Test de circuitos integrados
Diseño para test

Marisa López Vallejo
marisa@die.upm.es

Índice

1. Introducción al test
2. El test de fabricación
3. Diseño para test (DFT)
4. Estrategias de test

1. Introducción al Test

- Es una de las etapas más caras del diseño de un chip
  - La verificación lógica supone más del 50% del coste de diseño en muchos circuitos
  - El tiempo de depuración tras fabricación supone un coste enorme
  - Producir un circuito defectuoso puede hundir una empresa
- Ejemplo: Intel FDIV bug
  - Error lógico no descubierto hasta > 1M unidades vendidas
  - Coste de recuperación $450M (!!!)

Tipos de test de un sistema

- Test Funcional: verifica la integridad del sistema:
  - Comprueba que un sistema funciona como debe
  - Comprueba las especificaciones del sistema
  - Ejecuta funciones seleccionadas (críticas)
- Test Diagnóstico o de Fabricación,
  - Identifica unidades con fallos:
    - Cortos entre capas
    - Conexiones discontínuas
    - Perforaciones en el óxido fino

Verificación funcional

- ¿Simula el chip correctamente?
  - Normalmente se hace a nivel HDL
  - Los ingenieros de verificación escriben test bench en HDL
    - No se pueden probar todos los casos
    - Buscar casos extremos
    - Intentar romper el diseño: divide y vencerás
- Ex: Sumador 32-bit
  - Comprobar todas las combinaciones de casos extremos como entradas:
    - 0, 1, 2, 231-1, -1, -231, y unos cuantos números aleatorios
- Tests buenos requieren ingenuidad

Silicon Debug

- Comprueba los primeros chips de fabricación
  - Si tienes suerte, funcionan a la primera
  - Si no…
- Errores lógicos vs. fallos eléctricos
  - Los fallos en chip suelen deberse a errores lógicos por simulación inadecuada
  - Algunos son fallos eléctricos
    - Crosstalk
    - Nodos dinámicos: fugas, reparto de carga, etc.
    - Fallos de dimensionamiento (ratio)
  - Muy pocos fallos se deben a herramientas o metodología (e.g. DRC)
- Se arreglan los bugs y se fabrica una versión corregida del chip

Test de fabricación

- Una mota de polvo en una oblea puede inutilizar un chip
- El rendimiento (*Yield)* de cualquier chip es < 100%
  - Hay que comprobar los chips tras fabricar para entregar a los clientes sólo unidades válidas
- Los *testers* de fabricación son muy caros
  - Hay que minimizar el tiempo en tester
  - Cuidada selección de *vectores de test*

Proceso de test

- El diseñador escribe parte/todos los test que el circuito debe pasar: estímulos
- Simulación de estímulos: se vuelca el resultado en el fichero de actividad
- Se saca el programa de test
  - Cada línea un clk y las entradas determinadas

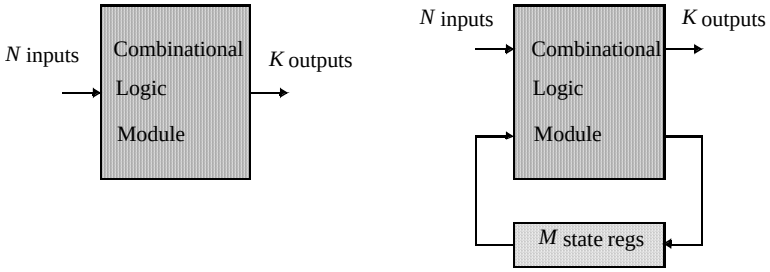
Validación y test de circuitos fabricados

- Objetivos de Diseño para Test (DFT)
  - Hacer el test de unidades fabricadas rápido y comprensivo
- Principio básico de DFT
  - Proporcionar controlabilidad y observabilidad
- Estrategia para DFT
  - Proporcionar circuitería para facilitar el test
  - Proporcionar patrones de test que garanticen una cobertura razonable

Clasificación del test de fabricación

- Test Diganóstico
  - Utilizado en depuración de chip/placa
  - Localización de defectos
- "go/no go" o test de producción
  - Utilizado en producción de chip
- Test paramétrico
  - $x \varepsilon [v,i]$ versus $x \varepsilon [0,1]$
  - Comprobar parámetros como NM, V_t, t_p, T

3. Diseño para *Testabilidad* (DFT)



(a) Combinational function

(b) Sequential engine

2^N patterns

2^{N+M} patterns

©Rabaey

El test exhaustivo es imposible o no practicable

Diseño para test

- Diseñar el chip para aumentar la observabilidad y controlabilidad
- Si cada registro puede ser observado y controlado, el problema del test se reduce a comprobar la lógica combinacional entre registros.
- Mejor aún, los bloques lógicos podrían entrar en un "modo test" en el que generan patrones de test y reportan el resultado automaticamente.

Problema: Controlabilidad/Observabilidad

- Circuitos Combinacionales:
  controlable y observable - relativamente fácil determinar patrones de test
- Circuits Secuenciales: ¡Estados!
  Volverlos combinacionales o utilizar auto-test
- Memorias: requieren patrones complejos
  Utilizar auto-test

©Rabaey

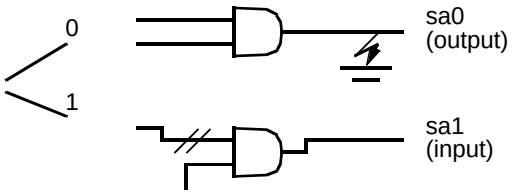
Generación y validación de vectores de test

- Automatic test-pattern generation (ATPG)
  - Para un fallo determinado, determinar el vector de entrada (test vector) que propaga el error a una salida primaria (observable)
  - Mayoría de herramientas disponibles: sóno redes combinacionales
  - ATPG secuencial sólo disponible en entornos de investigación académica
- Simulación de fallos
  - Determina la cobertura de test de un conjunto de vectores de test

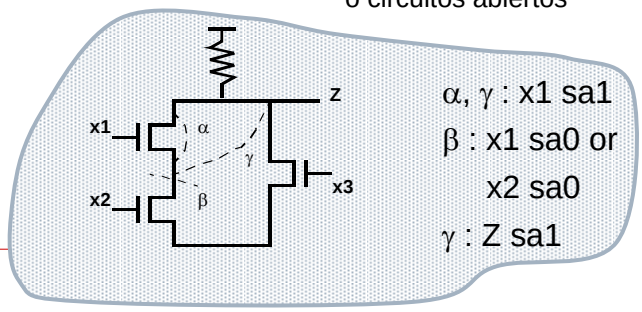
    cobertura = #nodos detectados / #nodos totales
  - Simula en paralelo la red correcta y la red con fallo
- Ambos requieren modelos de fallos adecuados a circuitos integrados CMOS

Modelos de Fallos

Más Popular - Modelo "Stuck - at"

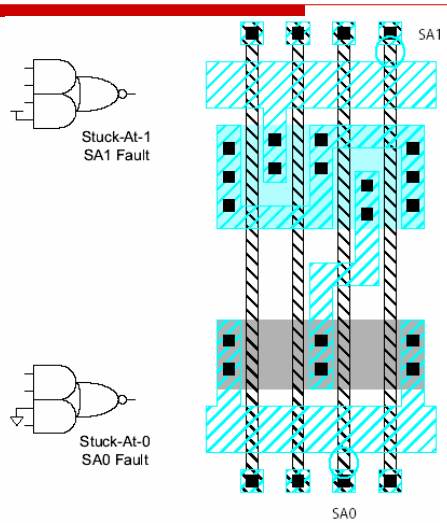


Cubre casi todos los fallos que se producen, como pueden ser los cortos o circuitos abiertos



©Rabaey

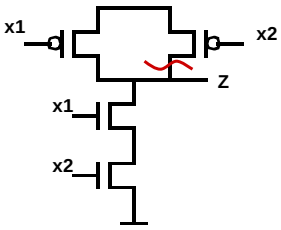
Ejemplos



©Weste&Harris

Problema con el modelo stuck-at: Fallo de circuito abierto en CMOS

x1	x2	Z
0	x	1
1	1	0
1	0	Z_{n-1}



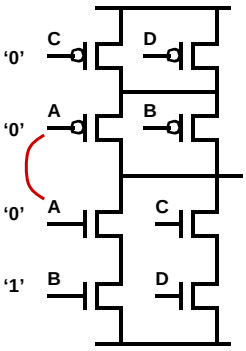
Efecto secuencial

Necesita dos vectores para asegurar detección!

Otras opciones: utilizar modelos stuck-open o stuck-short
Require simulación de fallos y análisis a nivel de conmutación o transistor -> ¡Muy caro!

©Rabaey

Problema con el modelo stuck-at: Fallo de corto circuito en CMOS



Causa corto circuito entre
Vdd y GND para A=C=0, B=1

Enfoque posible:
Medida de corriente de alimentación (IDDQ)
pero: no aplicable para gran escala de integración

©Rabaey

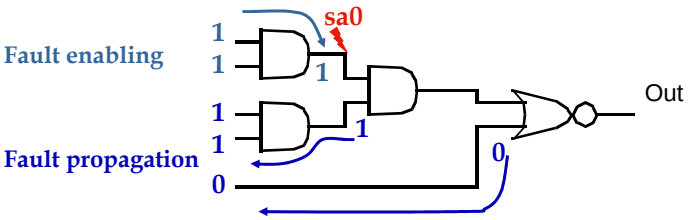
Generación de vectores de test

- El test de fabricación idealmente comprobaría cada nodo del circuito para demostrar que no está bloqueado (stuck)
- Se aplica la menor secuencia de vectores de test necesarios para probarque cada nodo no está bloqueado
- Buena observabilidad y controlabilidad reduce el número de vectores de test necesarios para el test de fabricación
  - Reduce el coste del test
  - Motiva el diseño-para-test

©Rabaey

Path Sensitization

Objetivo: Determinar el patrón de entrada que hace un fallo controlable (activa el fallo, y hace su impacto visible en los nodos de salida)

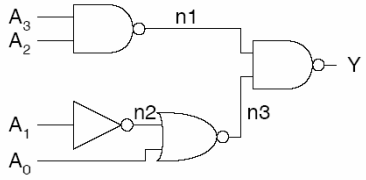


Técnicas: D-algorithm, Podem

©Rabaey

Ejemplo de test

	SA1	SA0
A_3	{0110}	{1110}
A_2	{1010}	{1110}
A_1	{0100}	{0110}
A_0	{0110}	{0111}
n1	{1110}	{0110}
n2	{0110}	{0100}
n3	{0101}	{0110}
Y	{0110}	{1110}

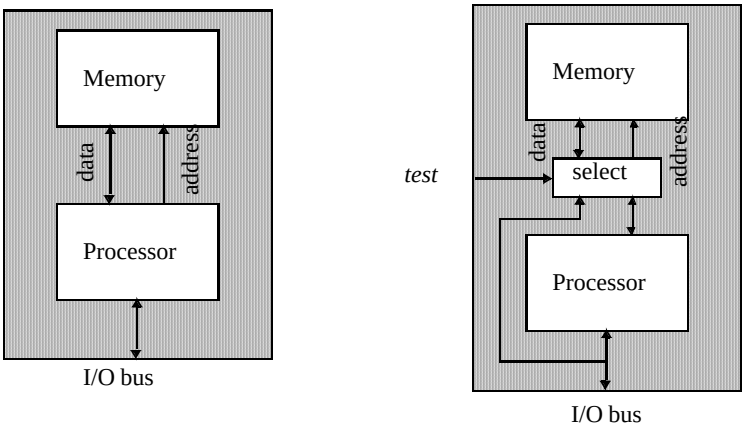


Minimum set: {0100, 0101, 0110, 0111, 1010, 1110}

Índice

1. Introducción al test
2. El test de fabricación
3. Diseño para test (DFT)
4. Estrategias de test

Ad-hoc Test

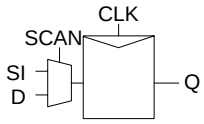


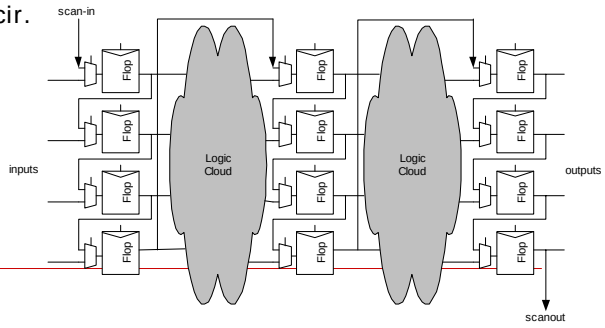
Insertar multiplexor mejora testabilidad

©Rabaey

Scan-based Test

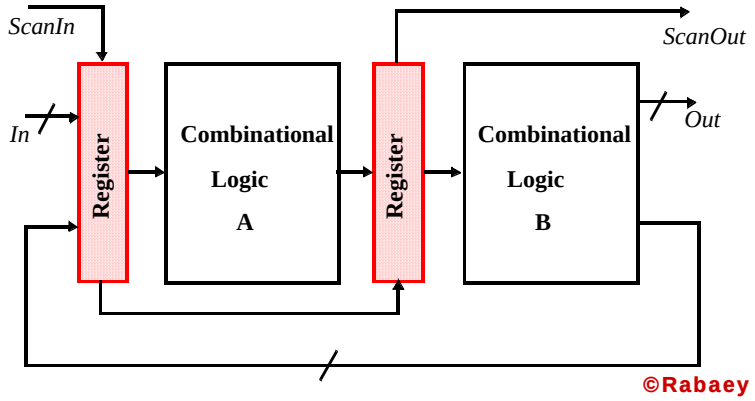
- Convierte cada flip-flop en un registro de scan
  - Coste: sólo un mux extra
- Modo Scan: los flip-flops se comportan como un registro de desplazamiento
  - Su contenido se puede leer (scanned) en un pad de salida y nuevos contenidos se pueden introducir.





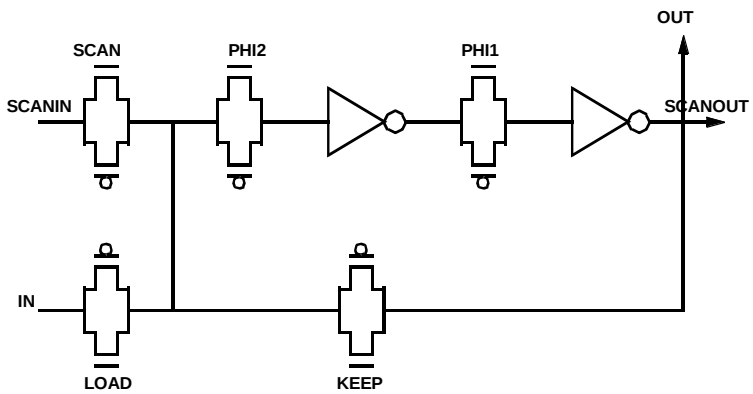
©Weste&Harris

Scan-based Test



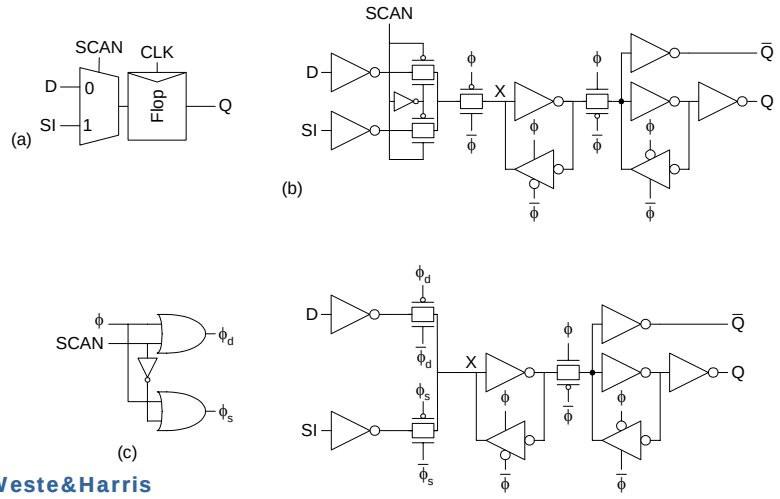
©Rabaey

Scan-Path Registro base



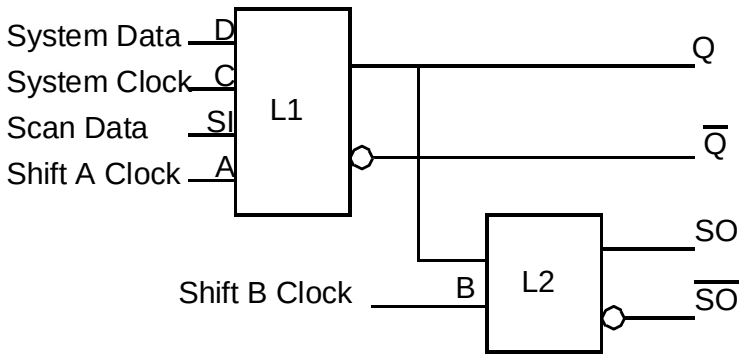
©Rabaey

Scan-Path Registros



©Weste&Harris

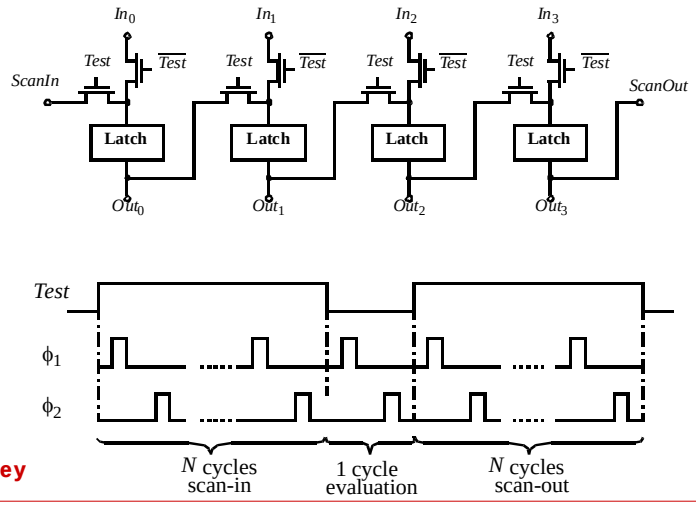
Polarity-Hold SRL (Shift-Register Latch)



©Rabaey

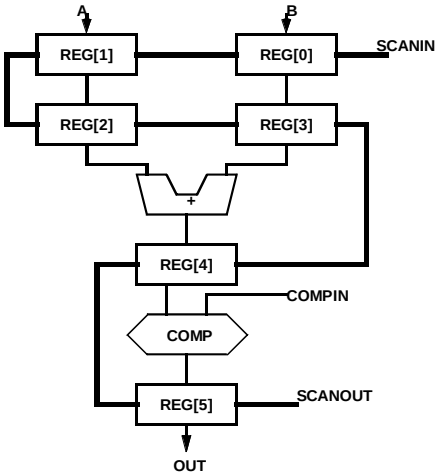
Utilizado en IBM fijado como política de la empresa

Scan-based Test —Operation



©Rabaey

Scan-Path Testing



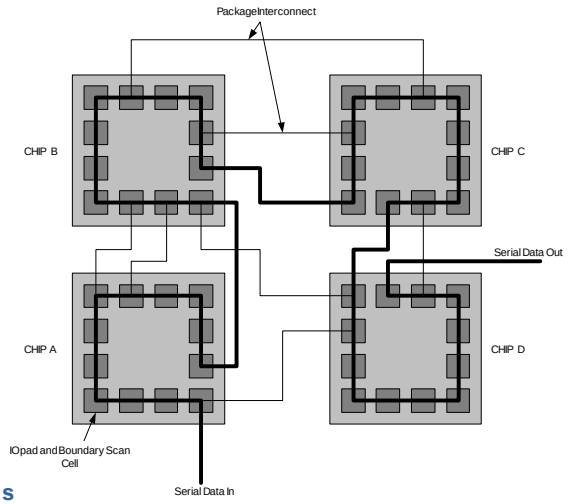
©Rabaey

Scan parcial puede ser más efectivo en datapaths con pipeline

Boundary Scan

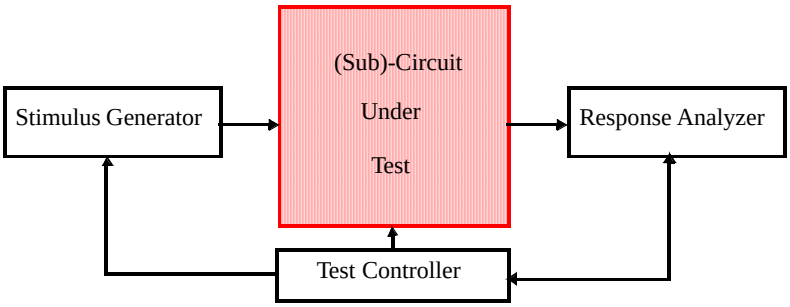
- El test de placas es también complicado
  - Necesario verificar soldaduras
    - Excitar un pin a 0, luego a 1
    - Comprobar que a todos los pines conectados les llega el valor
- Se crea la capacidad de observar y controlar los pines de cada chip para hacer más sencillo el test de la placa

Boundary Scan (JTAG)



©Weste&Harris

Auto-test

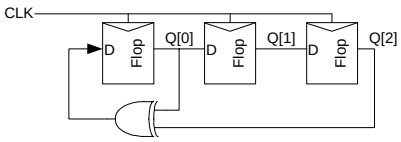


©Rabaey

Adquiere mayor importancia con las crecientes complejidades y los módulos mayores

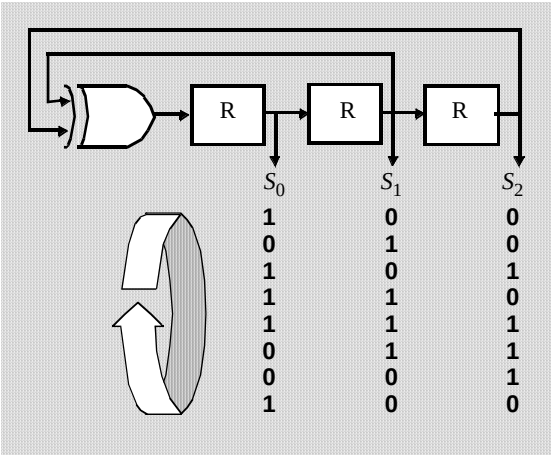
Linear Feedback Shift Register

- Shift register with input taken from XOR of state
- *Pseudo-Random Sequence Generator*



Step	Q
0	111
1	110
2	101
3	010
4	100
5	001
6	011
7	111 (repeats)

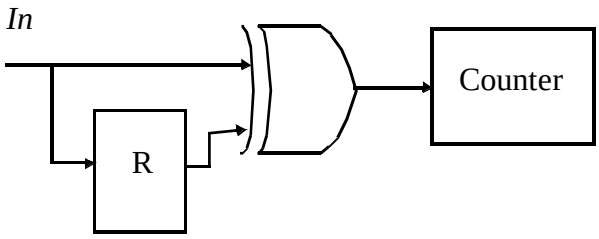
Linear-Feedback Shift Register (LFSR)



©Rabaey

Pseudo-Random Pattern Generator

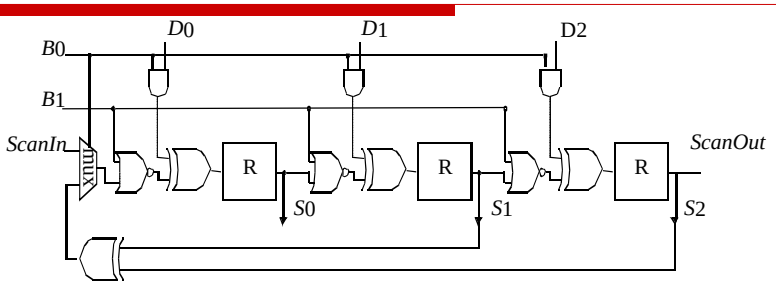
Análisis de firma



Cuenta transiciones en un flujo de bits
≡ Compresión en el tiempo

©Rabaey

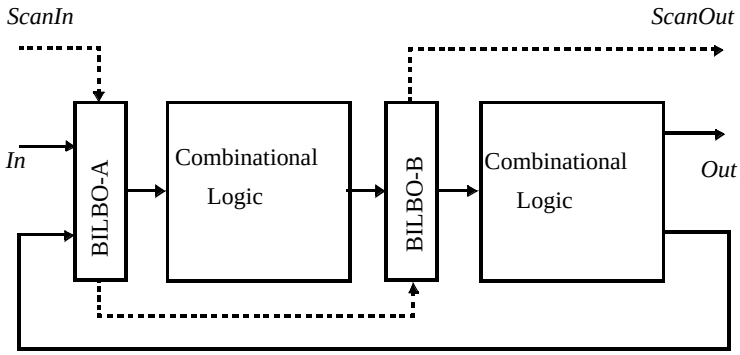
BILBO (Built-in Logic Block Observer)



Combina scan con LFSR & análisis de firma

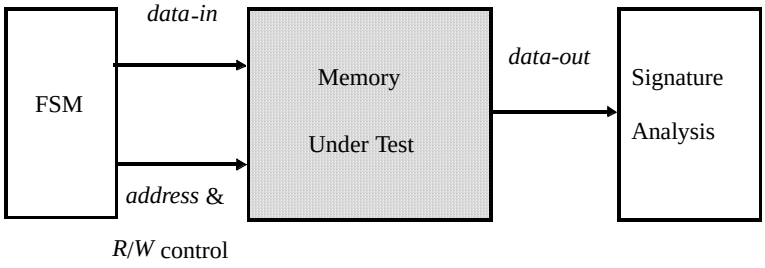
B_0	B_1	Operation mode
1	1	Normal
0	0	Scan
1	0	Pattern generation or Signature analysis
0	1	Reset

Aplicación BILBO



©Rabaey

Auto-Test de memorias



Patterns: Writing/Reading 0s, 1s,
Walking 0s, 1s
Galloping 0s, 1s

©Rabaey