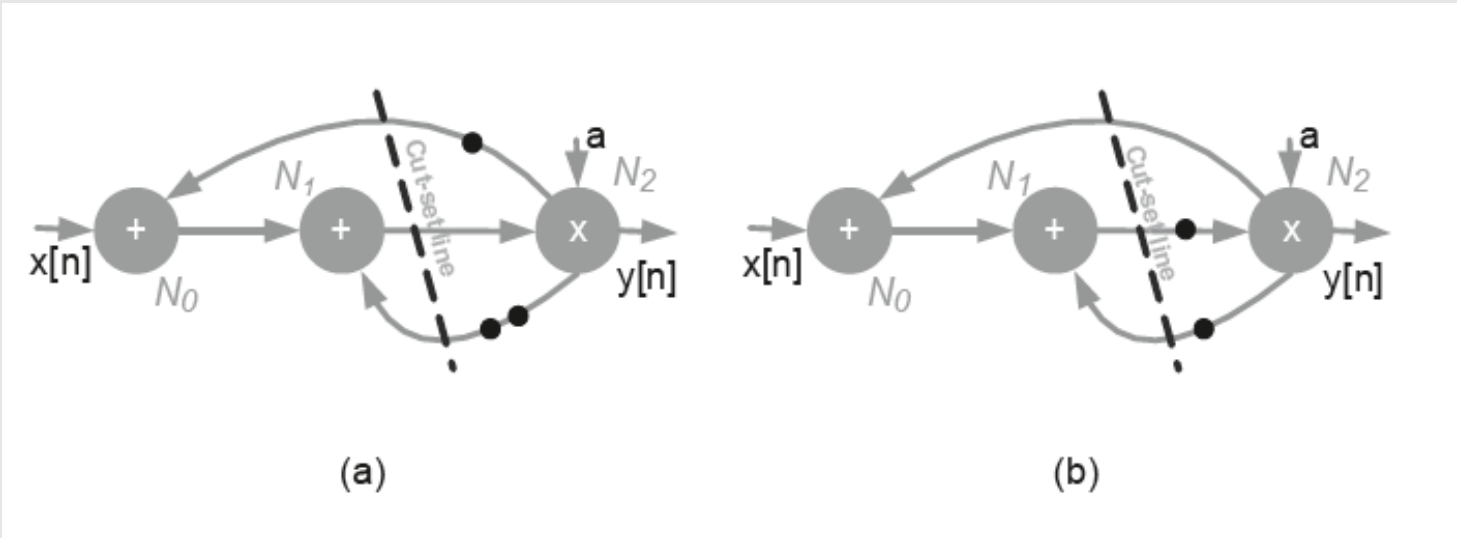
Conceptos Básicos

- En general, en un sistema con L niveles de etapas de pipelines, el número de elementos de retardo en cualquier camino de entrada a salida es $L-1$ veces mayor que en el sistema original.
- Incorporar registros de pipelining reduce el camino crítico pero incrementa la latencia del sistema y la salida $y[n]$ corresponde al procesamiento de una muestra de entrada anterior.
- Agregar registros al camino crítico que se desea mejorar y los caminos paralelos que deban compensar estos retardos, también aumentan el área y la carga del árbol de clocks. Se deben agregar pipelines únicamente cuando sea necesario.
- En el caso de Pipelining se agregan registros adicionales al sistema, mientras que en el retiming los registros se re-ubican para un objetivo particular:
  - Reducir el camino crítico
  - Reducir el número de registros
  - Reducir potencia
  - Aumentar la testeabilidad

Cut-Set Retiming

- Este técnica involucra el retiming de un gráfico de flujo de datos aplicando una línea de corte.
- Un corte válido se realiza en un conjunto de lazos backward y forward en un DFG cortados por una línea de forma que si estos elementos se eliminaran del gráfico, éste quedaría inconexo.
- El retiming implica transferir una cantidad de retardos desde los lazos con la misma dirección sobre la línea de corte del DFG a todos los lazos de direcciones opuestas sobre la misma línea.
- Estas transferencias de delays no alteran la función de transferencia del DFG.

Ejemplo



Ejemplo

La linea en el ejemplo, parte el DFG en dos gráficos distintos. Uno consiste en los nodos N0 y N1 y el otro con el nodo N2. $N1 \rightarrow N2$ es un corte forward mientras que $N2 \rightarrow N0$ y $N2 \rightarrow N1$ son cortes backward. El nuevo DFG con retiming se muestra en (b).

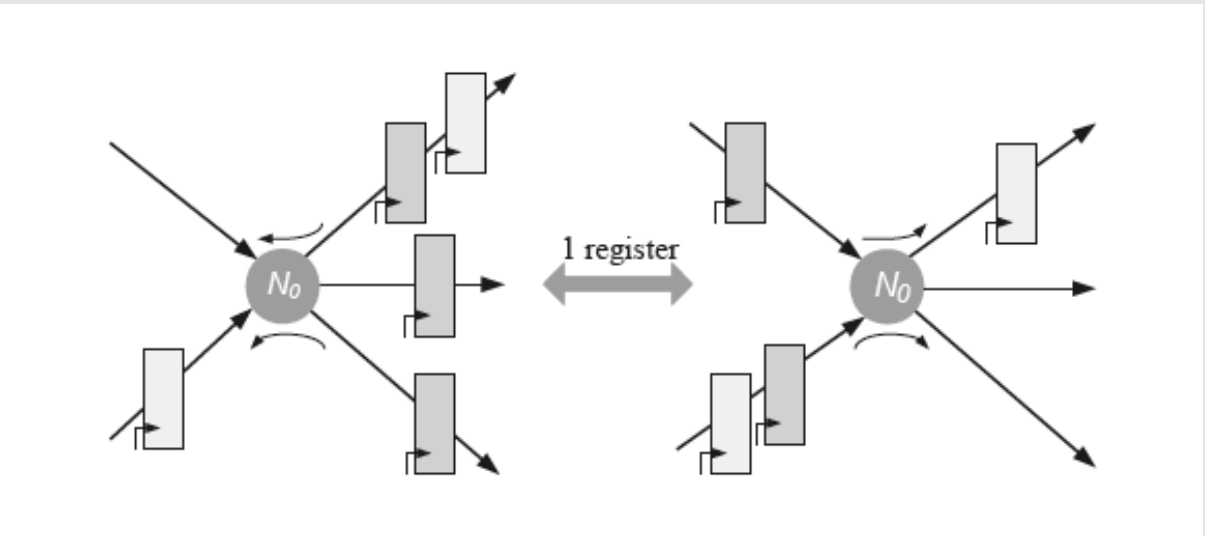
Pipelining and Retiming

Teorema de la transferencia de retardos

Transferencia de retardos

- El teorema indica que se pueden transferir N registros de cada flecha entrante en un nodo de un DFG a todas las flechas salientes del mismo nodo o viceversa, sin afectar a la función original.
- Caso particular del método de corte en el cual la linea se ubica para separar un nodo en el DFG.

Ejemplo



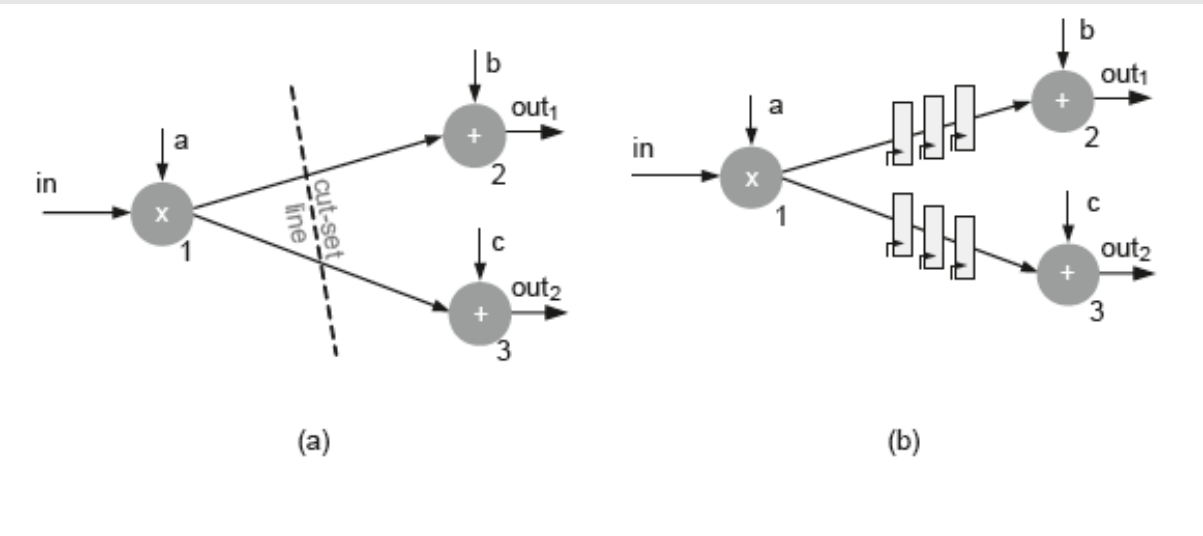
Sistemas sin Realimentación

Sistemas sin realimentación
Feedforward

Introducción

- Se pasa de un gráfico G a uno con pipelines G_r, cuyas funciones de transferencia, solo difieren en un retardo puro z^{-L} siendo L el número de etapas de pipeline agregadas
- Se debe tener cuidad de mantener la coherencia entre los paths y considerar la latencia extra en el sistema.

Ejemplo



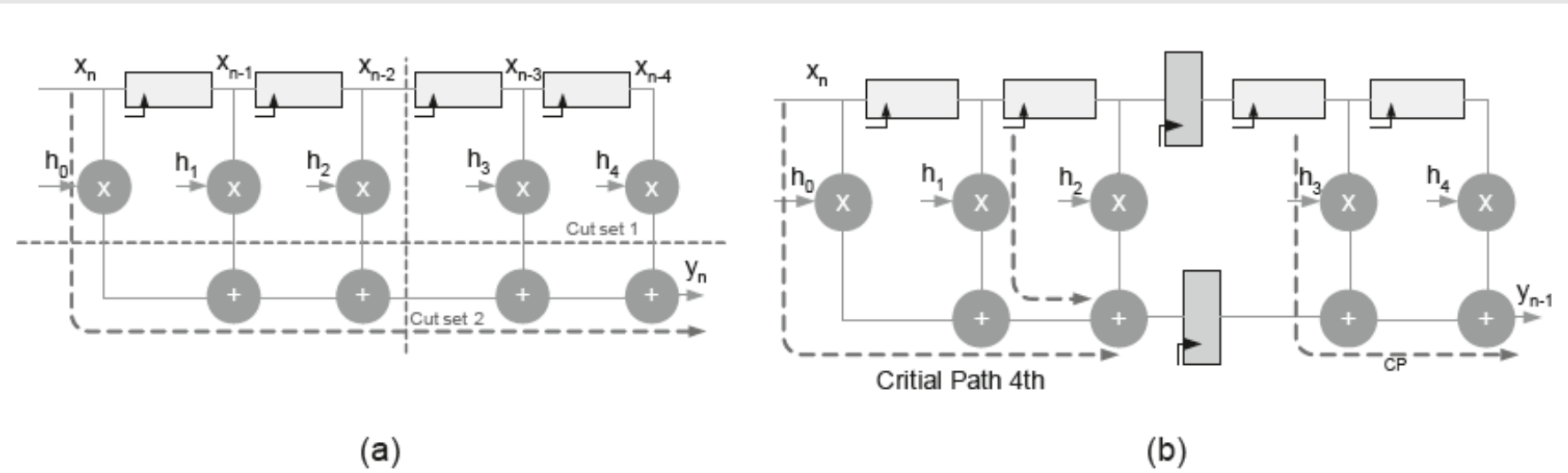
Sistemas sin realimentación
Uso de método de corte FIR

Ejemplo

Ecuaciones en diferencia y Función de transferencia de Filtro FIR de 5 coeficientes:

$$y_n = h_0 x_n + h_1 x_{n-1} + h_2 x_{n-2} + h_3 x_{n-3} + h_4 x_{n-4}$$

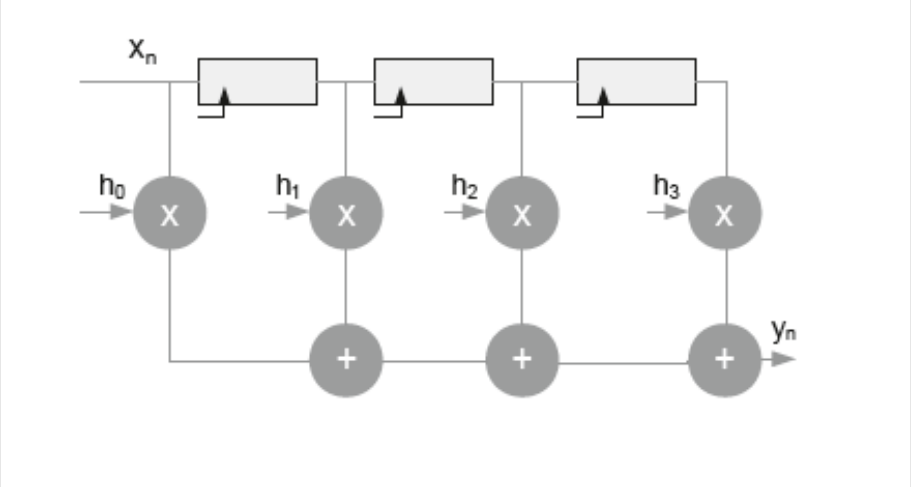
$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + h_4 z^{-4}$$

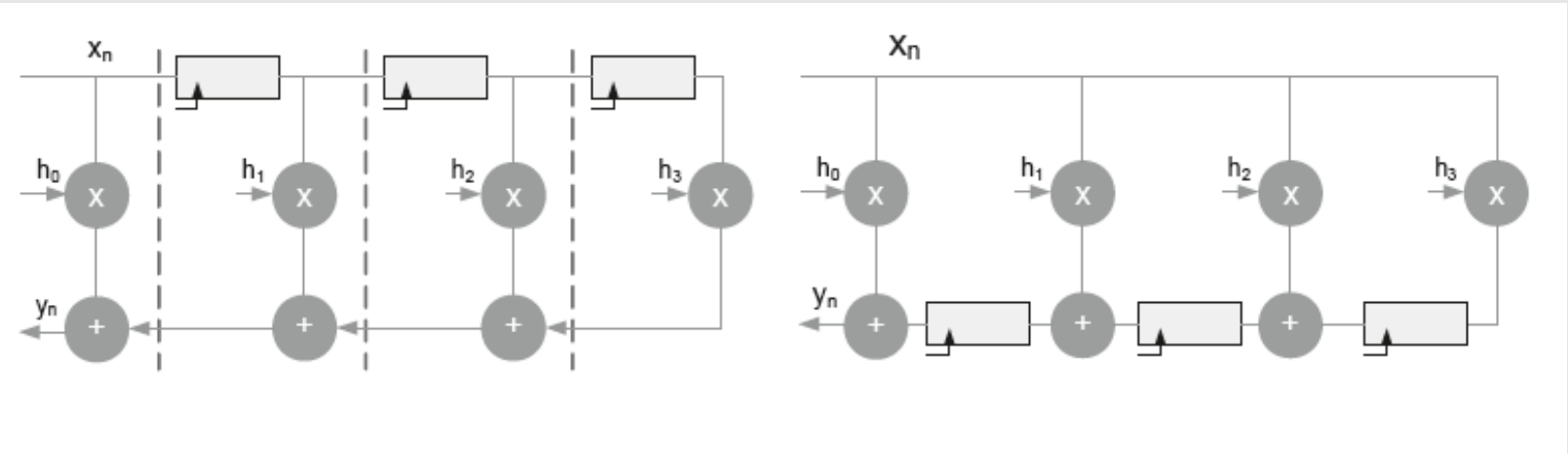


Sistemas sin realimentación

Uso de método de corte - FIR a forma transpuesta directa

Ejemplo TDF FIR

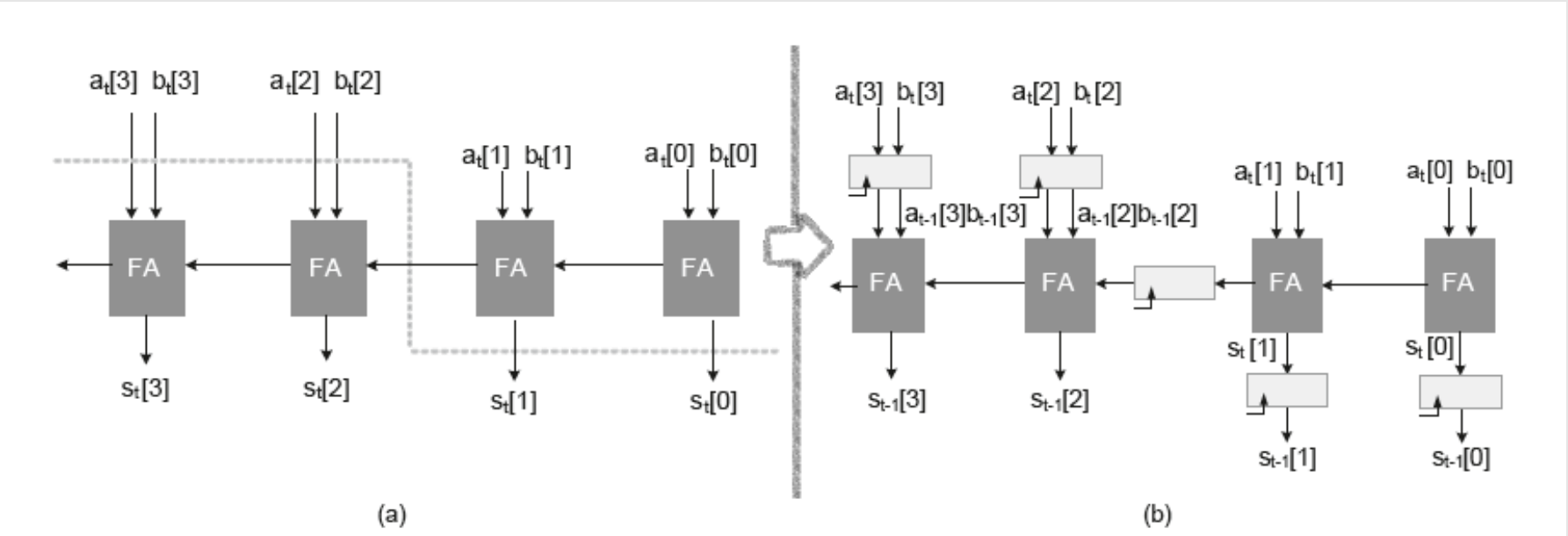




Sistemas sin realimentación
Uso de método de corte - carry adder (RCA)

Se pretende reducir el camino crítico de la lógica en la mitad.

4-bit ripple carry adder (RCA)



Sistemas sin realimentación
Uso de método de corte - Carry Adder (RCA)

Verilog

```
module pipeline_adder(
  input clk,
  input [3:0] a, b,
  input cin,
  output reg [3:0] sum_p,
  output reg cout_p);
  reg [3:2] a_preg, b_preg;
  reg [1:0] s_preg;
  reg c2_preg;
  reg c2;
```

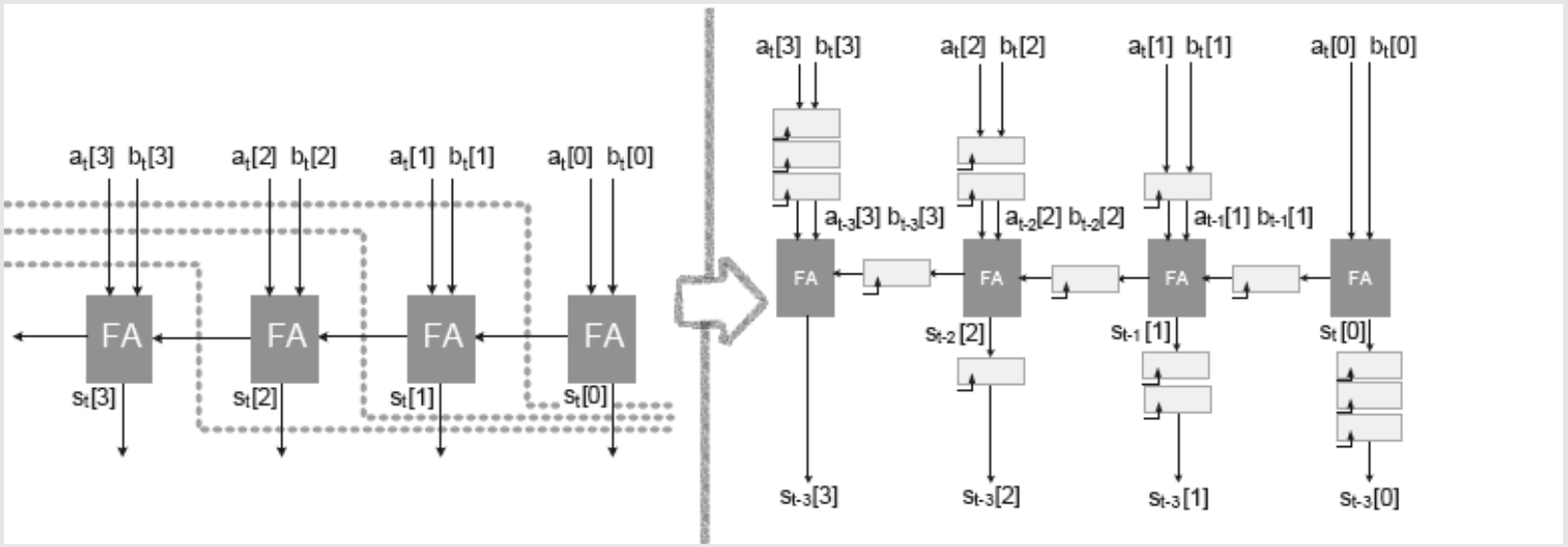
```
  always @(*) begin
    {c2, s[1:0]} = a[1:0] + b[1:0] + cin;
    {cout_p, s[3:2]} = a_preg + b_preg + c2_preg;
    sum_p = {s[3:2], s_preg};
  end
  always @(posedge clk) begin
    s_preg <= s[1:0];
    a_preg <= a[3:2];
    b_preg <= b[3:2];
    c2_preg <= c2;
  end
endmodule
```

Este pipeline RCA de dos etapas tiene dos FA en el camino crítico y una latencia de un ciclo. Esto significa que el resultado y el carry estarán disponibles luego de un período de clock. Ahora supongamos que deseamos reducir el camino crítico a un único full adder. Esto requiere agregar registros luego del primer, segundo y tercer sumador.

Sistemas sin realimentación
Uso de método de corte - carry adder (RCA)

En este ejemplo, se aplican 3 líneas de corte al DFG original. Para mantener la coherencia de los datos los cortes aseguran que todos los paths de entrada a salida tengan tres registros.

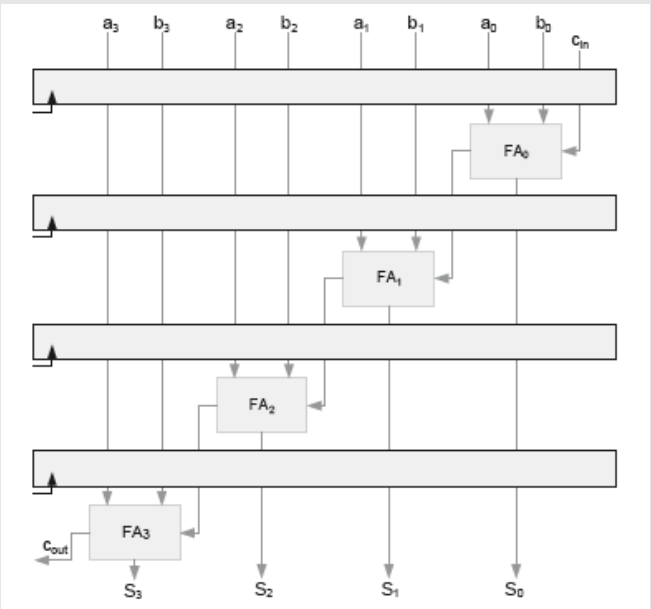
4-bit ripple carry adder (RCA)



Sistemas sin realimentación
Aplicación Teorema de transferencia de Registros

Una buena práctica para incorporar pipelines es agregar el número deseado de registros a todos los caminos de entrada y luego aplicando de forma recursiva el teorema, mover sistemáticamente los registros para cortar el camino crítico.

Teorema de Transferencia de Registros



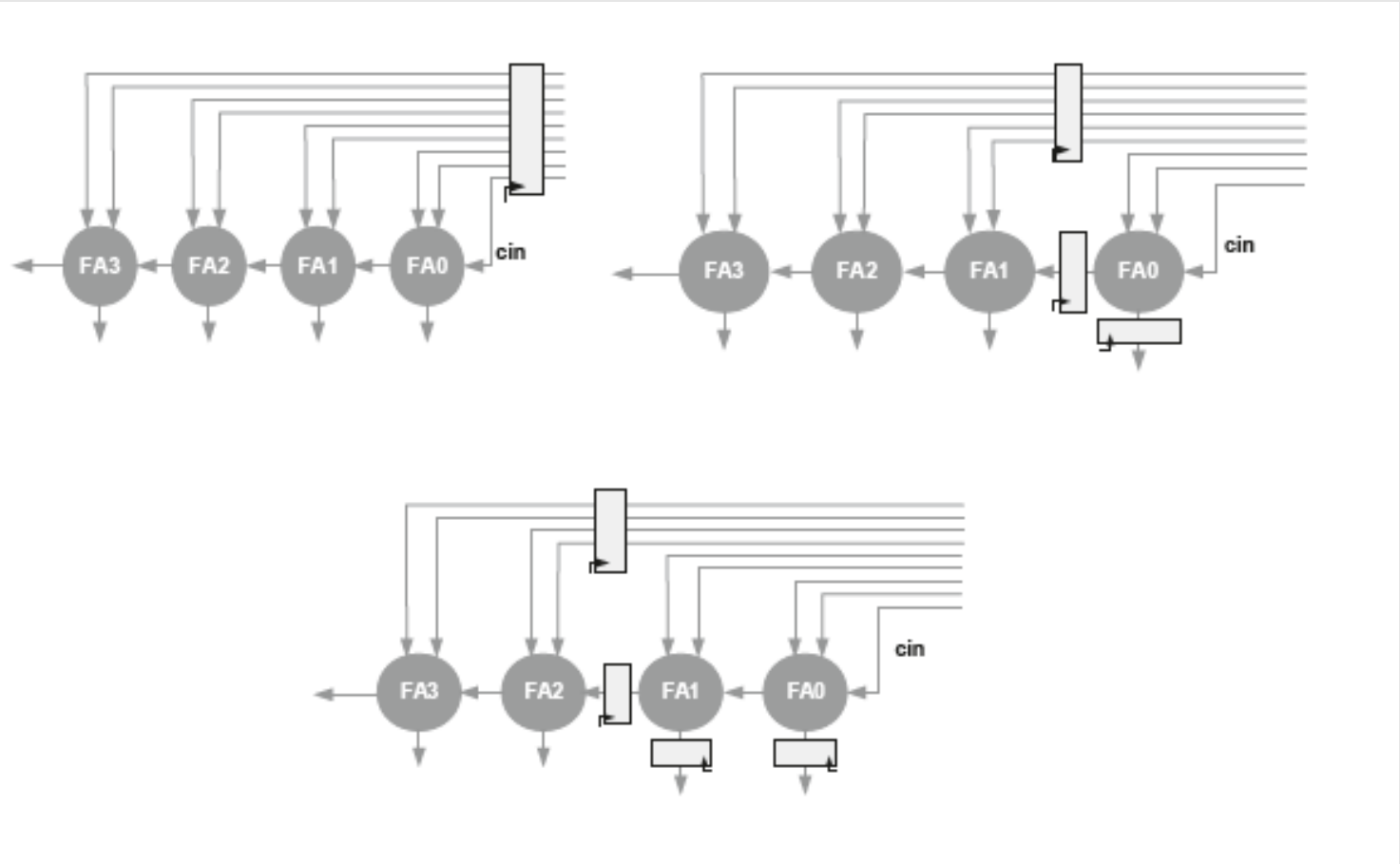
Sistemas sin realimentación
Aplicación Teorema de transferencia de Registros - Ejemplo 1

- En la figura se muestra un ejemplo de estrategia de pipelining aplicada al ejemplo anterior de un RCA de 4 bits.
- El objetivo es cortar el camino crítico que representa la propagación del acarreo desde el nodo FA1 hasta el FA2 en el DFG.
- Para comenzar, se agrega un registro a todos los caminos de entrada del DFG. El teorema de transferencia del nodo se aplica al nodo F0 y se genera un nuevo DFG.
- A continuación se aplica la transferencia al nodo FA1. Nuevamente, se transfiere cada retardo de todos los puntos de ingreso al nodo FA1 hacia los puntos de salida.
- Esto provoca el traslado del pipeline en el camino crítico del DFG mientras se mantiene la coherencia

Sistemas sin realimentación

Aplicación Teorema de transferencia de Registros - Ejemplo 1 - Cont.

Teorema de Transferencia de Registros Ejemplo

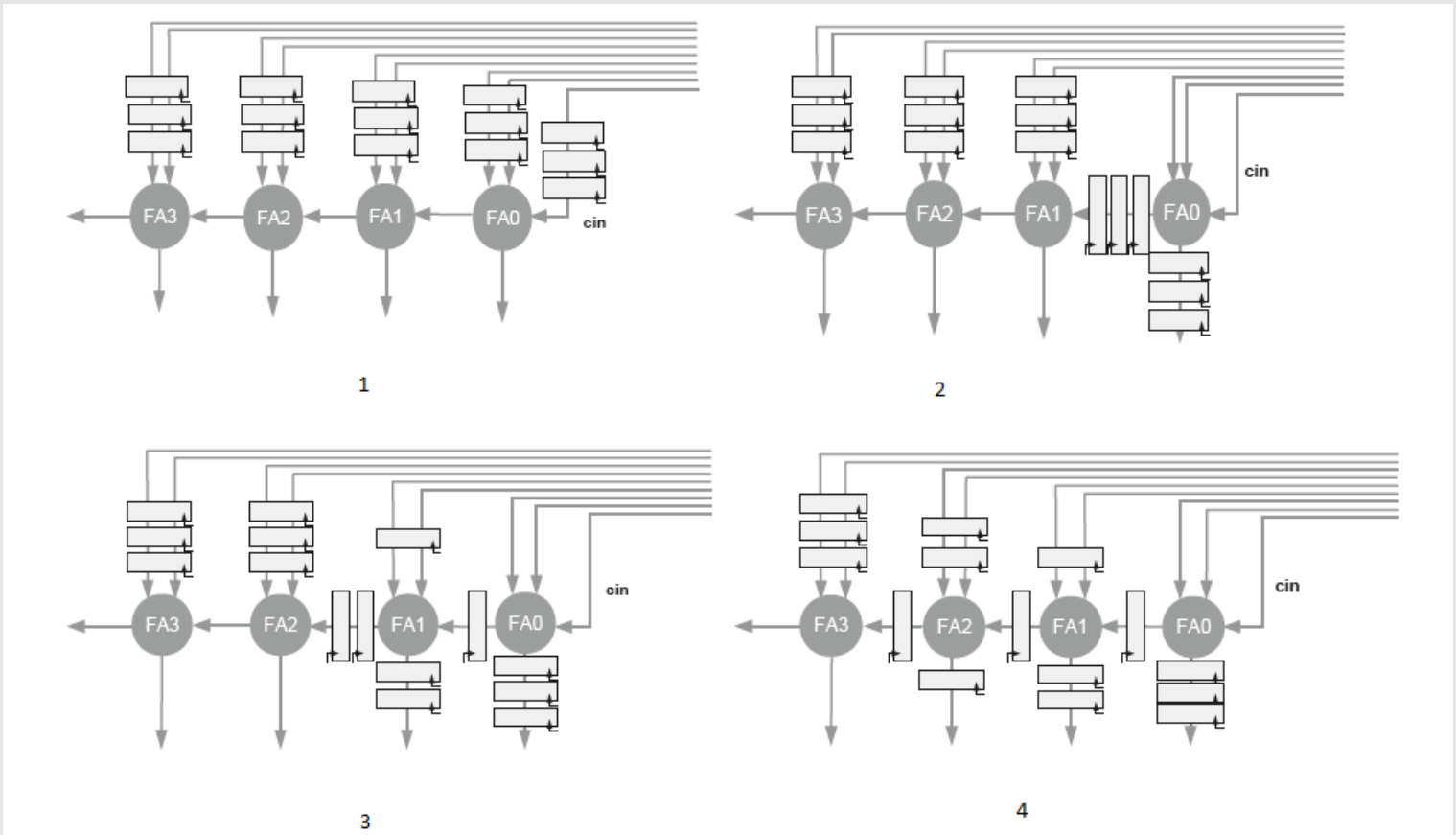


Sistemas sin realimentación
Aplicación Teorema de transferencia de Registros - Ejemplo 2

- Este segundo ejemplo agrega 3 registros de pipeline aplicando el teorema de transferencia de registros al RCA de 4 bits.
- La primera parte muestra el teorema aplicado de forma repetitiva para ubicar tres registros de pipeline en los distintos caminos de entrada. Se comienza por agregar tres registros de pipeline en todos los caminos de entrada del DFG.
- Luego se trasladan tres registros del primer nodo FA0 hacia los caminos de salida y los registros en la otra entrada se mantienen sin cambios.
- Se mueven dos registros de los puntos de ingreso del nodo FA1 hacia la salida.
- Finalmente un registro de cada entrada al nodo FA2 se mueve a todos los puntos de salida.

Sistemas sin realimentación
Aplicación Teorema de transferencia de Registros - Ejemplo 2

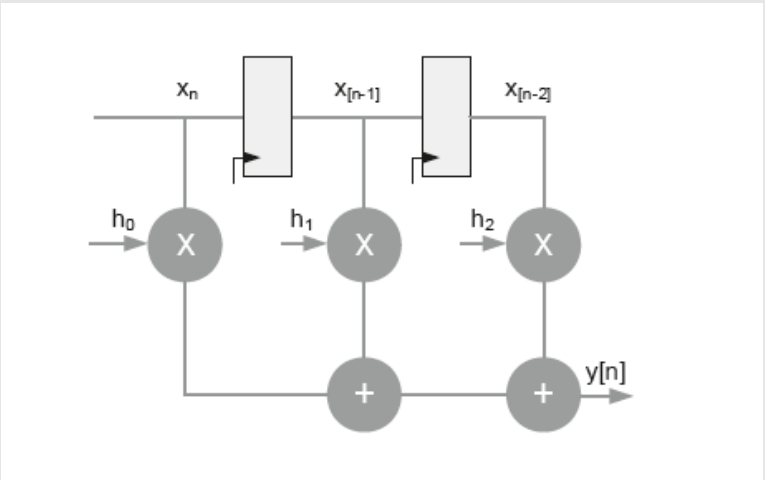
Teorema de Transferencia de Registros Ejemplo



Sistemas sin realimentación
Pipelining en DFG optimizados - Filtro FIR

- Anteriormente se explicó que los multiplicadores de propósito general no suelen utilizarse para multiplicación por constantes y se evitan los CPA (sumadores de propagación de acarreo) en el datapath. Se utiliza con frecuencia la codificación canónica (CSD). Se implementa la multiplicación utilizando operaciones de desplazamiento.
- Cada multiplicador genera un máximo de Productos Parciales (PPs). Se computa un Vector de Corrección para eliminar la extensión de signo y una lógica de complemento a dos para agregar 1 a la posición del LSB en los PPs negativos.
- Este CV también se agrega en el árbol de reducción.

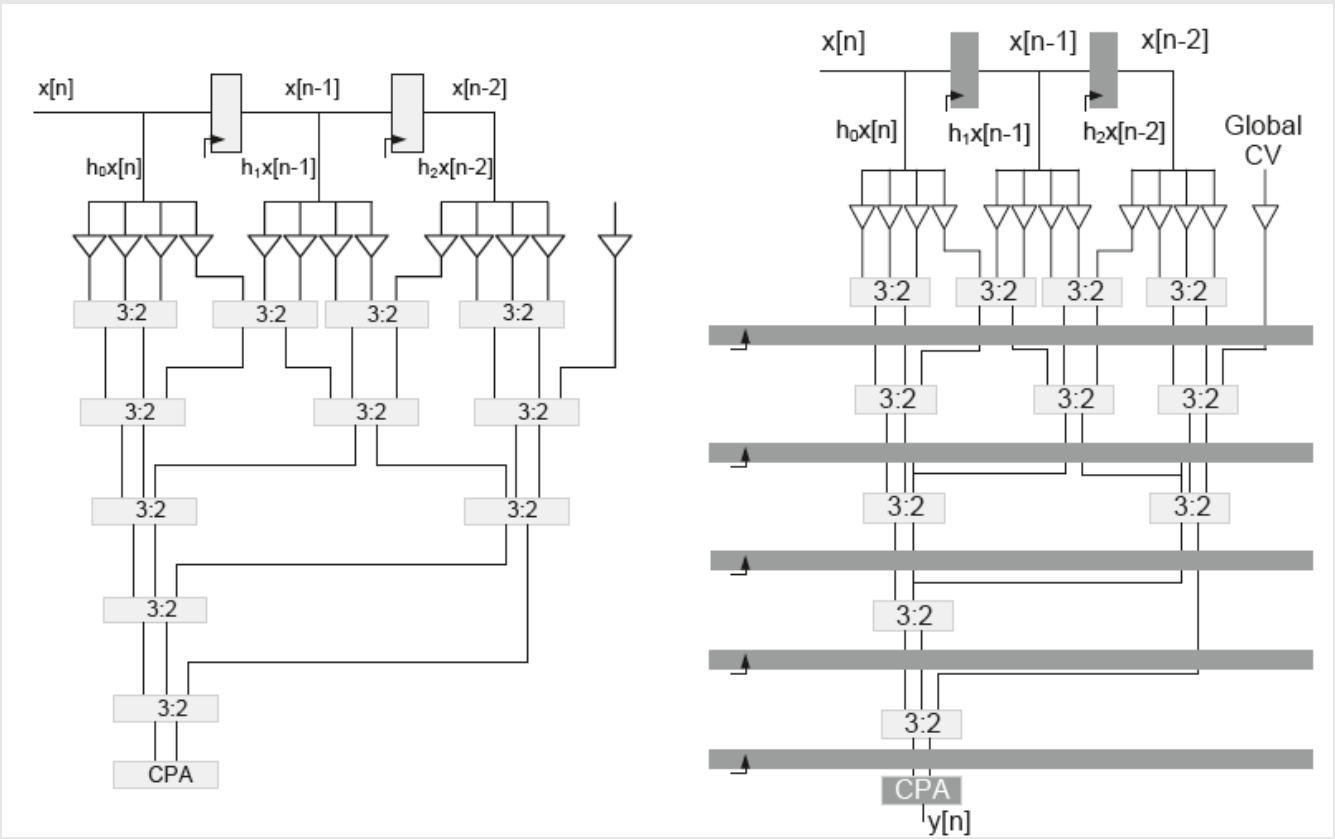
Ejemplo Filtro FIR de 3 coeficientes



Sistemas sin realimentación
Pipelining en DFG optimizados - Filtro FIR

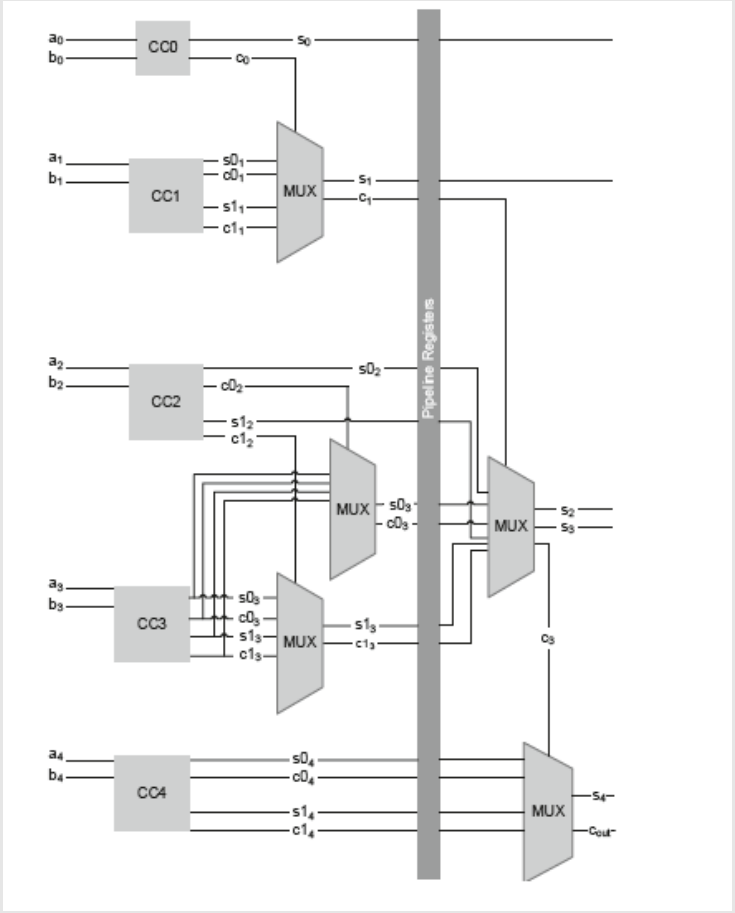
Pipelining por medio de cut-set en Filtro FIR optimizado con multiplicaciones CSD, Reducción de Productos parciales y CV en dos etapas y CPA para reducción final.

Alternativas de pipelining en FIR optimizado



Sistemas sin realimentación
Pipelining de CSA

Alternativas de pipelining en FIR optimizado



Sistemas sin realimentación
Retiming en Herramientas de Síntesis

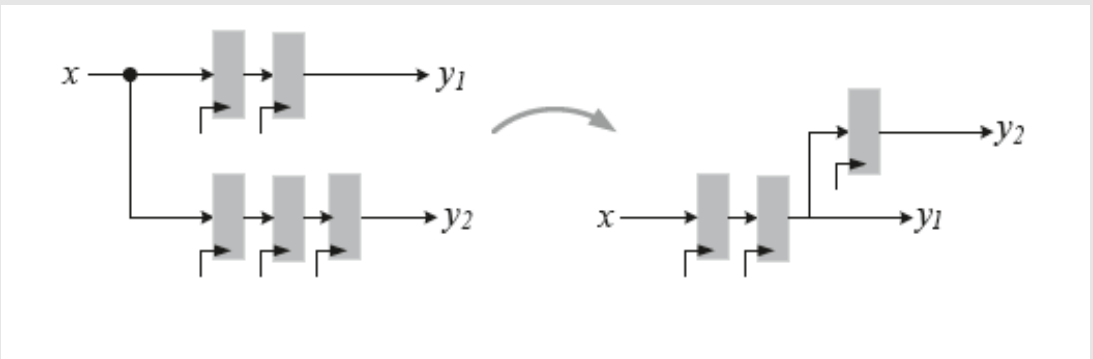
Retiming

- La mayoría de las herramientas de diseño digital tienen soporte para retiming automático o semi-automático.
- La técnica retiming se aplica a nivel de síntesis.
- Esta opción no mueve registros en los sistemas feedback.
- Una forma de utilizar esta técnica es colocar todos los registros en el nodos de entrada y dejar que la herramienta lo distribuya sobre los caminos críticos.
- En un VLSI podemos considerar que movemos el registro a otra posición mientras que en una FPGA estaríamos seleccionando un registro en otra posición del hardware dedicado.

Sistemas sin realimentación
Minimizar el número de registros y retardo del camino crítico

- La técnica del retiming suele utilizarse para minimizar el número de registros en el diseño.
- Para gráficos más complejos, la minimización de registros se modela como un problema de optimización. El problema se resuelve utilizando algoritmos incrementales en los cuales se mueven registros de manera recursiva a los vértices en donde se minimiza el número de registros y el gráfico satisface los requerimientos de tiempo

Ejemplo de Retiming para minimizar registros



Reducción de un path con 5 registros con retiming y reducción a 3 registros

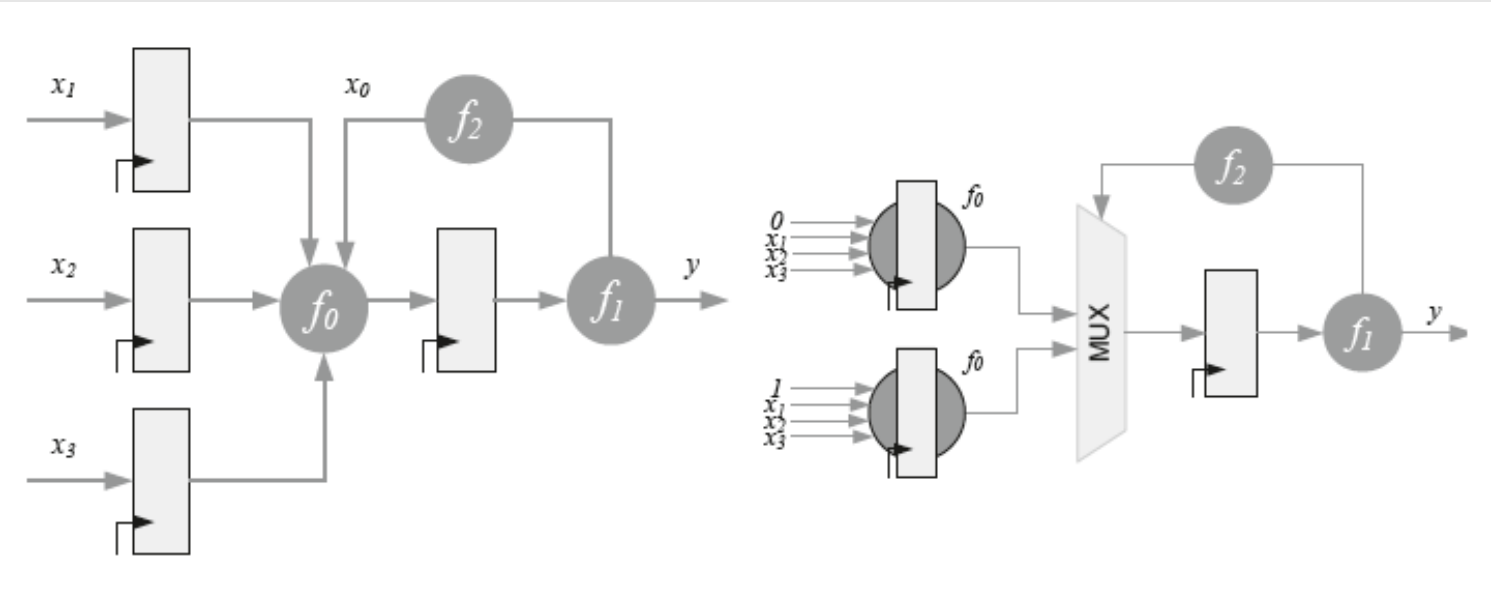
Sistemas sin realimentación
Descomposición de Shannon

- La descomposición de Shannon es una transformación que permite extender el alcance del retiming. Separa una función multivariable booleana en una combinación de dos funciones booleanas equivalentes:

$$f(x_0, x_1, \ldots, x_{N-1}) = \overline{x_0} f(0, x_1, \ldots, x_{N-1}) + x_0 f(1, x_1, \ldots, x_{N-1})$$

- La técnica identifica una señal con un retardo en la llegada al bloque y duplica la lógica del bloque con x_0 asignando valores de 0 y 1. Luego, un multiplexor de 2:1 selecciona la salida correcta de la lógica duplicada.
- El CSA (Carry Select Adder) es un ejemplo de descomposición de Shannon. El camino del carry es la parte más lenta en el RCA. La lógica en cada bloque se duplica con un carry fijo en 0 y en 1 y el valor correcto se selecciona con un multiplexor de 2:1.
- De manera genérica, la descomposición de Shannon puede funcionar de forma jerárquica en lógica con múltiples entradas, como en un CSA (Carry Select Adder) jerárquico o el sumador condicional.

Descomposición de Shannon



Diseño con una entrada más lenta x_0 al nodo f_0. La dependencia de f_0 con x_0 se remueve utilizando descomposición de Shannon que duplica la lógica en f_0 para los dos posibles valores de entrada de x_0, un multiplexor selecciona la salida correcta. Los registros en esta descomposición pueden ser trasladados de forma efectiva.

Sistemas Realimentados

Período y Período de Iteración

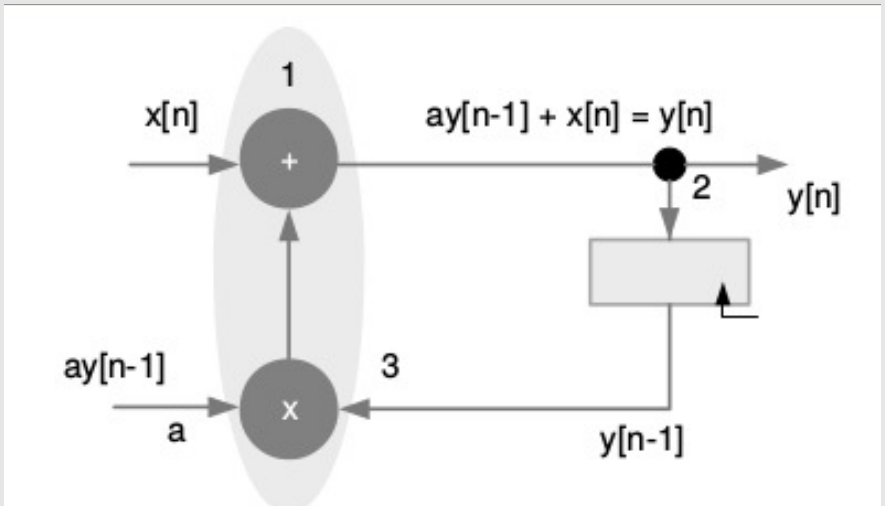
- Un sistema realimentado procesa una muestra de salida basándose en muestras anteriores de la salida y muestras actuales y anteriores de la entrada.
- Para este tipo de sistemas la iteración se define como la ejecución de todas las operaciones en el algoritmo requeridas para obtener una muestra de salida.
- El período de iteración es el tiempo requerido para la ejecución de una iteración del algoritmo.
- En los sistemas síncronos, el sistema debe completar la ejecución de iteración actual antes de que se requiera la siguiente muestra de entrada. Esto implica un límite superior en el período de iteración para que sea menor o igual a la tasa de muestreo del dato de entrada.
- En la filmina siguiente la figura muestra un sistema IIR implementando la siguiente ecuación en diferencias:

$$y[n] = ay[n-1] + x[n]$$

Ejemplo período y Período de Iteración

- El algoritmo necesita ejecutar una multiplicación y una suma para entregar una muestra de salida en una iteración. Si el tiempo de ejecución del multiplicador y el sumador son T_m y T_a respectivamente, el período de iteración para el ejemplo es $T_m + T_a$. Con respecto al período de muestreo del dato de entrada T_s, se debe satisfacer $T_m + T_a \leq T_s$.

Iteración en filtro IIR



Lazo y relación del lazo

- Un lazo se define como el camino directo que comienza y termina en el mismo nodo. En el ejemplo anterior del IIR, el camino atraviesa los nodos $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ es un lazo.
- La relación de lazo del lazo i esta definida por T_i/D_i, Donde T_i es el tiempo de procesamiento del lazo y D_i es el número de retardos en el lazo. Para el caso del ejemplo la relación de lazo es $T_m + T_a/1$.

Lazo crítico y relación de iteración

- Un lazo crítico en un DFG se define como el lazo con mayor relación de lazo. El período de iteración del lazo crítico se denomina período de relación de iteración IPB (Iteration Period Bound). Matemáticamente se describe por:

$$IPB = max_{all-L_i}\{T_i/D_i\}$$

Camino crítico y retardo del camino crítico

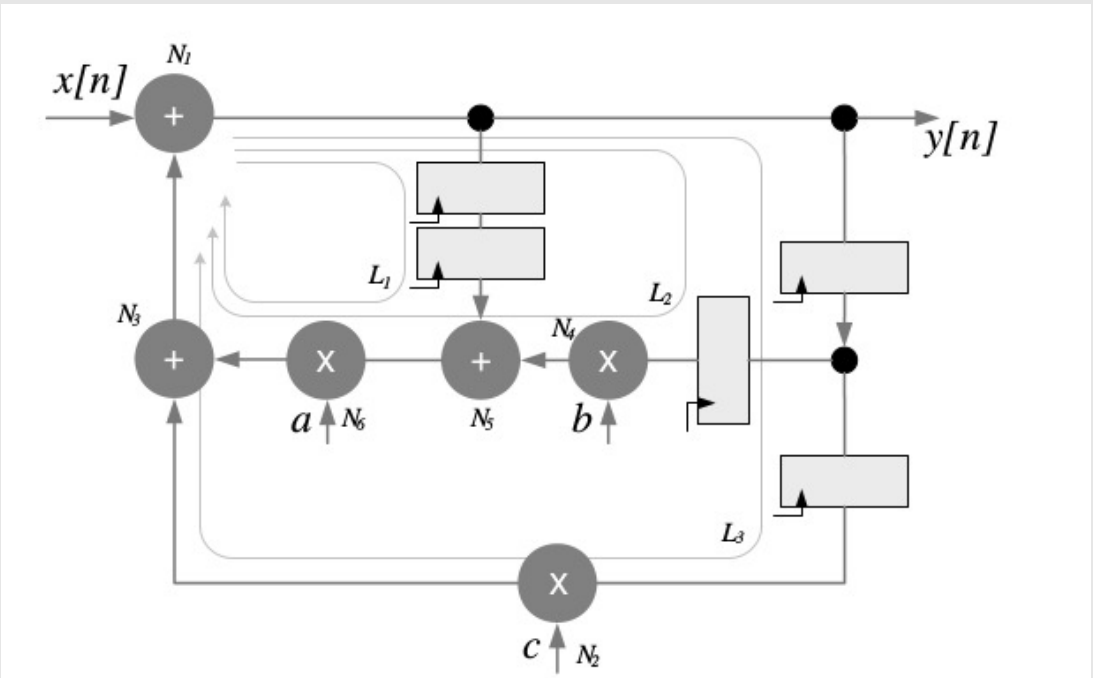
- El camino crítico del un DFG se define como el camino con el mayor tiempo de retardo de procesamiento entre todos los caminos que no contienen registros. El tiempo de procesamiento en ese camino se denomina retardo del camino crítico de un DFG.
- El camino crítico tiene gran importancia en el diseño digital ya que el período de clock y la posibilidad de disminuirlo se encuentra vinculado al retardo en el camino crítico. Un objetivo importante en el diseño es disminuir el retardo del camino crítico. Para esto se emplean técnicas como la selección de bloques de sumadores, multiplicadores y barrel shifters de alto desempeño en el procesamiento así como técnicas de retiming y transformaciones.
- Para los diseños sin realimentación los registros pueden reducir el camino crítico. En el caso de los sistemas con realimentación, la utilización de pipelining no se puede emplear de forma tan directa, pero se puede hacer uso del retiming para reducir el retardo del camino crítico. El IPB en estos diseños es el mejor tiempo que se puede obtener utilizando retiming sin generar trasformaciones de pipelining complejas.

Retiming y Pipelining en Sis. Realimentados

Ejemplo de Retiming en sistemas realimentados

- El DFG que se muestra en el ejemplo contiene tres lazos, L_1. L_2 y L_3.
- Asumimos que la multiplicación consume 2 unidades de tiempo y la adición 1.

Ejemplo de cálculo de lazos



Ejemplo

- La relación de lazo de cada uno de los lazos del ejemplo está dada por:

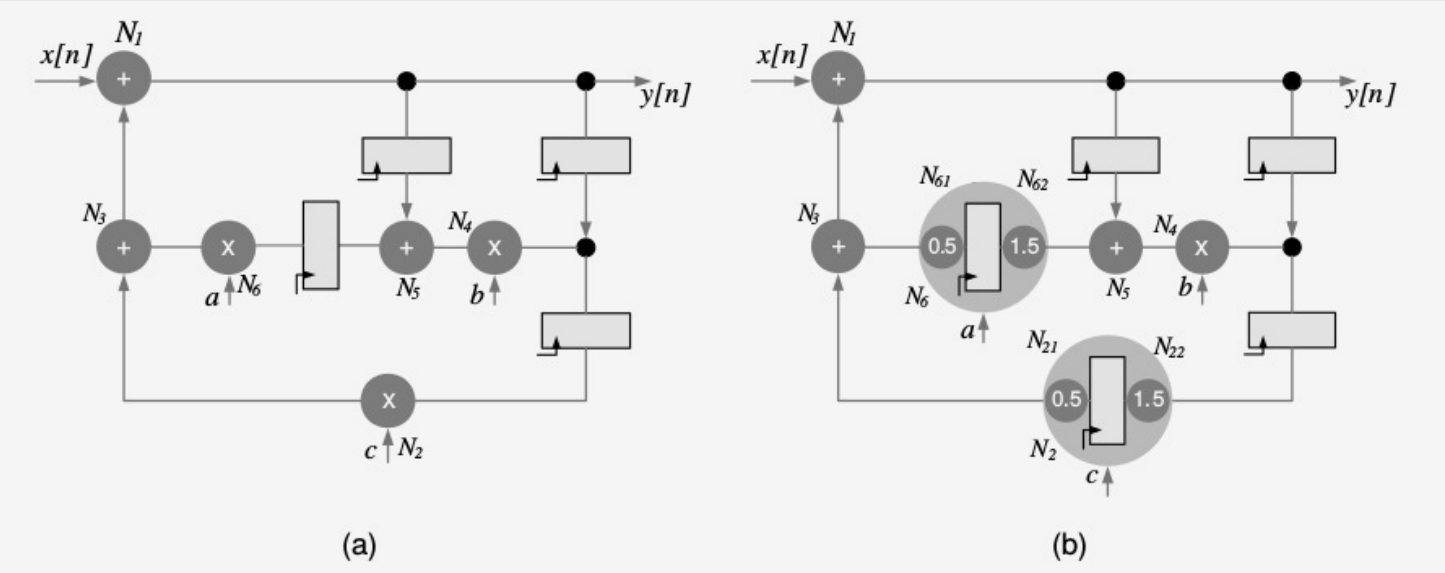
$$LB1 = T_1/D_1 = (1+1+2+1)/2 = 2.5$$
$$LB2 = T_2/D_2 = (1+2+1+2+1)/2 = 3.5$$
$$LB3 = T_3/D_3 = (1+2+1)/2 = 2$$

- L_2 es el lazo crítico ya que tiene la máxima relación de lazo. Esto es $IPB = max\{2.5, 3.5, 2\} = 3.5$ unidades de tiempo.
- El camino crítico o el camino más largo del DFG es el que va de $N4 \rightarrow N5 \rightarrow N6 \rightarrow N3 \rightarrow N1$ y el retardo de ese camino es de 7 unidades de tiempo.
- El IPB de 3.5 indica que aún es posible reducir el camino crítico.

Técnicas de Retiming

- Se puede optar por una solución de compromiso en la cual el path crítico no necesariamente se reduzca al IPB. En el gráfico de flujo de datos, se aplica de manera recursiva el teorema de la transferencia de registros al rededor del nodo N4 y luego de N5. El resultado se muestra en la figura (a).
- Otra optimización más fina divide los multiplicadores en N6 y N2 a sub-nodos N61 y N62 y N21 y N22 de forma que el comportamiento del multiplicador se separe en 1.5 unidades de tiempo y 0.5 unidades de tiempo respectivamente y se logra la reducción del camino crítico a 3.5 unidades de tiempo que equivale al IPB. El diseño final se muestra en la figura (b).

Retiming en path crítico para reducir IPB



Retming aplicado al ejemplo en (a) el camino crítico se redujo a 4 unidades de tiempo para un IPB de 3.5 unidades de tiempo. (b) Se realiza un retiming más fino para obtener un camino crítico con retardo de 3.5 unidades de tiempo equivalente al IPB.

Método de corte

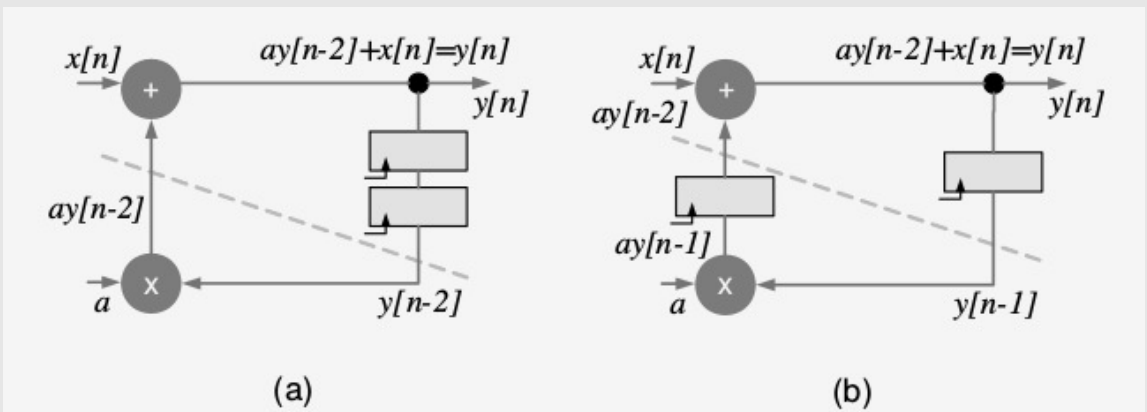
- En los sistemas realimentados, el método de corte puede utilizarse para desplazar sistemáticamente los delays propios del algoritmo en un diagrama de flujo de datos G desde un grupo de líneas hasta otros para maximizar los objetivos del diseño y mantener la función de transferencia sin cambios en el diagrama con retiming G_r.
- Los objetivos pueden ser
  - Reducir el retardo del camino crítico
  - Reducir el número de registros o el consumo de potencia o una combinación de estos.
- Se muestra un ejemplo de un filtro IIR que implementa la siguiente ecuación en diferencias:

$$y[n] = ay[n-2] + x[n]$$

El camino crítico del sistema es $T_m + T_a$ donde T_m y T_a son los retardos en el procesamiento de la multiplicación y el sumador, respectivamente.

- La figura (a) muestra la linea de corte para mover un retardo para cortar el camino crítico del DFG.
- El resultado se muestra en (b) con un camino crítico reducido al $max\{T_m, T_a\}$. El mismo retiming se puede aplicar utilizando el teorema de transferencia de retardos alrededor del nodo del multiplicador.

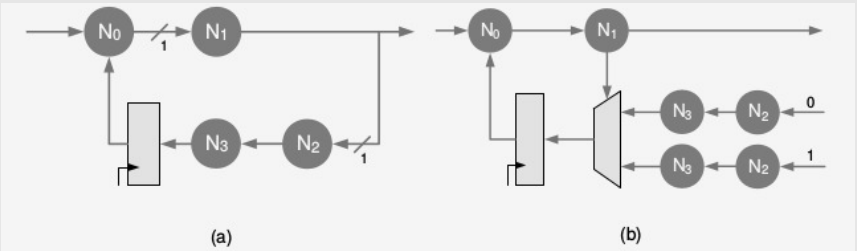
Ejemplo retiming en IIR



Descomposición de Shannon en sistemas realimentados

- Para sistemas recursivos el IPB define el mejor tiempo que puede obtenerse a través de retiming. La descomposición de Shannon puede reducir el IPB replicando algunos de los nodos críticos en el lazo y procesando las posibilidades en base a los bits de entrada, seleccionando la respuesta correcta cuando la salida del nodo anterior se encuentra disponible.
- En la filmina siguiente se muestra un lazo crítico con un registro y cuatro nodos combinacionales N0, N1, N2 y N3. Asumiendo que cada nodo requiere de 2 unidades de tiempo para ejecutarse, el IPB del lazo es 10 unidades de tiempo. Al llevar los nodos N2 y N3 fuera del lazo y computando los valores para las entradas 0 y 1, el IPB del lazo está reducido a cerca de 4 unidades de tiempo ya que solo dos nodos y un multiplexor permanecen en el lazo.

Ejemplo Descomposición de Shannon en sistemas realimentados



C-Slow Retiming

C-Slow Retiming

Introducción

Conceptos Básicos

- La técnica de C-Slow retiming remplaza cada registro en el flujo de datos con registros C.
- Estos pueden ser desplazados para reducir el camino crítico. El diseño resultante puede operar en C tramas distintas de datos.
- El uso óptimo de diseño C-slow requiere el multiplexado de C tramas de datos en la entrada y el respectivo demultiplexado de estas tramas en la salida.
- La técnica requiere C tramas de datos de entrada o emplear $C-1$ nulos luego de cada entrada valida al DFG. De esta forma el throughput efectivo no se ve afectado por la técnica C-slow, pero provee una forma efectiva para implementar el paralelismo sin agregar lógica combinacional redundante ya que solo los registros requieren ser replicados.

C-Slow Retiming

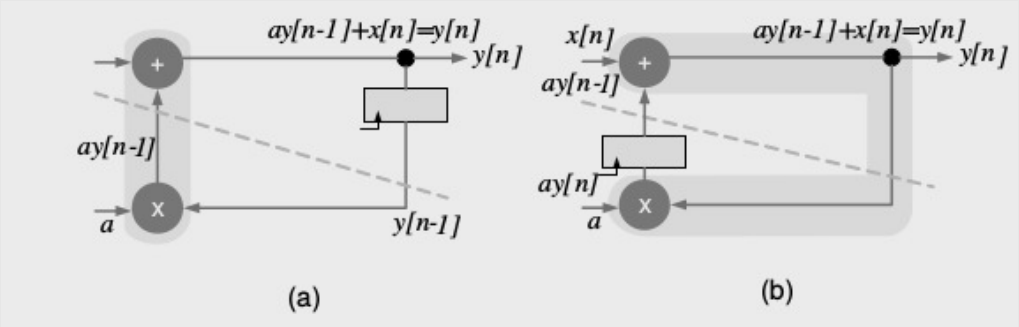
Introducción

Conceptos Básicos

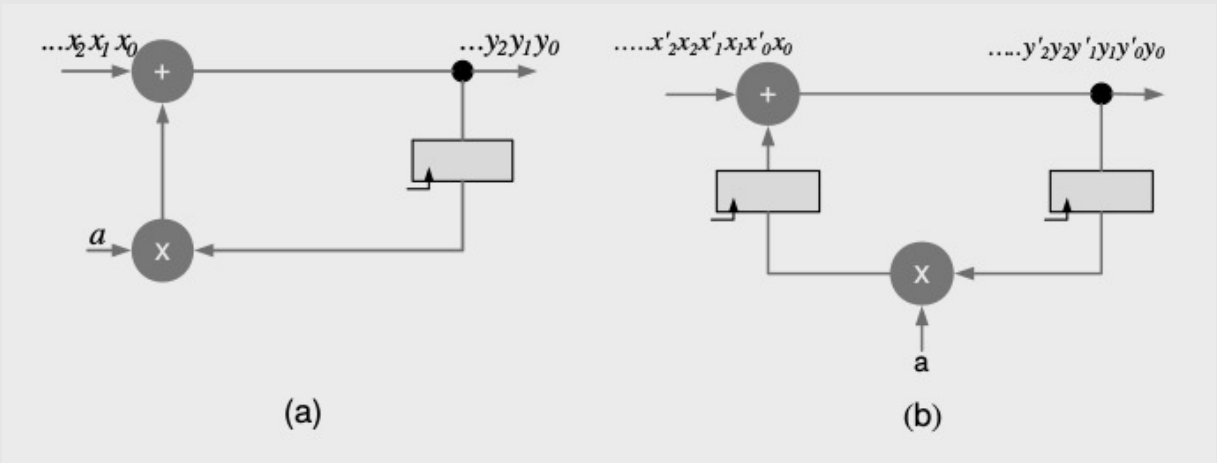
- En el siguiente ejemplo, se muestra el diseño de un filtro IIR modificado implementando 2-slow. El nuevo diseño alimenta a dos tramas de datos independientes. Este sistema funciona procesando dos tramas de entrada $x[n]$ y $x[n]'$ y produce dos tramas de salida $y[n]$ e $y[n]'$.
- Aunque un circuito puede ser reducido en tiempo en cualquier valor de C, la configuración del tiempo de registros y el retardo clock-to-Q se vinculan en número de registros que pueden ser agregados y cambiados en tiempo.
- Una segunda limitación es en diseños que pueden requerir muchos registros en el retiming y resultar en un incremento importante de área.

C-Slow Retiming
Ejemplo filtro IIR modificado

Ejemplo sin C-Slow Retiming



Ejemplo de aplicación de C-Slow Retiming en filtro IIR primer orden



C-Slow Retiming

C-Slow para Procesamiento de Bloques

- La técnica de C-slow funciona muy bien para bloques de procesamiento de algoritmos. Una cantidad C de bloques de una misma trama de datos pueden ser procesadas por una arquitectura C-slow.
- Un ejemplo es el cifrado AES (Advanced Encryption Standard) donde se encriptan bloques de 128 bits. Reemplazando cada registro en una arquitectura AES, con C registros, el diseño puede encriptar simultáneamente C bloques de datos.

C-Slow Retiming

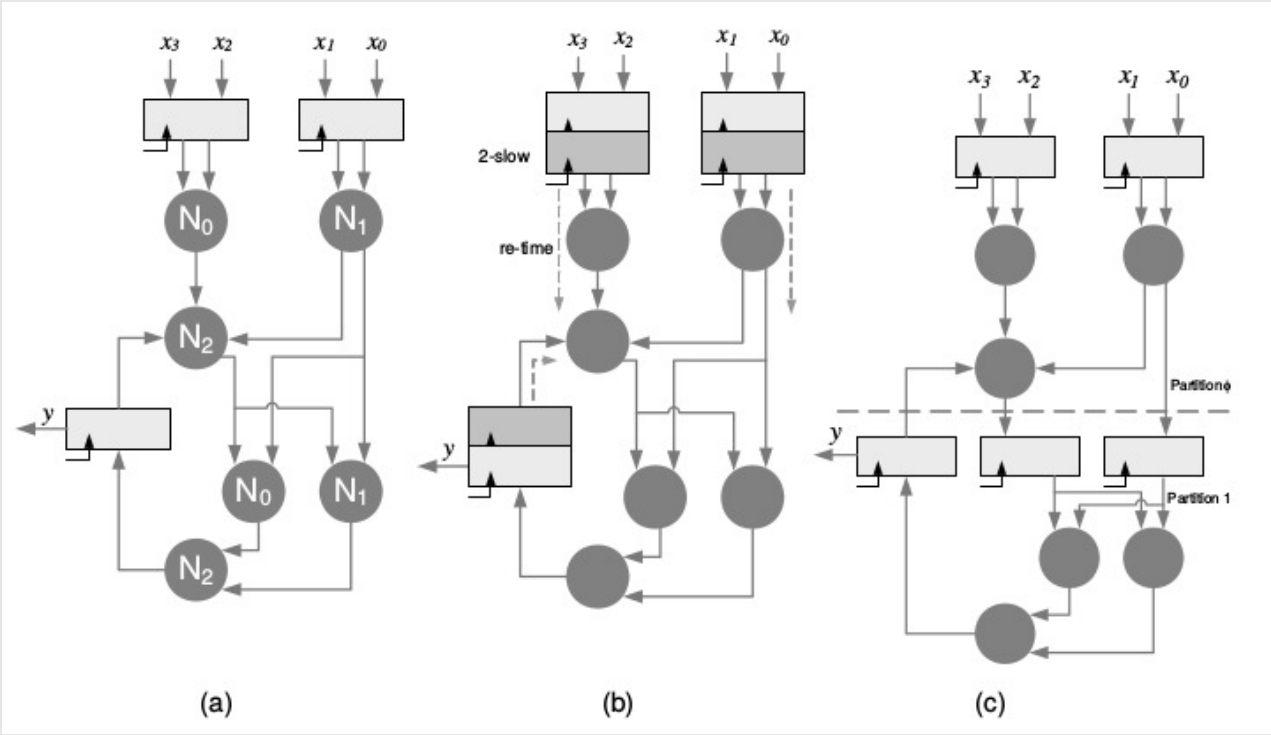
C-Slow para FPGA y Diseños reconfigurables multiplexados en el tiempo

- Debido a que las FPGA contienen gran cantidad de registros, la técnica C-slow puede utilizar de forma efectiva estos registros para procesar múltiples tramas de dato. En muchos diseños la frecuencia del reloj es fijada a un valor pre-definido, y la técnica C-slow resulta muy útil para lograr la frecuencia fija del reloj
- El retiming con C-slow también puede convertir arquitecturas en paralelo a arquitecturas multiplexadas en el tiempo donde se pueden mapear C particiones en una FPGA reconfigurable en tiempo de ejecución. En un diseño C-slow, siendo la entrada válida en cada Cth clock, por medio del retiming se pueden mover registros para dividir el diseño en particiones donde las particiones pueden ser creadas de manera óptima para dividir equitativamente la lógica en C partes.
- Un diseño multiplexado en el tiempo requiere menor área y resulta ideal para lógica reconfigurable en tiempo de ejecución.
- Luego de aplicar la técnica se mapea el DFG a lógica multiplexada en el tiempo donde el dato es conectado al diseño en un ciclo de clock y el resultado se almacena en los registros, estos valores son entradas al diseño que reutiliza la misma lógica computacional, reduciendo área.

C-Slow Retiming

- En la figura se muestra el concepto básico. El DFG de (a) tiene 2-slowed como muestra (b). Se asume que los tres nodos implementan alguna lógica combinacional que se repite dos veces en el diseño. Se efectúa retiming para ubicar los registros de forma óptima mientras se crean dos partes iguales. El resultado se muestra en (c).

C-Slow para multiplexado en el tiempo



Arquitecturas Folding y Unfolding

Arquitecturas Folding y Unfolding

Introducción

- Las principales decisiones en el diseño digital se basan en la relación entre la frecuencia de muestreo y la frecuencia del circuito.
- La frecuencia de muestreo es específica de una aplicación y se deriva del criterio de muestreo de Nyquist o de la restricción de muestreo del ancho de banda.
- La frecuencia del circuito, por otro lado, depende principalmente del diseño y la tecnología utilizada para la implementación.
- En muchas aplicaciones de alto rendimiento, el foco principal del diseño es hacer funcionar el circuito a la velocidad de reloj más alta posible para obtener el rendimiento deseado.
- Si no se puede sintetizar un mapeo simple del gráfico de flujo de datos (DFG) en el hardware a la velocidad de reloj requerida, el diseñador opta por usar varias técnicas.
- En los diseños forward, una transformación unfolding hace posible el procesamiento paralelo, así como también el pipeling es otra opción para mejorar el timing.
- En diseños realimentados, la transformación unfolding no da como resultado un verdadero procesamiento paralelo ya que la velocidad del reloj del circuito necesita reducirse (no mejora el IPB).
- El único beneficio de la transformación unfolding es que el circuito se puede ejecutar a un reloj más lento ya que cada registro es más lento por el factor de despliegue.

Arquitecturas Folding y Unfolding

Introducción

- En el diseño basado en FPGA, con un número fijo de registros y unidades computacionales integradas, unfolding ayuda a optimizar diseños que requieren demasiados registros algorítmicos.
- A diferencia de las arquitecturas dedicadas o paralelas, las arquitecturas compartidas en el tiempo se diseñan en instancias donde el reloj del circuito es al menos dos veces más rápido que el reloj de muestreo.
- El diseño que se ejecuta a velocidad de reloj de circuito puede reutilizar sus recursos de hardware, ya que los datos de entrada siguen siendo válidos para relojes de circuito múltiple.
- El Unfolding aumenta el área del diseño sin afectar el rendimiento.
- El intercambio de área-potencia debe estudiarse cuidadosamente si el despliegue se realiza con el objetivo de reducir la potencia.
- La transformación unfolding es muy efectiva si hay más registros en el DFG original que se puedan aplicar retiming de manera efectiva para reducir el camino crítica del diseño.
- Es importante señalar que, en muchas instancias de diseño, la opción de elección de pipeling y retiming da como resultado un área menor que un diseño unfolding.

Arquitecturas Folding y Unfolding

Consideraciones de la Frecuencia de Muestreo

Teorema de Muestreo

- Para la digitalización de una señal analógica, el teorema de muestreo de Nyquist define la restricción mínima en la frecuencia de muestreo de la señal analógica.
- La frecuencia de muestreo define la cantidad de muestras que el sistema necesita procesar cada segundo.
- Establece la frecuencia de muestreo f_s mínima para que sea mayor o igual al doble del contenido de frecuencia máxima de la señal f_N ($f_s \geq 2f_N$).
- La frecuencia de muestreo es la restricción más crítica en un diseño digital.
- Por lo general, las muestras de un conversor A/D se colocan en un FIFO para su procesamiento.
- El número requerido de muestras en el buffer genera una señal de activación para que el componente digital comience a procesar el buffer.

Arquitecturas Folding y Unfolding

Consideraciones de la Frecuencia de Muestreo

Teorema de Muestreo

- Para un diseño que procesa la señal discreta muestra por muestra, el reloj de muestreo también se utiliza como reloj de circuito para el diseño del hardware.
- La frecuencia de muestreo impone restricciones estrictas incluso si los datos muestreados se almacenan primero en un buffer.
- El diseñador diseña el HW para procesar este buffer de datos antes de que el siguiente buffer esté listo para el procesamiento.
- Si el reloj de muestreo es más rápido que un reloj de circuito, entonces el diseñador necesita explorar opciones de procesamiento paralelo o pipeling.

Loop Unrolling

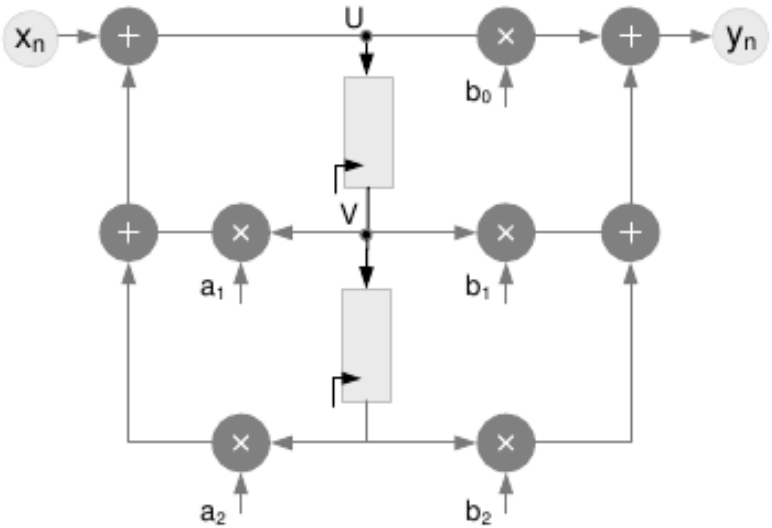
- En el contexto del software, la transformación unfolding es el proceso de "desenrollar" un ciclo, de manera que varias iteraciones del ciclo se ejecutan en una iteración desenrollada.
- Por esta razón, desplegar también se llama ’desenrollar bucle’.
- Para el diseño de hardware, el despliegue corresponde a la aplicación de una transformación matemática en un gráfico de flujo de datos para replicar su funcionalidad para el cálculo de múltiples muestras de salida para muestras de entrada múltiples relevantes dadas.
- El concepto también puede utilizarse al realizar una asignación SW a HW de una aplicación escrita en un lenguaje de alto nivel.
- El despliegue de bucles brinda más flexibilidad al diseñador de HW para optimizar el mapeo.

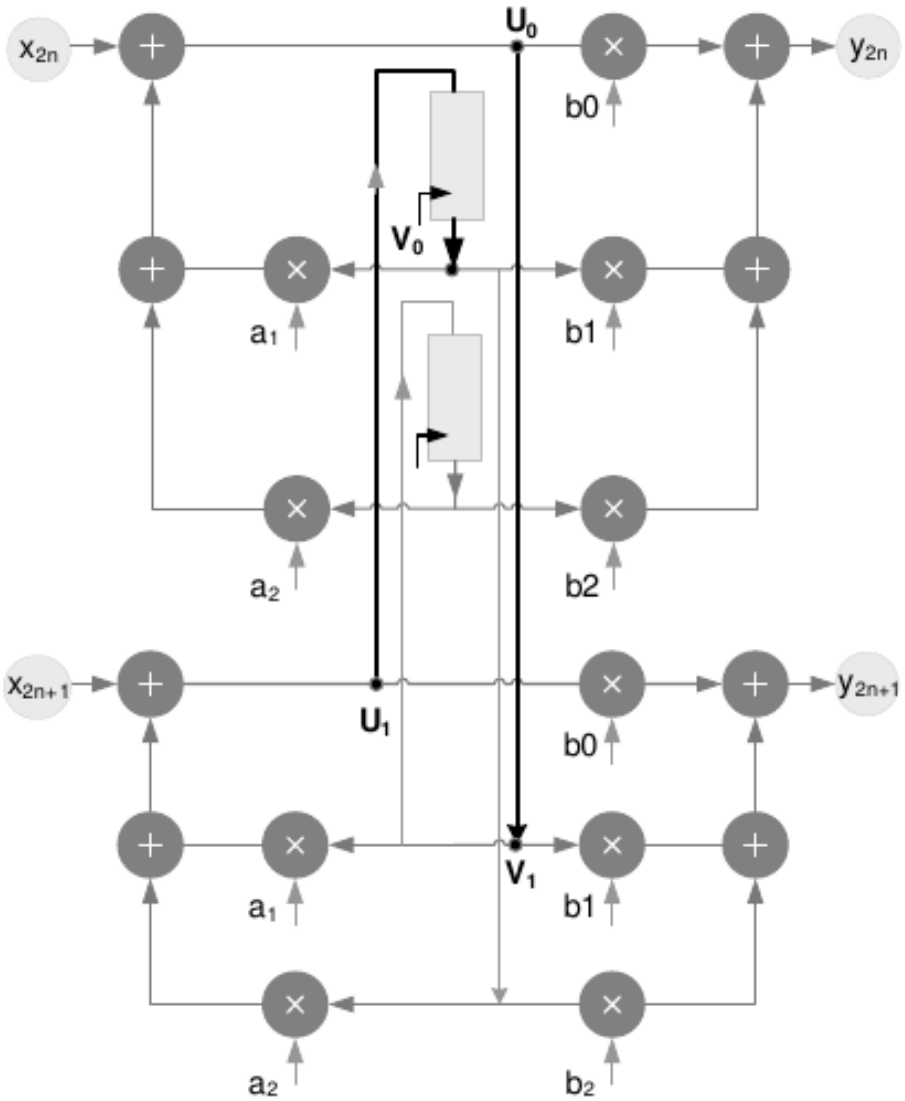
Transformación Unfolding

- Cualquier DFG puede desplegarse mediante un factor desplegable J usando los dos pasos siguientes
  - Para desplegar el gráfico, cada nodo U del DFG original se replica J veces como $U_0, \ldots, U_{J-1}$ en el DFG desplegado.
  - Para dos nodos conectados U y V en el DFG original con retardos w, dibuje J bordes de modo que cada borde $j(=0,\ldots,J-1)$ conecte los nodos U_j a los nodos $V(j+w)\%J$ con $\lfloor (j+w)/J \rfloor$ retardos, donde $\%$ y $\lfloor . \rfloor$ son el resto y la operación floor.

Arquitecturas Folding y Unfolding

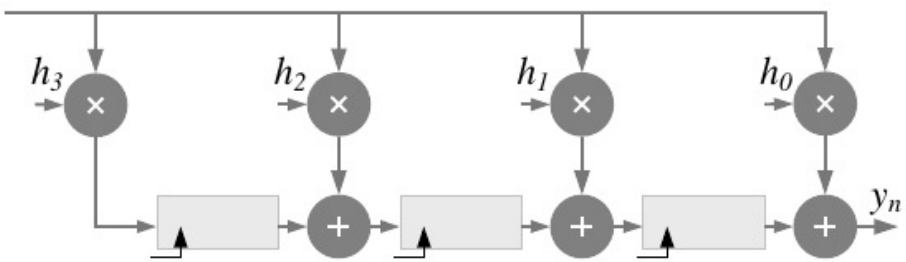
Técnica Unfolding

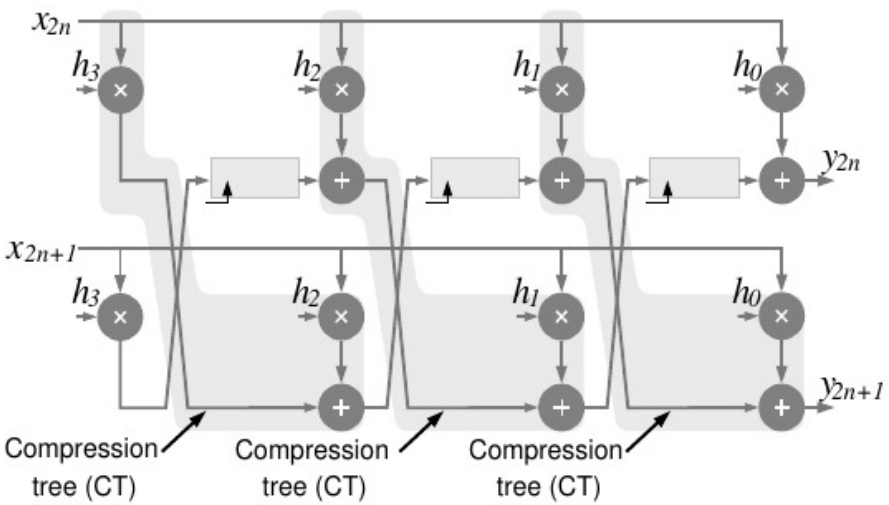




Arquitecturas Folding y Unfolding
Técnica Unfolding

- Para el diseño de hardware de las DFG forward, se pueden explorar las opciones de arquitectura efectivas utilizando la transformación desplegable.
- El diseñador puede diseñar una unidad computacional que consta de dos multiplicadores CSD y dos sumadores como una unidad computacional.
- Esta unidad se puede implementar como un árbol de compresión que produce una suma y un acarreo.



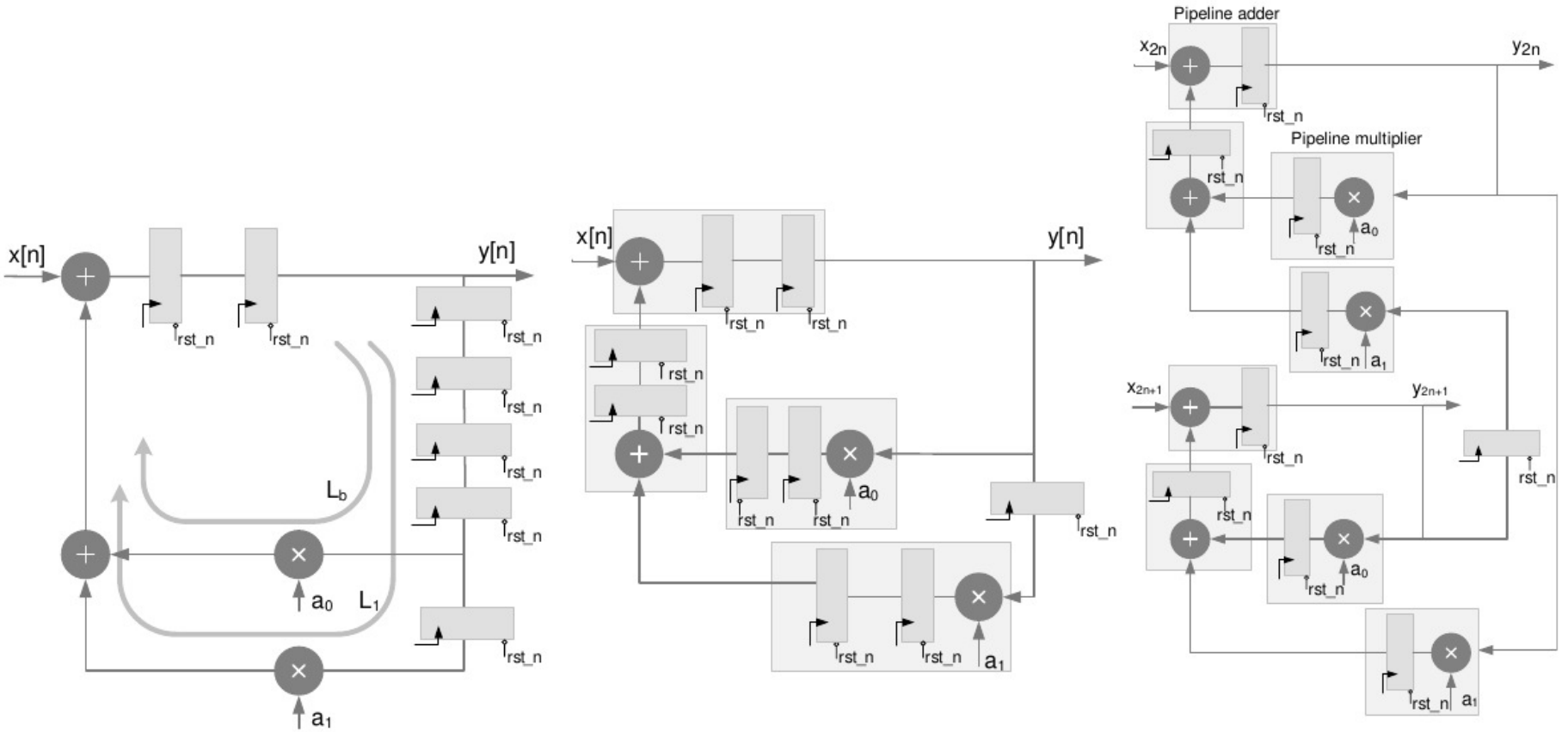


Diseños Realimentados

- Una transformación desplegada no mejora el tiempo; más bien, resulta en un aumento en la demora del camino crítico y, para diseños feedback, un aumento en IPB por el factor desplegable J.
- Este aumento se debe a que, aunque todos los nodos computacionales se replican J veces, aún el número de registros en el DFG desplegado sigue siendo el mismo.
- Para diseños realimentados, el despliegue puede ser efectivo para instancias de diseño donde hay abundantes registros algorítmicos para los nodos combinatorios en el diseño.
- En estos diseños, el desplegado seguido de retiming proporciona flexibilidad para colocar estos registros algorítmicos en el diseño desplegado mientras se optimiza el tiempo.

Arquitecturas Folding y Unfolding

Técnica Unfolding



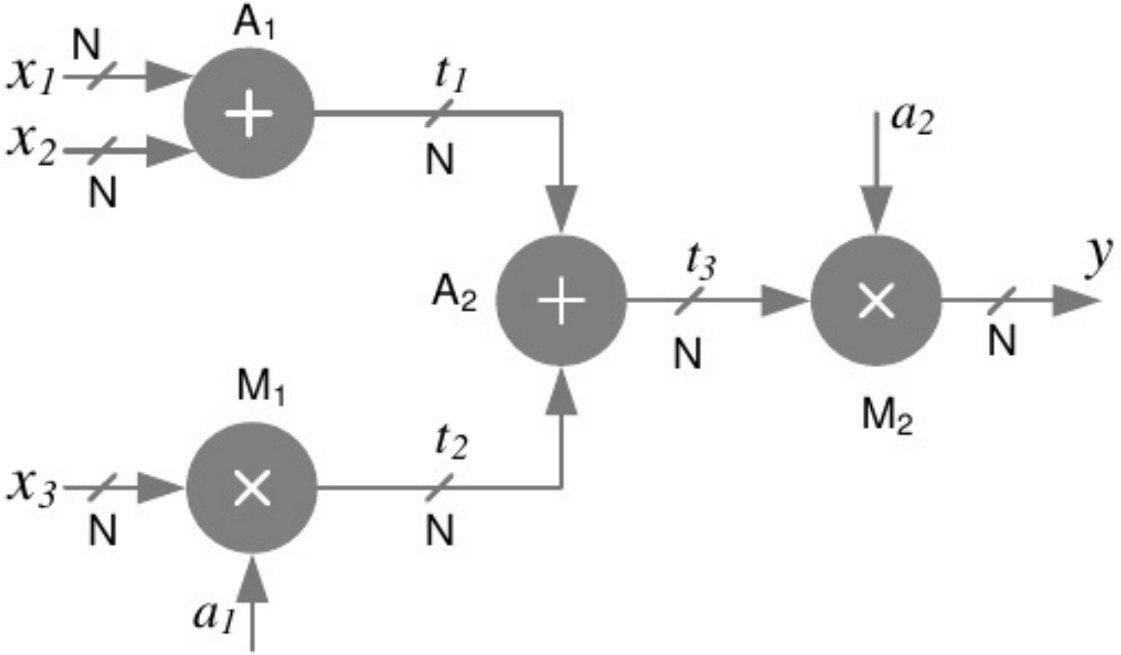
Unfolding filtro IIR.

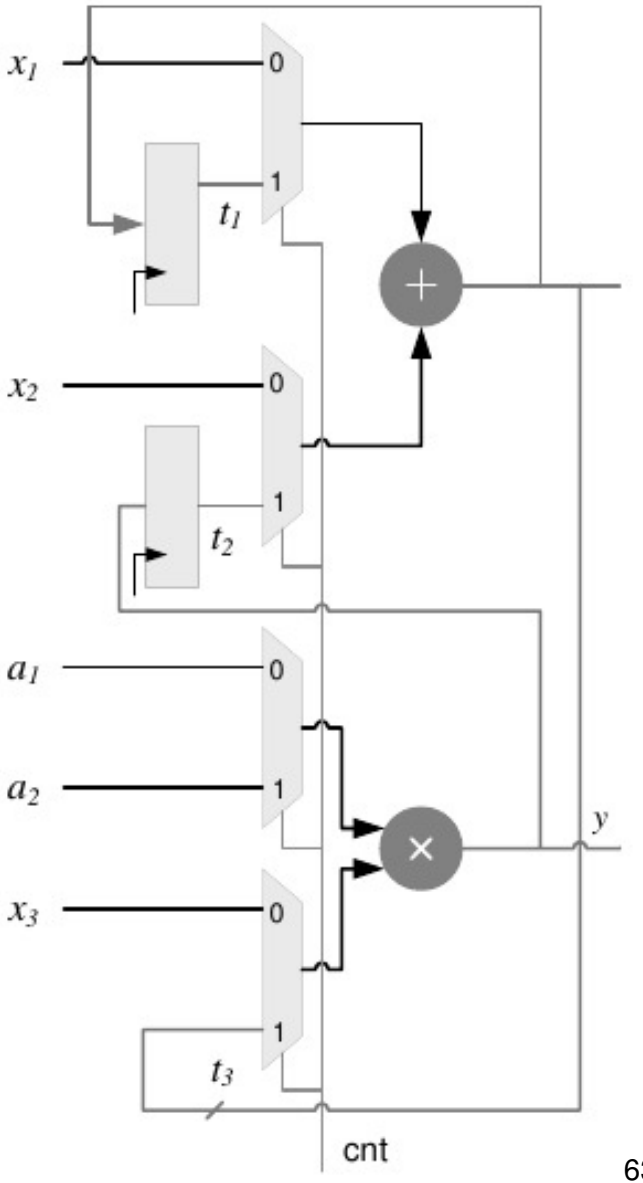
Transformación Folding

- El plegado es una técnica matemática para encontrar una arquitectura multiplexada en el tiempo y un cronograma de mapeo de múltiples operaciones de un gráfico de flujo de datos en menos unidades computacionales de hardware.
- El factor de plegado se define como la cantidad máxima de operaciones en un DFG mapeado en una unidad computacional compartida.
- Un conjunto plegable o planificador plegable es la secuencia de operaciones de un DFG mapeado en una única unidad computacional.
- La transformación de plegado tiene dos partes. La primera parte trata de encontrar un factor de plegamiento y la segunda el cronograma para mapear diferentes operaciones del DFG en unidades computacionales en el DFG plegado.
- El factor de plegado óptimo se calcula como $N = \lfloor f_c/f_s \rfloor$ donde f_c y f_s son frecuencias de reloj de circuito y de muestreo.
- La transformación plegable por un factor de N introduce latencia en el sistema.
- El intercambio de una unidad computacional por diferentes operaciones también requiere un controlador que programe estas operaciones en intervalos de tiempo (máquina de estados finitos (FSM)).

Arquitecturas Folding y Unfolding

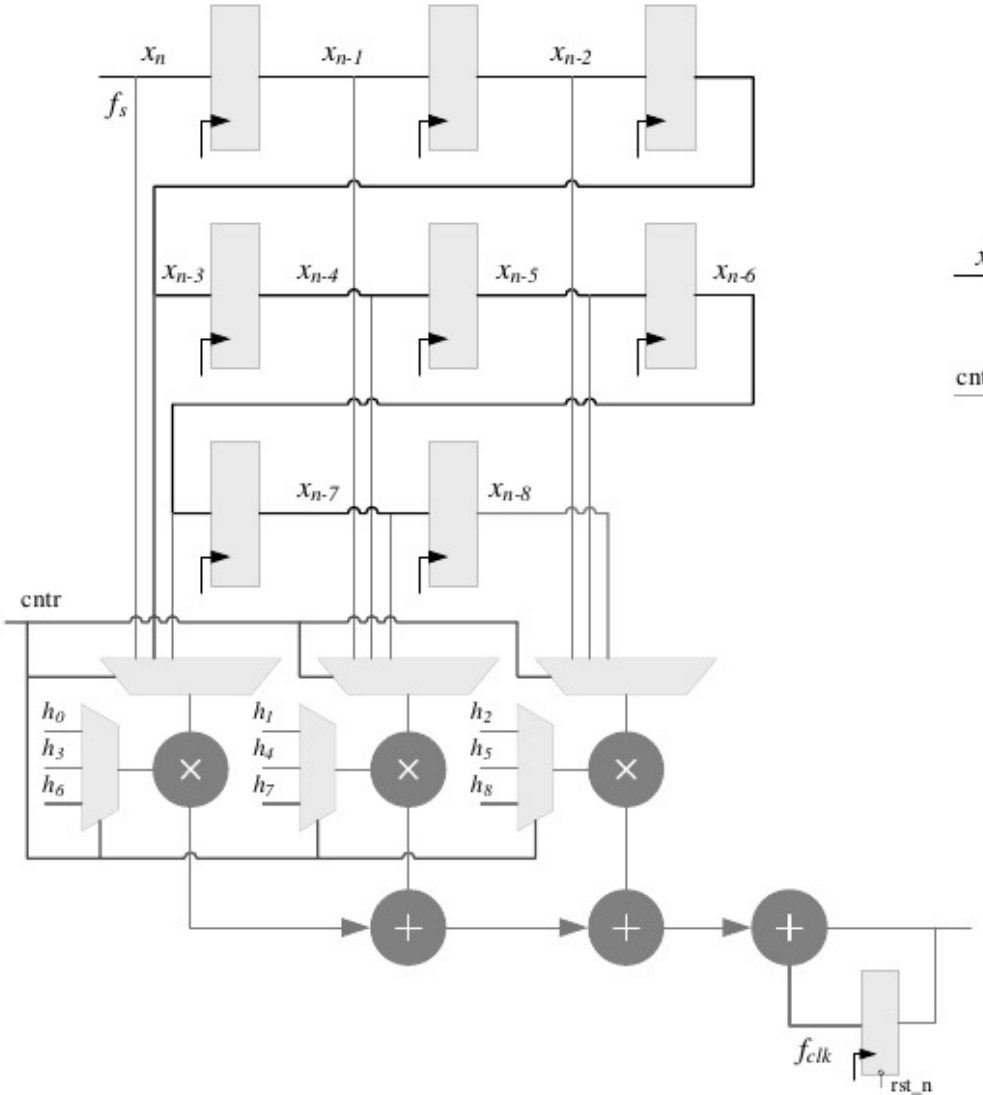
Técnica Folding

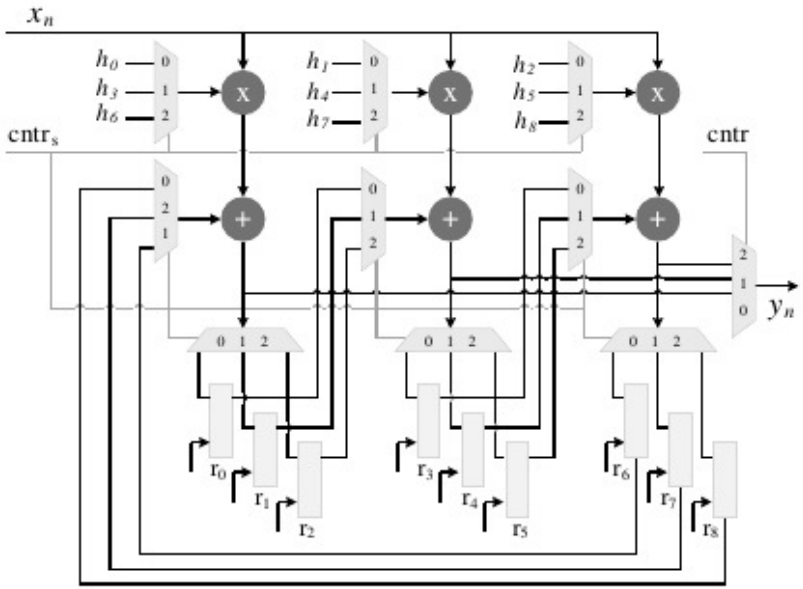




Arquitecturas Folding y Unfolding

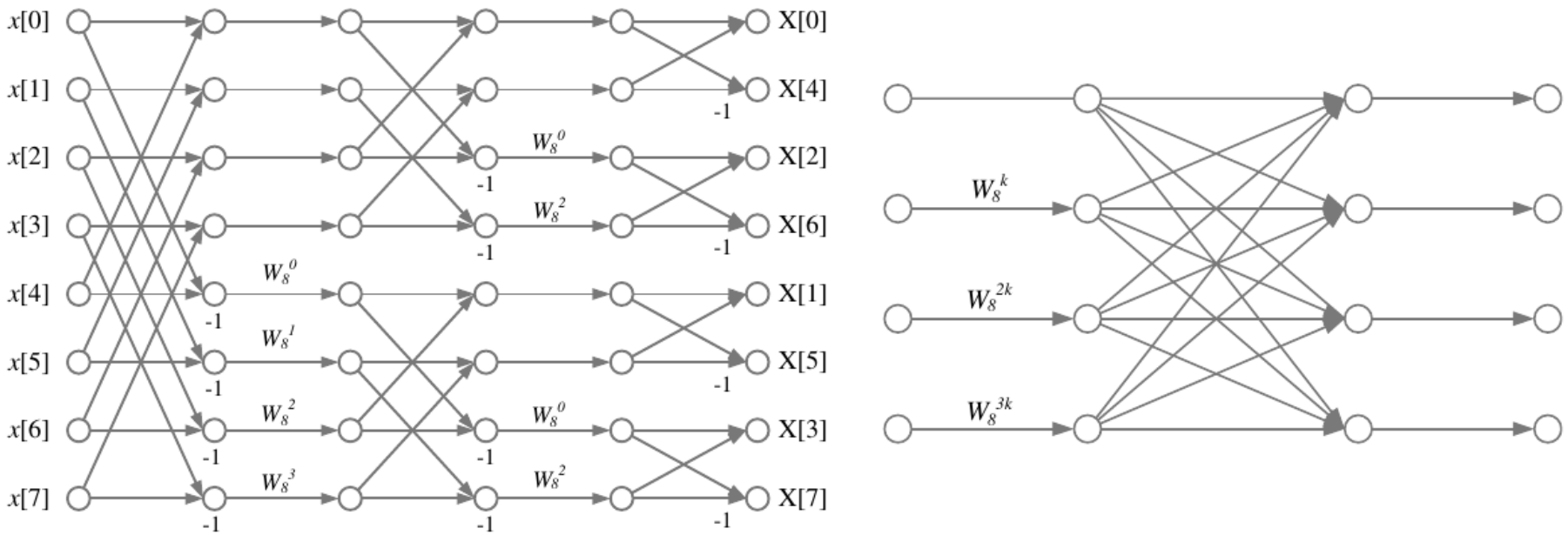
Técnica Folding - Estructuras Regulares FIR





Arquitecturas Folding y Unfolding

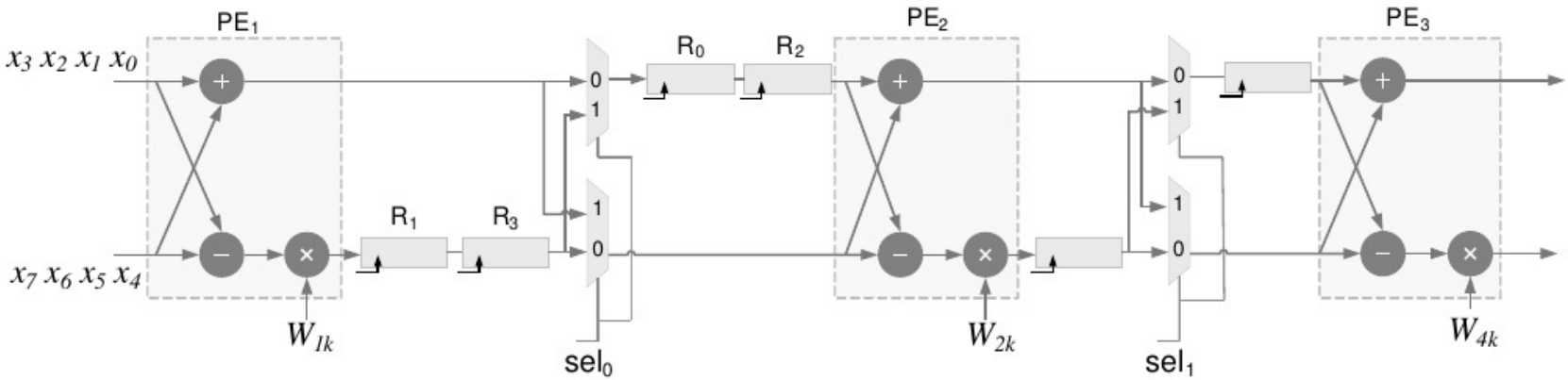
Técnica Folding - Estructuras Regulares FFT



a) FFT de 8 puntos usando Radix-2. b) Radix-4 butterfly.

Arquitecturas Folding y Unfolding

Técnica Folding - Estructuras Regulares FFT



Arquitectura sistólica MDC (multi-delay conmutator) que implementa una FFT de 8 puntos con desimación en frecuencia.