Tecnológico de Costa Rica

Área Académica de Ingeniería en Electrónica

Programa de Licenciatura de Ingeniería en Electrónica

Diseño de una implementación eficiente en área de latches pulsados para la técnica Boundary Scan

Informe de Trabajo Final de Graduación para optar por el título de

Ingeniero en Electrónica con el grado académico de Licenciatura

Luis Felipe Retana Corrales

Cartago, Noviembre de 2018

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Tesis de Licenciatura
Tribunal Evaluador

Tesis de licenciatura defendida ante el presente Tribunal Evaluador como requisito para optar por el grado académico de licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Dr. Ing. Renato Rímolo Donadío
Profesor Lector

M.Sc. Ing. Sergio Arriola Valverde
Profesor Lector

M.Sc. Ing. Ronny García Ramírez
Profesor Asesor

Los miembros de este Tribunal dan fe de que la presente tesis de licenciatura ha sido aprobada y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica.

Cartago, 27 de noviembre de 2018

Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

Luis Felipe Retana Corrales

Cartago, 27 de Noviembre de 2018

Céd: 1-1519-0946

Resumen

En este proycto se establece un punto de comparación entre dos implementaciones topológicas diferentes para el estándar 1149.1 de la IEEE. Se espera brindar todas las herramientas teóricas necesarias para comprender el diseño con la finalidad de establecer una equiparación de las topologías utilizadas con la finalidad de concluir cuál implementación es más eficiente en términos de consumo de área, consumo de energía y velocidad.

Palabras clave: Boundary Scan, Frecuencia máxima de operación, TAP, Celda de Scan, Cadena de Scan, Validación, Diseño Verificable, Latch, Latch Pulsado, Flip Flop, VLSI.

Abstract

This document establishes a comparison between two different topologies for the IEE 1149.1 standard implementation. It is expected to give all the theoretical tools required to understand the design as well as to establish a comparison of each of the topological implementations in order to conclude which implementation is more efficient in terms of area consumption, energy consumption and speed.

Keywords: Boundary Scan, Maximum Frequency Operation, TAP, Scan Cell,Scan Chain, Validation, Testable Design, Latch, Pulsed Latch, Flip Flop, VLSI.

a mis queridos padres

Agradecimientos

El resultado de este trabajo no hubiese sido posible sin el apoyo de mis padres Flory Corrales Segura y Rafael Retana Jiménez, a mi hermano Fabián Retana Corrales, profesor asesor Ronny García, profesor y amigo Roberto Molina Robles, profesores que me apoyaron durante la travesía por la carrera, Thevenin, Norton, Maxwell y Ohm.

Luis Felipe Retana Corrales

Cartago, 28 de noviembre de 2018

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Desde la concepción de la regla de Moore, la escala de los circuitos integrados (CI) ha aumentado,aproximadamente, el doble cada 18 meses esto implica una mayor integración de transistores manteniendo la proporción de área.
En la actualidad dispositivos VLSI con millones de transistores son utilizados comúnmente en ordenadores y aplicaciones de dispositivos electrónicos, esto es un resultado directo del constante decrecimiento dimensional del transistor, conocido como «feature size» o tamaño característico. Los dispositivos modernos poseen un tamaño característico menores a 50 nm, esta reducción del tamaño característico ha permitido aumentar la frecuencia de operación y velocidad de los relojes de los dispositivos, pero también ha incurrido en el aumento de la probabilidad de errores de manufactura. Un error diminuto en una tecnología inferior a los 50 nm puede incurrir en la fabricación de un transistor defectuoso o un error de intercomunicación entre transistores. Es suficiente un transistor defectuoso o una conexión no funcional, para generar que todo un chip falle en realizar sus funciones o no pueda operar a la frecuencia esperada. No obstante los errores debido al proceso de manufactura no pueden ser eliminados completamente, por lo que es esperable una cantidad de CI defectuosos. Es por esto que la validación e implementación de pruebas se convierte en una necesidad para garantizar un diseño libre de fallas para el usuario final. Debido a la variabilidad encontrada durante el proceso de fabricación se pueden encontrar CI defectuosos, por lo que la validación y verificación se convierten en herramientas esenciales para disminuir la cantidad de circuitos integrados con faltas de funcionamiento. El proceso de verificación de un circuito integrado se puede resumir en la figura 1.1.

Se dice, como regla general en la industria de la manufactura[8], «El costo de detectar un Circuito Integrado defectuoso se incrementa en un orden de magnitud cada vez que nos movemos por una etapa de manufactura», a esta regla se le llama «Regla de Diez», y a partir de esta regla se entiende que encontrar los defectos de manera temprana se convierte en una actividad vital para reducir costos de producción.
La verificación y validación de un CI se ha vuelto se tan relevante que desarrollaron un área en sí misma, y se conoce como DFT (Design For Testability), esta área se centra en el diseño de circuitos que permiten facilitar el proceso de verificación de los circuitos

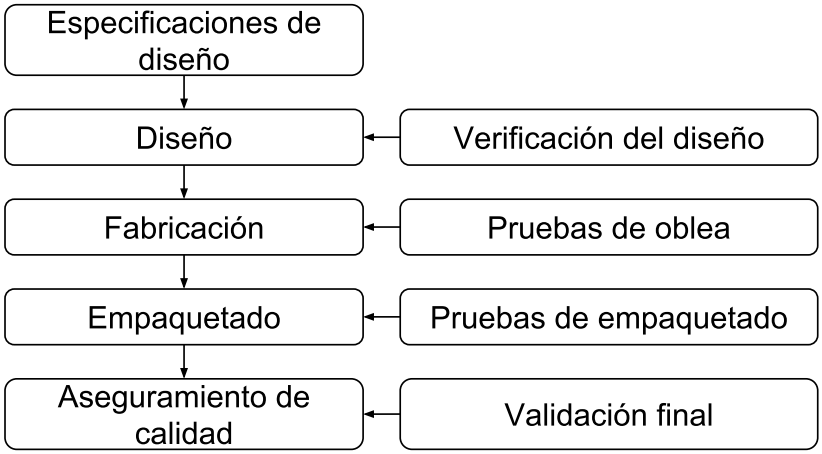

Figura 1.1: Proceso de diseño VLSI

integrados por medio de la adición de puntos de observación y la estimulación de las estructuras internas de la arquitectura.
Previo a la década de los 90, se trabajaba con equipos de prueba automáticos o ATE por sus siglas en inglés (Automatic Test Equipment), en dichos equipos se implementaban generadores automáticos de pruebas los cuales eran algoritmos que generaban automáticamente vectores de datos que permitían verificar el funcionamiento de los circuitos integrados [5], estos algoritmos se aseguraban de corroborar un alto porcentaje de los errores que tenían los CI, el problema es que aunque son equipos eficientes, también son los equipos de validación más costosos.
Debido a esto se diseñaron técnicas que permitieron reducir el costo de probar los circuitos integrados, una de ellas es el Built-In Self Test (BIST) y el Automatic Test Pattern Generator (ATPG).
En 1990 la metodología para validación de circuitos integrados creada por el grupo conocido como Joint Test Action Group (JTAG) se estandariza en la industria al ser acogida por la IEEE mediante la norma Std. 1149.1-1990, en ese mismo año sale al mercado el procesador 80486 de Intel, que es el primer procesador de la compañía en utilizar JTAG, lo que propició una rápida adopción del estándar. Finalmente un complemento que contiene Boundary Scan Description Language (BDSL) es añadido consolidando así el JTAG como la norma por excelencia en el mercado.

1.1. Objetivos y estructura del documento

El objetivo del proyecto es implementar una interfase de pruebas estándar en una tecnología de fabricación de 180nm que permita la integración de elementos estándar de comprobación de circuitos integrados digitales tales como BIST, Scan y Boundary Scan para ser usada en los diseños del DCILab del Instituto Tecnológico de Costa Rica.
Para cumplir con el objetivo del proyecto es necesario proponer al menos 2 diferentes opciones de implementación del estándar 1149.1 de la IEEE e implementarlos a nivel RTL

para después sintetizar los diseños propuestos usando bibliotecas estándar en tecnología de 180nm y comprobar su funcionamiento post síntesis lógica con la finalidad de generar una caracterización para cada uno de los diseños propuestos mediante pruebas que busquen el mayor consumo de potencia dinámica posible, el consumo de área y velocidad máxima del reloj.

En el capítulo 2 se esboza la teoría necesaria para comprender la solución realizada en el capítulo 3. Los resultados obtenidos se presentan en el capítulo 4 y finalmente las conclusiones se presentan en el capítulo 5.

Capítulo 2

Marco teórico

2.1. Técnica de Boundary Scan

El estándar 1149.1 de la IEE conocido como JTAG o simplemente Test Access Port (TAP) desarrolla la normativa para implementar un puerto de acceso que proporciona la capacidad de realizar pruebas a nivel de placa o sobre la lógica interna de un CI.

El estándar está diseñado con la intención de que impacte en varias áreas del ciclo de vida de un producto [4]. Dichas áreas son:

- A nivel de Circuito Integrado: Mediante soporte para técnicas de prueba internas del circuito.
- A nivel de placa: Facilita la validación de la placa, emulación de funciones, y pruebas de producción entre otros.
- A nivel de sistema: El estándar soporta validación para niveles superiores de manofactura, desde módulos o cajas hasta sistemas completos en conjunto con otros estándares como el 1149.5.

El JTAG tiene dos modos de operación:

- No invasivo: En este modo el TAP se comunica de manera asíncrona con el exterior para preparar pruebas o leer resultados. Este modo de operación es invisible al circuito integrado.
- Invasivo: Este modo de operación el TAP usurpa los pines de entrada/salida (I/O) del circuito integrado, desconectándolo del exterior. Este modo de funcionamiento tiene como finalidad probar la lógica de circuito integrado o aislar la lógica mientras se realizan pruebas sobre los pines I/O.

El TAP es un puerto de propósito general que provee acceso a varios tipos de pruebas soportadas por el CI. La figura 2.1 presenta la arquitectura básica del boundary scan.

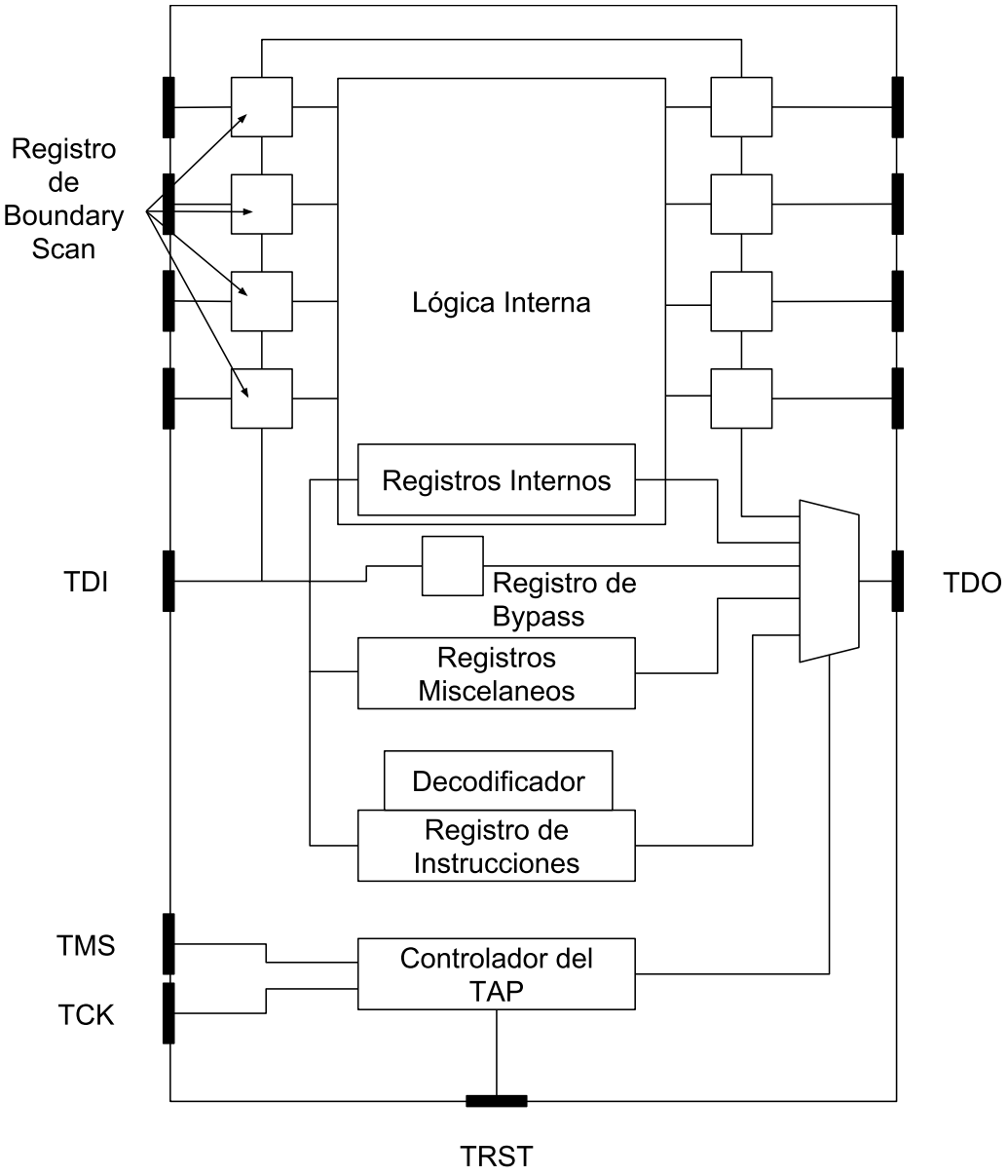

Figura 2.1: Arquitectura simplificada del estándar IEE 1149.1

La arquitectura básica del tap incluye 4 conexiones básicas Test Clock (TCK), Test Mode Selector (TMS), Test Data In (TDI) y Test Data Out (TDO), además de un quinto pin opcional (Test Reset) TRST [4].

- TCK (Test Clock): Provee de la señal de reloj a la lógica de validación. Si TCK adquire un valor LOW o 0, la lógica de prueba debe conservar el estado indefinidamente.
  Esta señal se puede controlar con un único driver, por lo que la carga vista por TCK debe ser pequeña.

La función principal de la señal es ser un garante para que el camino de datos seriales entre componentes pueda ser implementado de manera aislada.

- TMS (Test Mode Selector): El valor del TMS durante el flanco positivo o de subida (rising edge) de la señal TCK determina el estado de controlador del TAP. Esta señal debe ser muestreada cada flanco positivo de TCK, además, cuando el circuito alimentado por TMS permanece sin driver (TMS no está actuando), debe producir un efecto similar a un 1 conectado permanentemente al pin de entrada, esto para que se desplace lo más rápido posible por los estados del Controlador del TAP hasta alcanzar el estado inicial.
  En el caso de lógica Complementary Metal Oxide Semiconductor (CMOS) se recomienda el uso de una resistencia de pull-up en el pin de entrada con el fin de lograr el comportamiento deseado.

- TDI (Test Data In): Entrada serial de instrucciones y datos hacia la lógica de prueba. Esta señal se muestrea en el flanco positivo de TCK.
  Cuando el circuito alimentado por TDI permanece sin driver (TDI no está actuando), debe producir un efecto similar a un 1 conectado permanentemente al pin de entrada.
  Cuando los datos son transmitidos de TDI a TDO los datos deben aparecer sin inversión después de un número de flacos positivos determinado por el tamaño de la instrucción o del dato.

- TDO (Test Data Out): Es la salida serial de la lógica de prueba definida por el estándar IEE 1149.1.
  Cambios realizados en la señal manejada a través de TDO deben ocurrir únicamente en el flanco negativo de TCK. Este puerto debe ser desactivado a menos que esté transfiriendo datos. El comportamiento de captura en el flanco negativo se logra mediante un registro de salida, además la capacidad de desactivar el puerto de salida TDO, en tecnología CMOS, se puede lograr mediante el uso de un buffer tri-estado de salida.

- TRST (Test Reset): Es la señal de reinicio del controlador del sistema.

El dispositivo que controla el TAP durante la ejecución de pruebas se le llama bus maestro, el bus maestro contiene todas las conexiones con los diferentes pines del TAP definidos por el estándar, y al ser un producto estandarizado, cualquier bus maestro es compatible con cualquier configuración de TAP mientras siga las características propuestas por el estándar 1149.1. El equipo que realiza las pruebas es llamado Equipo de Pruebas Automatizadas (ATE), dicho equipo posee una conexión con los pines de entrada del TAP, si el estándar se implementa utilizando lógica CMOS, es conveniente tomar en cuenta las conexiones a nivel de placa con la finalidad de conectar adecuadamente el ATE durante el proceso de validación.
Para realizar interconexiones de componentes compatibles con este estándar se deben tener

en cuenta dos factores, el primero es que TMS y TCK son señales transmitidas de forma paralela desde el bus maestro hacia cualquiera de los dispositivos esclavos conectados y el segundo factor es que se debe generar un camino serial con una configuración de cadena de margaritas entre los puertos de datos de prueba (TDI y TDO) como muestra la figura 2.2.

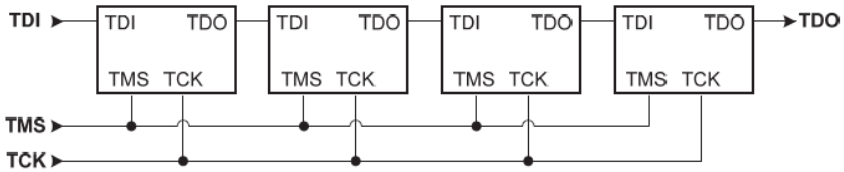

Figura 2.2: Conexión serial del TAP tomado de [1]

También se encuentran otras formas de interconexión compatibles como la conexión paralela 2.3.

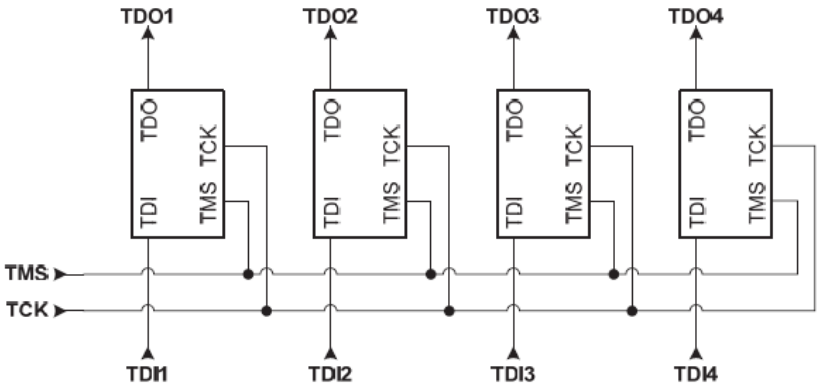

Figura 2.3: Conexión paralela del TAP tomado de [1]

Y la conexión serie paralelo 2.4.

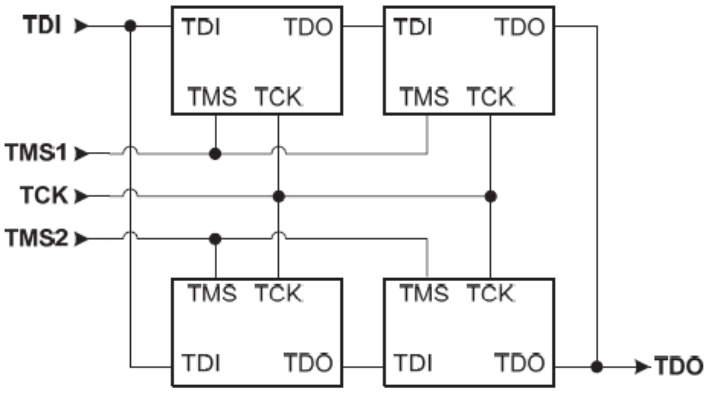

Figura 2.4: Conexión serie-paralela del TAP tomado de [1]

Al diseñar un JTAG, es necesario considerar la compatibilidad del equipo de validación disponible y que tipo de conexión es capaz de soportar, una conexión serial, paralela o serie paralela, ya que el el JTAG es un diseño personalizado conforme a las necesidades del circuito o a las directrices de los diseñadores, a razón de esta situación, el estándar manda que cualquier fabricante de dispositivos para validación como lo son los ATE, sean compatibles con la conexión serial mostrada en 2.2, esto significa que todo dispositivo que se compre para validación es capaz de comunicarse con el TAP de manera serial.

La arquitectura de la lógica de prueba debe contener tres elementos accesibles por el TAP los cuales son:

- El controlador del TAP.
- Registro de instrucciones.
- Registro de datos.

El controlador del TAP controla el flujo de datos desde TDI hasta TDO. El Registro de Instrucciones guarda los bit de instrucción; los bit de instrucción definen la funcionalidad del TAP, mediante el control del modo de operación de los Registros de Datos. El Registro de Datos es un banco de registros conformado por diferentes cadenas de scan como lo son el Registro de Boundary Scan, EL Registro de Bypass y cualquier otro registro definido por el diseñador, tienen como fin almacenar los vectores de prueba o los datos obtenidos una vez realizada alguna prueba.
Se puede utilizar como etapa de salida un buffer de tercer estado, debido a que si se da la conexión serial de varios dispositivos a un único bus de datos, es necesario desacoplar el sistema de la línea de transmisión de dicho bus. Esta etapa de salida se recomienda utilizar en caso de ser necesaria.

El Controlador del TAP El control del TAP se implementa mediante una máquina de estados que dependiendo de las señales TMS y TCK define su siguiente estado, además controla las funciones del registro de datos y del registro de instrucciones.

En la figura 2.5, la señal TMS al ser censada durante los flancos positivos de TCK produce un cambio de estado. Para reiniciar la máquina de estados, no importa su posición inicial, al introducir una secuencia de 5 1's (TMS = 11111), siempre vuelve al estado Test-Logic Reset, en caso de implementar el puerto de Test Reset, cuando el puerto está en alto, el controlador debe volver al estado de Test-Logic Reset.
La columna Data Column representa los estados en los cuales los registros de datos están activos, por su parte la columna Instruction Column representa los estados en que el registro de instrucciones se mantiene activo.

Los estados de la máquina de estados se pueden dividir en:

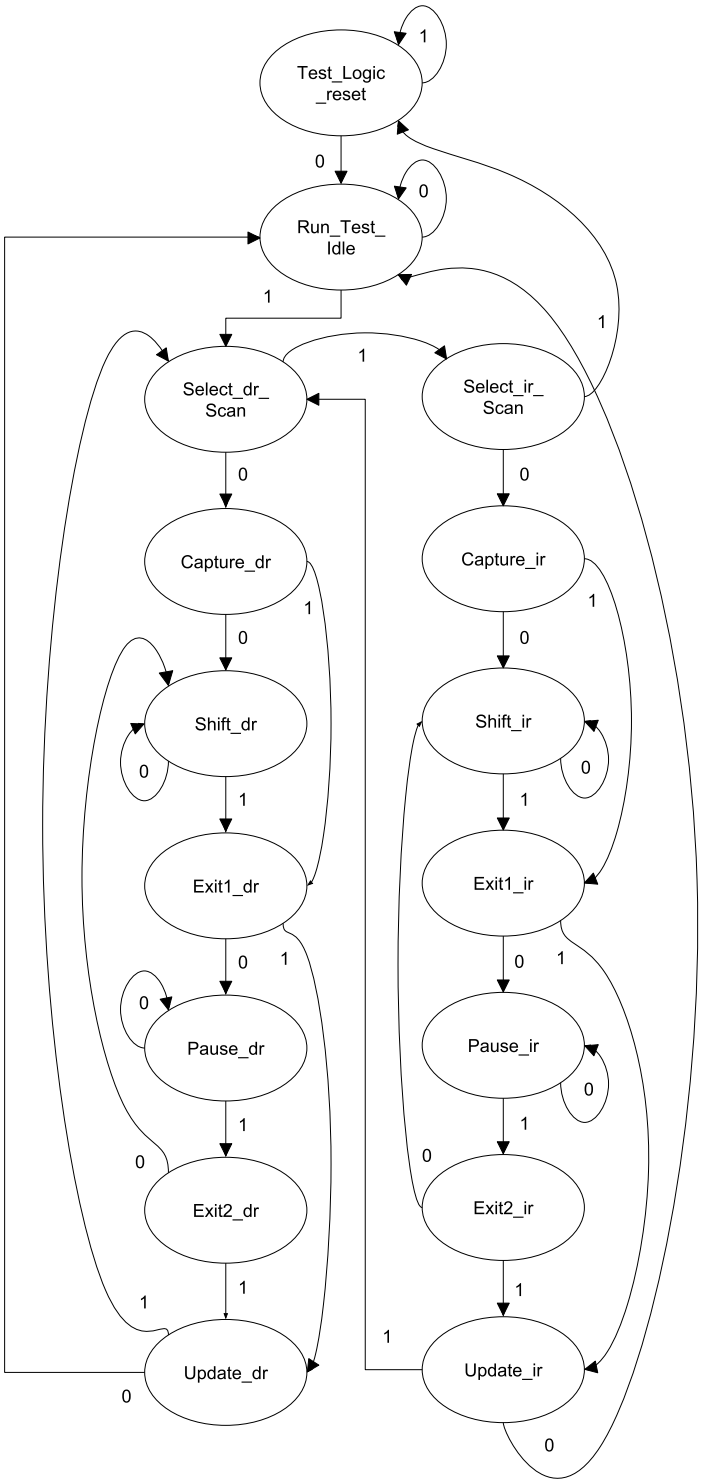

Figura 2.5: Control del TAP tomado de [1]

Test-Logic-Reset: Este estado no tiene una incidencia directa sobre la lógica de prueba o sobre el sistema, pero es el estado de reinicio del controlador del TAP.
Run-Test/Idle:En este estado ocurren las actividades seleccionadas en cualquier lógica de prueba mientras existan instrucciones que las activen, en otras palabras, las pruebas son implementadas, si no, el controlador simplemente entra en en este estado sin realizar

ninguna acción. Los registros de datos deben conservar sus valores y las instrucciones no deben cambiar.
Select-DR-Scan: Es un estado temporal de control en el cual el registro de datos retienen su estado anterior. En este estado se toma una decisión sobre si seguir hacia el estado Capture-DR o proseguir hasta el estado Select-IR-Scan.
Select-IR-Scan: Es un estado temporal de control en el cual el registro de datos retienen su estado anterior. En este estado se toma una decisión sobre si seguir hacia el estado Capture-IR o moverse hacia Test-Logic-Reset.
Capture-IR: En este estado el registro de instrucciones captura la entrada paralela, se recomienda que se cargue a la entrada paralela un patrón «01» a los bits menos significativos y en caso de que hayan más significativos pueden recibir valores específicos del diseñador. Este estado ve una transición hacia Exit1-IR o Shift-IR dependiendo del valor de TMS.
Shift-IR: En este estado el registro de instrucciones está conectado entre TDI y TDO y desplaza, en cada flanco positivo de TCK, la instrucción capturada una etapa hacia la salida. También desplaza las nuevas instrucciones desde TDI hacia el registro de instrucciones. En este estado se puede mover hacia Exit1-IR mientras TMS sea alto o mantenerse en el estado si TMS está bajo.
Exit1-IR: Este estado de control temporal que decide si continuar hacia Pause-IR o Update-IR.
Pause-IR: Este estado interrumpe y paraliza el desplazamiento de datos en el registro de instrucciones.
Exit2-IR: Este estado de control temporal que decide si continuar hacia Pause-IR o devolverse hacia Shift-IR.
Update-IR: En este estado la instrucción desplazada previamente hacia el Registro de Instrucciones es actualizada y cargada en la salida del registro por lo que pasa a ser la instrucción actual, definiendo un nuevo modo de operación. Este estado se mueve hacia el estado de Select-DR-Scan o Run-Test/Idle.
Capture-DR: En este estado el Registro de Datos captura los bits que se encuentren en la entrada paralela del Registro de Datos seleccionado durante el flanco positivo de TCK. Este estado ve una transición hacia Exit1-DR o Shift-DR dependiendo del valor de TMS.
Shift-DR: En este estado el registro de instrucciones está conectado entre TDI y TDO y desplaza, en cada flanco positivo de TCK, el patron capturado una etapa hacia la salida. También desplaza las nuevas instrucciones desde TDI hacia el registro de instrucciones. En este estado se puede desplazar hacia Exit1-DR mientras TMS sea alto o mantenerse en el estado si TMS está bajo.
Exit1-DR: Este estado de control temporal que decide si continuar hacia Pause-DR o Update-DR.
Pause-DR: Este estado interrumpe y paraliza el desplazamiento de datos en el registro de instrucciones.
Exit2-DR: Este estado de control temporal que decide si continuar hacia Pause-DR o devolverse hacia Shift-DR.
Update DR: En este estado el vector de pruebas o los vectores resultados de una prueba

desplazados, previamente hacia el Registro de Datos son actualizados y cargados en la salida del registro, capturando así un nuevo patrón de pruebas o los resultados de una prueba. Este estado se desplaza hacia el estado de Select-DR-Scan o Run-Test/Idle.

Los registros de datos e instrucciones debe seguir dos lineamientos, el primero es permitir desplazar los bit del vector de pruebas o los bit de instrucción sin que produzca un cambio en la etapa de cambie la instrucción y el segundo es mantener la instrucción o el dato introducido hasta que se desee actualizar por otro nuevo. Una implementación que cumple dichas características es la celda serie/paralelo.

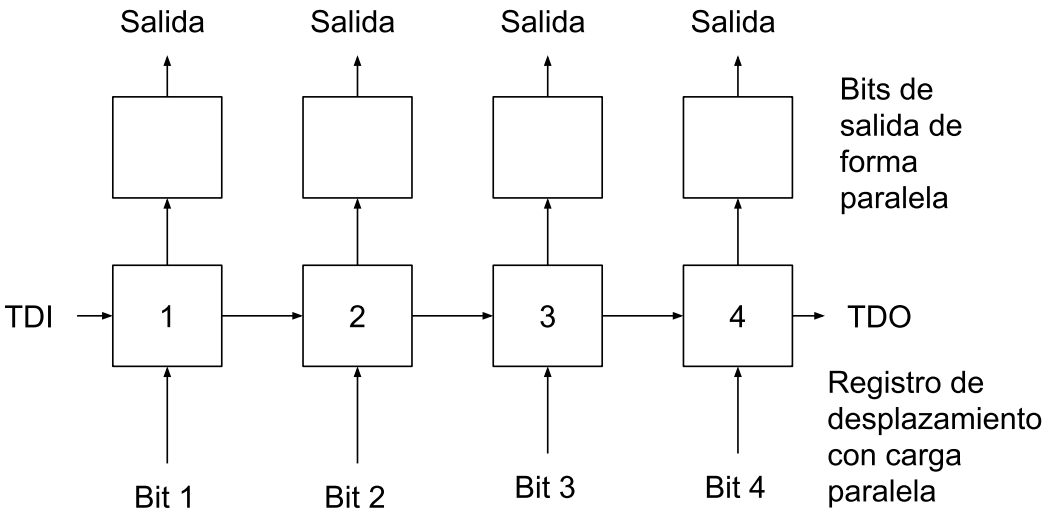

Figura 2.6: Configuración serie/paralelo

El registro de instrucciones: Es un registro serie/paralelo que contiene, al menos, dos celdas capaces de mantener las instrucciones, esto debido a que la mínima cantidad de instrucciones que el TAP debe soportar cunado se implementa son cuatro y la cantidad mínima de bits requeridos para representar cuatro elementos son dos bits 00, 01, 10 y 11. En caso de querer probar la integridad de la cadena de scan, el estándar 1149.1 provee de una solución muy sencilla, durante el estado Capture-IR los dos bits menos significativos deben capturar un «01», en otras palabras, desplazar los datos capturados por los bits menos significativos hacia el puerto de salida, TDO, con la finalidad de ver si existe algún error. Por ejemplo, un bit «11» obtenido después de desplazar los bits del registro de instrucción, cuando el resultado esperado es «01» indicaría que algún elemento de la cadena está con problemas. Otros bits de mayores ordenes de magnitud pueden capturar datos específicos dados por el diseñador.
Este paso no interrumpe el funcionamiento del registro de instrucciones, debido a que las instrucciones se cargan de manera serial utilizando el puerto TDI y se desplazan hacia las celdas en el estado de SHIFT-IR de controlador del TAP.
El estándar define una implementación sencilla para la celda de instrucciones como se ve en la figura 2.7.

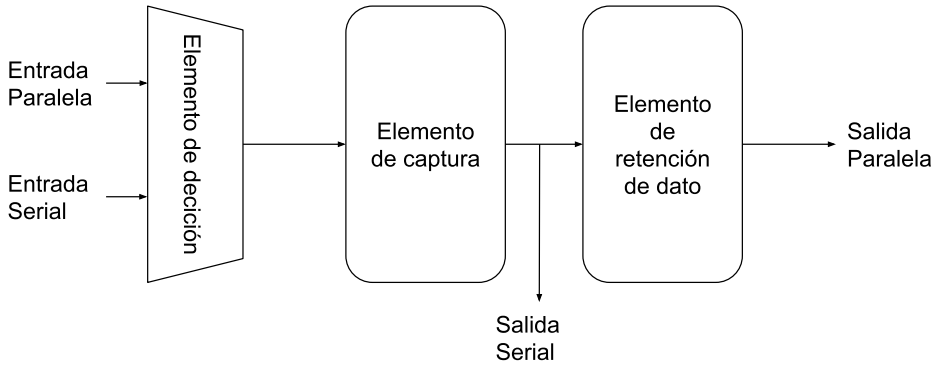

Figura 2.7: Arquitectura del registro de instrucciones

La etapa del banco de registro de instrucciones se compone del registro de instrucciones y el decodificador de dichas instrucciones visto en la figura 2.8.

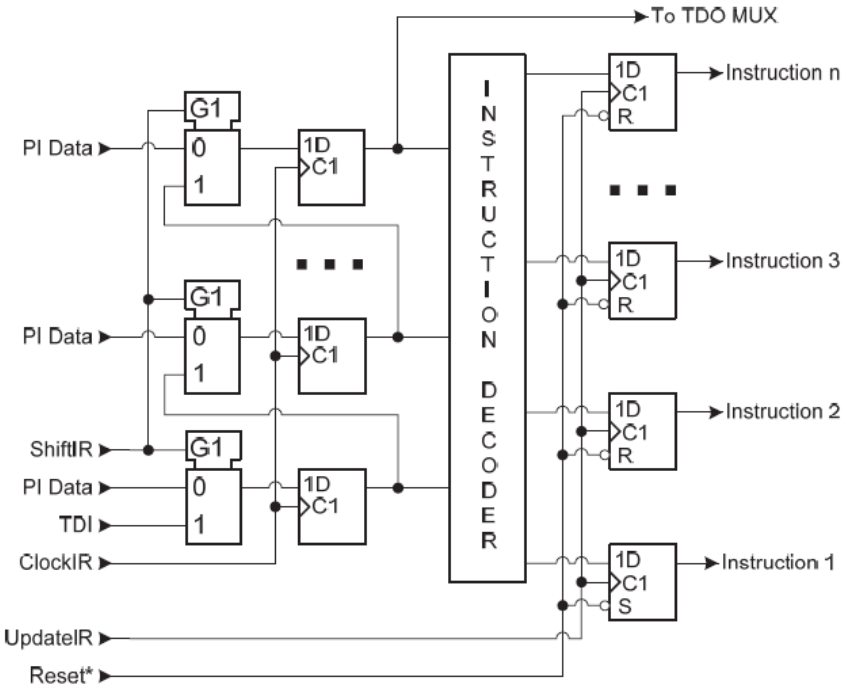

Figura 2.8: Banco de Registros de Instrucciones tomado de [1]

El decodificador determina el modo de operación del sistema dependiendo de la instrucción.

El registro de datos: Se encarga de almacenar los vectores de prueba y resultados obtenidos una vez realizadas las pruebas. El estándar incluye dos registros obligatorios: El registro de Bypass y el Registro de Boundary-Scan.

Un ejemplo del Banco de Registros de Datos se puede apreciar seguidamente.

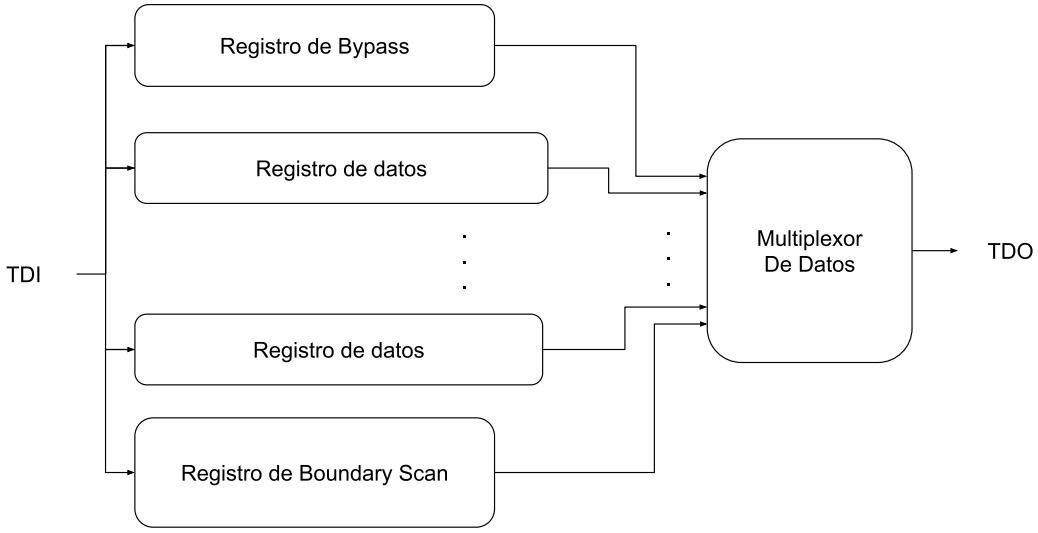

Figura 2.9: Banco de Registros de Datos tomado de [1]

El Registro de Bypass provee de la ruta mínima entre el puerto TDI y el puerto TDO, usualmente es un único bit y desconecta el DUT (Design Under Test) de la cadena de scan, la ventaja de este registro es que, el costo de desplazar los datos por el sistema cuando el registro de bypass está activo es un ciclo de la señal de reloj TCK.

El Registro de Bypass se puede implementarse como se observa en la figura 2.10.

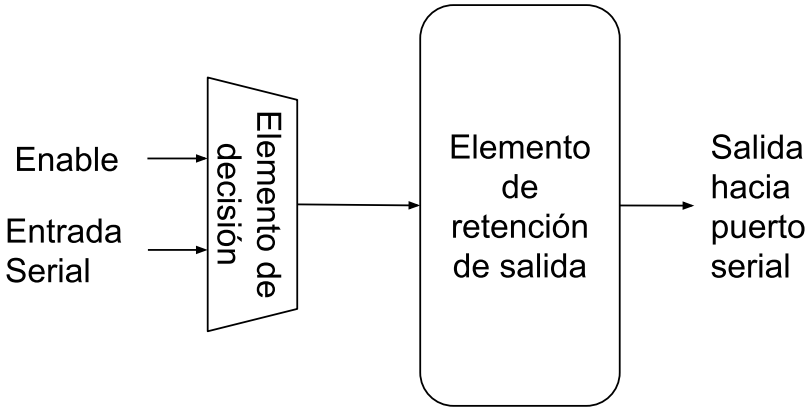

Figura 2.10: Arquitectura del Registro de Bypass

Mientras el Registro de Bypass sea seleccionado, ningún otro registro de prueba debe ser activad durante esta instrucción; tampoco debe afectar el funcionamiento de la lógica de prueba del sistema, en otras palabras ninguna celda debe de perder su estado anterior a menos que sean reiniciadas.
Estas características de operación se deben a que el estándar está en capacidad de interconectar varios circuitos integrados que incluyan Boundary Scan en cascada, como una

cadena para realizar pruebas.

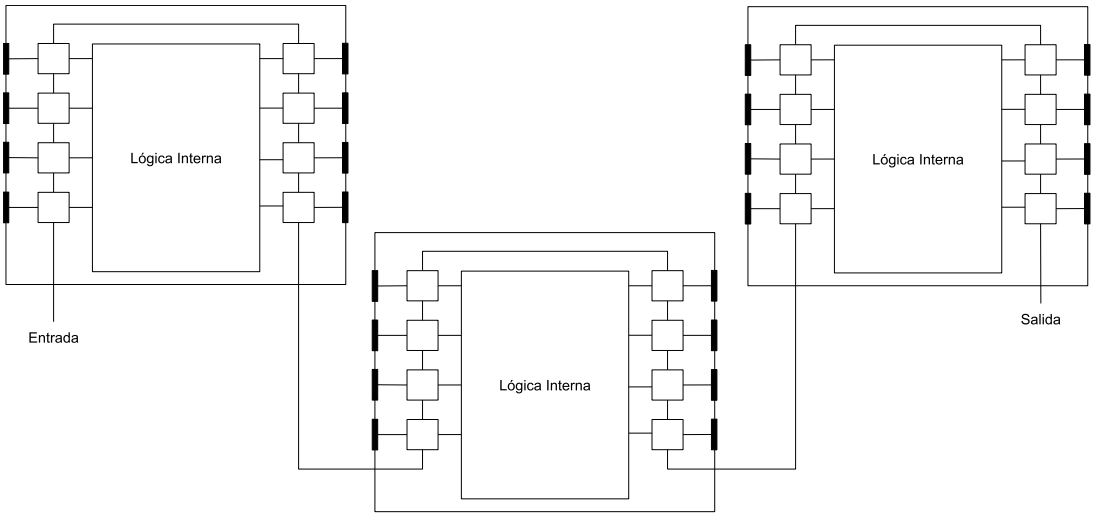

Figura 2.11: Cadena de Prueba tomado de [2]

En una cadena de pruebas, los datos y las instrucciones son enviados de forma serial por el puerto TDI hasta cada uno de los elementos deseados de la cadena, por lo que se necesita una cantidad de ciclos de reloj igual a la cantidad de celdas unitarias dentro de los registros por el cual el dato deba ser desplazado. Esta configuración se utiliza con la finalidad de realizar pruebas a nivel de placa en busca de fallas como lo son fallas de acoplamiento o si algún pin está cortocircuitado.
En uso del registro de Bypass, durante una conexión en cadena, dicho registro genera una ruta mínima para la transmisión del dato, mientras se encuentre desconectado alguno de los CI como se observa en la figura 2.12.

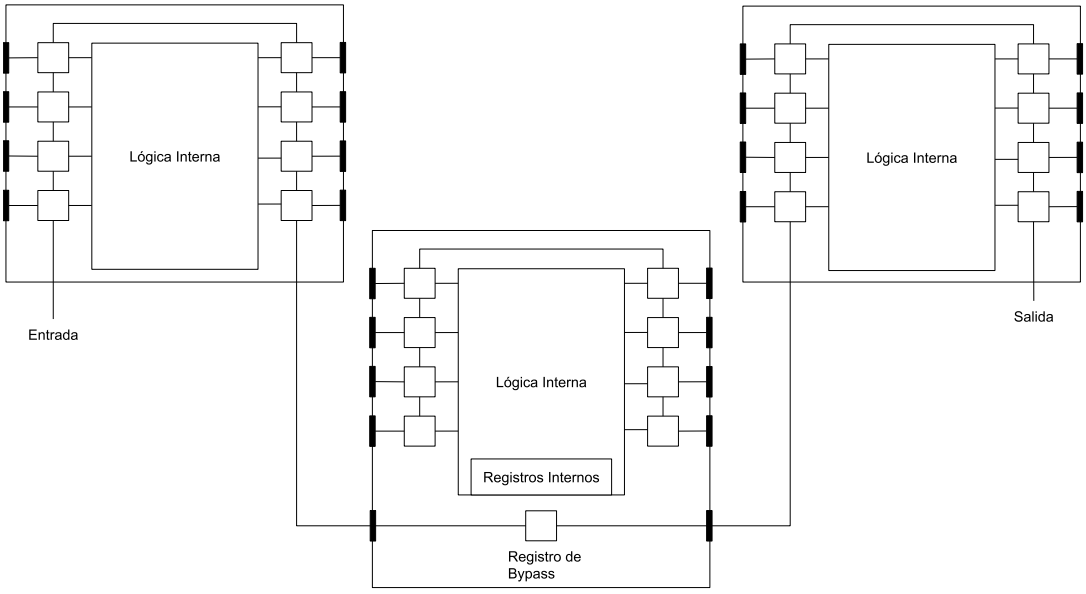

Figura 2.12: Uso del Bypass

El registro de Boundary Scan posee un celda adjunta que se encuentra entre las entradas de los elementos y los pines de entrada y salida del Chip con la finalidad de aumentar el grado de control y la capacidad de observación sobre los pines del chip. La implementación de este registro depende de cuantos pines de entrada y salida posea el sistema al cual se desea implementar Boundary Scan.

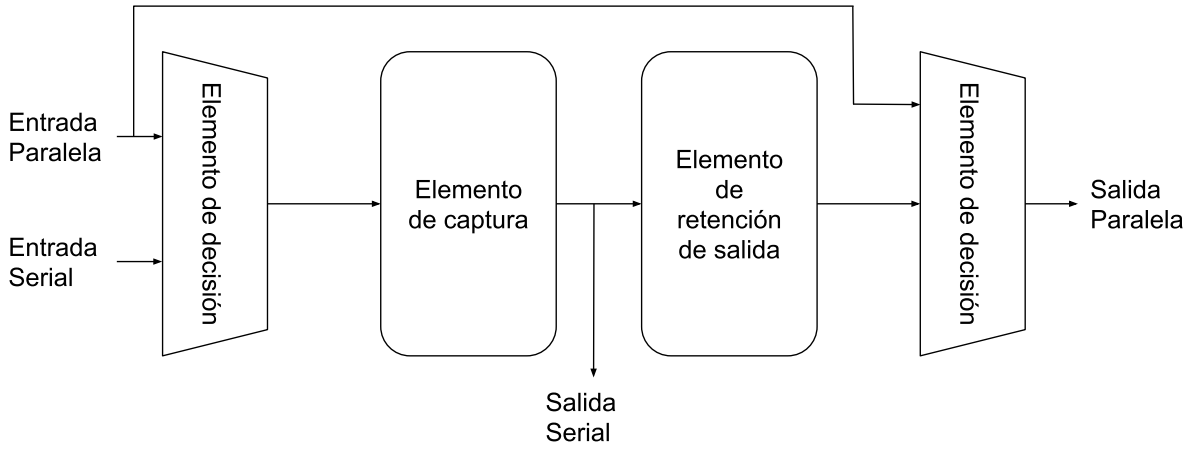

Figura 2.13: Arquitectura del Registro de Boundary Scan

En un diseño, el orden y número de los registro de Boundary Scan no debe cambiar por motivos de operación interna ya que esta celda se asocia únicamente con los pines de entrada y salida del CI.
Además cuenta con un elemento de selección de salida que determina el modo de operación del JTAG, dicho elemento permite desconectar los pines de entrada o de salida, de la lógica interna como se observa en 2.1, determinando así si se está en un modo de operación invasivo o no invasivo.

El estándar determina 4 instrucciones básicas EXTEST, SAMPLE, BYPASS y PRELOAD, además se utiliza una instrucción más INTEST.
BYPASS: Esta instrucción trabaja sobre el Registro de Bypass y tiene una asignación binaria fija, ya sea {111...1} o {000...0}.
Esta instrucción desconecta el equipo de validación de la lógica de prueba mientras está activa. Las siguientes instrucciones se centran en el registro de Boundary Scan, estas instrucciones determinan el modo de operación del JTAG, ya sea invasivo o no invasivo.
Primero se inicia con las instrucciones no invasivas.
SAMPLE: La instrucción permite capturar los datos que se encuentren de los pines de entrada y pines de salida del JTAG con la finalidad de desplazar los datos capturados hacia el puerto serial una vez realizada cada una de las pruebas. Esta instrucción no afecta el modo de operación del Registro de Boundary Scan.
PRELOAD: Al cargar esta instrucción se utiliza las celdas de captura del Registro de Boundary-Scan. Dichas celdas reciben la data de manera serial con el fin de precargar con valores determinados el Registro de Boundary Scan.

EXTEST: Una instrucción que trabaja en modo invasivo, ya que desconecta los pines de salida de la lógica interna y toma el control de los Registros de Boundary Scan asociados a los pines de salida. Esta instrucción se utiliza para realizar pruebas a nivel de placa debido a que conecta la salida del Registro de Boundary Scan con los pines de salida de CI. **INTEST**: Esta instrucción desconecta los pines de entrada y conecta la salida de los registros de Boundary Scan con la entrada de la lógica interna, por lo que se utiliza para realizar pruebas sobre la lógica interna del CI.

El funcionamiento del TAP, se puede explicar mediante el proceso de carga de instrucciones y de carga de vectores de prueba.
Para cargar instrucciones se sigue el proceso de la figura 2.14.

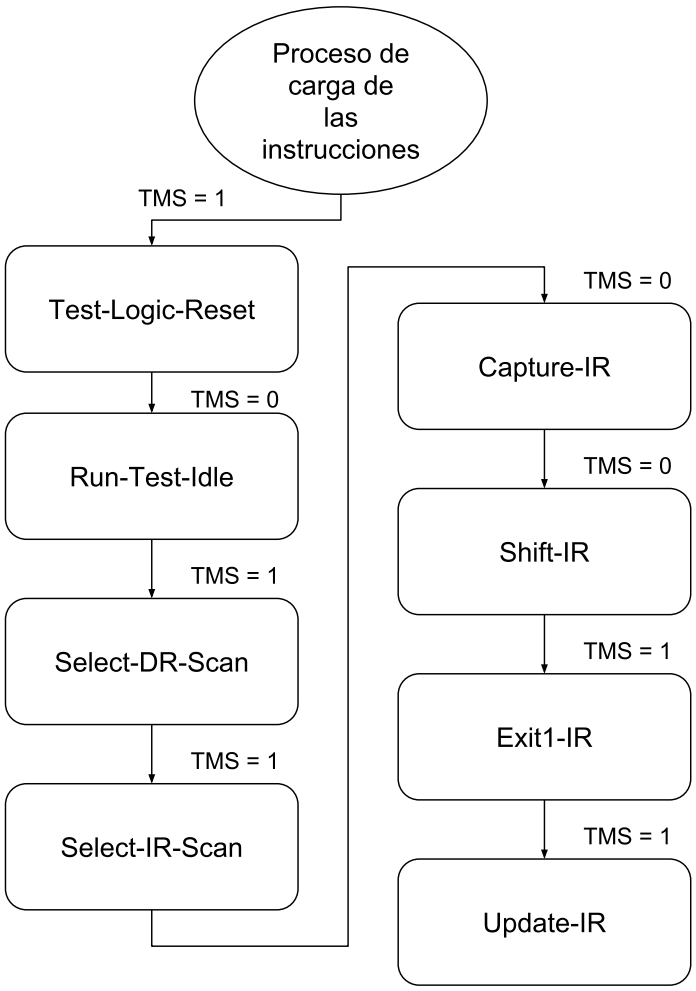

Figura 2.14: Proceso para cargar una instrucción

El proceso mostrado en la figura 2.14 sigue los siguientes pasos.
Primeramente se introduce una cadena de 5 unos, TMS = 11111, este proceso es serial y ocurre por cada ciclo de reloj de TCK, con la finalidad de volver al estado de Test_Logic_Reset.
El siguiente paso es asignarle un valor de 0 a TMS, para llevar al control al estado de Run_Test_Idle.

Tercer paso es cambiar el valor de TMS a 1 por dos ciclos de TCK, TMS = 11, con ello el controlador llega al estado de Select_IR_Scan.
Cuarto paso se cambia el valor de TMS a 0 con ello se cae al estado de Capture_IR, se mantiene por otro ciclo de reloj en 0 para llevar al control al estado Shift_IR.
El quinto paso es asignar un valor a TMS de 0 y comenzar a cargar las instrucciones en el registro, por lo que se carga el patrón binario en la entrada TDI, y se desplaza la cantidad de ciclos de reloj de TCK necesarios dependiendo del tamaño del registro de instrucciones para cargar la instrucción.
El sexto paso es poner TMS en 1 para pasar al estado de Exit1_IR.
El séptimo paso es cargar a TMS un valor de 1, con ello se leva al controlador a Update_IR, en este estado se actualizan los registros de salida de las celdas de instrucción, con lo que la instrucción es correctamente cargada en el sistema.

El proceso de carga de vectores de prueba es similar al de instrucciones como se observa en la figura 2.15.

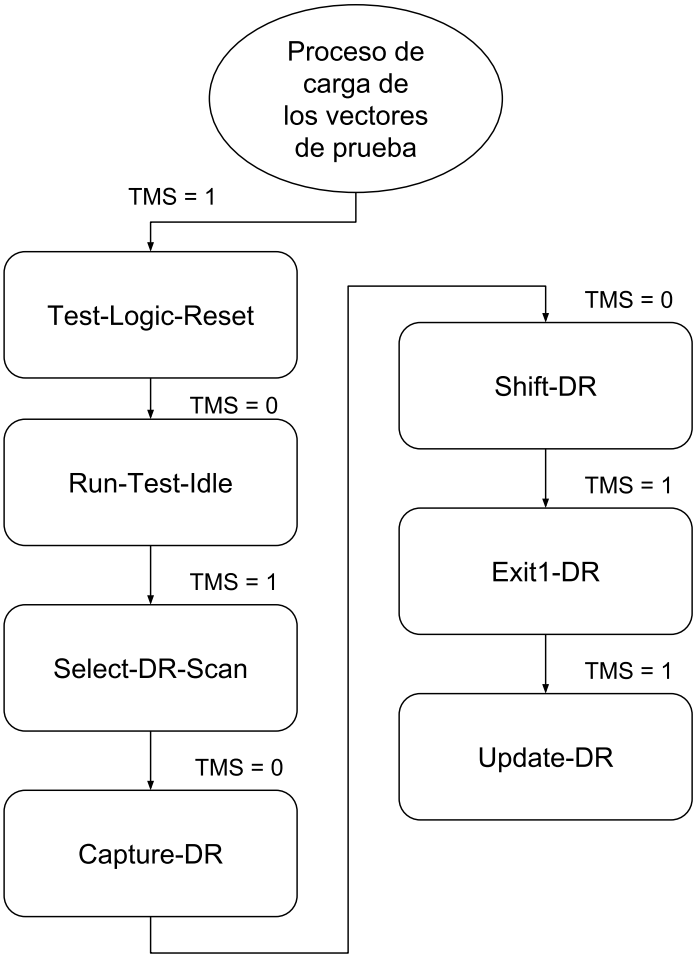

Figura 2.15: Proceso para cargar un vector de pruebas

El proceso para cargar un vector de prueba se puede dividir en 7 pasos.
Primeramente se introduce una cadena de 5 unos al pin TMS, TMS = 11111, este proceso es serial y ocurre por cada ciclo de reloj de TCK por lo que cada uno se censa durante los

flancos positivos de TCK, con la finalidad de volver al estado de Test_Logic_Reset, este proceso toma cinco flancos positivos de TCK.
El siguiente paso es asignarle un valor de 0 a TMS, para llevar al control al estado de Run_Test_Idle. Dura un ciclo de TCK.
Tercer paso es cambiar el valor de TMS a 1 por un ciclos de TCK, con ello el controlador llega al estado de Select_DR_Scan.
Cuarto paso se cambia el valor de TMS a 0 con ello se cae al estado de Capture_DR, se mantiene por otro ciclo de reloj en 0 para llevar al control al estado Shift_DR.
El quinto paso es asignar un valor a TMS de 0 y comenzar a cargar los vectores en el registro de datos asignado para dicha prueba, por lo que se carga el patrón binario en la entrada TDI, y se desplaza la cantidad de ciclos de reloj de TCK necesarios dependiendo del tamaño del registro de detos seleccionado para cargar la data.
El sexto paso es poner TMS en 1 para pasar al estado de Exit1_DR.
El séptimo paso es cargar a TMS un valor de 1, con ello se leva al controlador a Update_DR, en este estado se actualizan los registros de datos, con lo que un vector de pruebas es correctamente cargada en el sistema.

2.2. Registros

Un registro de N bits se define como una secuencia de flip-flops que comparten una misma señal de reloj de tal manera que todos los bits del registro sean actualizados en el mismo instante [3].

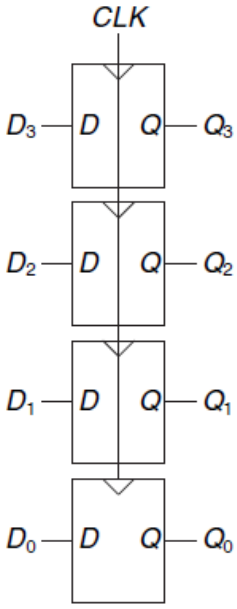

Figura 2.16: Boundary-Scan Cell de salida tomado de [3]

Los registros se pueden diseñar utilizando flops o latches, también pueden cambiar su configuración, la figura 2.16 es una configuración paralela, pero es posible conectar en cascada los registros, esta configuración es conocida como registro de desplazamiento, un registro de desplazamiento sería algo similar a lo visto en 2.4, donde dos elementos secuenciales se conectan en serie desplazando el dato conforme hay un cambio en el ciclo de reloj.
Un diseño para un flip flop se presenta en la figura 2.17.

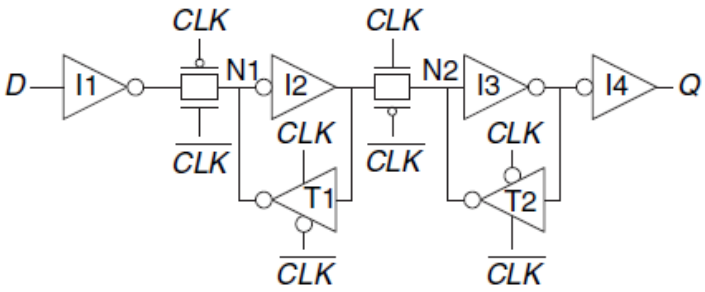

Figura 2.17: Estructura de un Flop tomado de [9]

Por su parte el diseño de un latch se presenta en la figura 2.18.

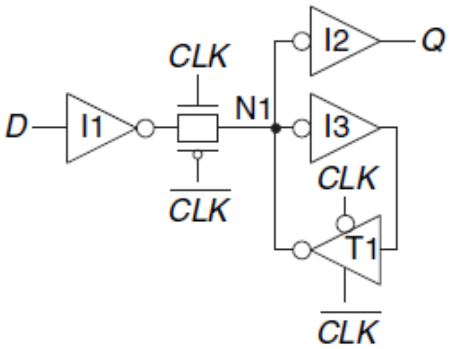

Figura 2.18: Estructura de un Latch tomado de [9]

Al comparar las figuras 2.17 y la figura 2.18 se observa que el flop es, aproximadamente, 50 % más grande que el latch. Debido a esto la implementación de los registros con latches o flops debería variar, en términos de consumo de área, en una proporción similar.
Otra diferencia radica en el comportamiento ante una señal de reloj, mientras que el latch es trasparente cuando el reloj está en alto y cerrado cuando el reloj está en bajo, el flop captura los datos durante los flancos de reloj, ya sean positivos o negativos dependiendo del diseño del flop y se mantiene cerrado en cualquier otro instante.

Una manera de generar que el latch capture en un flanco de una señal de reloj, similar al comportamiento de un flop, es utilizar un latch Pulsado.
El latch pulsado se compone de un latch y un circuito detector de flancos que genera un pulso cuando detecta un flanco positivo o negativo de la señal de reloj como se observa en la figura 2.20.

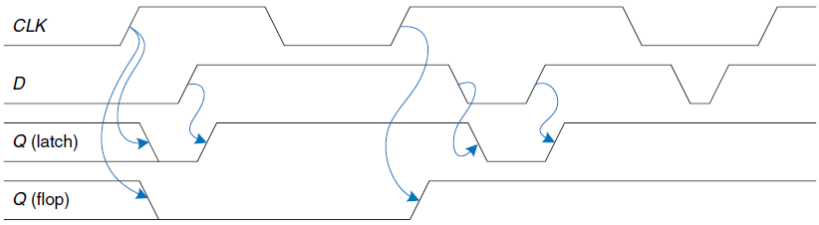

Figura 2.19: Diagrama de tiempos tomado de [9]

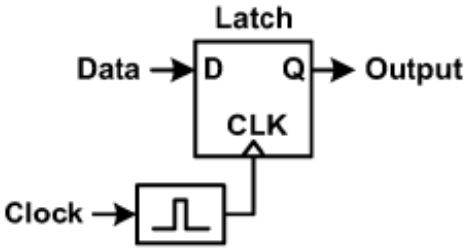

Figura 2.20: Latch Pulsado tomado de [10]

La finalidad de desarrollar los registros del TAP utilizando latches pulsados, es ahorrar área como se menciona en [10], otra razón es el ahorro de consumo energético.
Para que la cadena de latches pulsados se comporte como un registro de desplazamiento, como se observa en la figura 2.2, es necesario que los pulsos sean generados de manera ordenada y secuencial como se muestra en la figura 2.21.

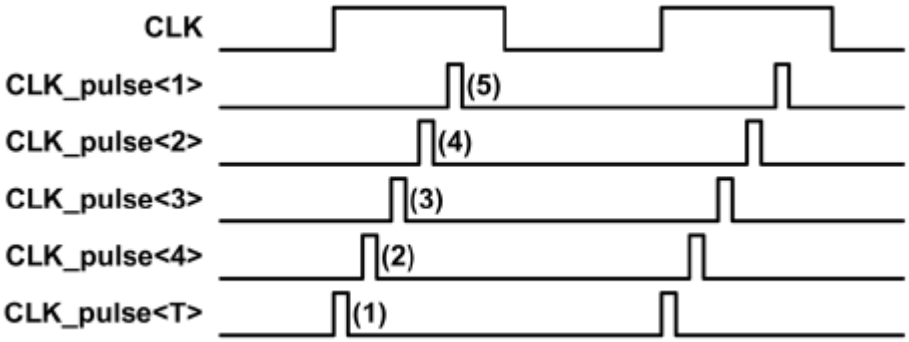

Figura 2.21: Habilitador de reloj en cascada tomado de [10]

Por lo que la principal diferencia entre un registro de desplazamiento implementado con flops y otro con latches pulsados radica en la implementación de un circuito generador de pulsos reloj para que los latches capturen en los flancos positivos o negativos de la señal de reloj.

Cuando hablamos de validación, tenemos que tomar en cuenta los registros de scan, un registro de scan es un registro que se utiliza para realizar procesos de validación, son celdas con una funcionalidad específica. Por ejemplo los registros de desplazamiento se utilizan para extraer datos de un punto de referencia específico, por lo que se consideran celdas

que aumentan la capacidad de observar elementos dentro de un chip. También existen otros tipos de celdas en este caso nos enfocaremos en dos celdas básica.
La primera es la celda D multiplexada vista en la figura 2.22.

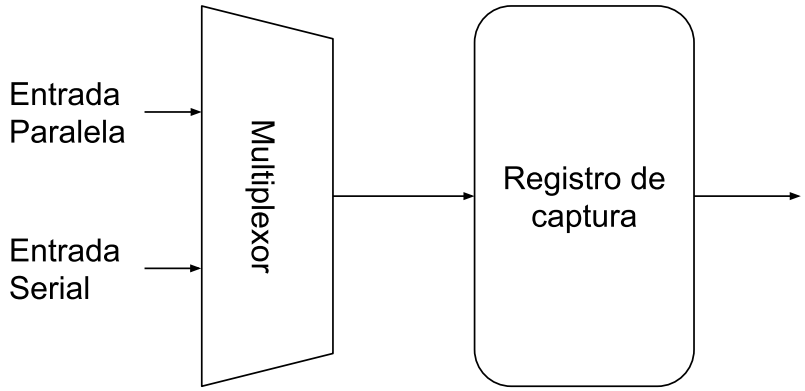

Figura 2.22: Celda de Scan D Multiplexada

Esta celda se compone de un multiplexor en la entrada, este multiplexor le otorga la capacidad de cargar datos de manera serial o de manera paralela dependiendo de cual entrada sea seleccionada.

La siguiente celda es la celda envolvente y se observa en la figura 2.23, dicha celda captura un dato serial y paralelo, pero también tiene un modo de operación donde la entrada paralela evade la lógica de captura de la celda.

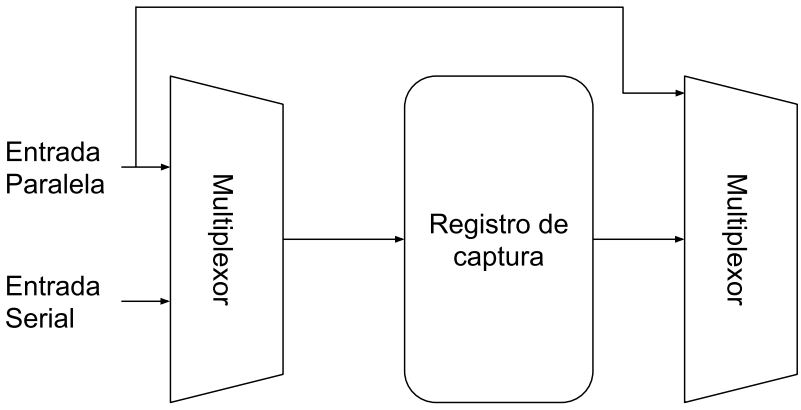

Figura 2.23: Celda de Scan Envolvente

2.3. Consumo de energía

La energía consumida por un circuito CMOS se puede explicar mediante el consumo dinámico y el consumo estático.

$$P_{total} = P_{conmutacion} + P_{corto\_circuito} + P_{fuga} \tag{2.1}$$

La potencia de conmutación es el consumo energético que ocurre cuando la capacitancia de compuerta o del cableado se carga o descarga, la potencia de conmutación está dada por.

$$P_{conmutacion} = \alpha \cdot f \cdot C_{eff} \cdot V_{dd}^2 \tag{2.2}$$

Donde α es el factor de actividad de la compuerta, f es la frecuencia de conmutación, C_{eff} es la capacitancia efectiva y V_{dd}^2 es el voltaje de alimentación al cuadrado.
La potencia de corto circuito ocurre cuando un transistor está realizando una transición de región de operación, durante la conmutación de corte a saturación hay un instante en el cual entra en la región lineal, en este momento existe un camino entre v_{dd} y tierra (Gnd), aunque el instante es corto, el consumo energético es muy ato, de hecho en un circuito CMOS, el mayor consumo energético ocurre debido a la potencia de corto circuito.

$$P_{corto\_circuito} = I_{cc} \cdot V_{dd} \cdot f \tag{2.3}$$

Donde I_{cc} es la corriente de corto circuito durante la conmutación, V_{dd} es la fuente de alimentación y f es la frecuencia con la cual conmuta la compuerta.
La potencia de fuga es una función que depende del tamaño del transistor, pero el efecto es usualmente despreciable cuando se tiene una baja densidad de transistores, que es el caso de la implementación realizada en este proyecto, por lo que no se toma en cuenta para reportar el consumo energético.

$$P_{fuga} = f(V_{dd},V_{th},WL) \tag{2.4}$$

En general la potencia de fuga depende del voltaje de la fuente V_{dd}, del voltaje de subumbral V_{th} y del tamaño del transistor W/L.

Capítulo 3

Diseño de un JTAG con base en el estándar IEE 1149.1

3.1. Implementación general

Un diseño con base en el estándar 1149.1 de la IEE, se detalla en la figura 3.1.

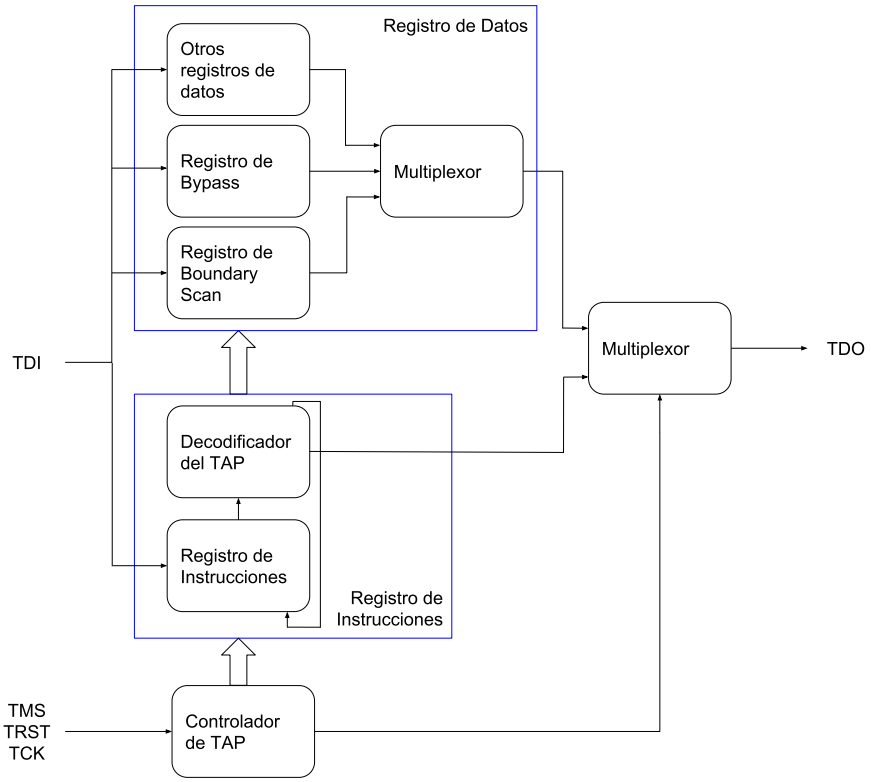

Figura 3.1: Arquitectura básica del TAP

El diseño se compone de 4 bloques:
Controlador del TAP: Controla el flujo de datos entre TDI y TDO, además de generar las señales de control del sistema.
Registro de datos: Este bloque se encarga de transferir datos desde TDI hasta TDO, permite capturar y desplazar los datos capturados de manera serial.
Registro de Instrucciones: Permite guardar instrucciones provenientes de TDI y desplazarlas de manera serial hacia TDO, además las instrucciones controlan el modo de operación del registro de Datos.
Multiplexor de salida: Permite decidir si la información a desplazar son instrucciones o datos provenientes de los respectivos registros.
Cada uno de los bloques tiene una función específica, por lo que se detallan utilizando un diagrama de bloques por módulo.
Los registros vistos en la figura 3.1, se desarrollan implementado flops [4] y latches [7].

Registro de Instrucciones El registro de instrucciones se divide en dos partes, el registro de instrucciones y el respectivo decodificador. Este módulo controla el modo de operación dependiendo de la instrucción que se cargue en el banco de registros que guardan los bits de la instrucción, el decodificador tiene asignada al menos una instrucción por cada una de las cadenas de scan conectadas en el registro de datos como se observa en la figura 3.2.

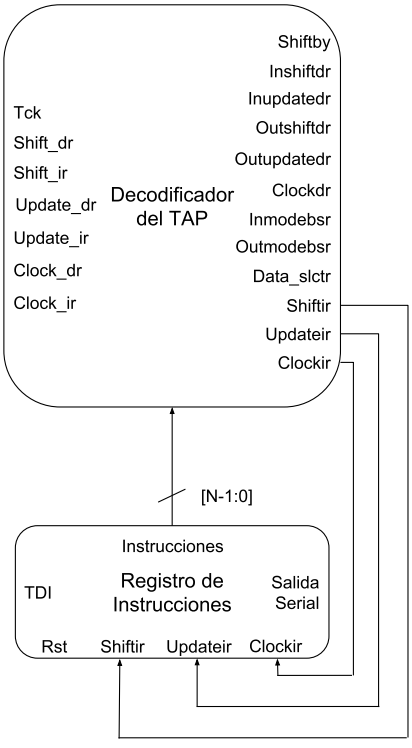

Figura 3.2: Registro de Instrucciones compuesto por el Registro de Instrucciones y el Decodificador

El Registro de Instrucciones permite guardar instrucciones provenientes de TDI y des-

plazarlas de manera serial hacia TDO, además las instrucciones controlan el modo de operación del registro de Datos.
Las entradas son:

- TDI: Entrada serial del TAP.
- Pines de entrada: Entrada proveniente de una carga paralela, recomendado por motivos de integridad de señal por el estándar
- shiftir: Señal que selecciona el dato a capturar.
- clockir: Señal que captura el dato en el registro de captura.
- updateir: Señal que actualiza el registro de salida.
- rst: Señal de reset.

Tiene como salidas las señales:

- Salida serial: Salida serial de la celda, puede estar conectada en cascada con otras celdas o directamente al mux de selección de salida.
- Salida de carga paralela o Instrucciones: Salida paralela del registro de instrucciones la cual actualiza los bits de la instrucción en los registros de salida.

Por su parte el Decodificador de Instrucciones decodifica las instrucciones y las señales de control y las redirige hacia los módulos correspondientes.
Recibe a las señales de entrada:

- Instrucciones: Son las instrucciones provenientes del registro de instrucciones que dictaminan como debe decodificarse las señales de control.
- Update_ir: Esta señal habilita al registro de salida del registro de instrucciones para capturar el dato.
- Clock_ir: Esta señal es un reloj que se permite desplazar los datos serialmente en los registros de captura del registro de instrucciones.
- Update_dr: Esta señal habilita al registro de salida de los registros de datos para capturar el dato.
- Clock_dr: Esta señal es un reloj que se permite desplazar los datos serialmente en los registros de captura de los registros de datos.
- Shift_ir: Señal de control que determina si el dato a capturar por el registro de instrucciones proviene de la entrada serial o entrada paralela.

- Shift_dr: Señal de control que determina si el dato a capturar por el registro de datos proviene de la entrada serial o entrada paralela.

- TCK: Es la señal de reloj del TAP.

Las señales de salida del bloque son:

- Updateir: Esta señal habilita al registro de salida del registro de instrucciones para capturar el dato.

- Clockir: Esta señal es un reloj que se permite desplazar los datos serialmente en los registros de captura del registro de instrucciones.

- Inupdatedr: Esta señal habilita al registro de salida de los registros de datos para capturar el dato de la celda de Boundary Scan de entrada.

- Outupdatedr: Esta señal habilita al registro de salida de los registros de datos para capturar el dato de la celda de Boundary Scan de salida.

- Clockdr: Esta señal es un reloj que se permite desplazar los datos serialmente en los registros de captura de los registros de datos.

- Shiftir: Señal de control que determina si el dato a capturar por el registro de instrucciones proviene de la entrada serial o entrada paralela.

- Inshiftdr: Señal de control que determina si el dato a capturar por el registro de datos proviene de la entrada serial o entrada paralela para la celda de Boundary Scan de entrada.

- Outshiftdr: Señal de control que determina si el dato a capturar por el registro de datos proviene de la entrada serial o entrada paralela para la celda de Boundary Scan de salida.

- Data_Slctr: Señal de selección del mux de datos, se diseñó e implementó de forma «One-Hot».

- Out_mode_dr: Señal que controla el modo de operación de las celdas de Boundary-Scan a la salida del circuito.

- In_mode_dr: Señal que controla el modo de operación de las celdas de Boundary-Scan a la entrada del circuito.

- Shift_by: Señal de habilitación para el registro de Bypass.

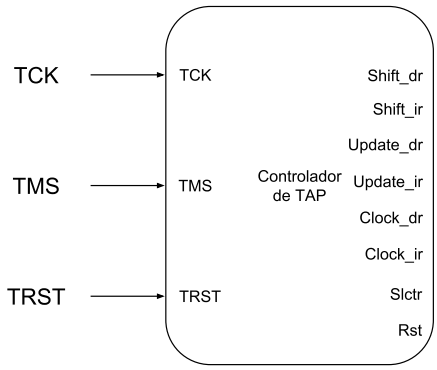

Figura 3.3: Controlador del TAP

Controlador del TAP Se compone del control del TAP y tiene como función controlar el flujo de datos entre TDI y TDO. La figura 3.3 muestra la implementación del controlador.

Las señales de entrada son:

- TMS: Esta es la señal de control de la máquina de estados del controlador del tap.
- TCK; Es la señal de reloj del TAP.
- TRST: El estándar no incluye esta señal de entrada pero está permitida, es una señal que permite establecer una condición de reinicio a la máquina de estados.

Las señales de salida del bloque son:

- Update_ir: Esta señal habilita al registro de salida del registro de instrucciones para capturar el dato.
- Clock_ir: Esta señal es un reloj que se permite desplazar los datos serialmente en los registros de captura del registro de instrucciones.
- Update_dr: Esta señal habilita al registro de salida de los registros de datos para capturar el dato.
- Clock_dr: Esta señal es un reloj que se permite desplazar los datos serialmente en los registros de captura de los registros de datos.
- Shift_ir: Señal de control que determina si el dato a capturar por el registro de instrucciones proviene de la entrada serial o entrada paralela.
- Shift_dr: Señal de control que determina si el dato a capturar por el registro de datos proviene de la entrada serial o entrada paralela
- Selector: Esta señal controla cual entrada selecciona el Multiplexor de salida del TAP.

- Rst: Señal de reinicio para los registros del sistema.

Registro de Datos El registro de datos está compuesto por dos registros obligatorios, el registro de Bypass, el registro de Boundary-Scan, un elemento de selección para los registros de datos como lo es el Multiplexor de Datos y cualquier otra cadena de scan a implementarse, estas cadenas pueden ser registros opcionales implementados por el estándar, como lo es el registro de ID, que contiene un código de identificación para el chip que se va a validar así como cualquier otro tipo de técnica que implemente cadenas de scan compatibles con el TAP, estas técnicas pueden ser, desde registros de «shadow» hasta técnicas para validar elementos específicos dentro de la lógica interna del chip como el Built-In Self Test dicha etapa se observa en la figura 3.4.

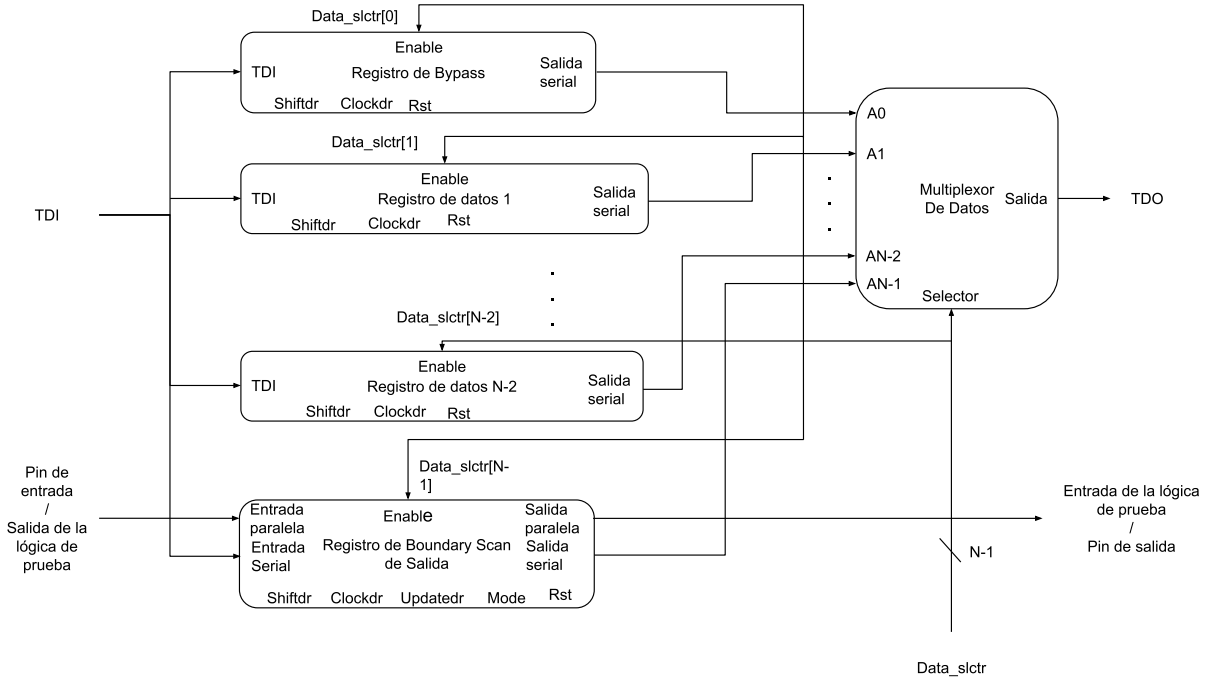

Figura 3.4: Registro de Datos

Para la implementación inicial se considera únicamente el registro de Bypass y el registro de Bpundary Scan, pero para realizar preubas de área y consumo de energía se agregan otras cadenas de scan.

Registro de Boundary Scan El Registro de Boundary-Scan posee un celda adjunta que se encuentra entre las entradas de los elementos y los pines de entrada y salida del Chip.Una característica a destacar que posee este registro es que permite observar y controlar actividades de los pines de entrada y salida del CI (Circuito Integrado).
Tiene como entradas:

- TDI: Entrada serial del TAP.

- Pines de entrada: Entrada proveniente de la placa de desarrollo, si son los registros de entrada.
- Salida de la lógica interna: Entrada proveniente de las salidas de la lógica bajo prueba, si son los registros de salida.
- mode: Señal que determina el modo de operación de la celda.
- shiftdr: Señal que selecciona el dato a capturar.
- clockdr: Señal que captura el dato en el registro de captura.
- updatedr: Señal que actualiza el registro de salida.
- Enable: esta señal permite habilitar el funcionamiento del Registro de Boundary Scan, proviene de la señal Data_slctr.
- rst: Señal de reset.

Tiene como salidas las señales:

- Salida serial: Salida serial de la celda, puede estar conectada en cadena con otras celdas o directamente al mux de selección de datos.
- Salida Paralela: Salida hacia la entrada de la lógica bajo prueba o a la salida de la placa.

El registro de Boundary Scan es un banco de registros que envuelve a la lógica interna del chip, debido a que se ubica entre las entradas y salidas de la lógica o los pines de entrada y salida de la placa donde se monta el circuito integrado como se observa en la figura 2.1.

Registro de Bypass El Registro de Bypass provee de la ruta mínima entre el puerto TDI y el puerto TDO, usualmente es un único bit y permite desconectar el chip en caso de no utilizar ninguna función del TAP.
Tiene como entradas:

- TDI: Entrada serial del TAP.
- shiftdr: Señal que selecciona el dato a capturar, en el caso de esta celda es una señal de habilitación.
- clockdr: Señal que captura el dato en el registro de captura, desplaza serialmente los datos.
- Enable: esta señal permite habilitar el funcionamiento del Registro de Bypass, proviene de la señal Data_slctr.

- rst: Señal de reset.

Tiene como salidas las señales:

- Salida serial: Salida serial de la celda, puede estar conectada en cadena con otras celdas o directamente al mux de selección de datos.

Multiplexor de Datos El multiplexor de datos permite decidir si la información a desplazar son datos provenientes de el Registro de Bypass, el Registro de Boundary-Scan o cualquier otro registro de datos.
Tiene como entradas:

- A_0 hasta A_N: Entrada correspondiente a cada una de las cadenas de scan implementads.
- Data_slctr: Señal que selecciona cual de los registros de datos va a ser multiplexado, corresponde a un bit asociado por cada una de las cadenas implementadas.

Tiene como salidas las señales:

- Salida: Señal de salida del multiplexor, se conecta con el multiplexor de salida.

Multiplexor Permite decidir si la información a desplazar son instrucciones o datos provenientes de los respectivos registros de datos como se observa en la figura 3.5.

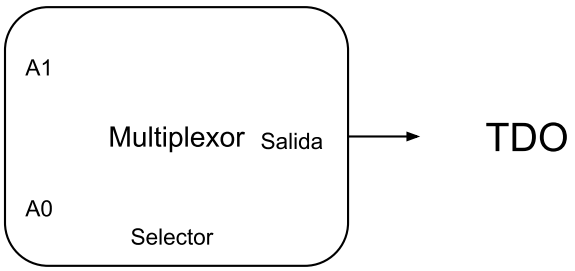

Figura 3.5: Multiplexor de salida

Tiene como entradas:

- A_0 o A_1: Señal proveniente del registro de datos o del registro de instrucciones.
- Selector: Señal que selecciona cual de los registros va a ser multiplexado.

Tiene como salidas las señales:

- Salida: Salida del multiplexor, se comunica con TDO, el puerto de salida serial del estándar TAP. Puede estar conectada en cadena con otras celdas o directamente al al dispositivo de análisis de datos.

Para la implementación utilizando latches pulsados los bloques mantienen la misma labor y las mismas señal, pero aparecen cambios en el bloque de decodificación y el controlador además de introducir un bloque de generación de pulsos, el cambio principal ocurre en el decodificador ya que se elimina la señal de Clock_ir como se observa en la figura 3.6.

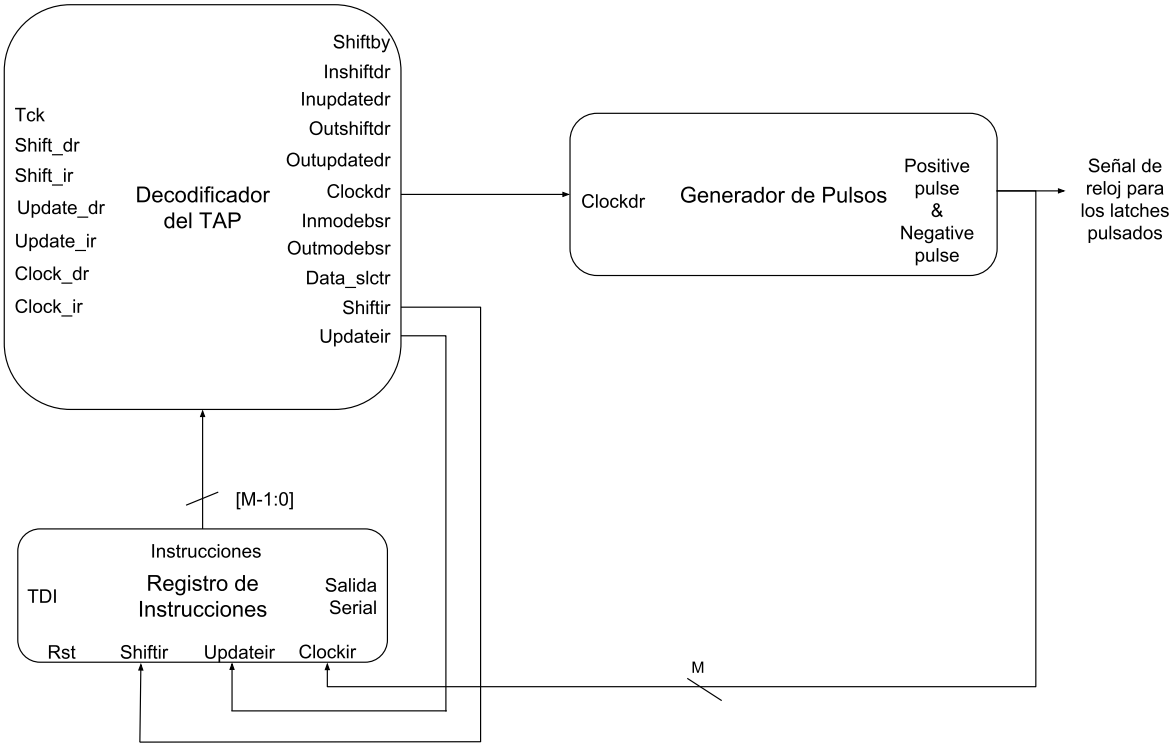

Figura 3.6: Registro de Instrucciones arquitectura de latch pulsado

El Generador de pulsos o circuito de reloj genera los pulsos necesarios para que los registros seriales implementados con latches pulsados realicen los desplazamientos, siguiendo la figura 2.21.
Recibe a las señales de entrada:

- Clockdr: Esta señal es un reloj que se permite desplazar los datos serialmente en los registros de captura de los registros de datos.

Las señales de salida del bloque son:

- Pulso Positivo: Es la señal de pulso que se genera durante un flanco positivo, del total del tamaño del bus este siempre es un único bit.

- Pulso Negativo: Esta señal se genera durante un flanco negativo, este bus tiene un tamaño igual a la cantidad de bits de la cadena de scan más larga implementada en el TAP.

Los diseños deben cumplir con una restricción de operación durante la operación de la señal TCK.
Esta restricción indica que todo cambio de estado o captura de datos en los registros debe ocurrir durante el flanco positivo de la señal TCK,y todo cambio en la operación o actualización en las salidas de los registros debe ocurrir durante el flanco negativo, esto para evitar errores de operación debido a cambios de estado durante una operación determinada.

3.2. Diseño de los módulos

3.2.1. Controlador del TAP

Para implementar dicho control se utiliza una máquina de estados, se utiliza una implementación de una máquina de estados de Moore como se observa en 3.7.

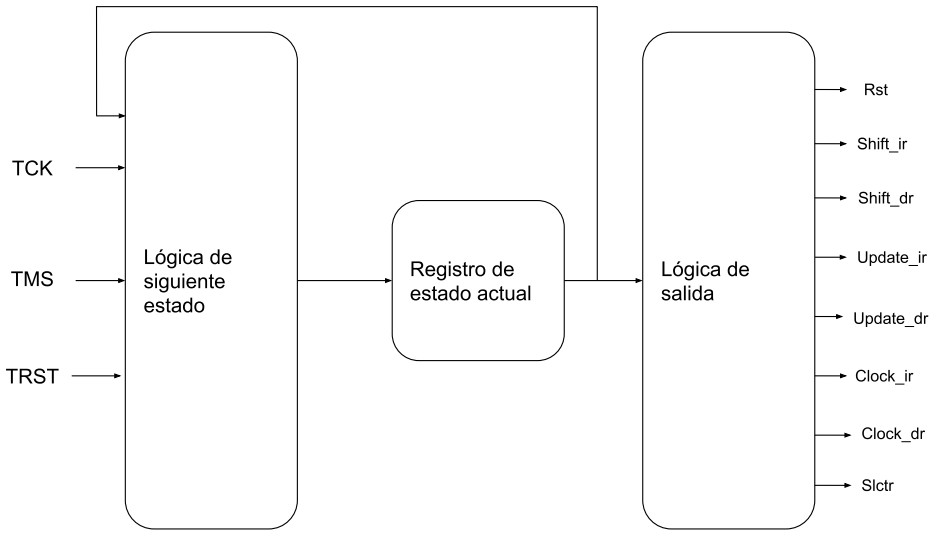

Figura 3.7: Máquina de estados del controlador del TAP

Dicha máquina tiene la característica de que las sus señales de salida dependen únicamente del estado en que se encuentra, por lo que las señales únicamente determinan el siguiente estado que la máquina de estados va a adquirir, este comportamiento es deseable debido a que una afectación en la señal de entrada no va generar un fallo en alguna de las señales de salida y afectar el funcionamiento de la máquina de estados.

El controlador del TAP funciona bajo 16 estados definidos por el estándar JTAG. La figura 3.8 menciona la distribución de los estados.

Una característica de la implementación de la máquina de estados es que no importa el estado inicial de la máquina o el estado actual, si es sometida a una secuencia de cinco unos a su entrada, siempre vuelve al estado de Test_logic_reset, el cual es un estado de reset del sistema.

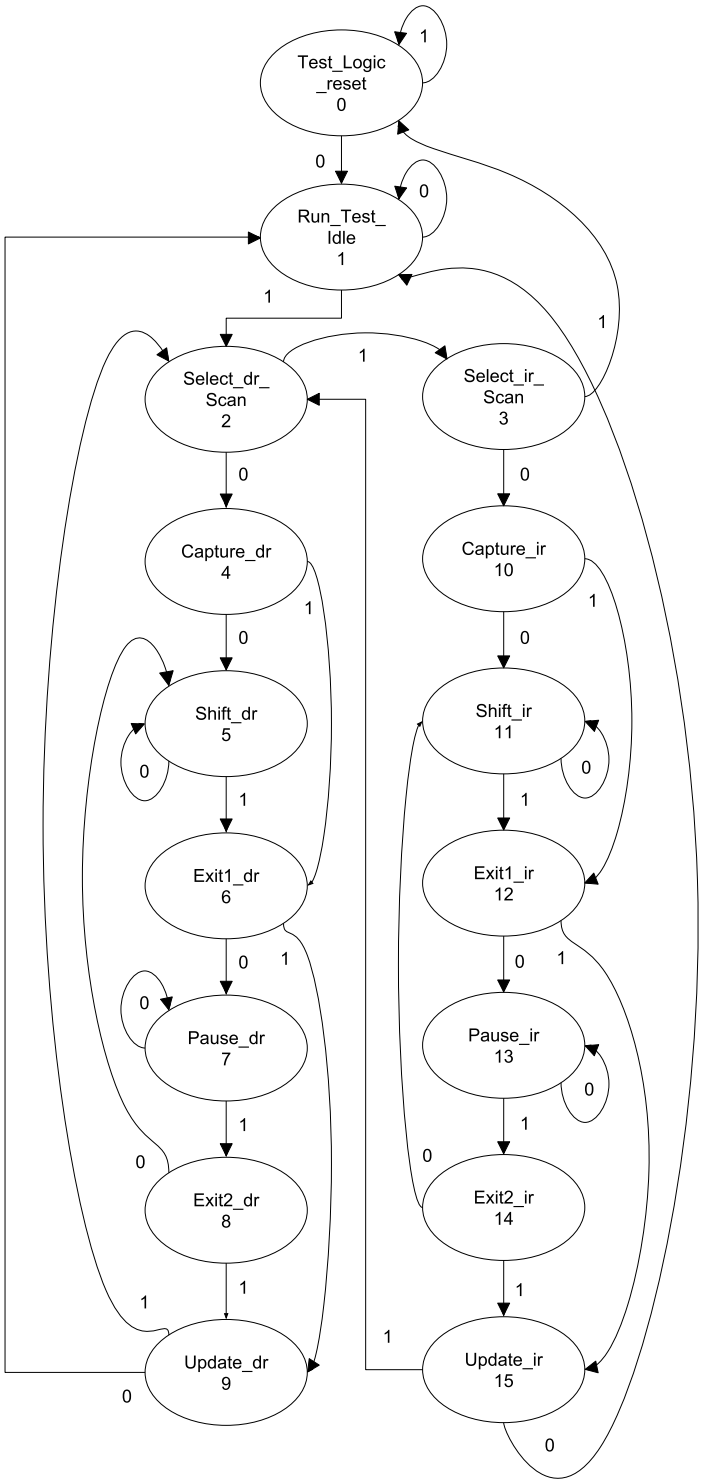

Figura 3.8: Diagrama de estados del controlador del TAP

Las señales que debe generar la máquina de estados durante cada estado se presentan a continuación para ambas topologías.
Para la implementación realizada con Flip Flops se obtiene:

Estado	Salidas
Test_Logic_Reset	rst = 1'b1
Select_DR_Scan	slctr = 1'b1
Capture_DR	slctr = 1'b1, clock_dr = 1'b0
Shift_DR	slctr = 1'b1, clock_dr = 1'b0, shift_dr = 1'b0
Exit1_DR	slctr = 1'b1
Pause_DR	slctr = 1'b1
Exit2_DR	slctr = 1'b1
Update_DR	slctr = 1'b1, update_dr = 1'b1
Capture_IR	clock_ir = 1'b0
Shift_IR	clock_ir = 1'b0, shift_ir = 1'b0
Update_IR	update_ir = 1'b1

Tabla 3.1: Salidas durante los estados para la implementación con Flip FLop

Para la toplogía de Latches Pulsados se debe seguir la siguiente tabla:

Estado	Salidas
Test_Logic_Reset	rst = 1'b1
Select_DR_Scan	slctr = 1'b1
Capture_DR	slctr = 1'b1, clock_dr = 1'b0
Shift_DR	slctr = 1'b1, clock_dr = 1'b0, shift_dr = 1'b0
Exit1_DR	slctr = 1'b1
Pause_DR	slctr = 1'b1
Exit2_DR	slctr = 1'b1
Update_DR	slctr = 1'b1, update_dr = 1'b1
Capture_IR	clock_dr = 1'b0
Shift_IR	clock_dr = 1'b0, shift_ir = 1'b0
Update_IR	update_ir = 1'b1

Tabla 3.2: Salidas durante los estados para la implementación con Latches Pulsados

Para la implementación con flops, se genera una señal de reloj para los registros de datos y otra para el registro de instrucciones como se observa en la tabla 3.1, pero debido al circuito generador de reloj, para la implementación de latches pulsados solo es requerida una señal de reloj como se observa en la tabla 3.2.

3.2.2. Registro de instrucciones

El registro de instrucciones es un registro serie/paralelo, que como ya se observa en la figura 3.2, consta de una etapa de captura y otra de carga, por lo que una implementación sencilla se plantea en la figura 3.9

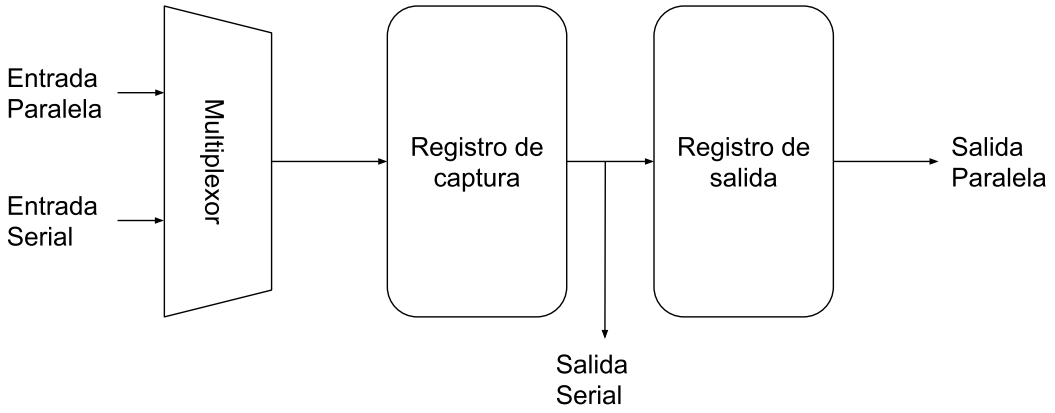

Figura 3.9: Bit para el Registro de Instrucciones

Implementación con Flip Flop

Para la implementación con Flip Flop, se utiliza una topología con Flip Flop D vista en la figura 3.10.

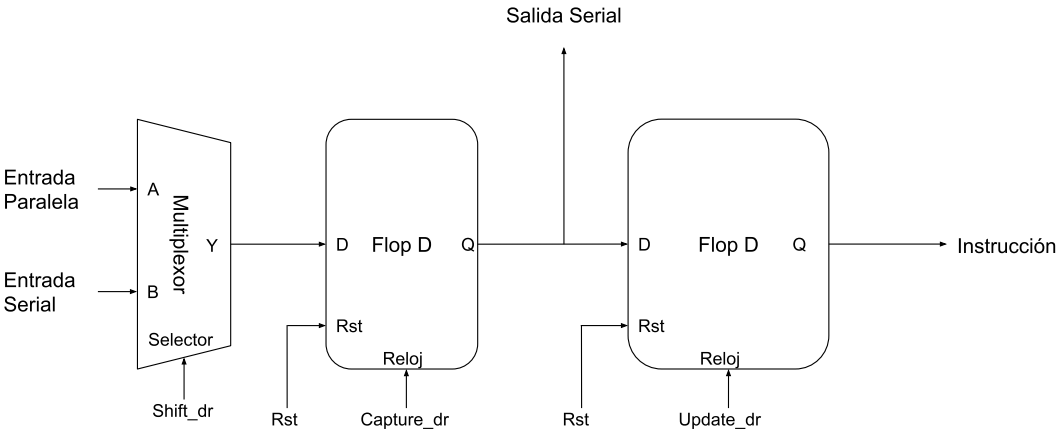

Figura 3.10: Celda de instrucción utilizando Flop

Se utiliza una celda Flip FLop D, esto debido al funcionamiento simple que tiene dicho Flop, cuando hay un flanco de reloj, la salida captura el dato presente en la entrada.

Implementación con Latch

Para la implementación con Latch, se utiliza una topología con Latch D como se observa en 3.11.

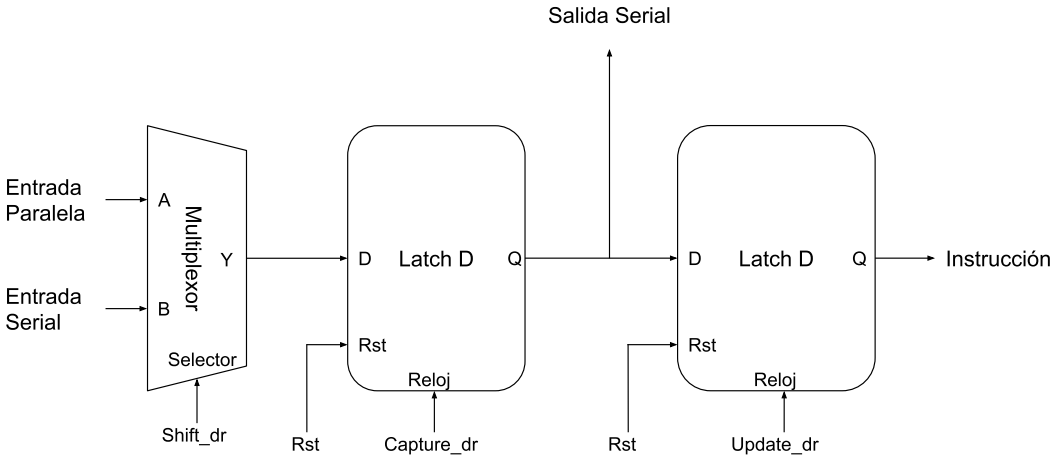

Figura 3.11: Celda de instrucción utilizando Latch

Los registros de instrucciones deben de tener la capacidad de transmitir de manera serial las instrucciones o capturar una instrucción mediante una carga paralela, para ello se utiliza un multiplexor de entrada cuya señal de selección está dada por Shift_dr. Los registros son controlados por las señales de Capture_dr, la cual permite capturar y desplazar los datos provenientes del multiplexor de entrada, y Update_dr, esta señal tiene como finalidad actualizar el registro de salida y cargar una nueva instrucción.

3.2.3. Decodificador

El decodificador tiene la funcionalidad de reasignar las señales dependiendo de la instrucción que está activa en el registro de instrucciones.
El decodificador de instrucciones se plantea como el driver de los demás bloques, incluyendo al registro de instrucciones, por ello recibe las señales de enable del controlador y dependiendo de las instrucciones envía las señales de selección para el mux de datos, la señal de captura y actualización para los demás registros.
Las instrucciones que recibe el decodificador se conforman de la siguiente manera BYPASS = 0, SAMPLE = 2, Preload = 1, INTEST = 3 y EXTEST = 7.
Se organiza en una tabla de comportamiento y se observa como se comporta el decodificador dependiendo de cada uno de los estados sigueindo la tabla 3.3.

El decodificador se implementa utilizando en una lógica de «One Hot» que controla el multiplexor de selección de datos, en caso de que sea necesario agregar una nueva instrucción, se añade una nueva salida al multiplexor y se asocia con la respectiva entrada del multiplexor con la finalidad de utilizar la salida Data_slctr como un habilitador de las

Instrucción	Salidas
0	Data_slctr = 0...01
1	Data_slctr = 0...10
2	Data_slctr = 0...100
3	Data_slctr = 0...1000
4	Data_slctr = 0...10000

Tabla 3.3: Decodificador de señales

celdas correspondientes.
Las señales de Updateir, Clockir, Shiftir, Shiftby, Clockdr, Inmodebsr, Outmodebsr, Inshiftdr, Outshiftdr se implementan con operacione y compuerta AND y Buffers.
En otras palabras Updateir = (Update_ir && tck) y Updatedr = (Update_dr && tck), estas señales se implementan debido a que debe durar un ciclo de reloj en alto, de igual forma las señales de reloj se implementan com compuertas AND Clockir = (Clock_ir && tck) y Clockdr = (Clock_dr && tck), como ya se mencionó Clock_dr y Clock_ir son señales de enable, por lo que para generar las señales de reloj para las celdas de captura de los registros, es necesario que sean producidas por la señal de reloj TCK.
En el caso de las señales Shiftir, Shiftdr, se implementan a manera de buffer de las señales de Shift_ir y Shift_dr.
En caso de la señal Outmodebsr e Inmodebsr, son señales generadas a partir de la instrucción que la utiliza, la instrucción Extest es la representación binaria de 4 o = {000...100}, por lo que la señal Outmodededebs toma el valor se le asigna el bit del vector correspondiente y de la misma manera para Inmodebsr.

3.2.4. Registro de Boundary Scan

Para la implementación de este registro, es necesario que pueda tener una salida serial y otra paralela, esto debido a que es una celda que se coloca entre los pines de entrada o salida y la entrada o salida de la lógica a la cual se le desea implementar esta técnica. Una implementación genérica de la celda se presenta a continuación.

El registro mostrado en 3.12 consta de dos registros para almacenar datos, los cuales son el registro de captura y el registro de salida, el registro de salida permite actualizar la salida paralela de la celda, mientras que el registro de captura permite tomar una captura de la entrada, ya sea serial o paralela, esta característica es definida por el modo de operación del controlador del TAP y de la instrucción implementada.
El registro de Boundary Scan permite capturar datos provenientes del pin de entrada mediante la conexión con la entrada paralela o desplazar datos de manera serial mediante la entrada seria. Además posee la capacidad de desconectar los pines de entrada de la lógica interna del CI, esta cualidad permite cargar a la salida paralela los vectores de prueba que se encuentren almacenados en los registros de salida, por ello se implementa con dos multiplexores, un multiplexor a la entrada permite capturar datos provenientes

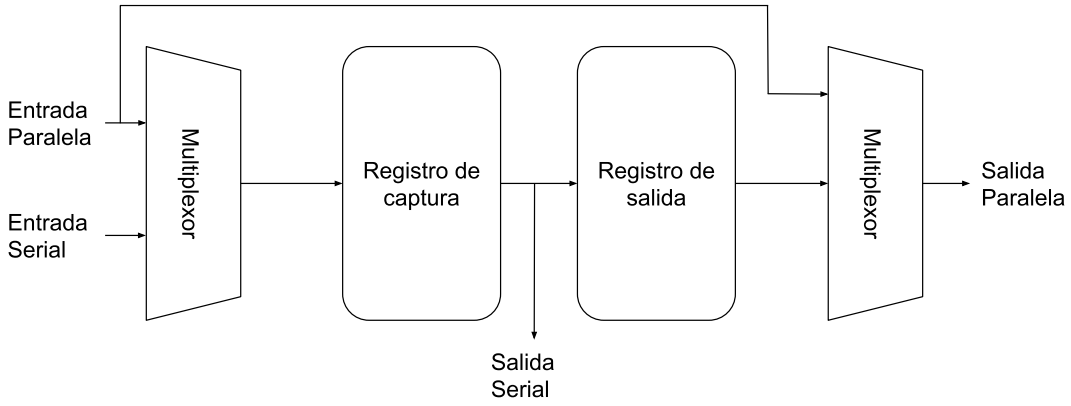

Figura 3.12: Celda del Registro de Boundary Scan

del pin de entrada mediante la entrada paralela o datos provenientes de una cadena serial de registros mediante la entrada serial, el multiplexor de salida permite desconectar el pin de entrada de la lógica de prueba durante la realización de pruebas o mantener una comunicación entre el pin y la lógica interna, en palabras más sencillas, permite evitar que la lógica de prueba influye sobre la conexión del pin con la lógica en tanto no se necesite realizar pruebas invasivas sobre la lógica interna, esta característica depende del tipo de instrucción con la cual se esté trabajando, ya sea una instrucción no invasiva o una instrucción invasiva.

Implementación con Flip Flop

Para la implementación con Flip FLop, se utiliza una topología con Flip FLop D como se observa en la fgura 3.13.

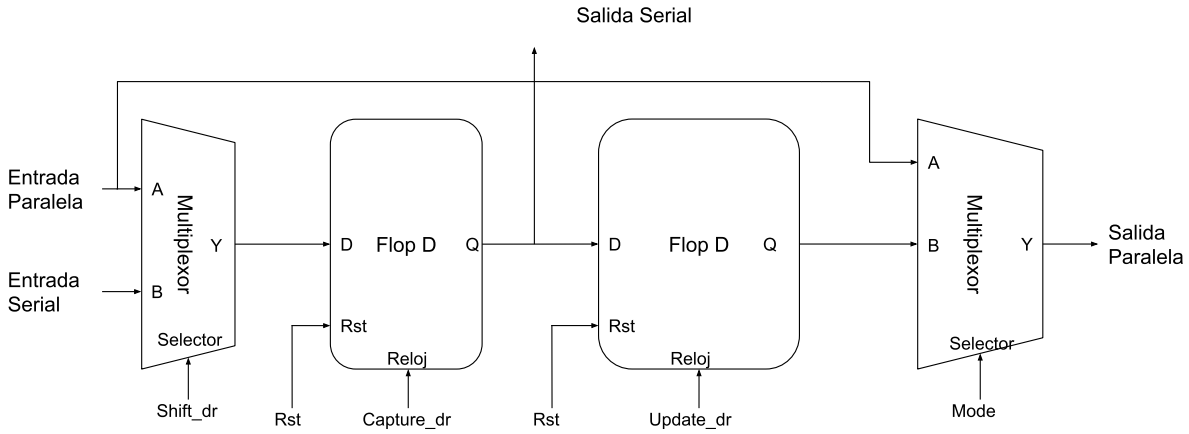

Figura 3.13: Registro de Boundary Scan utilizando flops

Implementación con Latch Pulsado

Para la implementación con Latch Pulsado, se utiliza una topología con Latch D como se observa en la figura 3.14.

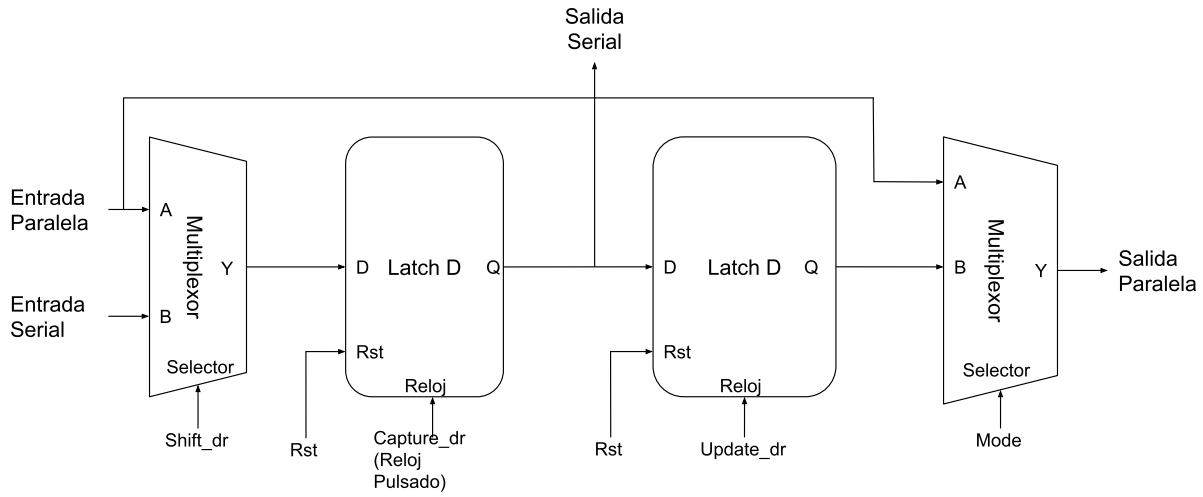

Figura 3.14: Registro de Boundary Scan utilizando latch

Para que las celdas cumplan con las restricciones de funcionamiento, el diseño consta de cuatro señales de control, Shift_dr, Mode, Capture_dr y Update_dr.
Shift_dr: selecciona cual entrada va a ser capturada, si la entrada paralela o la entrada serial.
Mode: esta señal se activa únicamente cuando una señal que requiera el modo de operación invasivo, prova que el Multiplexor de salida del registro de Boundary Scan seleccione, ya sea, la entrada paralela o la salida del registro de salida.
Capture_dr: permite capturar y desplazar los datos.
Update_dr: actualiza el registro de salida.

3.2.5. Registro de Bypass

El registro de bypass tiene como función principal establecer una ruta de menor longitud entre TDI y TDO.
Se implementa con una compuerta AND a la entrada con la finalidad de utilizarlo como un buffer controlado, esta característica se aprecia en la tabla 3.4. Esta configuración es relevante debido a que durante el tiempo en que permanece desconectado, el JTAG, no interactúa con los pines de entrada.
La arquitectura del registro de bypass se observa en la figura 3.15.

Esta celda debe tener la capacidad de registrar los datos cada flanco negativo del reloj TCK, esto implica que cada flanco negativo debe capturar un dato y transmitirlo hacia la salida del JTAG mediante el puerto TDO.
El registro de Bypass desconecta el CI de una cadena de pruebas vista en la figura 2.21, sin

A	B	Salida
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 3.4: Tabla de verdad de la compuerta AND

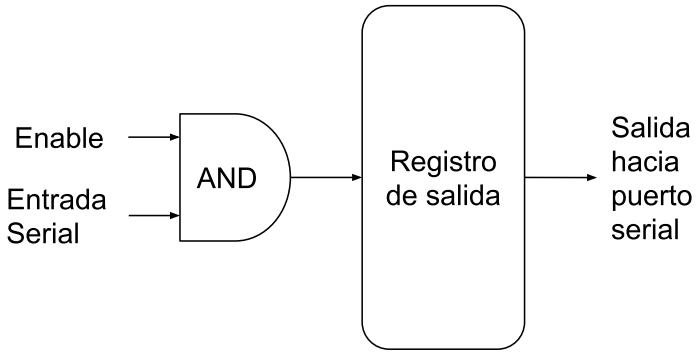

Figura 3.15: Bit para el Registro de Bypass

la necesidad de detener el funcionamiento de dicho CI, en otras palabras, si la instrucción de BYPASS está cargada en los registros de instrucciones, las pines de entrada y salida están conectados directamente a la lógica interna, esto implica que después de una prueba es posible mantener el funcionamiento del CI sin requerir un reinicio del equipo de pruebas debido a que la lógica interna y la lógica de pruebas se manejan con dominios de reloj independientes.

Implementación con Flip Flop

Para la implementación con Flip Flop, se utiliza una topología con Flip Flop D.

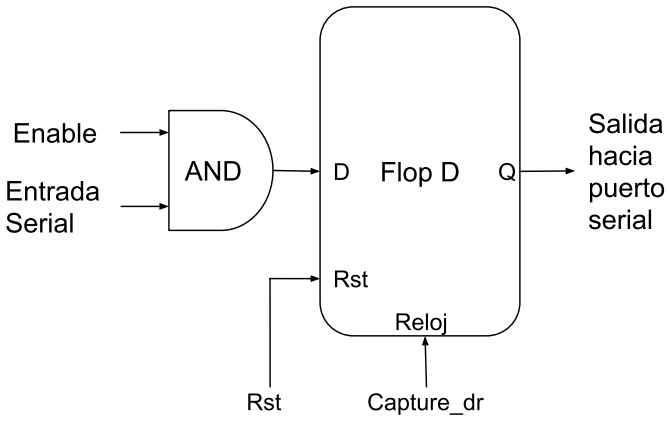

Figura 3.16: Registro de Bypass implementado con flop

Implementación con Latch Pulsado

Para la implementación con Latch Pulsado, se utiliza una topología con Latch D como lo muestra la figura 3.14.

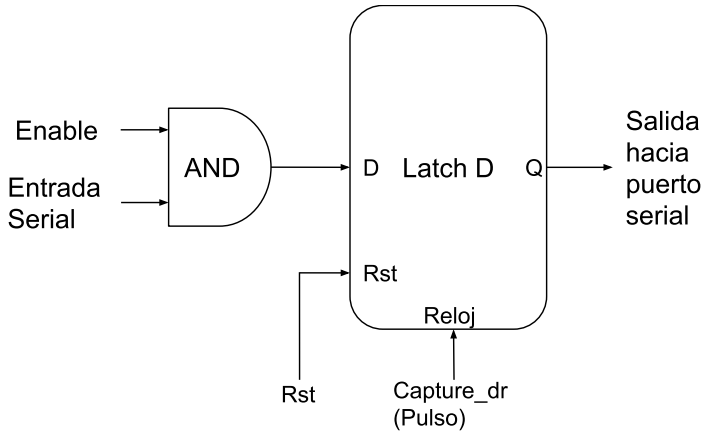

Figura 3.17: Registro de Bypass utilizando un latch

Se implementa con una una compuerta And y una señal de permiso que en este caso es la señal Shift_dr; se utiliza la compuerta And debido a que si en alguna de sus entradas es aplicada una señal en alto, se comporta como un buffer y si se aplica una señal en bajo impide el paso del otro puerto a la salida, la señal Capture_dr permite registrar los datos durante cada flanco de TCK.

3.2.6. Multiplexor de datos

El multiplexor de datos se implementa utilizando una lógica de One Hot, esto permite utilizar multiplexores de menor tamaño que los generados por compuertas lógicas como lo son los multiplexores utilizando compuertas NAND, dicha implementación se observa en la figura 3.18.

La implementación de este multiplexor permite que sea parametrizable, en otras palabras, que se defina un parámetro o constante que permita variar el tamaño de entradas dependiendo del valor que adquiera dicha constante, con esto es cuestión de definir cuantas entradas hay y un selector por cada una de las entradas.
La finalidad de este multiplexor es seleccionar la cadena de scan de acuerdo con la instrucción actual.

3.2.7. Multiplexor de salida

El multiplexor de salida se implementa comporta mentalmente bajo un simple condicional debido a que depende únicamente de una señal que es un 1 o un 0.

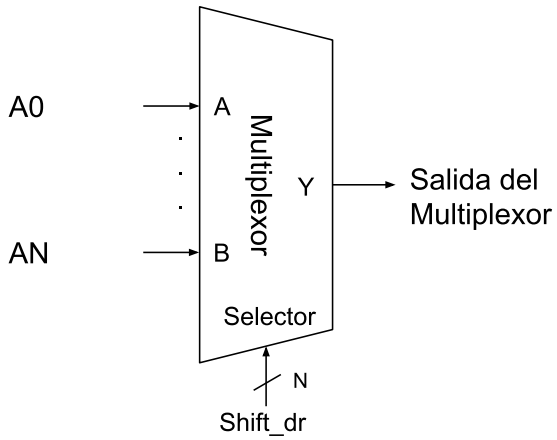

Figura 3.18: Multiplexor de datos «One Hot»

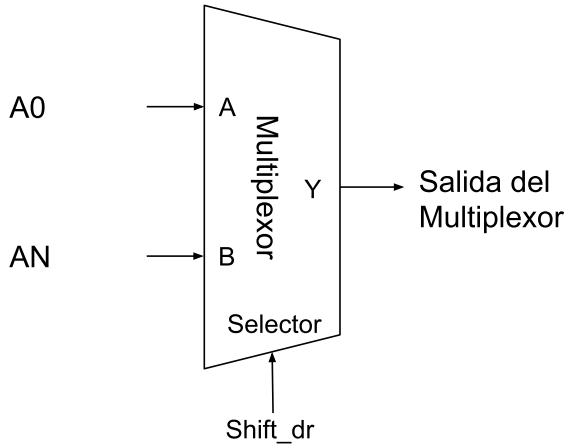

Figura 3.19: Multiplexor de salida

Un multiplexor de dos entradas y un selector. Este multiplexor decide si lo que se va a transmitir al puerto de salida TDO son instrucciones o datos.

3.2.8. Generador de pulsos de reloj para la topología del latch pulsado

El generador de pulsos es un circuito que emite un pulso cada vez que detecta un flanco de reloj, puede ser durante flancos negativos o durante flancos positivos.
Para un detector de flancos positivos se utiliza la compuerta And con una unidad de retraso, comúnmente un inversor como se observa en la figura 3.20.
Para un detector de flancos negativos se utiliza la compuerta Xor con una unidad de retraso, usualmente un inversor como se observa en la figura 3.21.

Así tenemos un circuito generador de pulsos conectado a múltiples latches con la finalidad de que funcionen como un registro de desplazamiento en configuración serial, con esto es

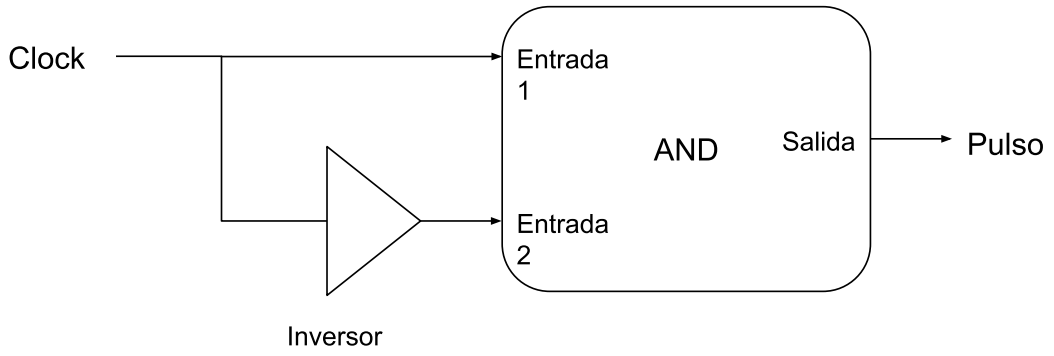

Figura 3.20: Circuito detector de flancos positivos utilizando compuerta AND

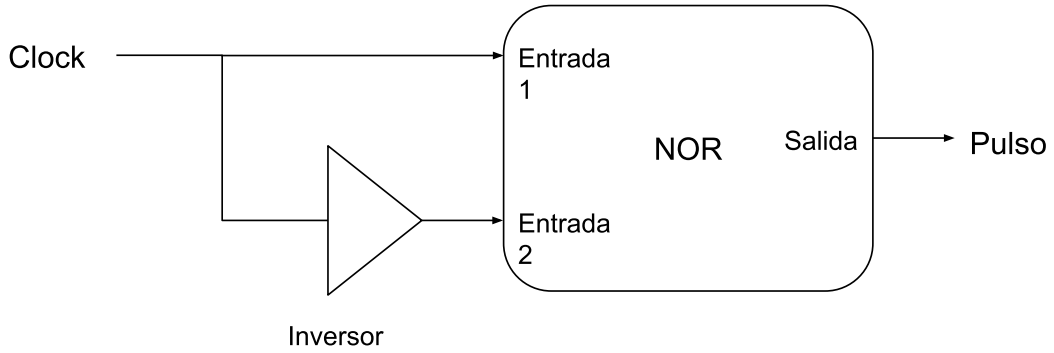

Figura 3.21: Circuito detector de flancos negativos utilizando compuerta XOR

posible conectar varios registros de datos en serie y compartir entre todos un circuito generador de pulsos de reloj como se observa en la figura 3.22. Este diseño tiene la cualidad de ser más lento, debido a que entre mayor sea en crecimiento en longitud de los registros, más pulsos es necesario generar para poder desplazar los datos, esto implica que en caso de que el periodo de la señal de reloj sea muy pequeño, los pulsos podrían seguir generándose con una dependencia del flanco anterior incluso después de transcurrido el periodo de la señal de reloj, esto implica que hay que aumentar el periodo de la señal de reloj para dar tiempo suficiente al circuito generador de pulsos de emitir todos los pulsos sin que se vea afectada la transmisión de los datos.

Otro efecto a considerar es que pasa si se captura un dato meta estable, el dato meta estable se genera debido a que el latch captura un dato que no se ha estabilizado po rlo que no se puede determinar el valor que va a adquirir el latch sea un «1» o un «0». La falta de estabilidad se produce debido a que o los datos no se estabilizan cuando el pulso de reloj es generado, o cuando el pulso de reloj se genera sin considerar la frecuencia máxima de operación por lo que los pulsos de reloj se genera más rápido que el proceso de estabilización del dato.

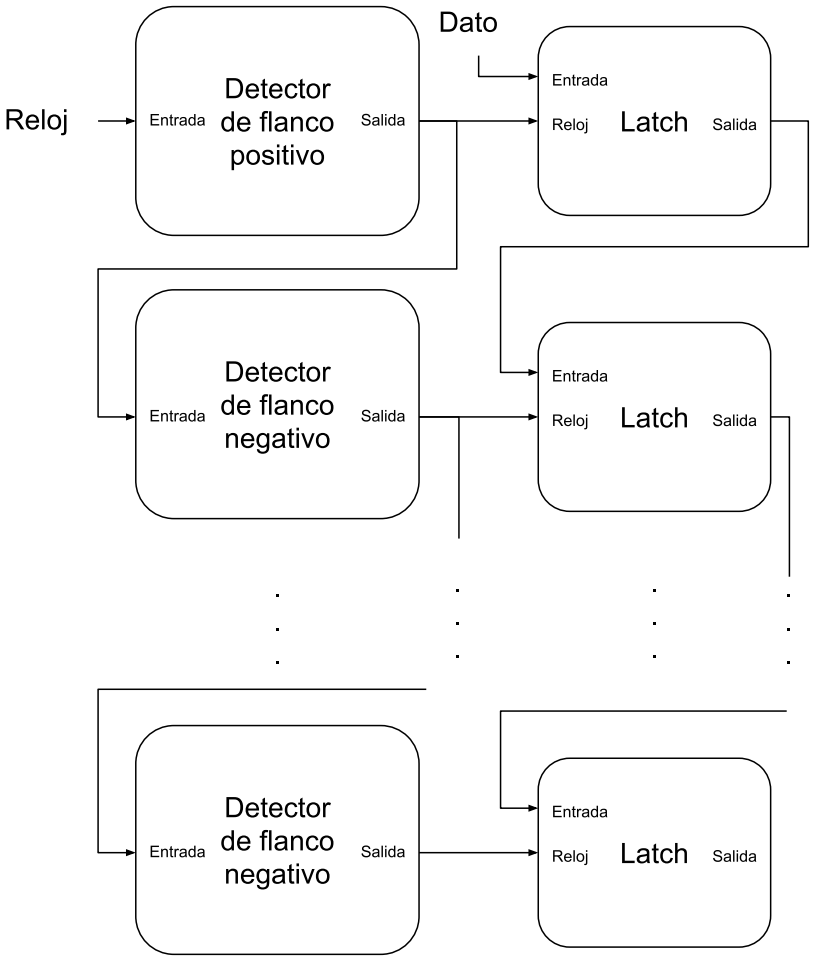

Figura 3.22: Registro de desplazamiento basado en Latches Pulsados

3.3. Implementación del Diseño

La implementación del diseño sigue un proceso sencillo de dos etapas como se observa en la figura 3.23, dichas etapas consisten en desarrollar la biblioteca mediante un lenguaje de descripción de hardware (HDL) y la implementación lógica utilizando una herramienta de síntesis adecuada.
El desarrollo de la biblioteca mediante el uso de un HDL permite obtener un modelo funcional a nivel lógico, en otras palabras, se desarrolla un modelo funcional descriptivo del circuito con la finalidad de emular el comportamiento a nivel lógico.
Una vez realizado el proceso de síntesis lógica se obtiene un biblioteca descrita a nivel de compuertas lógicas, a comparación de un modelo descriptivo, el modelado mediante compuertas lógicas describe el comportamiento de un circuito mediante un arreglo de compuertas lógicas mediante tecnología CMOS, lógica de paso o lógica compleja.

La implementación de los diseños se realiza mediante la herramienta de compilación Synopsys VCS y el simulador Simv de Synopsys, estas herramientas permiten compilar y simular HDL, en este caso SystemVerilog, y generar una estimación lógica del funcionamiento del circuito, esto debido a que se utiliza un modelo RTL para dicha aplicación,

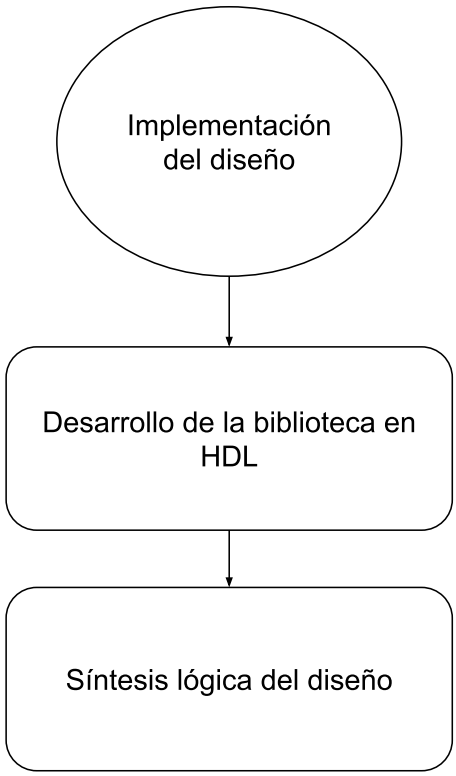

Figura 3.23: Implementación del diseño

aunque esta implementación es completamente funcional, permite estimar el comportamiento lógico del circuito planteado.
Para realizar el diseño se trabaja individualmente sobre cada uno de los módulos con la finalidad de describir el comportamiento deseado para cada uno, en otras palabras, se realiza un modelo funcional por cada uno de los módulos y se deja a discreción de la herramienta implementar la lógica correspondiente.
Una vez realizado el diseño comporta mental, es necesario seguir un proceso de síntesis lógica, esto significa que se realiza una estimación de consumo energético, de área y finalmente de retrasos temporales debido a efectos de retraso de compuerta, estimación de capacitancia de entrada de compuerta y fanout, esta estimación es realizada mediante Synopsys Design Compiler, esta herramienta realiza una síntesis lógica, esto es una reducción de las ecuaciones booleanas del circuito con la finalidad de optimizar en área, velocidad y consumo de energía, dicha síntesis se realiza con una biblioteca de celdas descritas utilizando el lenguaje de Verilog provistas por el fabricante, esta biblioteca contiene una descripción del comportamiento además de contener característica temporales de las celdas producidos por el proceso de fabricación, por lo que genera una estimación sobre una estadística de datos reales.
La síntesis lógica puede ser sometida a pruebas mediante la generación de un netlist, un netlist es una descripción del circuito hecha en Verilog resultante del proceso de síntesis, dicha descripción incluye efectos físicos por lo que es posible utilizar el netlist para estimar el comportamiento que tendrá el circuito una vez fabricado, pero nunca para tener una

estimación completamente real circuito.

El desarrollo de la biblioteca en HDL se puede subdividir procesos más pequeños con la finalidad de obtener un modelo descriptivo del funcionamiento del circuito implementado de manera modular como se observa en la figura 3.24. Durante este proceso se obtienen dos archivos, uno llamado Biblioteca, el cual contiene la descripción individual de cada uno de los módulos y un segundo archivo el cual consiste en la instanciación de caa uno de los módulos para generar el circuito diseñado en la figura 3.1.

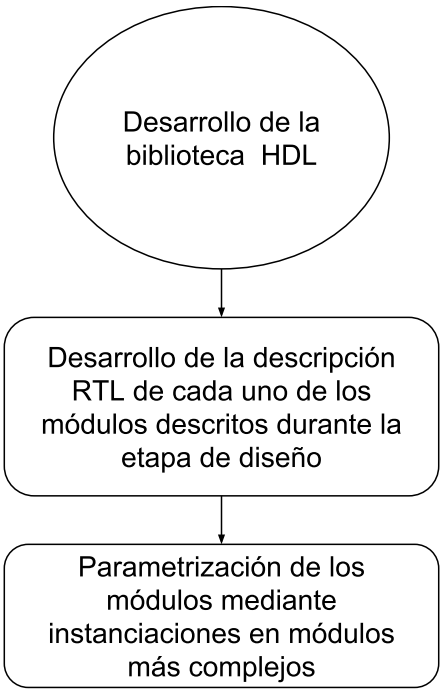

Figura 3.24: Desarrollo de la bilbioteca

Descripción RTL: La primer tarea a realizar es tomar cada uno de los módulos y analizar las característica las características que los módulos deben tener con la finalidad de generar un modelo de comportamiento del módulo individual [2], además de si son módulos que se mantienen estáticos o es necesario generar un parámetro para cambiar el tamaño de los módulos, una vez realizada esta tarea, se prosigue a realizar una descripción RTL de cada uno de los módulos buscando siempre una implementación individual, en el caso de los registros si es necesario conectar varios en serie, se desarrolla una celda unitaria que describe el funcionamiento del registro para su posterior uso como se observa en [6]. Instanciación en módulos complejos: Una vez terminado el desarrollo de las celdas básicas, se toma la celda y mediante un comando de compilación y una variable de generación, el comando de compilación es una función conocida como generate, esta función permite instanciar múltiples módulos mediante el uso de un bucle como lo es un For o un While, para contar cuantos módulos se van a instanciar se necesita generar una variable que permita dicha funcionalidad, esta variable se llama genvar a diferencia de un número entero, es una variable de síntesis, en otras palabras no es una constante, si no que se implementa

únicamente durante el proceso de síntesis con la finalidad de permitir varias instanciaciones mediante la función generate. Finalmente se genera dentro de la biblioteca los módulos cuya instanciación dependen de un parámetro, esto implica que, de las celdas individuales, se pueden obtener diseños más complejos debido a las múltiples instanciaciones del diseño.

El proceso de síntesis lógica también se puede subdividir en varias tareas como sigue la figura 3.25, con la finalidad de obtener reportes de área y consumo de energía para la implementación de la biblioteca con el fin de caracterizar los diseños como se observa en la figura 3.25.

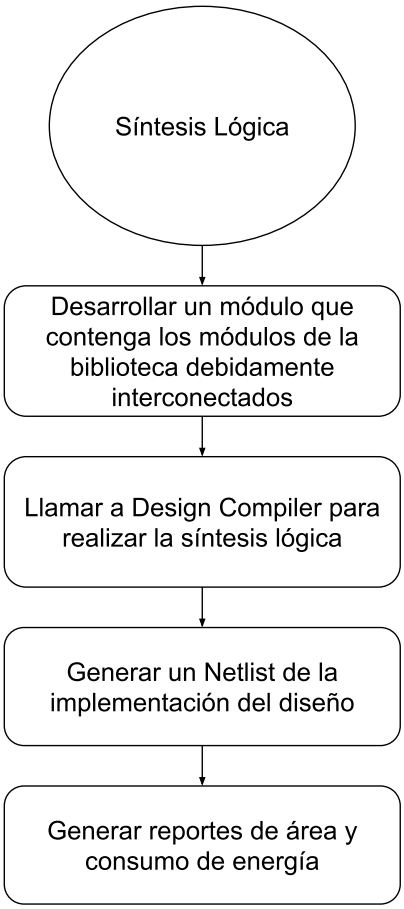

Figura 3.25: Desarrollo de la síntesis lógica

Módulo General: Esta tarea consiste en implementar el diseño esto implica tomar los módulos de la biblioteca y desarrollar las conexiones necesarias entre los módulos y la lógica interna que va a ser sometida a síntesis.
Llamar a la herramienta de síntesis: Abrir Design Compiler y correr los comandos necesarios para realizar la síntesis lógica del circuito. Generar un Netlist: Generar la descripción en Verilog del diseño, el netlist es generado por la herramienta de síntesis utilizando la biblioteca proporcionada por el fabricante, un netlist es una implementación utilizando modelos de celdas físicos que cumplen con el comportamiento descrito por la biblioteca

realizada por el diseñador.
Generar reportes: La generación de los reportes de consumo de energía y consumo de área, el reporte de área se genera utilizando la estimación de área proporcionada por el fabricante de cada una de las compuertas que la herramienta utiliza durante el proceso de síntesis, estas compuertas son las que se encuentran instanciadas en el netlist y son proporcionadas por el fabricante; en el caso del reporte de energía es necesario la generación de un saif, un saif es un reporte del factor de actividad de una compuerta, en otras palabras cuantas veces conmuta, cuanto tiempo permanece en 1 y cuanto tiempo permanece en 0, este reporte se genera simulando el funcionamiento del circuito y mediante un comando de simulación, el simulador genera el respectivo saif.

3.4. Validación del diseño

Para validar ambos diseños, es necesario realizar pruebas que permitan corroborar el funcionamiento realizado, debido a que el proyecto se implementa utilizando SystemVerilog, el cual es un Lenguaje de Descripción de Hardware, la validación se realiza mediante un banco de pruebas una vez hecha la síntesis lógica.
Por lo que se necesitan tareas que dictan el proceso a seguir, la validación es un proceso que se presenta en todos pasos a seguir durante diseño e implementación de una arquitectura. La figura 3.26 permite determinar si el diseño es funcional y adecuado o es necesario iterar el proceso para obtener una biblioteca con el funcionamiento adecuado, también se obtiene una comparativa en términos de área y consumo con la finalidad de determinar si las implementaciones realizadas son consistentes con la teoría que respalda la decisión de realizar dichas implementaciones.

Consulta de experto: Una vez realizado el diseño inicial debe ser sometido a revisión, esto con el fin de tener una segunda opinión con respecto al trabajo realizado, para este caso el experto es considerado el profesor asesor del proyecto, en otras palabras se realiza una revisión sobre el diseño presentado y una vez que se consigue una aprobación se procede a la realización e implementación del diseño, esta tarea implica describir la funcionalidad de cada uno de los módulos con la finalidad de encontrar errores de funcionamiento lógico o simplificaciones que se le puedan realizar al diseño inicial.
Pruebas de funcionamiento: Durante el proceso de implementación se habló de una descripción RTL para cada uno de los módulos esto implica que cada módulo debe llevar su banco de pruebas correspondiente con el fin de garantizar el correcto funcionamiento de las celdas, una vez realizada la verificación de cada uno de los módulos es necesario verificar el funcionamiento de los módulos más complejos, por lo que se deben generar pruebas más complejas, es aquí donde los testbench se deben tratar de automatizar con el fin de facilitar el proceso de validación, esto implica generar un módulo que describa una secuencia de estímulos para observar el comportamiento del circuito, la característica de este testbench es que debe poder cambiar acorde al parámetro cambia las características del diseño.

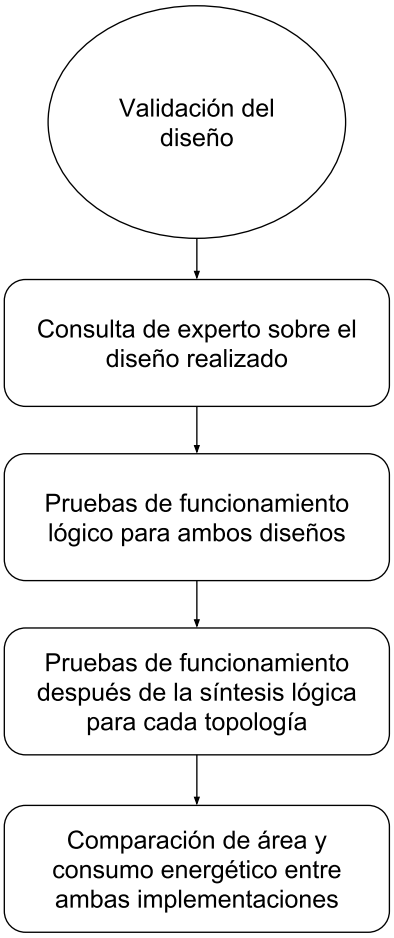

Figura 3.26: Validación del diseño

Pruebas con Netlist: Una vez realizada la síntesis lógica es necesario revisar el comportamiento de dicho módulo, esto debido a que en un Netlist se consideran los retrasos de compuerta, los testbench son los mismos que durante las pruebas de funcionamiento pero el enfoque y cuales resultados se buscan difiere del anterior, si bien es necesario observar el funcionamiento, ahora se busca buscar errores debido a retrasos temporales, bugs entre módulos, o glitches en las señales, esto debido a que estos tests indican si es necesario aumentar el periodo de reloj, si existen problemas de min o problemas de max, si es necesario agregar buffers entre módulos o cualquier otra necesidad que pueda surgir, como si es necesario implementar lógica sesgada.
Comparación área y consumo energético: Finalmente para validar el diseño, se necesita un modelo de referencia, en este caso el modelo de referencia es la implementación con Flip Flops, la idea es comparar la ganancia en términos de área implementando Latches pulsados contra los Flops, por lo que los reportes de área y consumo energético son necesarios, estos reportes se generan desde la herramienta Design Compiler y reportar jerárquicamente el consumo de área y de energía, esto es por cada módulo generado.

3.4.1. Verificación del controlador

El controlador controla el flujo de datos durante la operación del TAP, por lo que la validación de dicho módulo se convierte en una tarea relevante a la hora de desarrollar el proyecto para garantizar el funcionamiento adecuado.
Por lo que para verificar el controlador es necesario seguir dos pasos como se observa en la figura 3.27. La verificación del controlador tiene como resultado observar si el controlador funciona de manera adecuada o si es necesario realizar cambios de funcionamiento y reiterar la descripción en HDL del controlador.

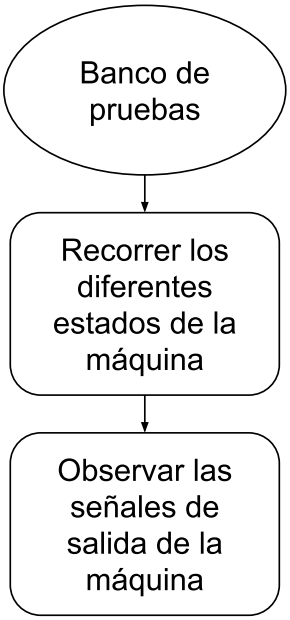

Figura 3.27: Pasos para verificar el controlador

El primer paso es realizar un banco de pruebas que garantice que un desplazamiento por todos los estados de la máquina con el fin de observar todas las transiciones y conmutaciones de los registros de estados y de la lógica de salida.
El segundo paso es observar dichos resultados una vez realizada la implementación y la simulación lógica además de realizar la simulación con netlist se obtiene la siguiente tabla comparativa con las señales respectivas que deben estár en alto y los resultados obtenidos por la simulación.

Estado	Simulación	Síntesis Lógica
Test_Logic_reset	rst	rst
Run_Test_idle	–	–
Select_dr_Scan	Slctr	Slctr
Select_ir_Scan	–	–
Capture_dr	Clock_dr & Slctr	Clock_dr & Slctr
Shift_dr	Clock_dr & Slctr & Shift_dr	Clock_dr & Slctr & Shift_dr
Exit1_dr	Slctr	Slctr
Pause_dr	Slctr	Slctr
Exit2_dr	Slctr	Slctr
Update_dr	Slctr & Update_dr	Slctr & Update_dr
Capture_ir	Clock_ir	Clock_ir
Shift_ir	Clock_ir & Shift_ir	Clock_ir & Shift_ir
Exit1_ir	–	–
Pause_ir	–	–
Exit2_ir	–	–
Update_ir	Update_dr	Update_dr

Tabla 3.5: Implementación del control utilizando una máquina de estados

3.4.2. Reporte de área y consumo

Para los reportes de área y energía se utilizan dos topologías para las cadenas de scan que se agregan a los registros de datos. La primera implementación se realiza con registros de desplazamiento, en otras palabras una cadena de registros utilizando latches pulsados o flops y otra utilizando una celda de scan multiplexada.
El registro de desplazamiento se observa en la figura 3.28.

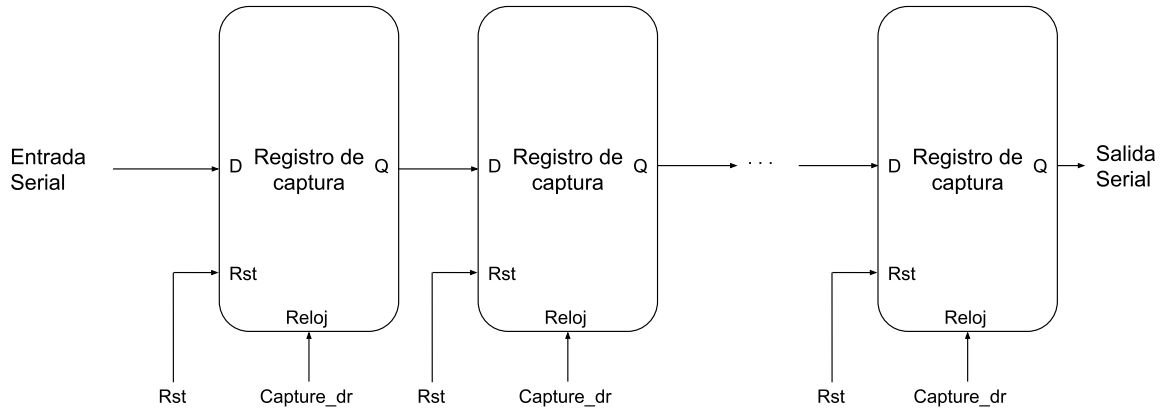

Figura 3.28: Registro de desplazamiento de N bits

Esta celda se implementa con la finalidad de establecer un punto de comparación para obtener la estimación de área, ya que al no utilizar lógica combinacional la diferencia radica únicamente en el área consumida por la lógica secuencial, permitiendo observar la

variación de área más grande entre ambas implementaciones.
La celda multiplexada se presenta en la figura 2.22.

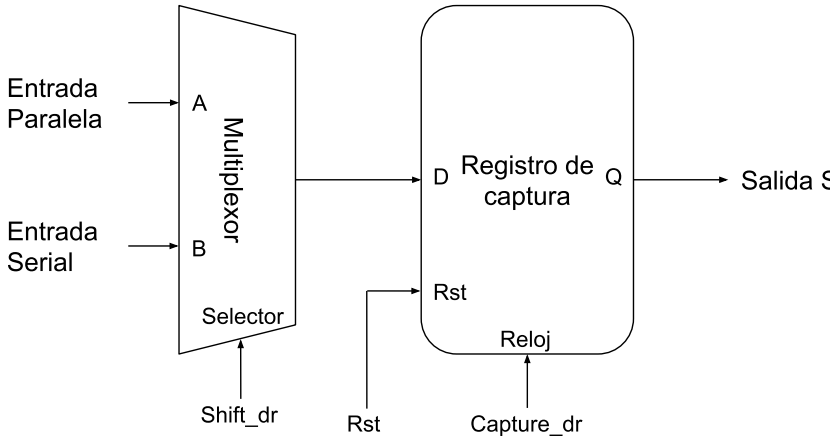

Figura 3.29: Celda de Scan D Multiplexada

Esta celda se conecta en cascada dependiendo del parámetro que determine la longitud de la cadena. Al utilizar lógica combinacional extra, el área aumenta, esto implica que aunque la diferencia en área se observe en la lógica secuencial, el valor total del área aumenta debido al proceso de síntesis, ya que no todas las celdas van a ser colocadas de tal manera que el proceso presenta variabilidad, por lo que la diferencia en área puede ser menor.
Para obtener los reportes de área se implementan una cantidad variable de cadenas de scan las cuales varían en el largo de sus bits con la finalidad de generar varios reportes en área y obtener los datos necesarios para realizar la comparación. Este modelo se define debido a que al aumentar las cadenas conectadas al estándar aumenta el área que consumen dichas cadenas la idea es generar un estudio comparativo que permita determinar el área utilizada por ambos diseños y cuales compromisos hay que considerar a la hora de implementar alguna de las topologías. En general entre mayor sea la densidad de los bits implementados en las cadenas de scan menos significativo va a ser el área del TAP, el objetivo es observar que la variación de consumo de área entre ambas topologías variando la cantidad de cadenas de scan conectadas.
En el caso de los reportes de energía es necesario realizar un banco de pruebas que permita estresar el diseño, esto permite generar un saif el cual se utiliza como medio para estimar el consumo energético.

El banco de pruebas tiene dos funciones, la primera es corroborar el funcionamiento adecuado del TAP y la segunda es estresar el diseño para generar un saif, con la finalidad de obtener una estimación del consumo de energía.

Por lo que el banco de pruebas debe seguir los pasos presentados en la figura 3.30.

Primeramente se carga la instrucción PRELOAD, esta instrucción permite precargar los

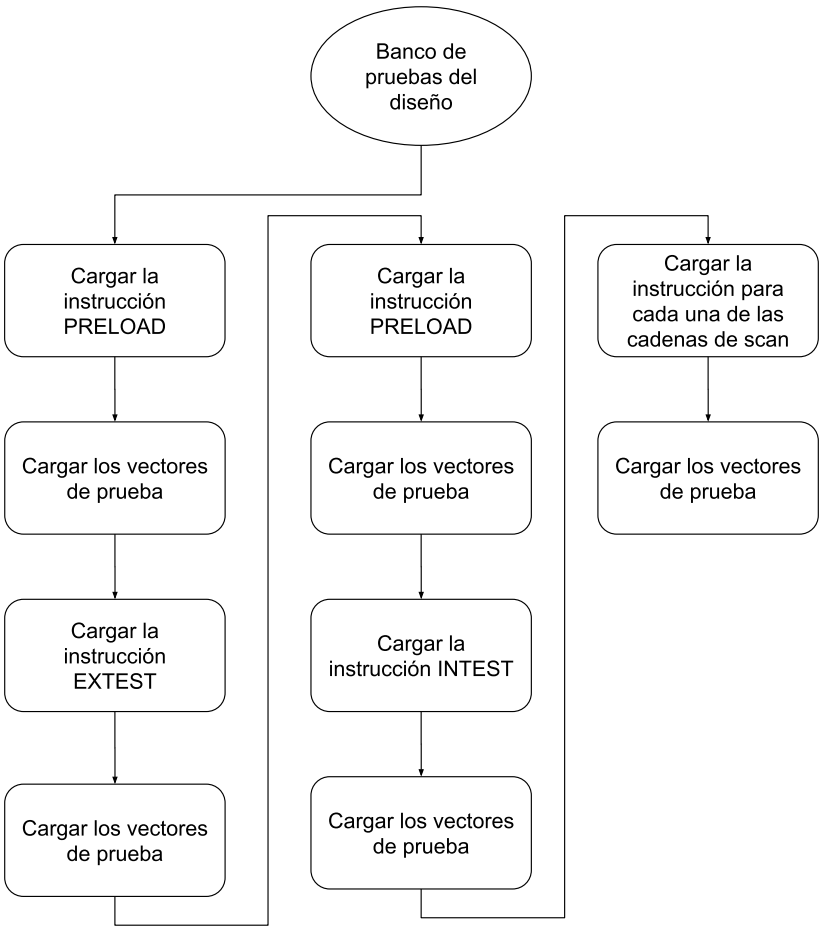

Figura 3.30: Banco de pruebas

registros de Boundary Scan con vectores de prueba en este caso se carga con 1 todos los bits del registro de Boundary Scan, después se carga la instrucción extest, esta instrucción permite cargar los registros de Boundary Scan de salida con los datos provenientes de los bits de salida de la celda de Boundary Scan, esta instrucción permite realizar pruebas a nivel de placa.

Despues se vuelve a cargar los vectores de prueba para realizar otra operación, en este caso se carga un 0 en todos los bits del registro de Boundary Scan.

Después se carga nuevamente la instrucción PRELOAD y se vuelve a cargar un 1 en todos los bits del registro de Boundary Scan, seguidamente se carga la instrucción INSTEST, esta función permite realizar pruebas de la lógica interna, lo que hace es que desconecta los pines de entrada y carga los bits de salida de la celda de bpundary scan del registro de Boundary Scan de entrada.

Después se vuelve a cargar otros vectores de prueba en este caso se carga el vector 0 en todos los bits del registro de Boundary Scan menos el último, en otras palabras se utilizan N-1 ciclos, donde N es el tamaño de la cadena del registro de Boundary Scan, de reloj para cargar 0 en todos los bits del registro.

Con este proceso se busca revisar el funcionamiento del TAP, pero para estresar el JTAG es necesario buscar el peor de los casos, el cual es cuando conmuta constantemente de 0

a 1 y de 1 a 0, esto debido a que genera más conmutaciones en los registros, aumentando así la potencia dinámica.
Para estresar el diseño, es necesario cargar una instrucción que selecciones las cadenas de scan necesarias y seguidamente se carga una secuencia 10....10 durante la cadena los ciclos de reloj necesarios para generar conmutaciones y estimar el factor de actividad de la compuerta a partir de dichas conmutaciones.

Capítulo 4

Resultados y Análisis

Para corroborar el funcionamiento se implementan las instrucciones de PRELOAD, SAMPLE INTEST y EXTEST. PRELOAD y SAMPLE son instrucciones que actúan sobre los registros de desplazamientos, la función de INTEST y EXTEST actúan sobre los registros de Boundary Scan y requieren permisos de pin para realizar pruebas invasivas, por lo que para corroborar el funcionamiento del TAP es necesario observar la salida paralela del circuito o Pout, si esta salida se mantiene con el mismo comportamiento a lo largo del Testbench, implica que el dato se desplazó y cargó adecuadamente sobre los registros de Boundary Scan y los registros de Instrucciones.
Los resultados de la comprobación de funcionamiento se ven en las siguientes imágenes de resultados de la simulación.
El primer resultado visto en 4.1 corresponde a la simulación lógica de topología implementada utilizando Flip Flops en los registros de scan .

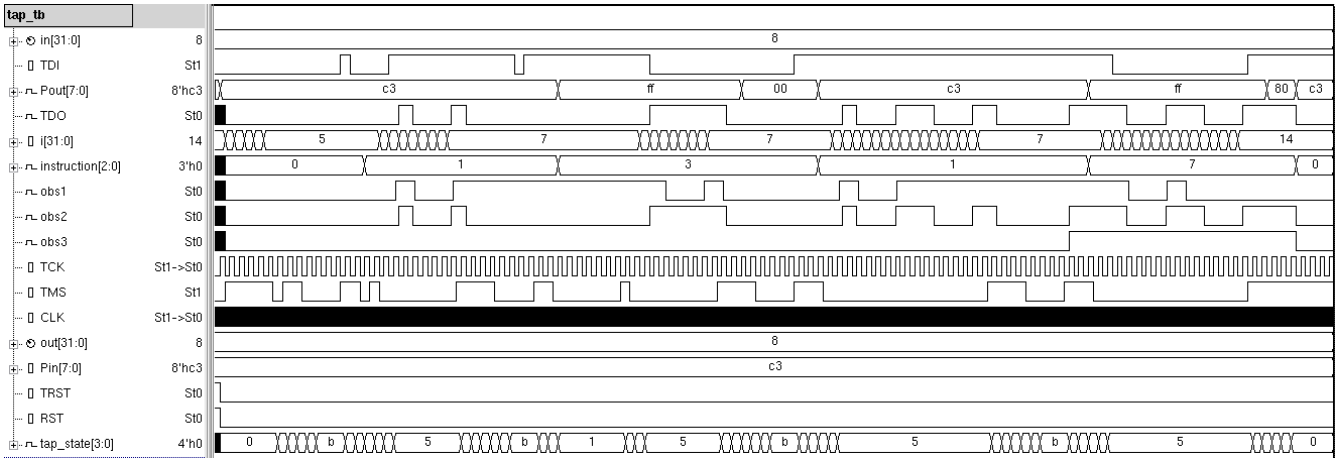

Figura 4.1: Simulación lógica para la implementación con Flops.

La figura 4.2 corresponde a la simulación utilizando el netlist obtenido una vez realizado el proceso de síntesis lógica.

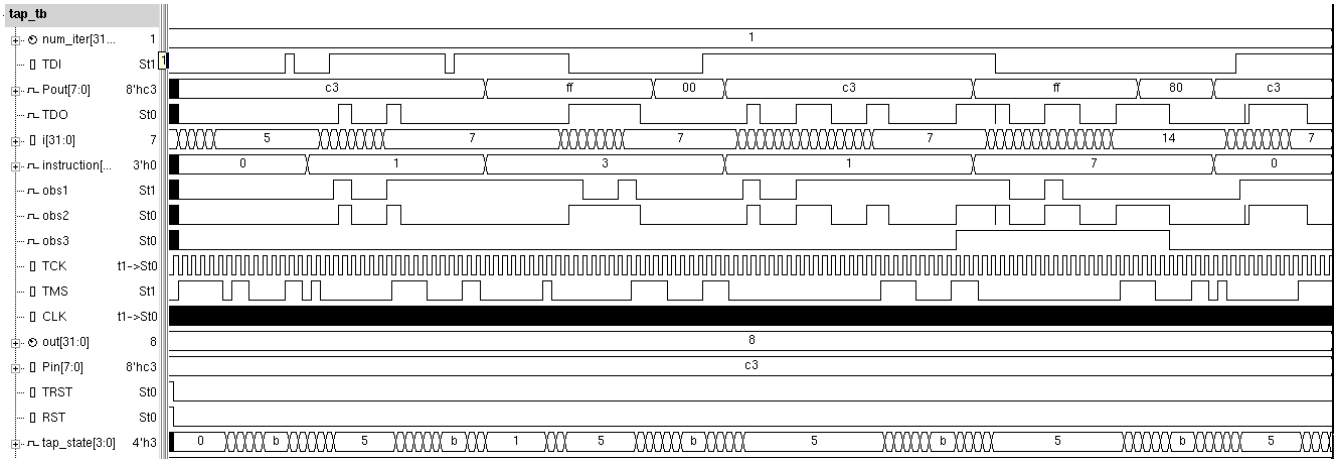

Figura 4.2: Simulación utilizando el netlist para la implementación con Flops.

El diagrama de ondas presentado a continuación corresponde al realizado con la implementación de Latches pulsdos.

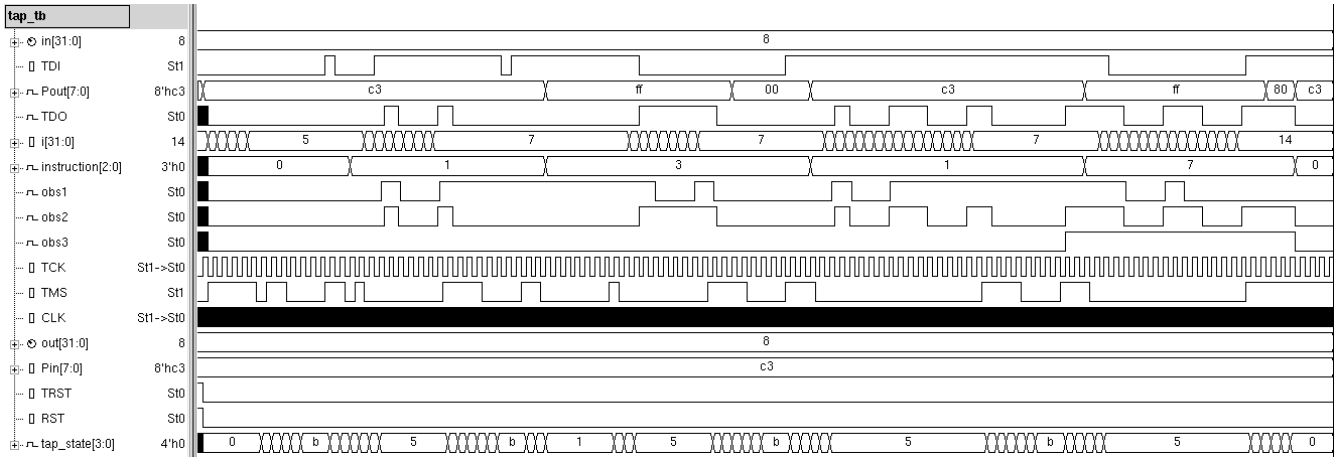

Figura 4.3: Simulación lógica para la implementación con Latches Pulsados.

Finalmente se observa la simulación realizada con el netlist para la síntesis lógica realizada con la topología de Latches Pulsados.

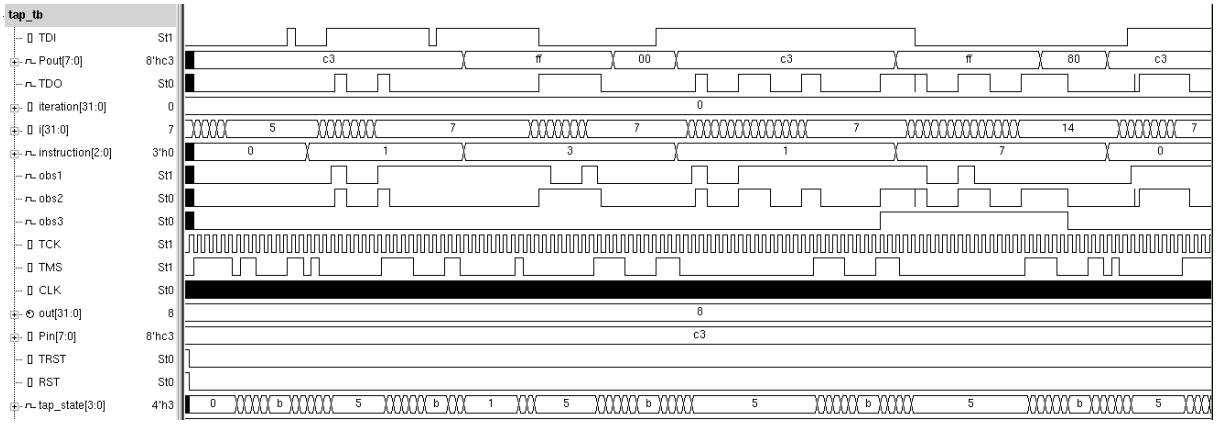

Figura 4.4: Resultado de la simulación utilizando el netlist de la síntesis lógica para la implementación con Latches Pulsados.

Como se observa en los resultados obtenidos por la simulación, el diseño mantiene el funcionamiento una vez realizada la síntesis lógica, también que el comportamiento no cambia entre las dos implementaciones, por lo que podemos decir que la implementación funciona adecuadamente, ya que el comportamiento se mantiene entre las dos implementaciones.
La tabla 4.1 resume el comportamiento obtenido por los pines de salida y los respectivos vectores cargados a las celdas de boundary scan y a los pines de entrada para la implementación utilizando flip flops a nivel de RTL .

Instrucción	Valor Hexadecimal de entrada	Salida
BYPASS	C3	C3
PRELOAD	C3	C3
INTEST	FF,00	FF,00
PRELOAD	C3	C3
EXTES	FF,80	FF,80

Tabla 4.1: Resultado de la simulación lógica flip flops

En el caso de la simulación del netlist los resultados se resumen en la tabla 4.2

Instrucción	Valor Hexadecimal de entrada	Salida
BYPASS	C3	C3
PRELOAD	C3	C3
INTEST	FF,00	FF,00
PRELOAD	C3	C3
EXTES	FF,80	FF,80

Tabla 4.2: Resultado de la simulación post síntesis lógica flip flops

Los resultados de los latches pulsados se resume en la tabla 4.3.

Instrucción	Valor Hexadecimal de entrada	Salida
BYPASS	C3	C3
PRELOAD	C3	C3
INTEST	FF,00	FF,00
PRELOAD	C3	C3
EXTES	FF,80	FF,80

Tabla 4.3: Resultado de la simulación lógica latches pulsados

Y para el netlist de la implementación de latches pulsados se tiene la tabla 4.4.

Instrucción	Valor Hexadecimal de entrada	Salida
BYPASS	C3	C3
PRELOAD	C3	C3
INTEST	FF,00	FF,00
PRELOAD	C3	C3
EXTES	FF,80	FF,80

Tabla 4.4: Resultado de la simulación post síntesis lógica latches pulsados

El banco de pruebas corrobora el funcionamiento del diseño, no busca realizar una análisis exhaustivo, por lo que realizar un proceso de verificación más completo se deja como tarea a futuro.

4.1. Reporte de área

Uno de los parámetros más relevantes para el proyecto es el consumo de área, por lo que observar la tendencia de ambas implementaciones permite comparar las topologías implementadas.
Para esta evaluación se utilizaron dos implementaciones para las cadenas de scan, la primera desarrollada con Celdas D Multiplexadas y otra utilizando registros de desplazamiento, para ambas implementaciones se utilizaron latches pulsados y flip flops.

Cadenas conectadas	Largo de la cadena (bits)	Flip Flop	Latch Pulsado
2	16	8480.715431	9548.267541
4	16	10194.25983	11997.44615
8	16	13788.27282	17182.67475
16	16	20837.27138	27169.47575
32	16	34745.10911	46950.07646
64	16	62574.52604	86344.23522
2	32	10941.25218	11889.87515
4	32	14283.92192	16669.00134
8	32	21111.43802	26525.03759
16	32	34566.64832	45853.93365
32	32	61512.82862	84424.71512
64	32	114950.2852	160945.5491
2	64	15895.00581	16576.08314
4	64	22482.50116	26036.35203
8	64	35721.71768	45210.27364
16	64	62201.97381	83330.01746
32	64	114801.4626	159029.0865
64	64	219839.7816	310305.2441
2	128	25880.64563	25925.08561
4	128	38861.41744	44746.1866
8	128	65062.63223	82701.74267
16	128	117172.2955	158051.5375
32	128	221328.9464	308623.2259
64	128	430790.0173	610356.398

Tabla 4.5: Resultados de área por cantidad de cadenas en paralelo y el largo de las cadenas

La tabla 4.5 permite organizar los resultados mediante cuántas cadenas en paralelo se implementan y el largo de dichas cadenas para cada una de las implementaciones realizadas. Para realizar dicha implementación se utilizan dos tipos de registros en las cadenas de scan paralelas, los registros seriales de desplazamiento y celdas d multiplexadas, para la

primera implementación se obtienen los siguientes datos.

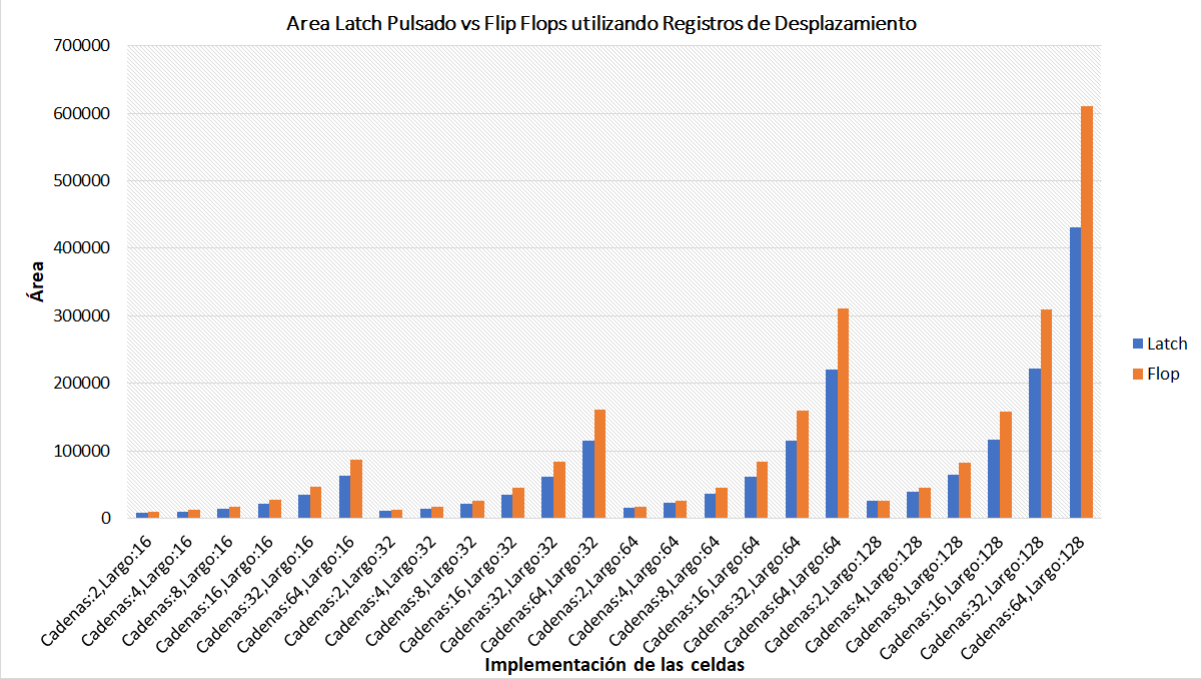

Figura 4.5: Gráfico de consumo de área para la implementación con registros de desplazamiento.

En el caso de la celda d multiplexada se obtiene.

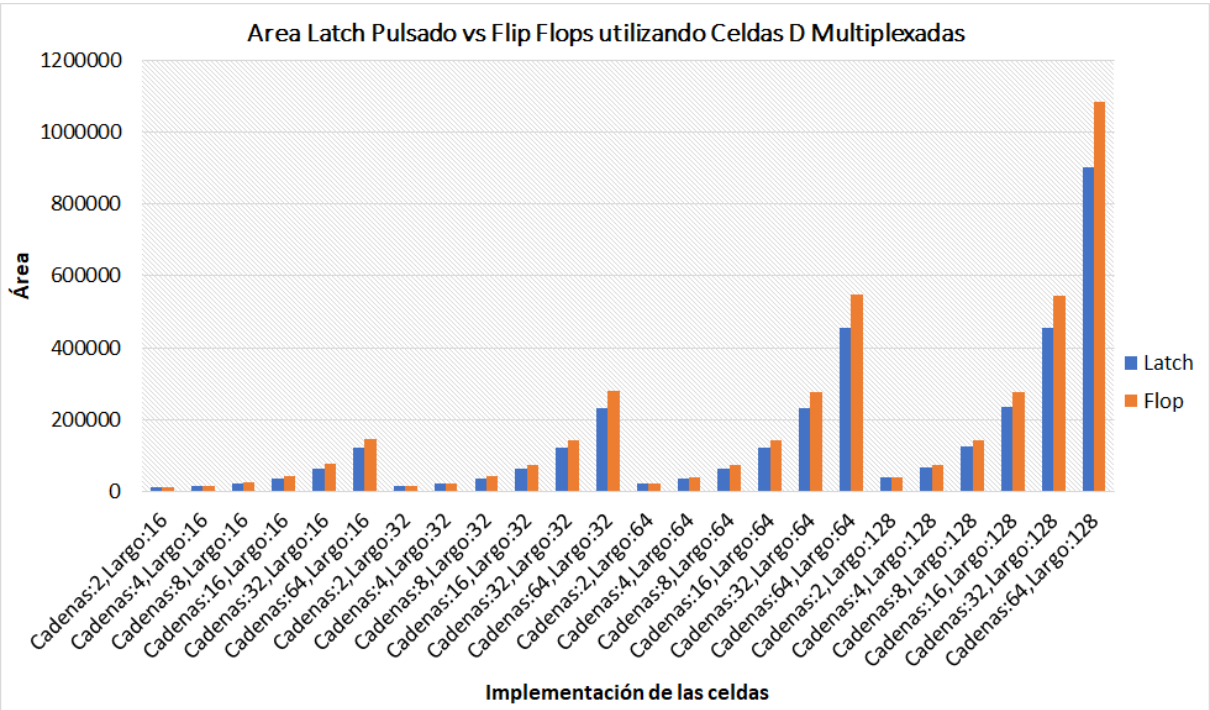

Figura 4.6: Gráfico de consumo de área para la implementación con celdas d multiplexada.

Los gráficos 4.5 y 4.6 muestran el crecimiento en área de las celdas conforme aumentan en longitud y se agregan más celdas en paralelo tiende a aumentar más rápidamente en

la implementación con flip flops que utilizando latches pulsados, básicamente entre mayor sea la longitud de las cadenas implementadas, mayor es la diferencia de área entre los latches y los flops cuando se aumenta la cantidad de cadenas utilizadas. Esto ocurre debido a que la lógica adicional del circuito generador de pulsos de reloj se distribuye entre los latches de las cadenas de scan en paralelo, por lo que al añadir cadenas de scan en paralelo se mejora el ahorro de área con respecto a la arquitectura que utiliza flops. Al añadir cadenas en paralelo, se bifurca el camino de los pulsos de reloj para que manejen varios bits a la vez, de esta forma es que se distribuye la lógica de generación de pulsos de reloj.

Cadenas conectadas	Largo (bits)	Detector flanco negativo	Detector flanco positivo	Total
2	16	9.5 %	0.2 %	9.7 %
2	32	15.2 %	0.2 %	15.4 %
2	64	21.2 %	0.1 %	21.3 %
2	128	26.2 %	0.1 %	26.5 %
64	16	1.3 %	0.03 %	1.33 %
64	32	1.4 %	0.02 %	1.42 %
64	64	1.5 %	0.009 %	1.51 %
64	128	1.6 %	0.005 %	1.61 %

Tabla 4.6: Porcentaje de área consumido por el circuito de reloj para la arquitectura de latches pulsados

La tabla 4.6 muestra que al aumentar la catidad de bits a lo largo de las cadenas de scan, se incrementa la lógica circuito generador de pulsos de reloj ya que es necesario un pulso por bit de la cadena de scan. Debido a que la arquitectura con flops no utiliza dicho circuito, la consecuencia inmediata es que entre más larga sea la cadena de scan, la diferencia entre el área consumida por la implementación con flops y la arquitectura con latches pulsados se va a ver disminuida.

Para realizar los reportes de área se utilizan dos topologías de registros de scan, una utilizando registros de desplazamiento y otra utilizando cadenas de Celdas D Multiplexadas, la diferencia entre ambas es el tamaño de los bits de la cadena de scan. La Celda D Multiplexada contiene un multiplexor a la entrada del circuito mientras que un registro de desplazamiento es una secuencia de flops o latches conectados en cascada.

La ganancia en área para la implementación con registros de desplazamiento se presenta en el gráfico 4.7.

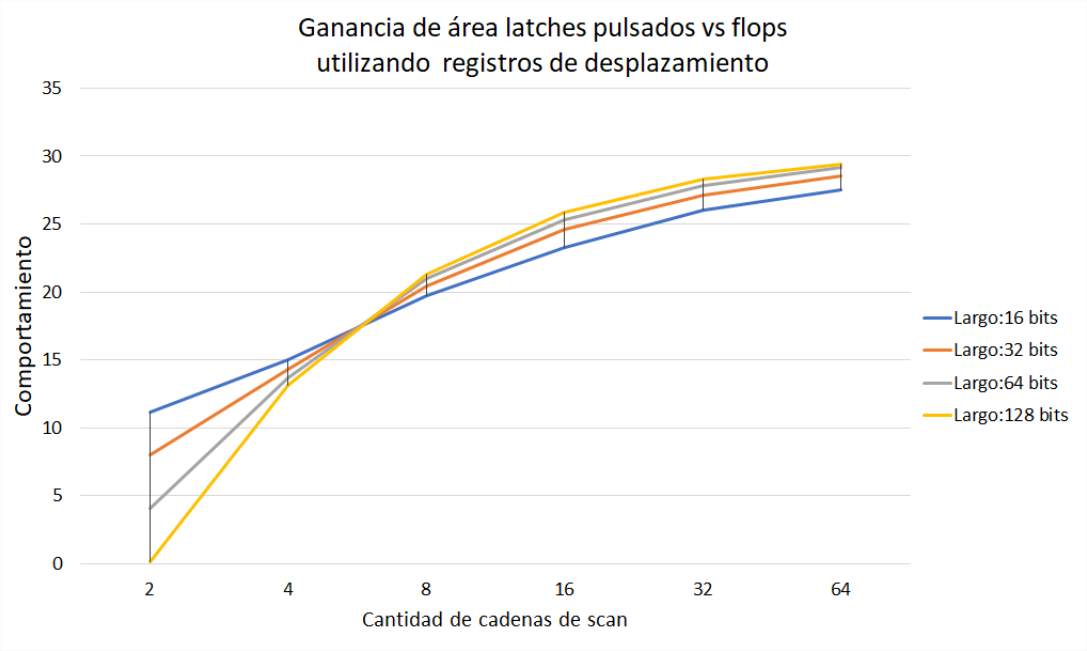

Figura 4.7: Diferencia de área implementado registros de desplazamiento.

Para el caso de la implementación con la celda d multiplexada se tiene el gráfico 4.8.

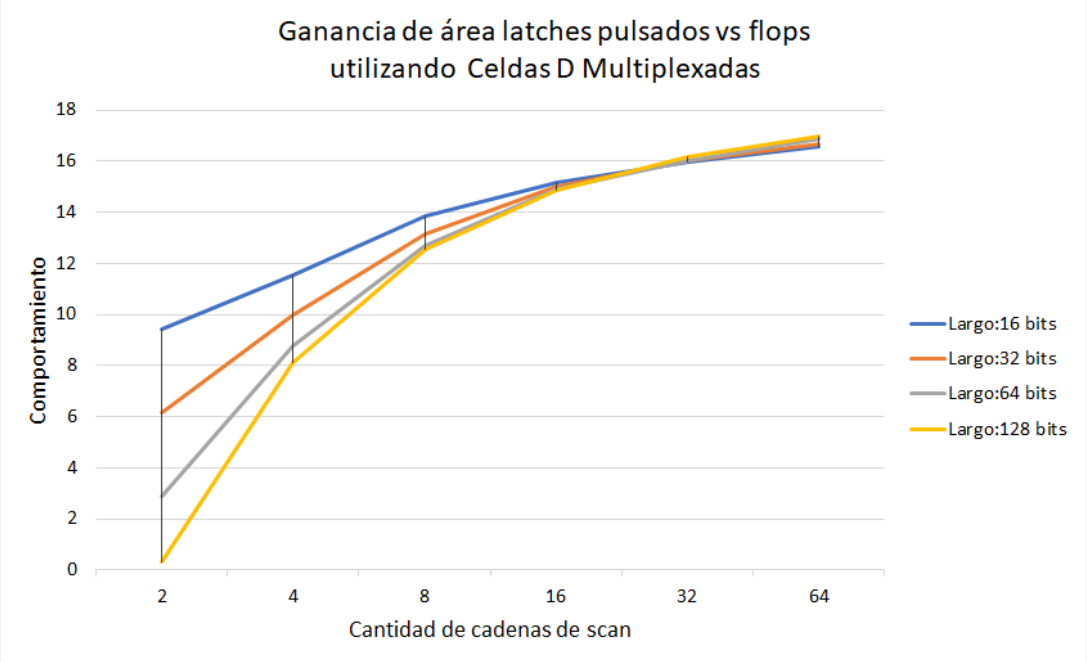

Figura 4.8: Diferencia de área implementado celdas d multiplexadas.

En el caso de la implementación de registros de desplazamiento la diferencia entre ambas celdas es de 32 % y en el caso de la celda d multiplexada es de 20 %, entonces se dice que

la ganancia en área utilizando la topología de latches pulsados contra la topología de flops es proporcional a la diferencia de tamaño que existe entre las cadenas de scan.

4.2. Reporte de consumo energético

El consumo energético se reporta de manera diferenciada entre la potencia total, la potencia de corto circuito y la potencia de conmutación. Para la implementación con latches pulsados se obtiene la tabla 4.7.

Cadenas de scan	Largo (bits)	Total (μW)	Conmutación (μW)	Corto circuito (μW)
2	16	2.81	0.00402	2.81
4	16	2.81	0.00486	2.81
8	16	2.82	0.00658	2.81
16	16	2.82	0.00995	2.81
32	16	2.83	0.0166	2.81
64	16	2.85	0.0304	2.81
2	32	2.81	0.00451	2.81
4	32	2.82	0.00624	2.81
8	32	2.82	0.00996	2.81
16	32	2.83	0.0184	2.81
32	32	2.85	0.0356	2.81
64	32	2.89	0.0689	2.81
2	64	2.82	0.00727	2.81
4	64	2.82	0.0113	2.81
8	64	2.83	0.0194	2.81
16	64	2.85	0.0361	2.81
32	64	2.89	0.0694	2.81
64	64	2.97	0.138	2.81
2	128	2.82	0.0115	2.81
4	128	2.83	0.0197	2.81
8	128	2.85	0.0377	2.81
16	128	2.89	0.0724	2.81
32	128	2.97	0.141	2.81
64	128	3.12	0.279	2.82

Tabla 4.7: Consumo de energía para la implementación con latches pulsados utilizando registros de desplazamiento como cadenas de scan

Para la implementación utilizando flip flops se obtiene la tabla 4.8.

Cadenas de scan	Largo (bits)	Total (μW)	Conmutación (μW)	Corto circuito (μW)
2	16	2.82	0.00768	2.81
4	16	2.82	0.0105	2.81
8	16	2.83	0.0166	2.81
16	16	2.85	0.0286	2.82
32	16	2.88	0.0506	2.82
64	16	2.94	0.098	2.84
2	32	2.82	0.01	2.81
4	32	2.83	0.0157	2.81
8	32	2.84	0.0278	2.81
16	32	2.87	0.0505	2.82
32	32	2.93	0.105	2.82
64	32	3.06	0.208	2.82
2	64	2.83	0.0172	2.81
4	64	2.84	0.0296	2.81
8	64	2.87	0.0543	2.81
16	64	2.93	0.105	2.82
32	64	3.04	0.206	2.82
64	64	3.28	0.422	2.84
2	128	2.85	0.0305	2.81
4	128	2.87	0.0557	2.81
8	128	2.93	0.11	2.82
16	128	3.05	0.216	2.82
32	128	3.27	0.426	2.83
64	128	3.74	0.86	2.85

Tabla 4.8: Consumo de energía para la implementación con flops utilizando registros de desplazamiento como cadenas de scan

La comparación entre consumo de energía total se muestra en el siguiente gráfico.

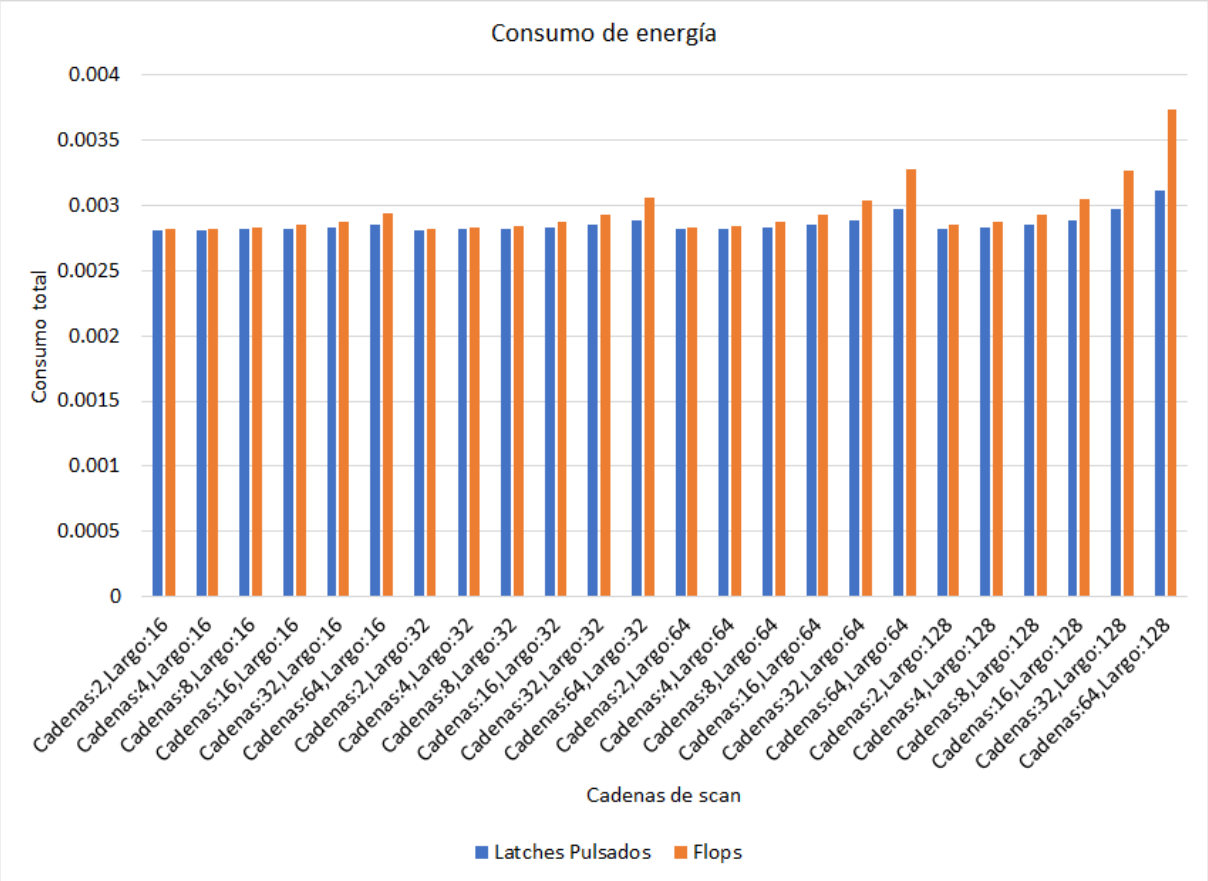

Figura 4.9: Comparación del consumo energético Latches Pulsados contra Flops.

El gráfico 4.9 permite observar el crecimiento del consumo energético entre ambas implementaciones, se observa que entre mayor sea la cantidad de cadenas de scan, mayor es el consumo de energía.

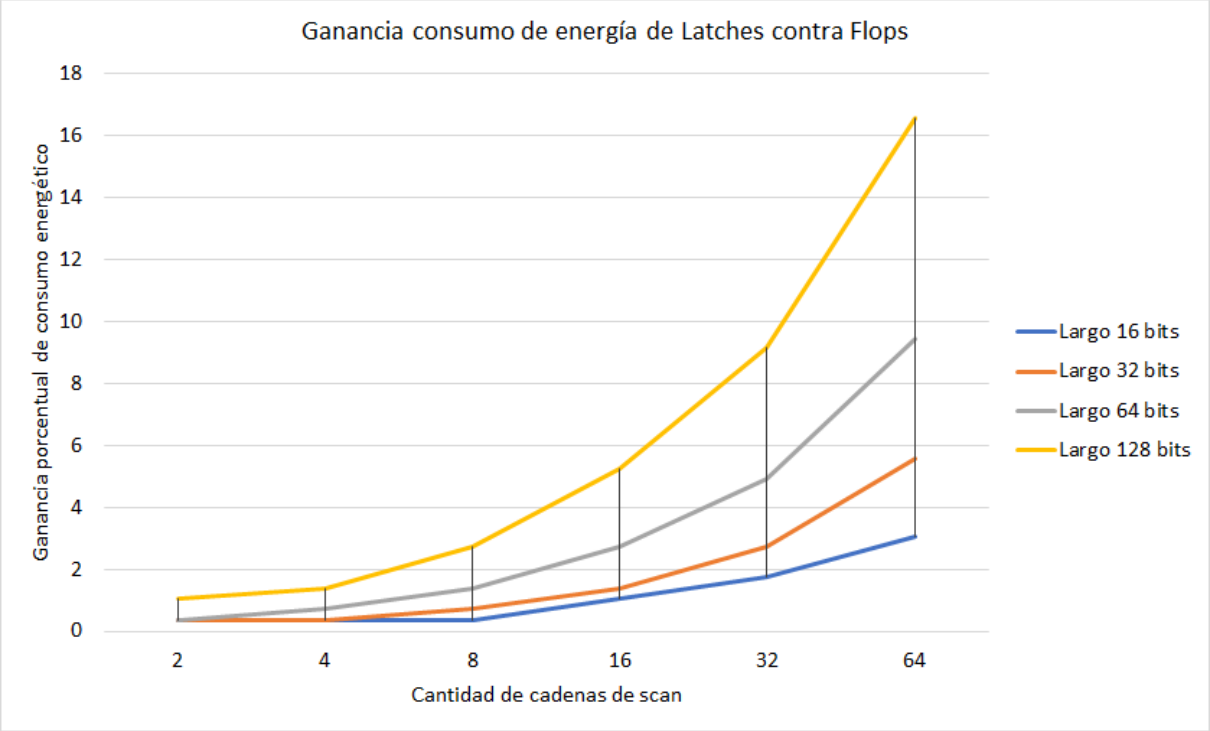

Figura 4.10: Ganancia en consumo energético Latches Pulsados contra Flops.

El gráfico 4.10, permite visualizar un comportamiento esperado, ya que entre mayor sea la longitud de las celdas y mayor sea la cantidad de cadenas de scan implementadas, mayor será el consumo de energía en ambas implementaciones, pero también mayor será la diferencia de consumo energético entre las topologías. Se observa como la tendencia se acentúa si se tiene una mayor longitud de las celdas, esto implica que la ganancia en consumo de energía es mayor en implementaciones con mayor cantidad de bits a lo largo de la cadena de scan que en topologías con menor cantidad de bits.

4.3. Frecuencia de operación

Si bien la implementación con latches pulsados permite ganar en área utilizada y en consumo energético, compromete la velocidad de trabajo, esto debido a que limita la frecuencia de operación debido al circuito generador de pulsos de reloj, ya que en una cadena de latches pulsados, es necesario generar un pulso de reloj por cada bit y como se observa en la figura 2.21, el periodo de la señal de reloj debe ser al menos del tiempo que le toma al circuito generador de pulsos de reloj, completar toda la cadena de pulsos.

El diseño de un registro de desplazamiento debe considerar la frecuencia de operación con la cual las cadenas de elementos secuenciales funcionan, dicha frecuencia se ve restringida por el tiempo de preparación y el tiempo de retención.

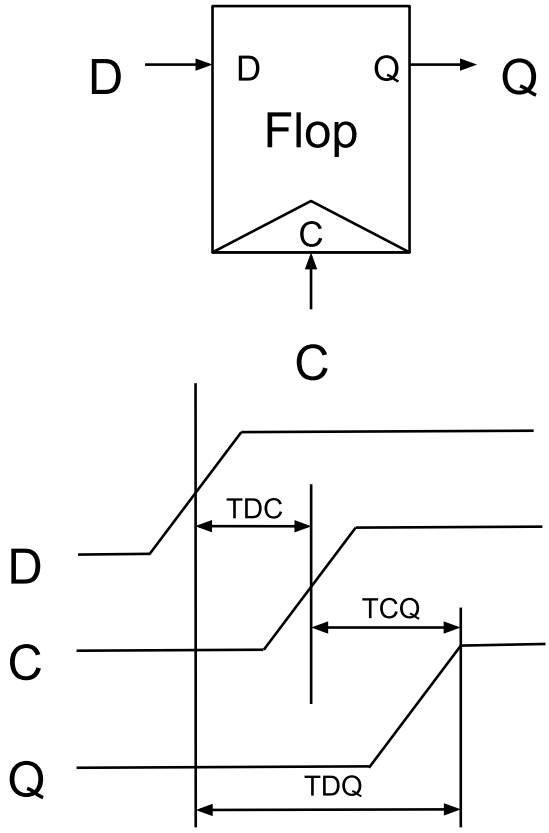

Figura 4.11: Tiempo de preparación y de retención.

La figura 4.11 utiliza un flip flop como ejemplo, pero el análisis es similar para el latch. El flip flop tiene tres tiempos característicos, T_{DQ} el cual es el tiempo de propagación de la señal desde la entrada hasta su salida, T_{DC}, es el tiempo de retraso entre la señal de entrada y la señal de reloj y T_{CQ} que es el tiempo de retraso entre la señal de reloj y la salida del circuito, en otras palabras $T_{DQ} = T_{DC} + T_{CQ}$, a partir de esta definición se establecen dos tiemposcuya referencia es la señal de reloj, el tiempo de preparación, que es el mínimo tiempo requerido para que la señal D se propague a la salida, está directamente relacionado con T_{DC} y el tiempo de retención, se define como el tiempo de retraso mínimo para que la salida conmute, tiene una relación muy estrecha con T_{CQ}, en otras palabras es el tiempo mínimo requerido para que la una vez capturado el dato en la entrada D sea propagado a la salida.
Para implementar un latch pulsado, durante el diseño del circuito generador de pulsos, es necesario considerar, que el tiempo de preparación y el tiempo de retención sumados es el periodo mínimo de la señal de reloj con que opera el latch, por lo que se sobredimensiona el tiempo entre pulsos para evitar que varios latches estén transparentes y evitar que la información se corrompa.

También hay que considerar otro factor relevante durante el proceso de diseño y es el visto en la figura 2.21, en otras palabras dependiendo de la cantidad de bits que se encuentren a lo largo del registro de desplazamiento, hay que generar un pulso por bit, y con la finalidad de no generar errores de traslape de datos o pérdida de información durante el desplazamiento es conveniente extender proporcionalmente el periodo de la señal de reloj. Por lo que una fórmula general para obtener el periodo de operación es.

$$P_{min} = (T_P + T_R) \cdot L \tag{4.1}$$

En dónde P_{min} es el periodo mínimo de operación, T_P es el tiempo de preparación de la señal, T_R es el tiempo de retención y L es la longitud de la cadena más larga a implementar.

Para el Latch, la biblioteca de las celdas estándar define:

Tiempo preparación (ps)	Tiempo de retención (ps)
20	20

Tabla 4.9: Tiempo promedio de retención y tiempo de preparación para el latch

En otras palabras el tiempo de propagación de la celda desde la entrada hasta su salida son 40 ps, y considerando el tamaño de la celda más grande se estima el periodo mínimo de operación de la señal de reloj.

Largo de la celda (bits)	Periodo (ps)
16	640
32	1280
64	2560
128	5120

Tabla 4.10: Periodo ideal de operación de la señal de reloj

Un flop está compuesto por dos latches, el tiempo de retraso del reloj a la entrada también es el mismo que el latch y del reloj a la salida conserva el mismo tiempo, por lo que el flip flop opera con frecuencias más altas que un latch pulsado, limitada por el tiempo de retención y el tiempo de preparación.
En el caso del Flop, la biblioteca de las celdas estándar define:

Tiempo preparación (ps)	Tiempo de retención (ps)
20	20

Tabla 4.11: Tiempo de retención y tiempo de preparación promedio para un flop

En las tablas 4.11 y 4.10 se observa que los tiempos de retención y preparación son el mismo, y considerando el funcionamiento del circuito generador de pulsos de reloj,

podemos observar que el periodo de operación mínimo del latch es más grande que el periodo de operación del flop entonces se dice que el latch es más lento que el flop.
Por lo que un circuito generador de pulsos debe considerar el tiempo de preparación y el tiempo de retención a la hora de implementarse como se observa en la figura 4.12.

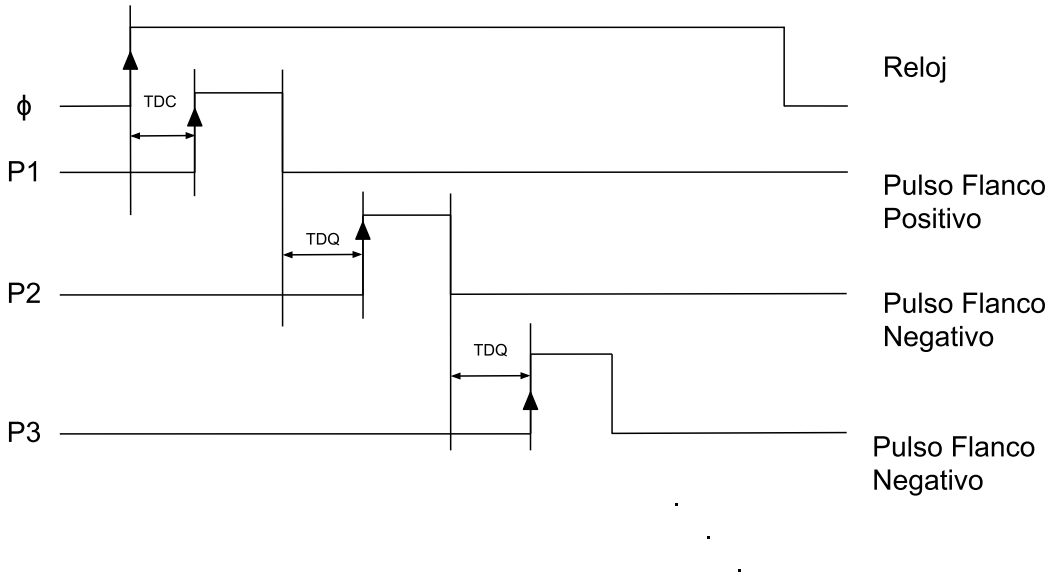

Figura 4.12: Pulsos generados por el circuito generador de pulsos de reloj.

El pulso positivo, se toma con el flanco positivo de la señal de reloj, se considera utilizar el retardo de compuerta debido a que también es de 20 ps, por lo que el puso se genera de la siguiente manera.

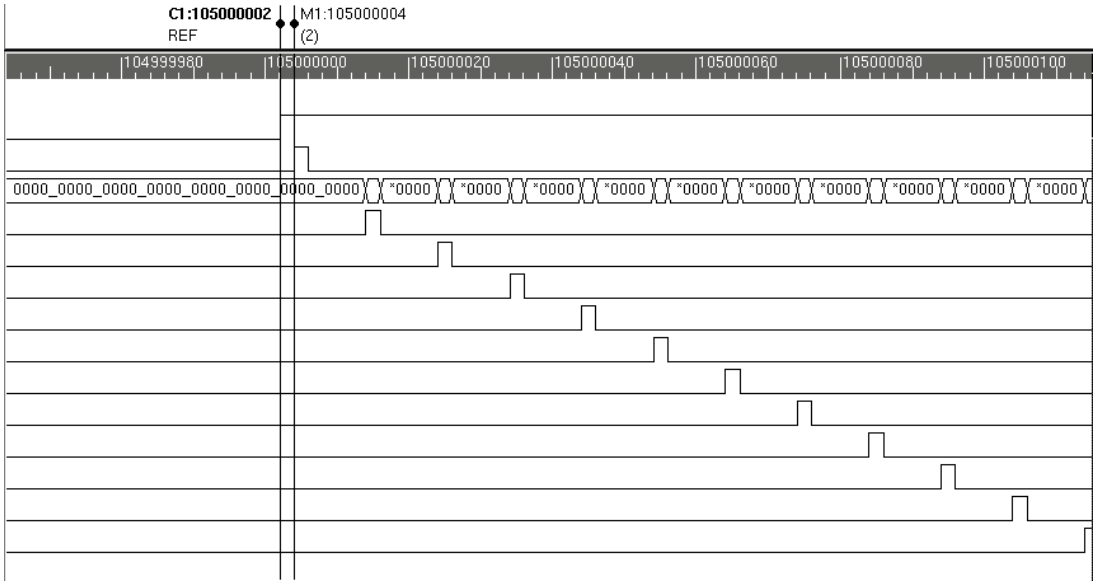

Figura 4.13: Pulso generado por el detector de flanco positivo.

En el caso del pulso negativo, el tiempo de preparación y de retención del latch son 20 ps, por lo que el periodo mínimo es de 40 ps, sobredimensionando el diseño el resultado

es un pulso que ocurre cada 80 ps, por lo que si cumple los requerimientos para que no se encuentren varios latches en transparencia durante una captura de datos.

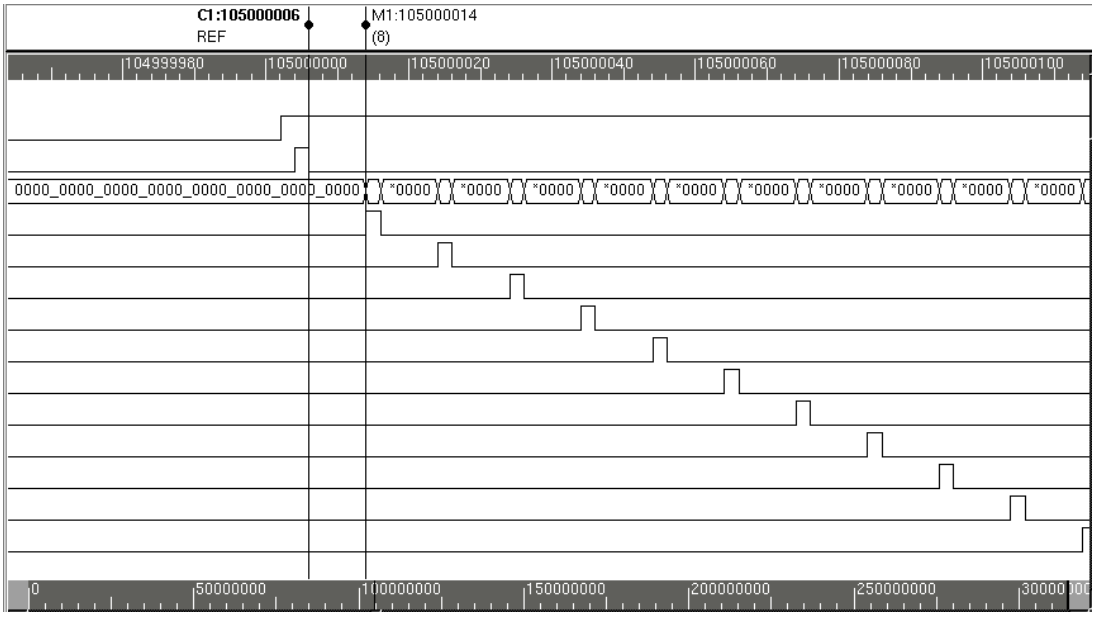

Figura 4.14: Pulse generado por el detector de flanco negativo.

Para una cadena de 128 bits, aplicando la fórmula [4.1] y considerando los datos obtenidos en las figuras 4.13 y 4.14 tenemos $(20 + 80) \cdot 128$, el periodo de operación máximo se encuentra en 12800 ps, para este caso el periodo simulado se encuentra en 12780, lo cual es muy cercano a la fórmula establecida.
Debido a la operación del circuito, la señal de reloj del TAP, se utiliza de manera lenta que la señal de reloj del DUT para evitar problemas de slack, esto es que la señal llegue mucho antes de que pueda ser capturada en la salida, además de que debido a que la biblioteca se va a implementar sobre un diseño de baja frecuencia, se utilizó una frecuencia de 20 MHz, esto equivale a un periodo de 500 ns.

Para observar mejor el incremento en el periodo mínimo de operación a la hora de implementar un latch pulsado, se utiliza la fórmula [4.1] se obtiene la tabla 4.12 variando la cantidad de bits de la cadena de scan.

Tabla 4.12: Velocidad de un latch pulsado

Largo de la cadena (bits)	Periodo mínimo de operación
16 bits	1600
32 bits	3200
64 bits	6400
128 bits	12800

Una gráfica generada a partir de los datos de la tabla 4.12 nos permite observar de mejor manera esta tendencia como se observa en la figura 4.15.

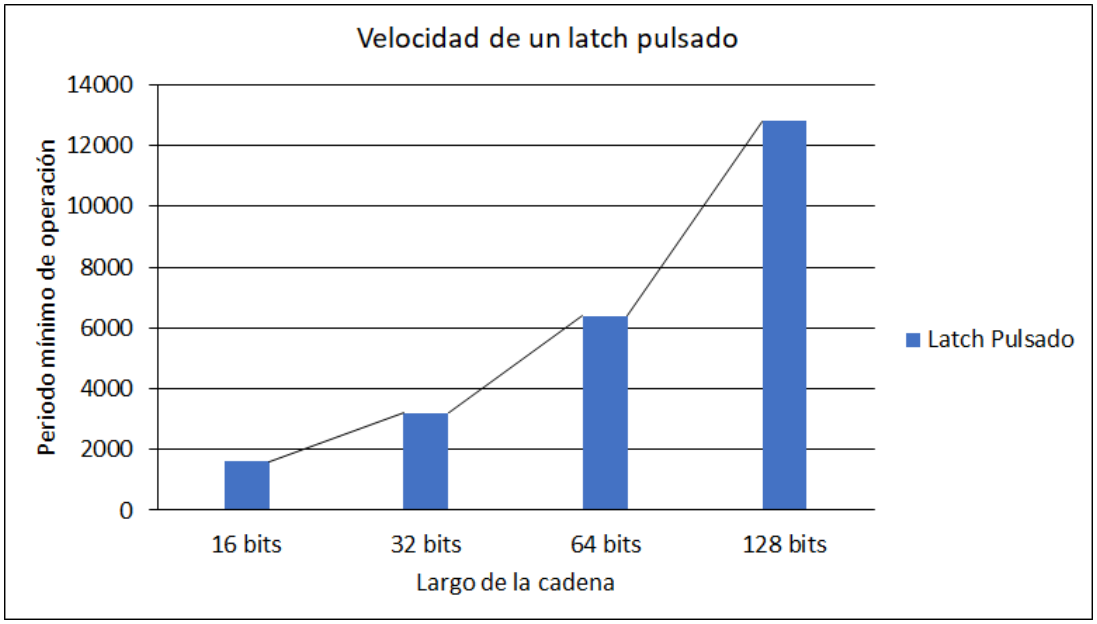

Figura 4.15: Aumento del periodo mínimo de operación para un latch pulsado.

Capítulo 5

Conclusiones

La ganancia en términos de área que se obtiene al implementar los latches pulsados para las cadenas de scan aumenta conforme se utilicen más cadenas de scan y conforme estas incrementen la cantidad de bits, esta ganancia es asintótica a la diferencia en área que existe entre la cadena implementada con flops y la cadena utilizando latches.

Entre mayor cantidad de bits se encuentre en las cadenas de scan mayor será la diferencia de consumo energético entre la arquitectura con flops y la arquitectura con latches pulsados, por lo que convierte a los latches pulsados en una mejor implementación si la meta es reducir el consumo energético.

La arquitectura implementada con latches pulsados tiene un frecuencia máxima de operación mayor que la arquitectura con base en el uso de flip flops, por lo que el compromiso de reducir el consumo energético y reducir el área utilizada conlleva una reducción en la velocidad de operación.

5.1. Recomendaciones

Para aumentar la ganancia en área aún más lo conveniente es implementar celdas personalizadas, con la finalidad de reducir el área de los latches y de los flops, dicho proceso se puede realizar utilizando la herramienta de Custom Compiler de Synopsys.

Realizar optimizaciones sobre la biblioteca hecha en HDL, esto implica reducir los módulos, por ejemplo, utilizar las señales «One Hot» generadas por el decodificador como habilitadores o controles de flujo.

Debido a la restricción de tiempo para realizar el proyecto, no se logró realizar la imple-

mentación con la biblioteca diseñada para validación, dicha biblioteca es proporcionada por el fabricante, pero para su implementación es necesario combinar dos herramientas de Synopsys, una llamada DesignWare y habilitando el modo de «Scan» para Design Compiler, con este proceso se habilita el uso de Flops, Latches y cualquier otra lógica secuencial o combinacional dimensionada para scan, por lo que es posible que se reduzca el tamaño de la implementación o realizar comparaciones más exhaustivas.

Realizar un análisis de esquinas para el diseño y los latches Pulsados, con el fin de obtener una caracterización de los latches pulsados para poder catalogar futuras fabricaciones además de tener una curva de rendimiento del diseño.

Realizar un proceso exhaustivo de verificación funcional que permita encontrar errores en la lógica, si es que los hay.

Realizar la síntesis necesaria de los árboles de buffers para los caminos con fan outs altos, los árboles se pueden desarrollar mediante el modo topográfico de la herramienta de síntesis, manualmente o utilizando herramientas de back-end.

Bibliografía

[1] J. Andrews. *IEEE Standard for Test Access Port and Boundary-Scan Architecture*. New York: The Institute of Electrical y Electronics Engineers, Inc., 2014.

[2] S. Chen y col. *Study on Design Method of Boundary-Scan Circuit Architecture Based on Verilog Language*. 2011.

[3] Sarah L. Harris y David M. Harris and. *Digital Design and Computer Architecture Arm Edition*. Morgan Kaufmann, 2015.

[4] P. P. Kenneth. *THE BOUNDARY-SCAN HANDBOOK SECOND EDITION Analog and Digital*. Kluwer Academic Publishers, 2002.

[5] Colin M. Maunder y Rodham E. Tulloss. *THE TEST ACCESS PORT AND BOUNDARYSCAN ARCHITECTURE*. IEEE Computer Society Press, 1990.

[6] Zainalabedin Navabi. *Digital System Test and Testable Design Using HDL Models and Architectures*. Springer, 2011.

[7] M. Tsukisaka, M. Imai y T. Nanya. *Asynchronous scan-latch controller for low area overhead DFT*. 2014.

[8] Laung-Terng Wang, Cheng-Wen Wu y Xiaoqing Wen. *VLSI Test Principles and Architectures*. 1ra. Morgan Kaufmann, 2006.

[9] Neil H. E. Weste y David M. Harris. *CMOS VLSI Design A Circuits and Systems Perspective*. New York: Addison-Wesley, 2010.

[10] Byung-Do Yang. *Low-Power and Area-Efficient Shift Register Using Pulsed Latches*. 2015.