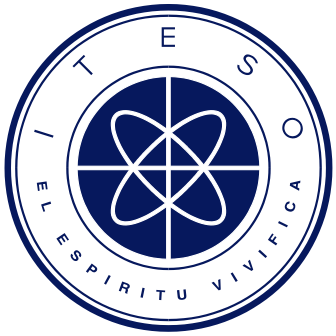
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

ESPECIALIDAD EN DISEÑO DE SISTEMAS EN CHIP



DISEÑO E IMPLEMENTACIÓN DE DIVISOR DE FRECUENCIA E INTEGRACIÓN DEL SoC CDR ADAPTATIVO A JITTER CON LFSR PARA PRUEBAS Y PLL INTERNO CON TECNOLOGÍA BiCMOS DE 130 nm

Tesina para obtener el grado de:

ESPECIALISTA EN DISEÑO DE SISTEMAS EN CHIP

Presenta: Francisco Javier Nuñez López

Director: Mtro. Cuauhtémoc Rafael Aguilera Galicia

San Pedro Tlaquepaque, Jalisco. Julio de 2018.

Agradecimientos

Quiero agradecer a todas aquellas personas que han sido parte fundamental en mi proceso de aprendizaje y que a su vez contribuyeron a la realización de este trabajo: compañeros, colegas, docentes y familia.

Principalmente a Conacyt por la confianza y el apoyo económico, sin el cual, nada de esto hubiera sido posible.

Y, por último, al ITESO por todas las facilidades prestadas a la ejecución de este trabajo.

Abstract

The Phase-Locked Loops are in a wide applications range such as wireless communication systems, digital circuits and hard disk drive electronics, generating blocked phase signals with external input signals. The phase-locked loops are composed of a phase frequency detector, a charge-pump, a loop filter, a voltage-controlled oscillator and a frequency divider in the feedback route. In this thesis, the design and the physical implementation, in BiCMOS8HP technology with a lithography process of 130 nm, of a frequency divider for the Phase-Locked Loop feedback is presented. This design is composed of asynchronous dividers and various multiplexers for the internal or external frequency references. The design flow of integrated circuits for specific applications used in this paper is described step by step, so that it can be replicated by any interested. During the flow, the used tools and files in each step, are presented. Besides, the applicated simulations at each stage are described in detail, showing the testbench code and the waveforms of the obtained results. At the end, the physical design and the prior verifications are showed.

Resumen

Los lazos de seguimiento de fase se encuentran en una amplia gama de aplicaciones tales como los sistemas de comunicaciones inalámbricas, circuitos digitales y electrónica de discos duros, generando señales de fase amarrada con señales de entrada externas. Los lazos de seguimiento de fase están compuestos de un detector de frecuencia de fase, una bomba de carga, un filtro de lazo, un oscilador controlado por voltaje y un divisor de frecuencia en la ruta de realimentación. En este trabajo se presenta el diseño y la implementación física en tecnología BiCMOS8HP con proceso de litografía de 130 nm de un divisor de alta frecuencia para la retroalimentación de lazo de seguimiento de fase, este diseño está compuesto por divisores asíncronos y diversos multiplexores para la selección de frecuencias externas o internas. El flujo de diseño de circuitos integrados para aplicaciones específicas implementado en este trabajo se describe paso a paso con detalle, para que el trabajo pueda replicarse por cualquier persona interesada. Durante el flujo se presentan las herramientas y archivos utilizados en cada uno de los pasos. Además del diseño, las simulaciones aplicadas en cada una de las etapas del diseño se describen a detalle mostrando el código de la cama de pruebas y las formas de onda de los resultados obtenidos. Al final se muestra el diseño físico realizado y las verificaciones realizadas previas a la generación de los archivos para su exportación.

Lista de Figuras

Contenido

Introducción

Este proyecto nace de la necesidad de generar un chip que ponga de manifiesto los conocimientos adquiridos durante la Especialidad de diseño de sistemas en chip.

El diseño engloba una serie de módulos, cada uno de ellos tiene una dependencia directa con al menos otro módulo, dando así la capacidad de generar un grupo de trabajo donde cada alumno desarrollará uno de estos módulos que, posteriormente, serán integrados para formar un único diseño.

El sistema aquí presentado consiste de los siguientes módulos: un lazo de seguimiento de fase o PLL (*Phase-Locked Loop*, por sus siglas en inglés), el cual proporciona ocho fases de reloj al siguiente módulo; un recuperador de reloj y datos o CDR (*Clock and Data Recovery*, por sus siglas en inglés) que se encarga de seleccionar una de esas ocho fases para recuperar datos provenientes de una entrada externa al módulo; el generador de números pseudo-aleatorios o LFSR (*Linear Feedback Shift Register*, por sus siglas en inglés), cuya finalidad es monitorear el correcto funcionamiento del CDR, sustituyendo la entrada externa del CDR por los datos generados internamente por el LFSR.

Otros módulos importantes son los contenidos dentro del PLL, que son de vital importancia para su funcionamiento adecuado. Dentro de estos módulos se encuentran el oscilador controlado por voltaje o VCO (*Voltage-controlled Oscillator*, por sus siglas en inglés) que convierte de un nivel de voltaje a una frecuencia determinada; el detector de frecuencia de fase o PFD (*Phase-frequency Detector*, por sus siglas en inglés), que determina si el PLL se encuentra operando a la frecuencia establecida de trabajo. Como en el caso de los módulos anteriores, este subconjunto también se dota de monitoreo, se añaden módulos divisores de frecuencia y algunos multiplexores; por medio de estos se pueden dirigir las señales a través de los módulos o evitar que pasen por los mismos, con la finalidad de determinar si los módulos base se encuentran trabajando de la manera correcta.

Para el desarrollo del diseño, se aplicó el flujo de diseño de circuitos integrados para aplicaciones específicas o ASIC (*Application-Specific Integrated Circuit*, por sus siglas en inglés). Para poder realizar el flujo de diseño se usaron diversas herramientas; ISE Simulator (ISim) de

Xilinx, Inc., Encounter RTL Compiler, NC-Verilog simulator, SimVision, Encounter Digital Implementation (EDI) System de Cadence Design Systems. Las celdas estándar fueron proporcionadas por GLOBALFOUNDRIES en tecnología BiCMOS8HP implementadas en un proceso de litografía de 130nm.

Aunado a todo esto, la meta concreta dentro de la especialidad es lograr la fabricación del chip diseñado. MOSIS, empresa dedicada a la fabricación de circuitos integrados, será la encarga de la fabricación. MOSIS presta sus servicios al ITESO gracias a convenios de colaboración anteriormente pactados, que favorecen la fabricación de chips didácticos y de investigación desarrollados por su comunidad docente y estudiantil.

1. Antecedentes

Hoy en día, la tendencia es desarrollar sistemas completos en un chip o SoC (*Systems on a Chip*, por sus siglas en inglés), como mencionan el Dr. T. Ananthapadmanabha et al., para integrar todas las funciones del sistema electrónico, incluidos los circuitos digitales y analógicos, en un solo dado de silicio. [1]

Para entender el desarrollo de un SoC debemos adentrarnos en el tipo de diseño que se puede usar y para esto nos ayuda D. Clein, que en su artículo define a los usuarios los diversos tipos de *full-custom layout* (diseño totalmente a medida). Clein presenta cuatro variantes distintas del diseño *full-custom layout* como una gama de posibilidades:

- Full-custom layout impulsado por limitaciones de área o necesidades especiales de aplicación.
- Full-custom layout para abordar el alto rendimiento o el diseño de circuitos analógicos.
- Full-custom layout que requiere mayor atención al área y rendimiento que el flujo digital.
- Full-custom layout para el desarrollo de celdas.

De estas cuatro ramas presentadas, se toman como pilares dos de ellas. Full-custom layout orientada al diseño de circuitos analógicos, y que incluye PLL, uno de los módulos contenidos en el chip. Y Full-custom layout para el desarrollo de celdas; los ejemplos incluirían las celdas dentro de una biblioteca de celdas estándar. [2]

Ya mostrados los métodos de diseño que se van a seguir, se pasa a dar sustento de otro de los módulos del chip, en este caso el LFSR. Los sistemas digitales se prueban aplicando estímulos apropiados y verificando las respuestas. La generación de tales estímulos, junto con el cálculo de sus respuestas esperadas, se llama generación de patrones de prueba. Los patrones de prueba se generan en la práctica mediante una herramienta de generación automatizada; por lo general, se aplican al circuito utilizando equipos de prueba automática. [3]

Los estímulos y patrones de prueba serán generados por el LFSR. Estas prácticas de DFT (*Design-for-testability*, por sus siglas en inglés) generalmente se conocen como autocomprobación incorporada BIST (*Built-in self-test*, por sus siglas en inglés). La técnica de autocomprobación incorporada BIST, por otro lado, implementa todos los recursos de prueba dentro del chip. De ahí la necesidad del desarrollo de dicho módulo. [3]

Existe una necesidad cada vez mayor de transmisión y recuperación de datos digitales [4]. Explotando esa necesidad se incluye en el diseño del chip un módulo CDR para realizar la función de recuperación de datos y reloj dentro del sistema.

Se ha escogido el CDR, pues como lo sustenta P. S. Corp. en su publicacion [4]. Un elemento central de cualquier sistema de transmisión y recuperación de datos es la recuperación del tiempo (o el reloj) de la información digital [4]. El proceso de recuperación de datos y reloj se realiza típicamente a través del uso de un sistema de control de realimentación como un lazo de seguimiento de fase [4], o como también se le conoce un PLL, lo cual lleva al siguiente pilar de este diseño.

Los lazos de seguimiento de fase o PLL son circuitos funcionales que generan señales de fase bloqueada con señales de entrada externas [1]. En la arquitectura general del sistema transceptor, la síntesis de frecuencia a partir de una frecuencia de referencia fija (es decir, desde un oscilador de cristal) es un bloque esencial. Debido a su mejor rendimiento de ruido, en comparación con otras soluciones y disponibilidad de selección de canal mediante la modulación de relación de división de bucle, el PLL es la elección natural para dicha aplicación [5].

Un PLL puede contener diversos elementos en implementación, pero existen algunos básicos e inamovibles en todo diseño. Diferentes referencias ilustran este concepto. Para Ananthapadmanabha los PLL generalmente están compuestos de un detector de frecuencia de fase, una bomba de carga, un filtro de paso bajo, una palanca de cambios de nivel, un oscilador controlado por voltaje y un divisor de frecuencia en la ruta de realimentación [1]. Analog Devices menciona como bloques básicos del PLL, el Detector de errores (compuesto por un detector de frecuencia de fase y una bomba de carga), Filtro de bucle, VCO y un Divisor de realimentación [6]. Los cuatro componentes básicos para Texas Instruments de un circuito PLL son el VCO, el detector de frecuencia de fase, el filtro de bucle y los divisores principal y de referencia [7].

Todos ellos coinciden en la importancia de los divisores de frecuencia, uno de los objetos de estudios de este trabajo. Los divisores constituyen una función principal en los circuitos PLL. Un circuito PLL necesita cubrir una amplia gama de divisiones continuas para la referencia de cristal y para el VCO, es lo que dice Texas Instruments respecto a los divisores de frecuencia [7]. Francesco Barale et al. mencionan como parte crítica de cada PLL la cadena divisora de

retroalimentación, que implementa la división de frecuencia de la salida del oscilador controlado por voltaje (VCO) para que sea igual a la frecuencia de la señal de referencia [8].

Un divisor de frecuencia digital se implementa comúnmente con un flip-flop de alternancia (T-FF) que consiste en un D-FF con la salida negada realimentada a la entrada D. El T-FF es una solución simple; sin embargo, está limitada a las proporciones de división de potencia-de-2 [8]. Esta implementación es el ancla del diseño para el divisor de frecuencia; si bien es simple, cumple con todos los requerimientos del SoC CDR adaptativo a jitter con LFSR para pruebas y PLL interno y puede ser fácilmente modificable para adecuarse a las diferentes proporciones de división demandadas.

2. Marco Teórico

Los lazos de seguimiento de fase (PLL) encuentran una amplia aplicación en áreas de comunicaciones, inalámbricas, sistemas, circuitos digitales y electrónica de discos duros. Dos factores explican esta tendencia: la demanda de mayor rendimiento y menor costo en los sistemas electrónicos, y el avance de las tecnologías de circuito integrado (IC) en términos de velocidad y complejidad. [9]

2.1. ¿Por qué hacer uso de un lazo de seguimiento de fase?

Consideramos cuatro problemas de diseño que pueden resolverse eficientemente con la ayuda de PLL como se menciona en [9].

2.1.1. Reducción de jitter

Las señales a menudo experimentan fluctuación de tiempo cuando viajan a través de un canal de comunicación o cuando se recuperan de un medio de almacenamiento. El *jitter* se manifiesta como una variación del período de una forma de onda, un tipo de corrupción que no se puede eliminar mediante amplificación y recorte, incluso si la señal es binaria. Se puede usar un PLL para reducir el jitter.

2.1.2. Supresión de skew

Un reloj del sistema, CKs, ingresa a un chip desde una placa de circuito impreso (PCB) y se almacena temporalmente (en varias etapas) para reducir el tiempo de subida y bajada sus transiciones y controlar la capacidad de carga con un mínimo de retraso. La principal dificultad en tal disposición es que el reloj en chip, CKc, típicamente impulsa varios nanofaradios de capacitancia, exhibiendo un retraso significativo con respecto a las CKs. El *skew* resultante reduce el presupuesto de tiempo para operaciones en chip e inter-chip. Para reducir el skew, la memoria intermedia del reloj puede colocarse en un PLL, alineando así CKc con CKs.

2.1.3. Síntesis de frecuencia

Muchas aplicaciones requieren la multiplicación de frecuencia de señales periódicas. Por ejemplo, en el caso anterior, la limitación del ancho de banda de las placas de PCB restringe la frecuencia del CKs, mientras que la frecuencia del reloj en chip puede necesitar ser mucho mayor. Esto ejemplifica la necesidad de la "síntesis de frecuencia", una tarea que se puede realizar de manera eficiente utilizando sistemas de fase bloqueada.

2.1.4. Recuperación de reloj

En muchos sistemas, los datos se transmiten o recuperan sin ninguna referencia de temporización adicional. En las comunicaciones ópticas, por ejemplo, una corriente de datos fluye sobre una única fibra sin reloj asociado, pero el receptor debe procesar los datos de forma sincrónica. Por lo tanto, la información de temporización (por ejemplo, el reloj) debe recuperarse de los datos en el extremo de recepción. La mayoría de los circuitos para recuperación de reloj emplean un PLL.

Estos cuatro problemas son, precisamente, requerimientos que deben ser cubiertos por el SoC CDR adaptativo a jitter con LFSR para pruebas y PLL interno desarrollado en el proyecto. Dado que todos esos requerimientos (posteriormente expuestos en detalle) son solucionados gracias a la implementación del PLL, es de vital importancia describirlo detalladamente y mostrar los elementos que lo componen.

2.2. Los lazos de seguimiento de fase (PLL)

La implementación de la síntesis de frecuencia en una aplicación multi GHz, se realiza a menudo conectando un oscilador controlado por voltaje (VCO) en un bucle, hasta el punto de amarrar la fase de la frecuencia de salida a una señal de frecuencia de referencia de entrada. El motivo es que no existen medios prácticos para generar una frecuencia de referencia pura (en términos de ruido de fase) y estable (libre de variaciones) por encima de unos pocos GHz. Tal sistema de realimentación, capaz de amarrar una salida del VCO de alta frecuencia a una señal de

referencia de frecuencia mucho más baja, se denomina lazo de seguimiento de fase (PLL). La disposición típica para un sistema PLL se muestra en la Figura 2-1. [5]

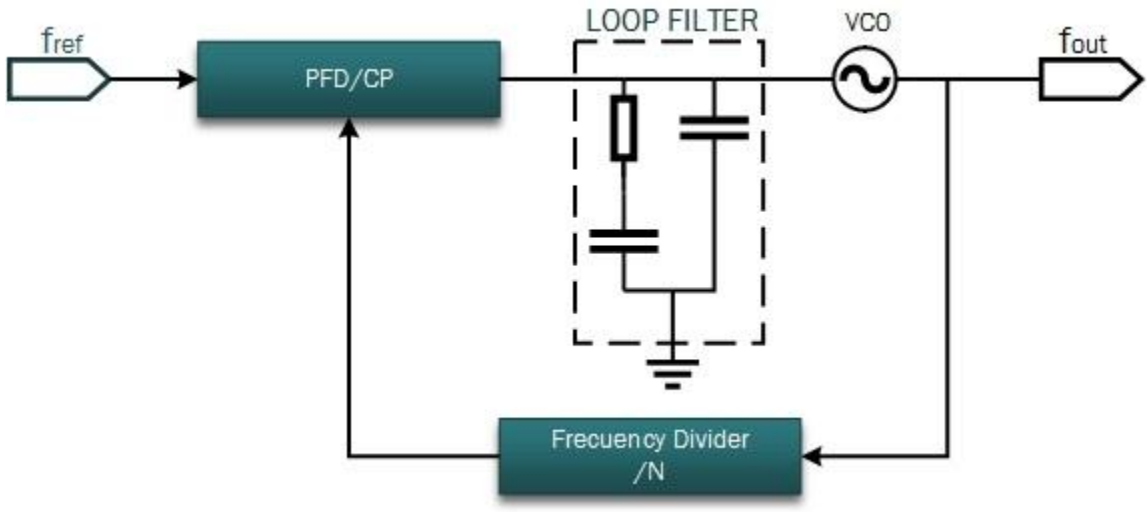


Figura 2-1. Disposición típica para un sistema PLL. [5]

Para describir el comportamiento de un lazo de seguimiento de fase PLL, a menudo se usa un modelo linealizado. Además, hasta el punto de enfatizar el mecanismo de amarre de fase, el dominio de la frecuencia se elige para el análisis. Para continuar con el análisis, cada bloque de construcción del PLL necesita modelarse en el dominio de la frecuencia. La descripción de estos bloques se toma de [5].

El detector de frecuencia de fase o PFD (*Phase-frequency detector*, por sus siglas en inglés) se puede interpretar como un bloque funcional con señal de salida proporcional a la diferencia de fase entre las dos señales de entrada. El factor de proporcionalidad se establece en 1 (sin unidades).

La bomba de carga o CP (*Charge-pump*, por sus siglas en inglés) se puede modelar como un bloque transconductor, transformando la diferencia de fase en una corriente proporcional. El factor de proporcionalidad k_ϕ se define como en (2-1).

$$\begin{cases} k\phi = \dfrac{I_p}{2\pi} \qquad \left[\dfrac{\mu A}{rad}\right] \\ I_p : \ máxima\ corriente\ de\ salida\ del\ charge-pump \end{cases} \tag{2-1}$$

El bloque, después del CP, es el filtro de lazo: su propósito es integrar los pulsos de corriente generados por los bloques PFD/CP. La función de transferencia del filtro de lazo tiene las unidades de impedancia (Ω), siendo el voltaje de salida generado a partir de la corriente de salida del CP.

En el dominio de la frecuencia, el VCO se puede modelar como un integrador que tiene un factor de proporcionalidad de k_{vco}. Las unidades del factor de proporcionalidad son (Hz / V).

Cada bloque divisor de frecuencia que realiza una división de frecuencia por un factor de N, también dividirá la fase de la señal de entrada por N. Por lo tanto, un bloque divisor de frecuencia puede modelarse con un simple bloque divisor-por-N.

El diagrama de bloques resultante para el modelo del PLL linealizado se muestra en la Figura 2-2. [5]

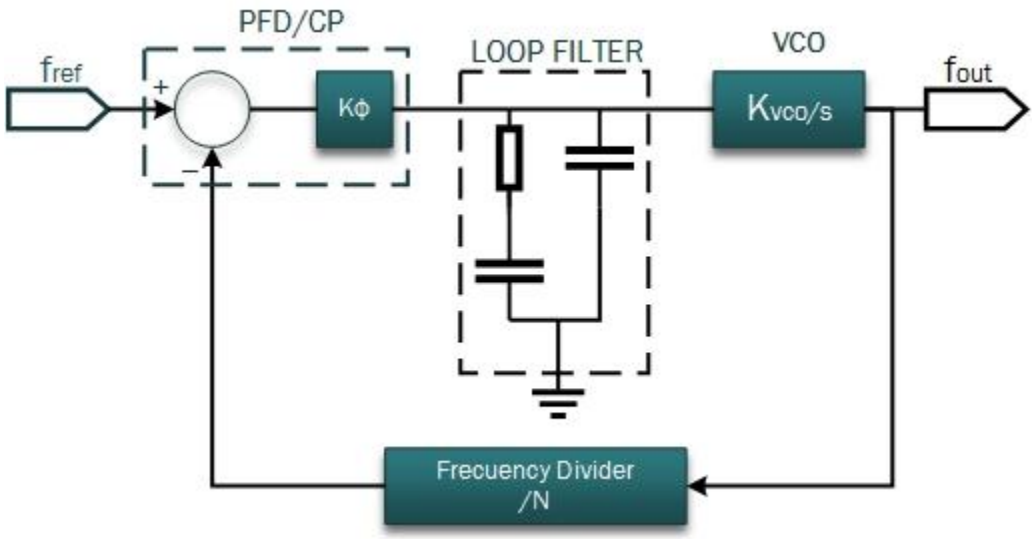


Figura 2-2. Modelo linealizado del PLL. [5]

2.3. Divisor de frecuencia

El divisor de frecuencia, componente del PLL, y otro de los objetivos de análisis de esta tesina.

2.3.1. Divisores de frecuencia básicos

En su implementación más básica, los divisores de frecuencia están compuestos básicamente por compuertas lógicas y *flip-flops*. Se pueden agrupar en tipos síncronos y asíncronos, según cómo se realice la sincronización de estos flip-flops. En los divisores síncronos, cada flip-flop se activa mediante la señal de entrada del divisor (reloj). Por otro lado, en los divisores asíncronos, la señal de entrada del divisor alimenta el primer flip-flop, que activa el segundo y así sucesivamente. Por lo tanto, los divisores de frecuencia sincrónicos logran una transición completa más rápido que los asincrónicos. [10]

La Figura 2-3 (a) muestra un divisor asincrónico de tres etapas. Como se describe en [10], cada etapa consiste en un divisor por dos, cuya salida es la entrada de la siguiente etapa. Esto hace que la tercera etapa sea asíncrona con respecto a la entrada de reloj.

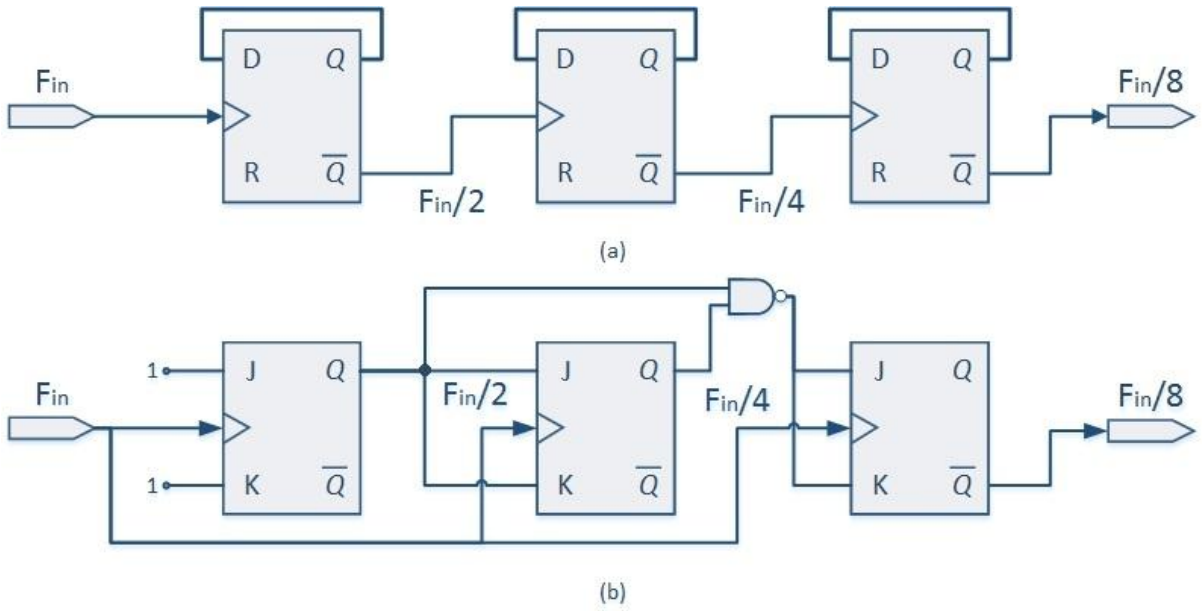


Figura 2-3. Divisores de frecuencia (a) asíncronos y (b) síncronos. [10]

Si las salidas de las tres etapas se combinan junto con la entrada en una compuerta “AND”, la señal de salida en esta puerta se sincronizará con el reloj (con un pequeño retraso causado por la puerta). La principal desventaja de los divisores asincrónicos es la acumulación de la fluctuación (parámetro de tiempo equivalente del ruido de fase) de una etapa a la otra. Esto se puede reducir colocando un flip-flop de sincronización al final de la cadena. En este caso, la fluctuación de la salida es solo la generada por el sincronizador. [10]

Los divisores síncronos se pueden implementar usando flip-flops- JK, como se ilustra en el ejemplo dado en la Figura 2-3 (b). Cada etapa cambia casi simultáneamente en los bordes del reloj. Esto hace que la última etapa responda más rápidamente que en el caso del divisor asíncrono que se muestra en la Figura 2-3 (a). Se debe tener especial cuidado cuando las salidas intermedias de cada una de las etapas se combinan en cualquier puerta lógica AND. Esto se debe al hecho de que pueden aparecer fallas técnicas no deseadas cuando cualquiera de las señales cambia más rápido o más lentamente que las otras. [10]

Aunque estos fundamentos podrían usarse para lograr muchas reducciones de frecuencia diferentes, el divisor de frecuencia básico puede sufrir limitaciones para las operaciones de alta frecuencia debido a la presencia de transistores PMOS y a su propósito de enfoque digital. En esos casos, el sistema debe incluir más bloques de construcción para acomodar la señal a las limitaciones de estos divisores. [10]

Dado las características del PLL interno del SoC CDR adaptativo a jitter con LFSR para pruebas y PLL interno, se implementarán divisores asíncronos dentro del lazo de retroalimentación, además se incluirán arreglos de divisores para lograr diversos factores de división, las especificaciones de los divisores y distintos factores de división se presentan la sección **4.1. Especificaciones del diseño**.

Como lo destacan en [11], un mal funcionamiento en el divisor de realimentación de una respuesta transitoria de lazo puede ocasionar que un PLL se pierda y suspenda el voltaje de control de VCO.

La recuperación de esta condición requiere una señal de reinicio o de control de encendido para reiniciar el circuito. Un divisor de retroalimentación que es más rápido que el VCO no requiere una señal de reinicio o de control de encendido porque nunca se pierde. En consecuencia, la frecuencia de salida más alta del VCO en un PLL determina el requisito de velocidad para el

divisor de realimentación. Este es otro de los requerimientos que condicionan el diseño del divisor de frecuencia. [11]

Por lo general, un flip-flop "divisor-por-2" tiene una respuesta de alta frecuencia y no limita las aplicaciones de PLL. Para las frecuencias que estén por encima de esta respuesta, puede que se tengan que diseñar flip-flops especiales [11]. Es por todo esto que se optó por una topología de divisor asincrónico como diseño base para lograr un buen desempeño.

3. Metodología

El diseño de ASIC se basa en un flujo que utiliza un lenguaje de descripción de hardware o HDL (*Hardware Description Language*, por sus siglas en inglés) como nivel de entrada, para este diseño se hizo uso de Verilog. La siguiente descripción en la Figura 3-1 detalla el flujo desde la especificación del diseño hasta la salida de un GDSII, que es el formato de archivo que se envía a la fundición de silicio para su fabricación.

El flujo se divide en dos grandes partes:

3.1. *Front-End Design*

Front-End Design incluye especificaciones del diseño, arquitectura del diseño, descripción comportamental, verificación funcional, descripción a Nivel de Transferencia de Registros o RTL (Register-Transfer Level, por sus siglas en inglés), compilación y verificación RTL y síntesis lógica.

3.2. *Back-End Design*

Back-End Design incluye todo el diseño físico, así como *partitioning*, *floor-planning*, *placement*, síntesis de árbol de reloj, *signal routing and timing closure*.

Cada uno de los elementos del flujo de diseño es descrito a continuación en un capítulo específico.

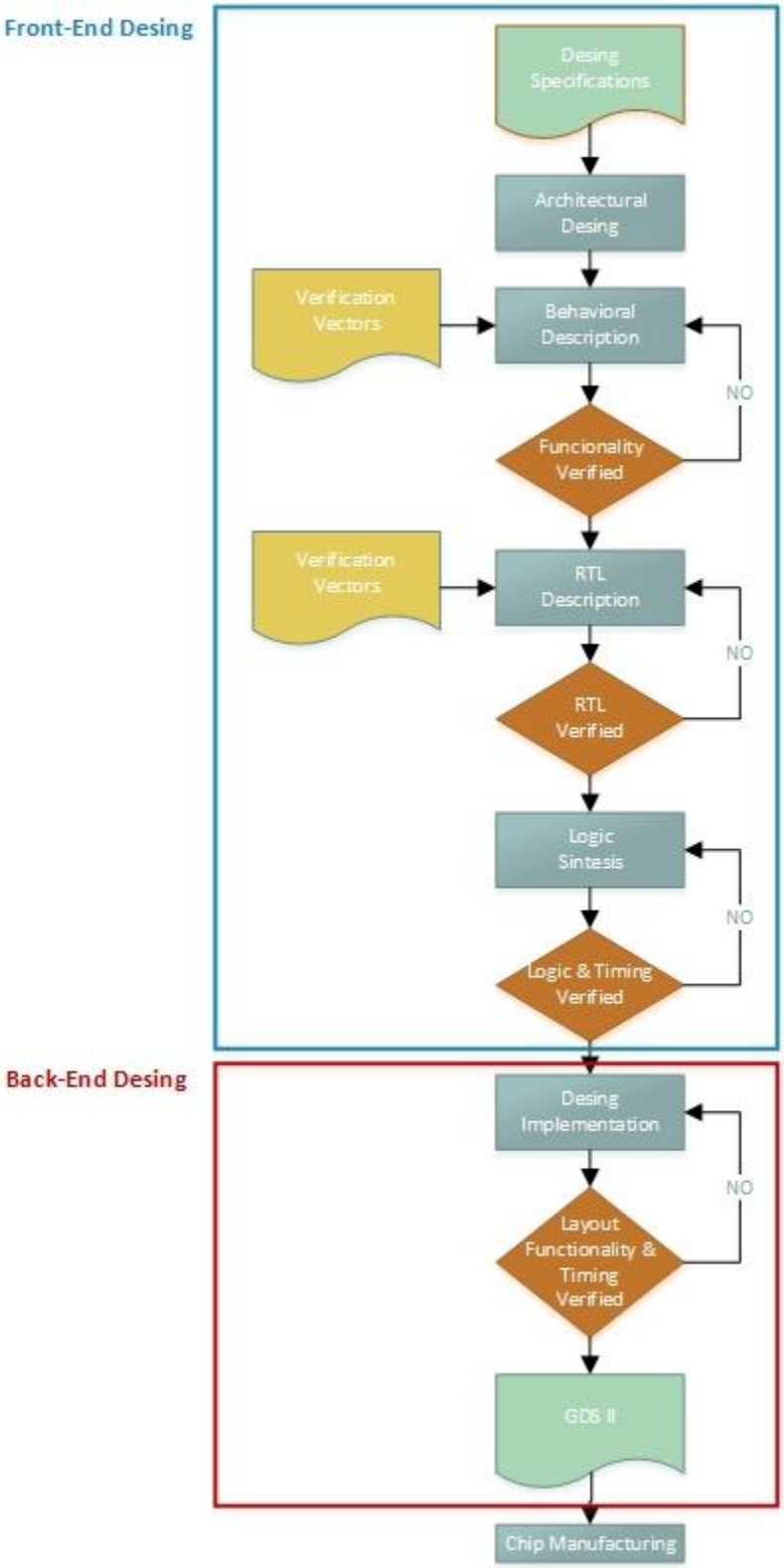


Figura 3-1. Flujo de diseño de circuitos integrados para aplicaciones específicas (ASIC).

4. Desarrollo del módulo Divisores de retroalimentación y multiplexores de frecuencia

4.1. Especificaciones del diseño

El módulo Divisores de retroalimentación y multiplexores de frecuencia diseñado en esta tesina debe cumplir con una serie de funcionalidades que se detallan a continuación:

- Tomar la frecuencia de 800 MHz – 1.3 GHz generada por el VCO y dividirla ya sea por un factor de división de 8 o 16 para, posteriormente, ser retroalimentada al PFD del PLL.
- Tomar la frecuencia de 800 MHz – 1.3 GHz generada por el VCO y generar la frecuencia de operación de 200 MHz del LFSR.
- Poder seleccionar entre una Frecuencia de Retroalimentación externa o la generada a partir de la división de la frecuencia del VCO.
- Generar una serie de frecuencias a partir de los divisores con factores de división de 2, 4 y 8 para uso externo.
- Poder seleccionar si esa serie de frecuencias será generada a partir de la frecuencia del VCO o una Frecuencia externa.

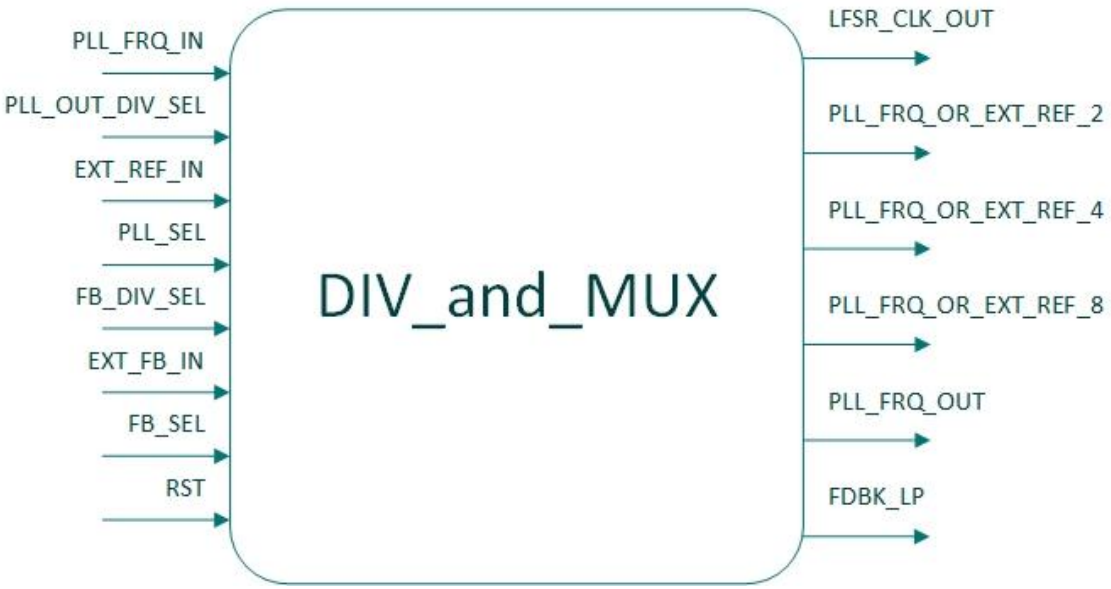


Figura 4-1. Interfaz del módulo Divisores de retroalimentación y multiplexores de frecuencia.

La Figura 4-1 muestra la interfaz del módulo necesaria para cumplir con esas funcionalidades. A continuación, se describen cada una de las entradas y salidas de dicha interfaz. La frecuencia de salida del PLL generada por el VCO es introducida al módulo por medio de la entrada PLL_FRQ_IN. Esta entrada alimenta una serie de multiplexores y divisores de frecuencia, además de estar conectada a una salida PLL_FRQ_OUT, que es una manera de observar desde el exterior del chip el funcionamiento del VCO. PLL_OUT_DIV_SEL es el selector del multiplexor que permite tener la frecuencia del VCO, o bien la VCO/2. EXT_REF_IN se direcciona a otro multiplexor controlado por PLL_SEL, que está conectado a una serie de divisores que generan la serie de frecuencias de las que se habló antes, las cuales están presentes en las salidas PLL_FRQ_OR_EXT_REF_2, PLL_FRQ_OR_EXT_REF_4 y PLL_FRQ_OR_EXT_REF_8. En el puerto de entrada EXT_FB_IN se introduce una frecuencia que puede ser utilizada en lugar de la frecuencia del VCO. FB_SEL permite seleccionar entre estas dos frecuencias. FB_DIV_SEL controla los factores de división en el lazo de retroalimentación, pudiendo seleccionar dos factores diferentes. FDBK_LP es la salida que se conecta al PFD del PLL para brindar la frecuencia de retroalimentación para este último. LFSR_CLK_OUT es conectada al LFSR como frecuencia de operación y por último RST, señal de entrada para el reinicio de toda la lógica secuencial que conforma el módulo.

4.2. Arquitectura del diseño

El hardware se compone de 9 módulos que, a través de su interacción, cumplen con las especificaciones de hardware mencionadas en la sección anterior. Estos módulos se seleccionaron de esta forma debido al análisis de las arquitecturas analizadas anteriormente en la sección **2.3.1. Divisores de frecuencia básicos**. El diagrama que ilustra su interacción se presenta en la Figura 4-2, donde se muestran los elementos de la arquitectura, mismos que serán explicados en la sección **4.3. Descripción comportamental** para dejar clara su inclusión y funcionamiento particular.

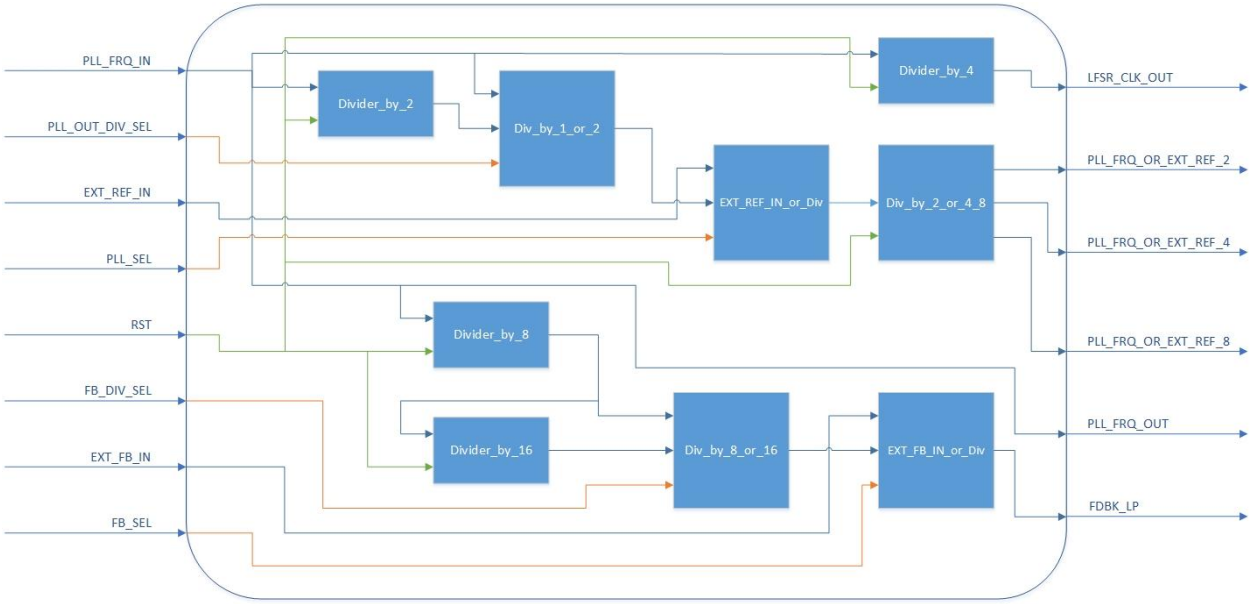


Figura 4-2. Arquitectura del módulo Divisores de retroalimentación y multiplexores de frecuencia.

4.3. Descripción comportamental

4.3.1. Módulo Divisor de frecuencia entre 2

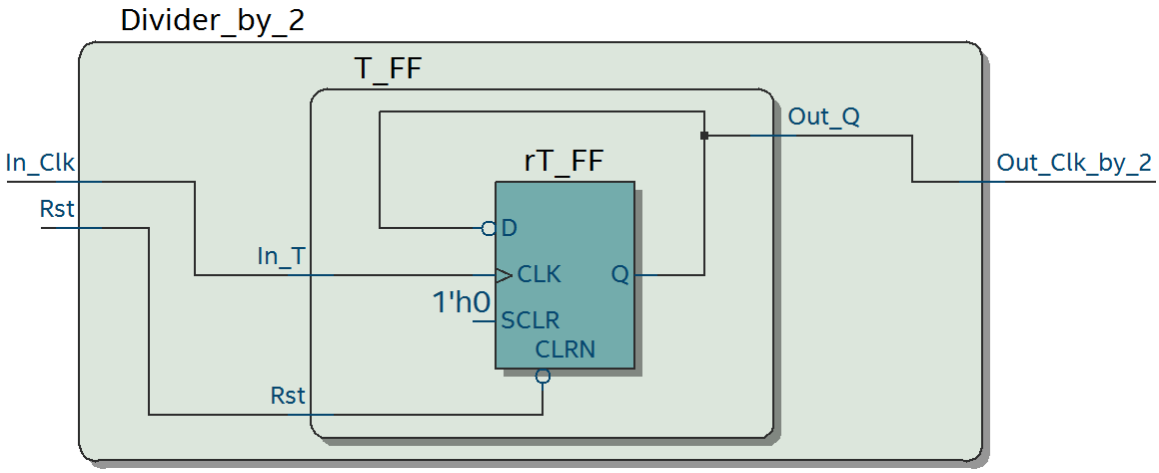


Figura 4-3. Arquitectura del módulo Divider_by_2.

El módulo Divider_by_2 es el responsable de tomar la frecuencia generada por el VCO presente en la entrada PLL_FRQ_IN del módulo Top y entregar una señal de la mitad de frecuencia. Como se pude apreciar en la Figura 4-3, el módulo está compuesto por una instancia

del módulo T_FF, el cual como se menciona en la sección **2.3.1. Divisores de frecuencia básicos**, es un divisor asíncrono, base de todos los divisores de este diseño.

Otro de los divisores de frecuencia presentes en el lazo de retroalimentación del PLL es el módulo **Divisor de frecuencia entre 16**. Toma la frecuencia generada por el módulo Divisor de frecuencia entre 8 ya que el objetivo es generar una señal de un dieciseisavo de la frecuencia del VCO a la salida y como la entrada ya es un octavo de la frecuencia del VCO, solo se divide por un factor de 2 y se logra el objetivo, reduciendo el número de divisores totales.

4.3.2. Módulos Selectores de frecuencia

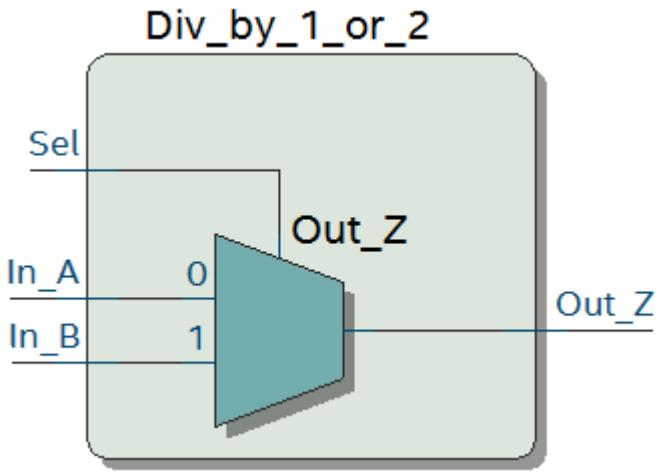


Figura 4-4. Arquitectura de los módulos Selectores de frecuencia.

Como se pude apreciar en la Figura 4-4, estos módulos están compuestos por un multiplexor 2 a 1. Se instanciaron cuatro de estos módulos para diversas tareas, a continuación, se detallan cada uno de ellos:

Div_by_1_or_2, en una de sus entradas está presente la frecuencia del VCO por medio de la entrada PLL_FRQ_IN del módulo Top y en otra está conectada la salida del módulo anterior (Divider_by_2). A la salida del cual se puede seleccionar la frecuencia del VCO o la frecuencia del VCO/2, por medio del selector PLL_OUT_DIV_SEL.

EXT_REF_IN_or_DIV, en una de sus entradas está presente una frecuencia externa por medio la entrada EXT_REF_IN del módulo Top y en otra está conectada la salida del módulo

anterior (Div_by_1_or_2). A la salida se puede seleccionar la frecuencia del VCO o la frecuencia del VCO/2 debido al funcionamiento del módulo anterior, o la ya mencionada frecuencia externa, por medio del selector PLL_SEL.

Div_by_8_or_16 encargado del factor de división de frecuencias en el lazo del PLL; en una de sus entradas está presente la frecuencia de **Divider_by_8 (**VCO/8) y en otra de sus entradas esta la frecuencia de **Divider_by_16 (**VCO/16). A la salida se puede seleccionar la frecuencia del VCO/8 o la frecuencia del VCO/16, por medio del selector FB_DIV_SEL.

EXT_FB_IN_or_DIV, se pueden retroalimentar en el lazo las frecuencias generadas internamente en el Top o una frecuencia externa. En una de sus entradas están presentes las frecuencias que puede seleccionar el módulo **Div_by_8_or_16** y en otra esta una frecuencia externa por medio la entrada EXT_FB_IN del módulo Top. A la salida se puede seleccionar la frecuencia del VCO/8 o la frecuencia del VCO/16 por medio del módulo anterior o una frecuencia de retroalimentación externa, por medio del selector FB_SEL.

4.3.3. Módulo Divisor de frecuencia entre 2, 4 u 8

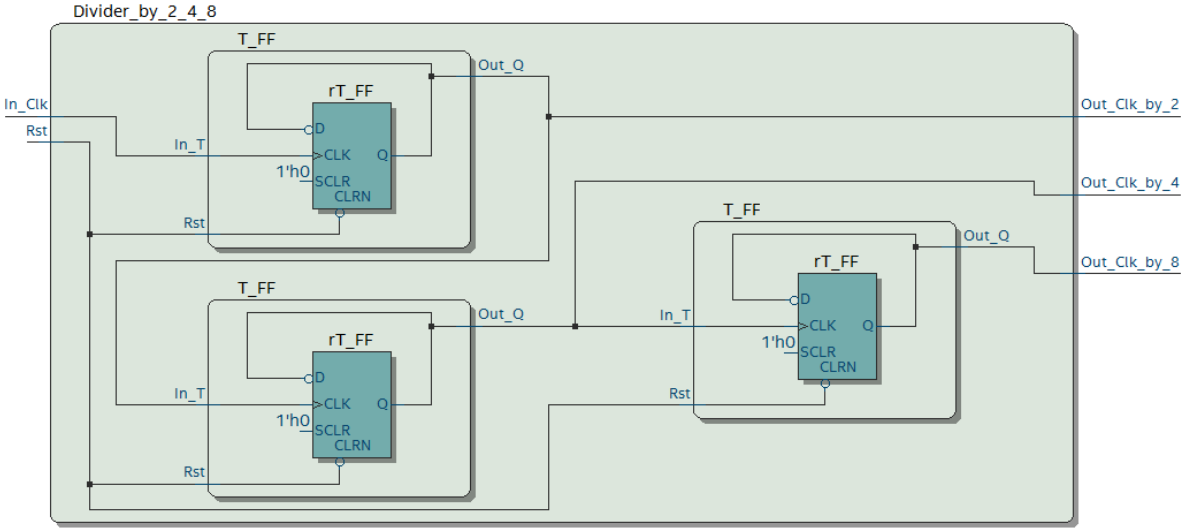


Figura 4-5. Arquitectura del módulo Divider_by_2_4_8.

El módulo Divider_by_2_4_8 genera una serie de frecuencias para uso externo. Al estar conectado a los módulos anteriores se pueden tener a la salida una variedad de frecuencias útiles

para el monitoreo del funcionamiento del módulo Top. La Figura 4-5 muestra la cadena de divisores que componen el módulo de este apartado. Cada uno de los divisores de esta cadena divide en dos la frecuencia otorgada por su antecesor, y como resultado de todo lo anterior se pueden tener a la salida frecuencias como: la frecuencia del VCO con factores de división de 2, 4, 8 y 16 o una frecuencia externa con factores de división de 2, 4 y 8.

4.3.4. Módulo Divisor de frecuencia entre 4

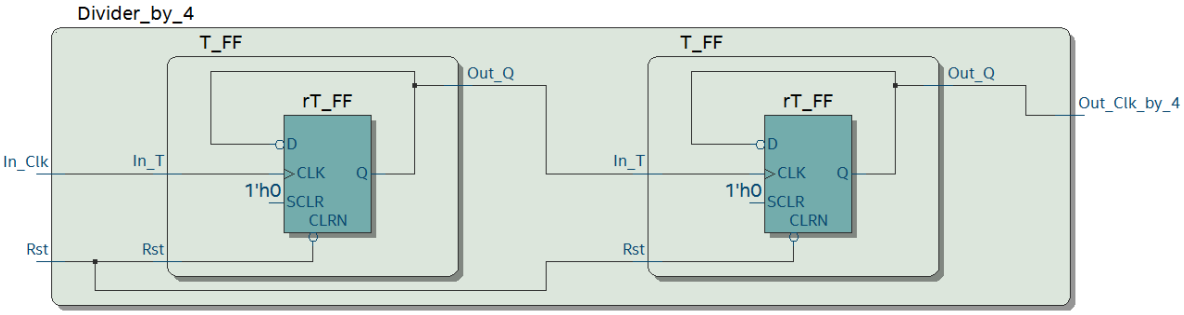


Figura 4-6. Arquitectura del módulo Divider_by_4.

El módulo Divider_by_4 es el generador de la frecuencia de operación del LFSR. Toma la frecuencia generada por el VCO presente en la entrada PLL_FRQ_IN del módulo Top y así entregar una señal de un cuarto de frecuencia de entrada. La Figura 4-6 muestra la composición del bloque. Usando de nuevo la base de todos los divisores logra su funcionamiento mediante una cadena de dos divisores, cada uno dividiendo en dos la entrada precedente.

4.3.5. Módulo Divisor de frecuencia entre 8

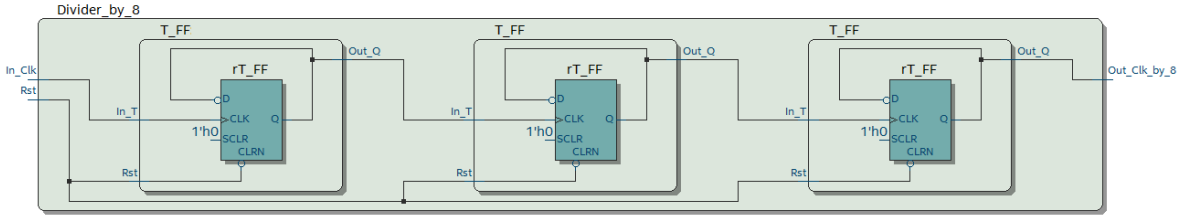


Figura 4-7. Arquitectura del módulo Divider_by_8.

Uno de los divisores de frecuencia presentes en el lazo de retroalimentación del PLL es el módulo Divider_by_8. Toma la frecuencia generada por el VCO presente en la entrada PLL_FRQ_IN del módulo Top y entregar una señal de un octavo de frecuencia de entrada. La Figura 4-7 muestra la composición del bloque, usando de nuevo la base de todos los divisores logra su funcionamiento mediante una cadena de tres divisores, cada uno dividiendo en dos la entrada precedente.

4.4. Descripción RTL

La microarquitectura a nivel de especificación en este punto del flujo de diseño se transformó como se muestra en la Figura 4-8 en una descripción RTL, para así dar comienzo a la fase del diseño donde se realiza la síntesis lógica. Como se espera para el diseño de un chip, debe ser un código RTL sintetizable, este código puede ser consultado en el Apéndice A.

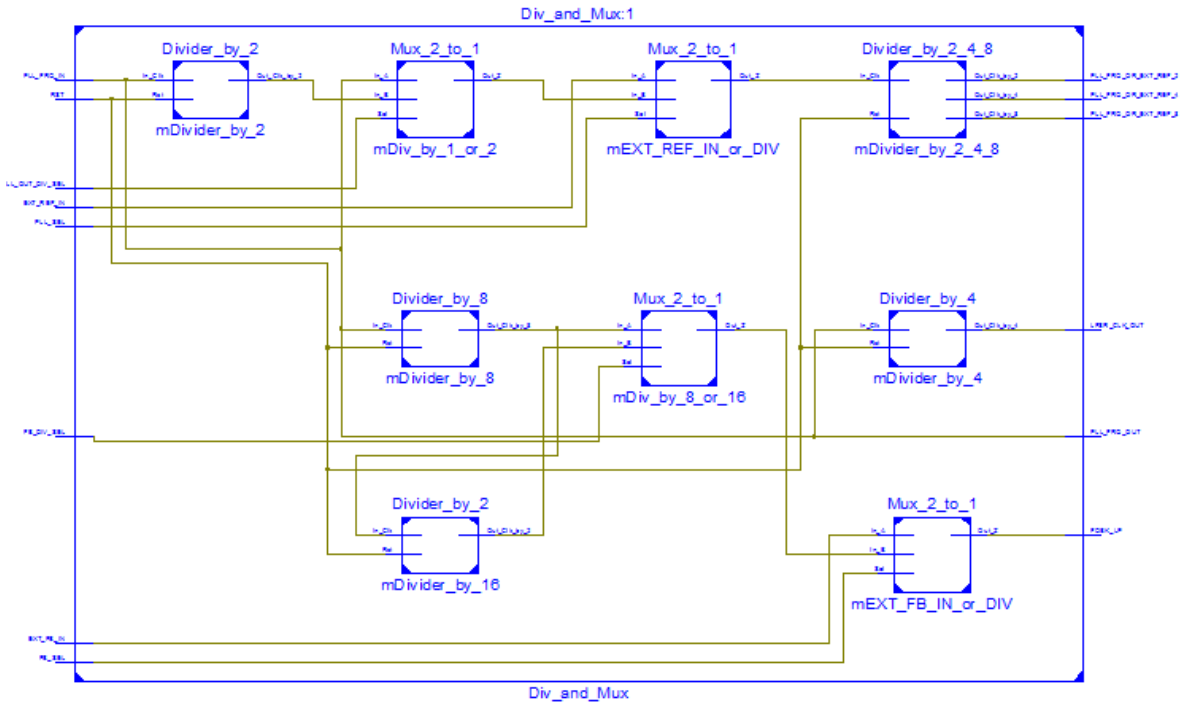


Figura 4-8. Descripción RTL del diseño.

4.5. Verificación RTL

El código RTL y la cama de pruebas o *testbench* (disponible en el Apéndice B) se verificaron usando simuladores HDL para comprobar la funcionalidad del diseño. La herramienta disponible en este paso fue ISE Simulator (ISim) de Xilinx, Inc.

De acuerdo con las especificaciones presentadas en la sección **4.1. Especificaciones del diseño**, se presenta una serie de imágenes que ilustran cada una de las camas de pruebas realizadas para verificar dichas especificaciones.

En la Figura 4-9 se aprecia como a los 20 ns de simulación los módulos **Divider_by_8** y **Divider_by_16** salen del estado de reset y comienzan a dividir la frecuencia del VCO presente en la entrada PLL_FRQ_IN, al estar los selectores FB_DIV_SEL en "0" y FB_SEL en "1" en la salida FDBK_LP se tiene la frecuencia del VCO dividida en un factor de 8, a los 150 ns de simulación se cambia el valor de FB_DIV_SEL a "1" y la salida FDBK_LP se tiene la frecuencia del VCO dividida en un factor de 16, verificando que la frecuencia generada por el VCO es dividida para, posteriormente, ser retroalimentada al PFD del PLL. A los 280 ns de simulación se cambia el valor de FB_SEL a "0" y la salida FDBK_LP se tiene la frecuencia externa presente en la entrada EXT_FB_IN verificando la selección entre una frecuencia de retroalimentación externa o la generada a partir de la división de la frecuencia del VCO.

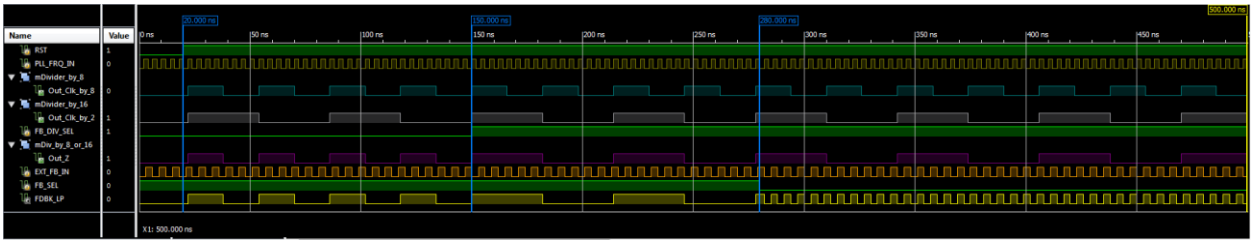


Figura 4-9. Simulación división de frecuencia del VCO y selección de retroalimentación.

En la Figura 4-10 se aprecia como a los 20 ns de simulación el módulo **Divider_by_4** sale del estado de reset y comienza a dividir la frecuencia del VCO presente en la entrada PLL_FRQ_IN, verificando que la frecuencia generada por el VCO es dividida y se genera la frecuencia de operación del LFSR. Adicionalmente se puede ver la salida PLL_FRQ_OUT, que es una manera de observar desde el exterior del chip la frecuencia del VCO.

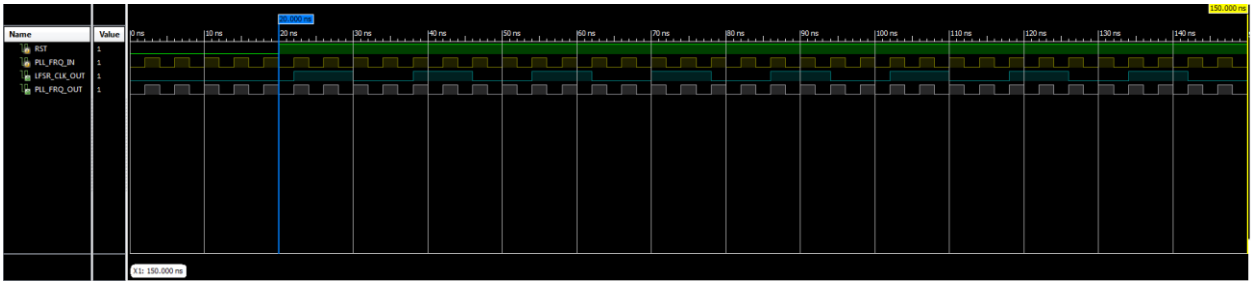


Figura 4-10. Simulación generación de frecuencia de operación del LFSR.

Por ultimo en la Figura 4-11 se aprecia como a los 20 ns de simulación los módulos **Divider_by_2** y **Divider_by_2_4_8** salen del estado de reset y comienzan a dividir la frecuencia del VCO presente en la entrada PLL_FRQ_IN, al estar los selectores PLL_OUT_DIV_SEL en "0" y PLL_SEL en "1" en las salidas PLL_FRQ_OR_EXT_REF_2, PLL_FRQ_OR_EXT_REF_4 y PLL_FRQ_OR_EXT_REF_8 se tiene la frecuencia del VCO dividida en un factor de 2, 4 y 8 respectivamente, a los 150 ns de simulación se cambia el valor de PLL_OUT_DIV_SEL a "1" y la salidas antes mencionadas tienen la frecuencia del VCO dividida en un factor de 4, 8 y 16, verificando que la frecuencia generada por el VCO es dividida para, posteriormente, ser dividida por el módulo **Divider_by_2_4_8**. A los 280 ns de simulación se cambia el valor de PLL_SEL a "0" y las salidas PLL_FRQ_OR_EXT_REF_2, PLL_FRQ_OR_EXT_REF_4 y PLL_FRQ_OR_EXT_REF_8 tienen la frecuencia externa presente en la entrada EXT_REF_IN dividida en un factor de 2, 4 y 8 respectivamente, verificando la selección entre una frecuencia de retroalimentación externa o la generada a partir de la división de la frecuencia del VCO.

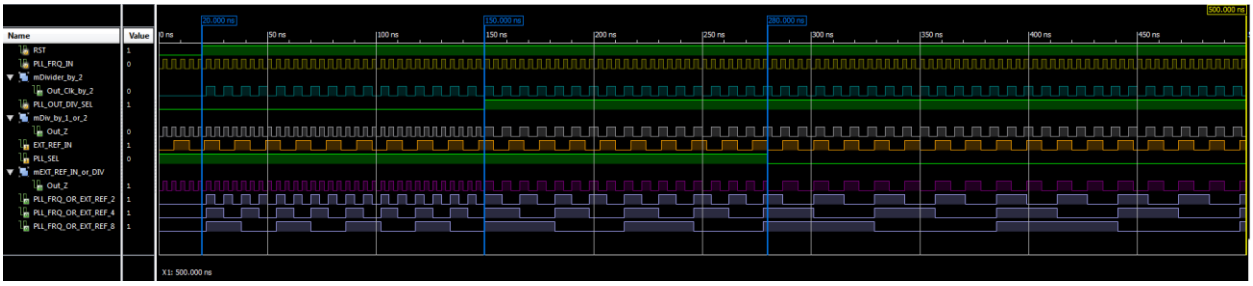


Figura 4-11. Simulación generación y selección de frecuencias para uso externo.

Los resultados de la simulación concordaron con la función esperada, el archivo testbench y el código RTL se revisaron para evitar que fueran la causa de un posible fallo.

5. Proceso de síntesis lógica

El software Encounter RTL Compiler fue utilizado a lo largo de todo el proceso de síntesis lógica. La Figura 5-1 ilustra la serie de pasos que se siguieron en el proceso.

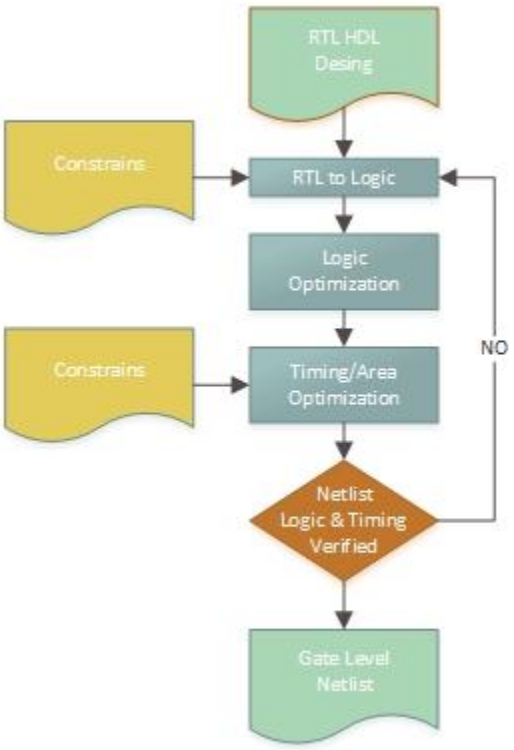


Figura 5-1. Proceso de la síntesis lógica.

5.1. Conversión de modelo RTL a modelo de lógica genérica.

Primer paso de la síntesis donde se recibió el modelo RTL y tradujo este archivo HDL de nivel medio de abstracción a lógica combinatoria y elementos de memoria, esto se ilustra en la Figura 5-2.

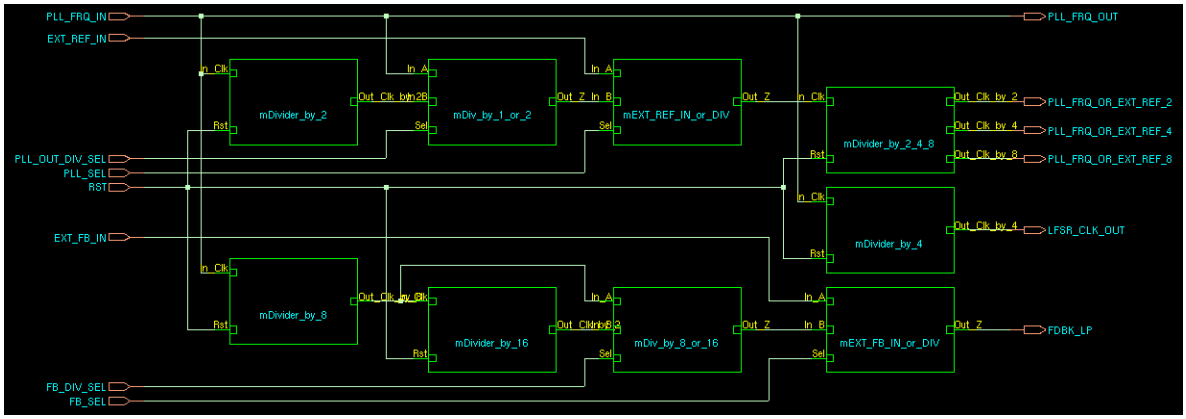


Figura 5-2. Lógica combinatoria y elementos de memoria traducida.

La elaboración solo fue necesaria para el diseño *top-level*. Durante la elaboración Encounter RTL Compiler realizo las siguientes tareas:

- Creo estructuras de datos
- Infirió registros en el diseño
- Realizo una optimización de HDL de alto nivel, como la eliminación de código inactivo
- Verifico la semántica

Al final de la elaboración, el compilador RTL muestra cualquier referencia no resuelta, los resultados se muestran en la Figura 5-3.

```
============================================
The RUNTIME after Elaboration is 12 secs
and the MEMORY_USAGE after Elaboration is 257.00 MB
============================================
Checking the design.

Check Design Report
-------------------

Unresolved References & Empty Modules
-------------------------------------
No unresolved references in design 'Div_and_Mux'

No empty modules in design 'Div_and_Mux'

Done Checking the design.
```

Figura 5-3. Resultados de la elaboración del diseño top-level.

En la Figura 5-4 se muestra como después de la elaboración, el compilador RTL tiene una estructura de datos creada internamente para todo el diseño para que pueda aplicar restricciones y realizar otras operaciones.

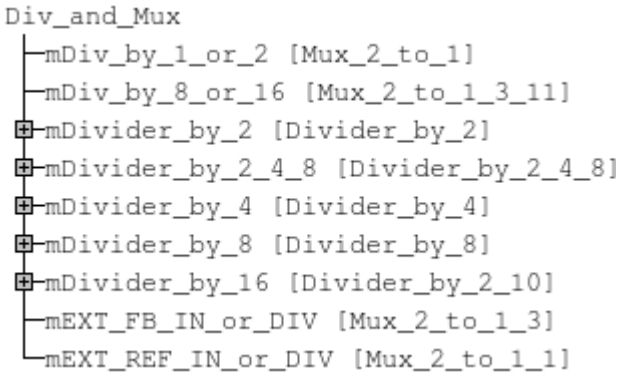


Figura 5-4. Estructura de datos.

5.2. Optimización lógica

En el siguiente paso de Optimización Lógica, las ecuaciones que representan los circuitos lógicos se minimizaron, aplanaron y factorizaron.

Después de cargar y elaborar el diseño, se especificaron restricciones. Las restricciones incluyeron:

- Condiciones de funcionamiento
- Formas de onda de reloj
- temporización de E / S

Estas restricciones se aplicaron mediante la inclusión del archivo de restricciones Div_and_Mux.sdc, este archivo puede ser consultado en el Apéndice C.

Además de aplicar restricciones de diseño, se usaron estrategias de optimización adicional para obtener los objetivos de rendimiento deseados a partir de la síntesis. Estas estrategias fueron, eliminar jerarquías creadas en el diseño (desagrupación) y crear grupos de costos personalizados para las rutas en el diseño (Entrada a Salida (I2O), Entrada a Registro (I2C), Registro a Registro (C2C) y Registro a Salida (C2O)).

5.3. Conversión de modelo de lógica optimizada a tecnología de celdas estándar

En el paso de Lógica a Tecnología se tradujo la descripción a nivel de lógica optimizada a una descripción a nivel de compuertas lógicas, utilizando celdas estándar de una librería de tecnología específica, este resultado se puede ver en la Figura 5-5. La librería .lib (*liberty format*) proporcionó información de definición funcional, tiempo, potencia y ruido para cada celda.

La ruta a las librerías .lib que se usaron son:

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/synopsys/typ_v150_t025/PnomV1p50T025_STD_CELL_8HP_12T.lib

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/synopsys/typ_v150_v150_t25/IBM_CMOS8HP_BASE_WB_IO_TYP_V150_V150_T25.lib

También se usaron librerías .lef (Library Exchange Format) que contienen el nombre de celda y las dimensiones de celda, capas, definiciones de vías, capacitancias, layout de pines y bloqueos.

La ruta a las librerías .lef que se usaron son:

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/BICMOS8HP_SC_1P2V_12T_RVT.lef

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/bicmos8hp_7AM_41_tech.lef

Este paso proporciona la implementación más pequeña posible del diseño, eso satisface los requisitos de tiempo.

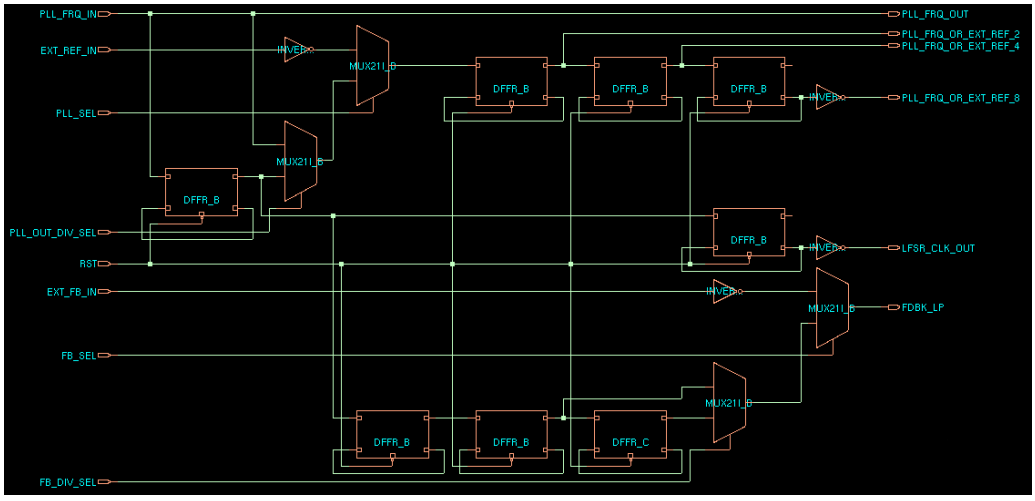


Figura 5-5. Descripción a nivel de compuertas lógicas utilizando celdas estándar.

5.4. Optimización de tiempo y área

El paso Optimización de tiempo y área optimiza la descripción a nivel de compuertas, utilizando la sustitución de compuertas para cumplir con el área específica y las restricciones de tiempo. La optimización final que el compilador RTL Compiler realiza es una optimización incremental, optimizaciones realizadas para mejorar el tiempo y el área y corregir las violaciones de DRC. De forma predeterminada, el compilador RTL Compiler no arregla las violaciones de DRC si hacerlo causa violaciones de tiempo.

El resultado de las optimizaciones es ilustrado en la Figura 5-6.

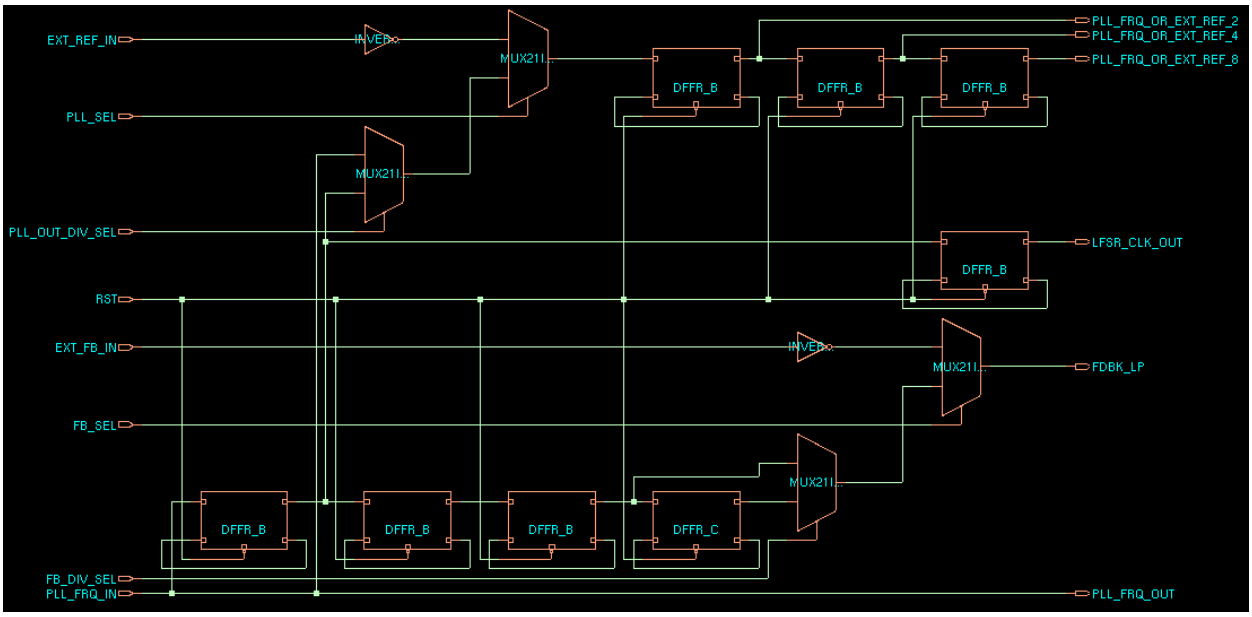


Figura 5-6. Resultados de la optimización de tiempo y área.

5.5. Simulaciones post-síntesis

La síntesis lógica generó una lista de conexiones a nivel de compuerta del circuito optimizado con información precisa de tiempos de la celda. Esta etapa finalizó con simulaciones *post-síntesis* para verificar que el circuito a nivel de compuerta proporcionará la funcionalidad deseada y cumpliera con los requisitos de tiempo apropiados.

Para las simulaciones post-síntesis se usaron las herramientas NC-Verilog simulator y SimVision, el archivo a nivel de compuerta Div_and_Mux_m.v, el archivo Verilog de las celdas estándar cmos8hp_sc_1p2v_12t_rvt.v, el cual contiene información necesaria para la simulación y por último el archivo testbench Div_and_Mux_TB.v, fue el mismo que se usó en la verificación RTL. Debido a esto, se hace una breve descripción de las simulaciones, se puede entrar más a detalle consultando la sección **4.5. Verificación RTL**.

Se presenta una serie de imágenes que ilustran cada una de las camas de pruebas realizadas para la verificación post-síntesis. En la Figura 5-7 se verificó que la frecuencia generada por el VCO es dividida para, posteriormente, ser retroalimentada al PFD del PLL. Además, se verificó la selección entre una frecuencia de retroalimentación externa o la generada a partir de la división de la frecuencia del VCO.

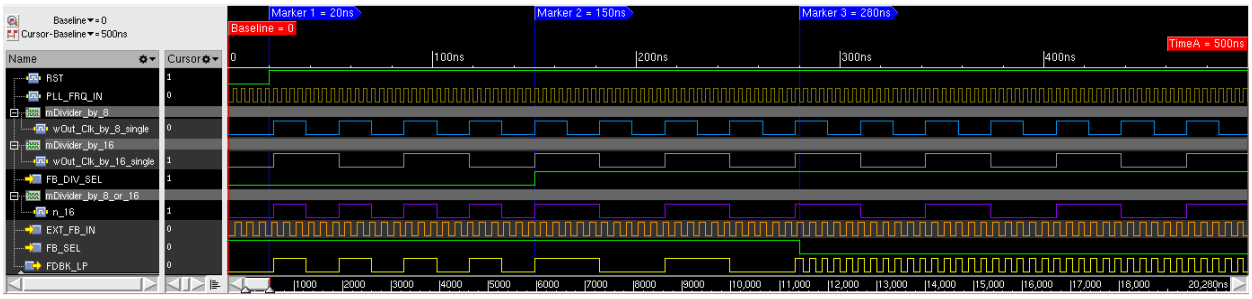


Figura 5-7. Simulación post-síntesis división de frecuencia del VCO y selección de retroalimentación.

En la Figura 5-8 se verificó que la frecuencia generada por el VCO es dividida y se genera la frecuencia de operación del LFSR. Adicionalmente se puede ver la salida PLL_FRQ_OUT.

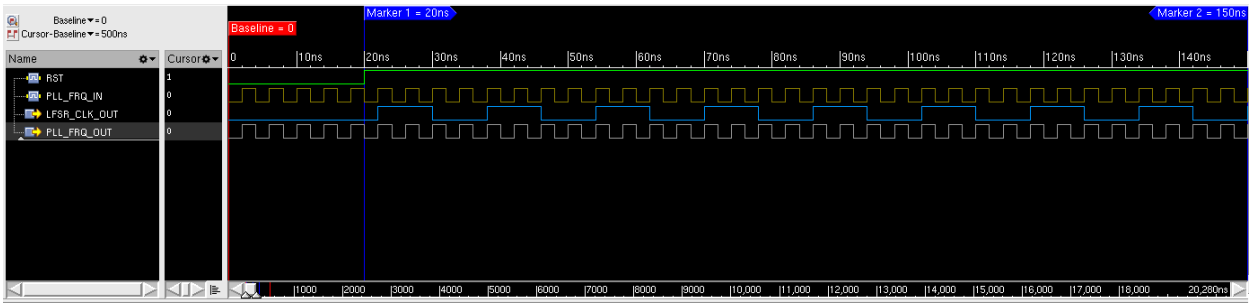


Figura 5-8. Simulación post-síntesis generación de frecuencia de operación del LFSR.

Por último, en la Figura 5-9 se verificó la generación de frecuencias para uso externo y la selección entre una frecuencia de retroalimentación externa o la generada a partir de la división de la frecuencia del VCO.

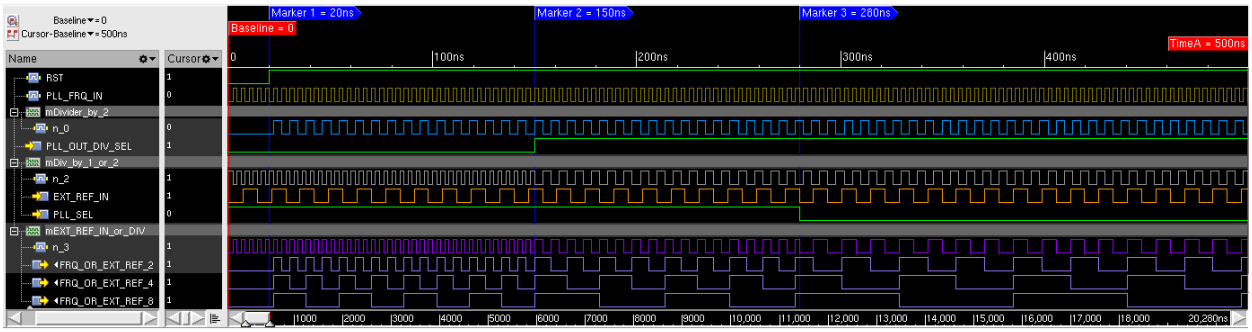


Figura 5-9. Simulación post-síntesis generación y selección de frecuencias para uso externo.

5.6. Archivos de salida

En este apartado se muestran los archivos de salida generados y los resultados del proceso de síntesis lógica, además se describen algunos archivos que se utilizaron durante el proceso y su ubicación dentro de los apéndices para su consulta.

Comenzando con los archivos de salida generados tenemos, el archivo Div_and_Mux_m.v resultado de la síntesis lógica (*modelo estructural a nivel de compuerta)*, contiene celdas estándar e interconexiones eléctricas requeridas. Para el proceso de Síntesis física se requieren restricciones de tiempo en formato .sdc, cada uno de estos archivos corresponde a una esquina proceso-voltaje-temperatura, Div_and_Mux_m_Typ.sdc corresponde a un proceso típico y Div_and_Mux_m_WC.sdc a un proceso lento. Todo el proceso de síntesis lógica se llevó a cabo con un *script* que automatizo el flujo de trabajo, este archivo es Div_and_Mux.tcl y puede ser consultado en el Apéndice D.

Como resultado de los procesos de optimización durante la síntesis lógica, mostrado en la Figura 5-10, se presenta el *timing* de cada una de las fases donde se puede observar como la última fase tiene un *Slak* positivo mayor a las fases anteriores, esto quieres decir que se logró una optimización como tal.

Metric	generic	map	incremental
Slack (ps):	2.1	2.3	2.3
R2R (ps):	569.5	566.5	566.5
I2R (ps):	no_value	no_value	no_value
R2O (ps):	527.5	303.7	303.7
I2O (ps):	166.7	37.8	37.8
CG (ps):	2.1	2.3	2.3
TNS (ps):	0	0	0
R2R (ps):	0.0	0.0	0.0
I2R (ps):	no_value	no_value	no_value
R2O (ps):	0.0	0.0	0.0
I2O (ps):	0.0	0.0	0.0
CG (ps):	0.0	0.0	0.0
Failing Paths:	0	0	0
Area:	845	399	388
Instances:	36	16	14
Utilization (%):	0.00	0.00	0.00
Tot. Net Length (um):	no_value	no_value	no_value
Avg. Net Length (um):	no_value	no_value	no_value
Total Overflow H:	0	0	0
Total Overflow V:	0	0	0
Route Overflow H (%):	no_value	no_value	no_value
Route Overflow V (%):	no_value	no_value	no_value

Figura 5-10. Resultado de los procesos de optimización durante la síntesis lógica.

En la Figura 5-11, se detallan las celdas que se describen en el modelo estructural a nivel de compuerta, así como el área total de las mismas.

```
================================================================================

Gate            Instances          Area                  Library
--------------------------------------------------------------------------------------------
DFFR_B                  7       282.240   PnomV1p50T025_STD_CELL_8HP_12T
DFFR_C                  1        40.320   PnomV1p50T025_STD_CELL_8HP_12T
INVERT_A                1         5.760   PnomV1p50T025_STD_CELL_8HP_12T
INVERT_C                1         5.760   PnomV1p50T025_STD_CELL_8HP_12T
MUX21I_B                4        53.760   PnomV1p50T025_STD_CELL_8HP_12T
--------------------------------------------------------------------------------------------
total                  14       387.840

  Type          Instances          Area          Area %
------------------------------------------------------------
sequential              8       322.560            83.2
inverter                2        11.520             3.0
logic                   4        53.760            13.9
------------------------------------------------------------
total                  14       387.840           100.0
```

Figura 5-11. Reporte de las celdas que se describen en el modelo estructural a nivel de compuerta.

Por último, en la Figura 5-12, se reportan los diferentes consumos de potencia que tiene el modelo estructural a nivel de compuerta final.

```
================================================================================

                          Leakage         Dynamic         Total
  Instance       Cells    Power(uW)       Power(uW)       Power(uW)
--------------------------------------------------------------------------------
Div_and_Mux      14           0.048         498.140         498.189
```

Figura 5-12. Reporte de consumos de potencia del modelo estructural a nivel de compuerta final.

6. Proceso de síntesis física

6.1. Importado de diseño

Este apartado presenta todos los archivos necesarios para comenzar con el proceso de la síntesis física. Para realizar la importación del diseño son necesarios los siguientes archivos:

Las librerías de *Timing*, que contienen información de tiempo de las celdas estándar y de los pines de E/O, el software Cadence® Encounter® Digital Implementation (EDI) System utilizado a lo largo de todo este proceso, lee estos archivos en formato .lib.

La ruta a las librerías .lib que se usaron son:

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/synopsys/typ_v150_t025/PnomV1p50T025_STD_CELL_8HP_12T.lib

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/synopsys/typ_v150_v150_t25/IBM_CMOS8HP_BASE_WB_IO_TYP_V150_V150_T25.lib

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/synopsys/slow_v108_t125/PwcV1p08T125_STD_CELL_8HP_12T.lib

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/synopsys/slow_v108_v140_t125/IBM_CMOS8HP_BASE_WB_IO_SLOW_V108_V140_T125.lib

Los archivos de tecnología, que proporciona al software EDI las reglas de diseño para ubicación y enrutamiento, datos de resistencia de interconexión y capacitancia para generar valores de RC y modelos de *wireload* para el diseño. También contiene información de proceso para la interconexión de capas de metal.

La ruta a las librerías .lef que se usaron son:

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/bicmos8hp_5AM_21_tech.lef

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/BICMOS8HP_SC_1P2V_12T_RVT.lef

/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/lef/CMOS8HP_BASE_WB_IO_5LM.lef

El archivo Div_and_Mux_m.v resultado de la síntesis lógica, que también es necesario en el proceso de la síntesis del árbol de reloj (**6.5. Síntesis del árbol de reloj.)**.

Los archivos de restricciones de tiempo en formato .sdc, cada uno de estos archivos corresponde a una esquina proceso-voltaje-temperatura, Div_and_Mux_m_Typ.sdc corresponde a un proceso típico y Div_and_Mux_m_WC.sdc a un proceso lento.

Todo el proceso de importación de diseño se guardó en los archivos Div_and_Mux_Typ_WC.globals y Div_and_Mux_Typ_WC_analysis.view, ambos pueden ser consultados en el Apéndice E.

6.2. Definición del floorplan

En este paso de realizo el *Floorplanning* o definición del floorplan, donde se realizó un *pre-placement* de los bloques, módulos y submódulos. Además de definió la tecnología del proceso, en este caso 130 nm y se declararon las medidas tanto del *Core*, los márgenes con el *I/O Boundary*, dimensiones del *Die* y el espaciamiento entre columnas. Todo esto se encuentra especificado en el archivo create_power_grid4.tcl, archivo en el cual se automatizo el proceso, el cual se encuentra disponible en el Apéndice F.

El resultado de esta definición es ilustrado en la Figura 6-1.

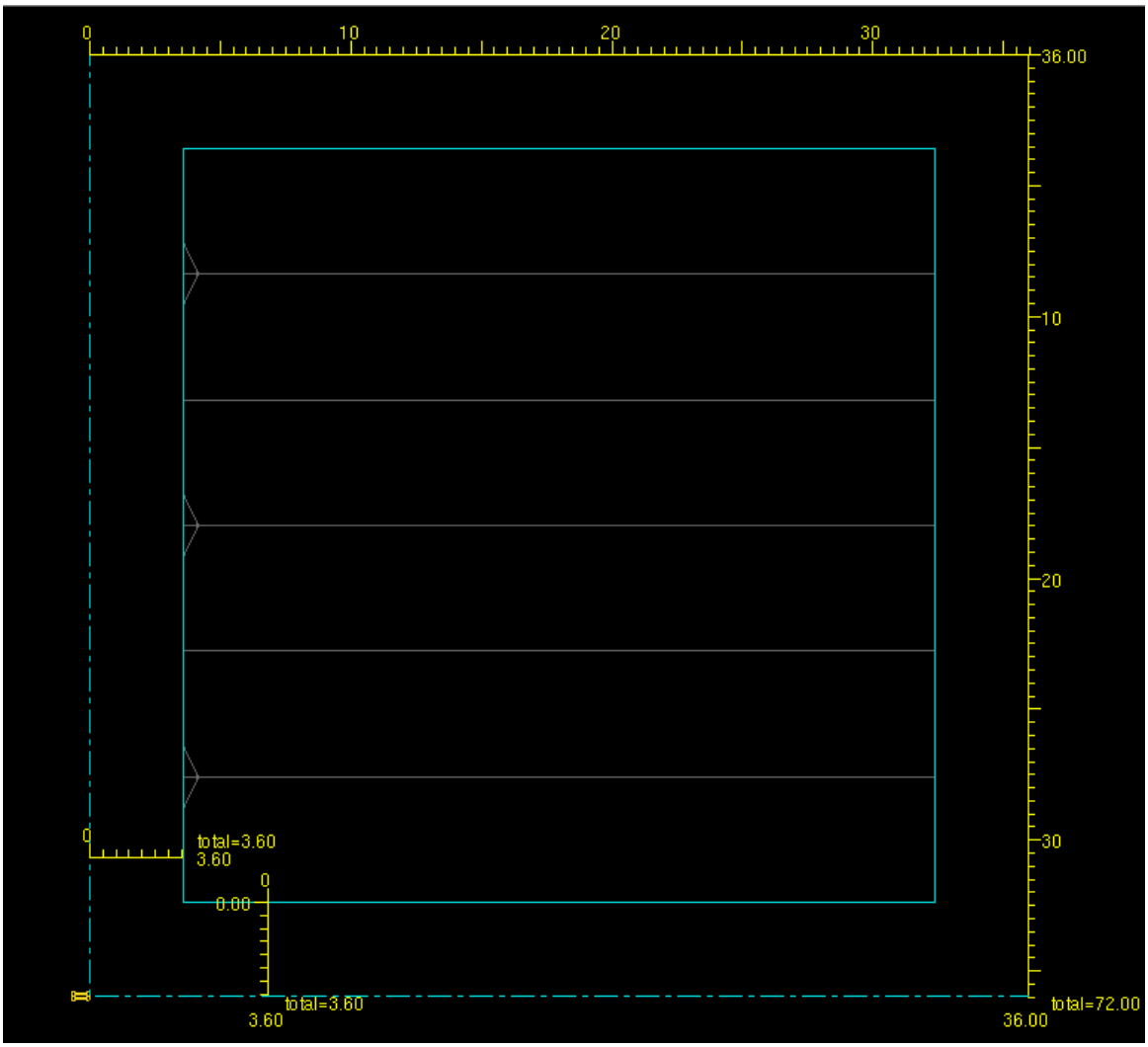


Figura 6-1. Floorplan después de la definición.

6.3. Creación power grid

Para la creación del power grid se definieron primeramente los *global nets*, se hizo la definición del *core ring* de acuerdo a los requerimientos de potencia y para concluir el grid se insertaron los *stripes* horizontales y verticales, todo esto de acuerdo a las dimensiones de las celdas estándar. Todo esto se encuentra especificado en el archivo create_power_grid4.tcl, archivo en el cual se automatizo el proceso, el cual se encuentra disponible en el Apéndice F.

El resultado de esta definición es ilustrado en la Figura 6-2.

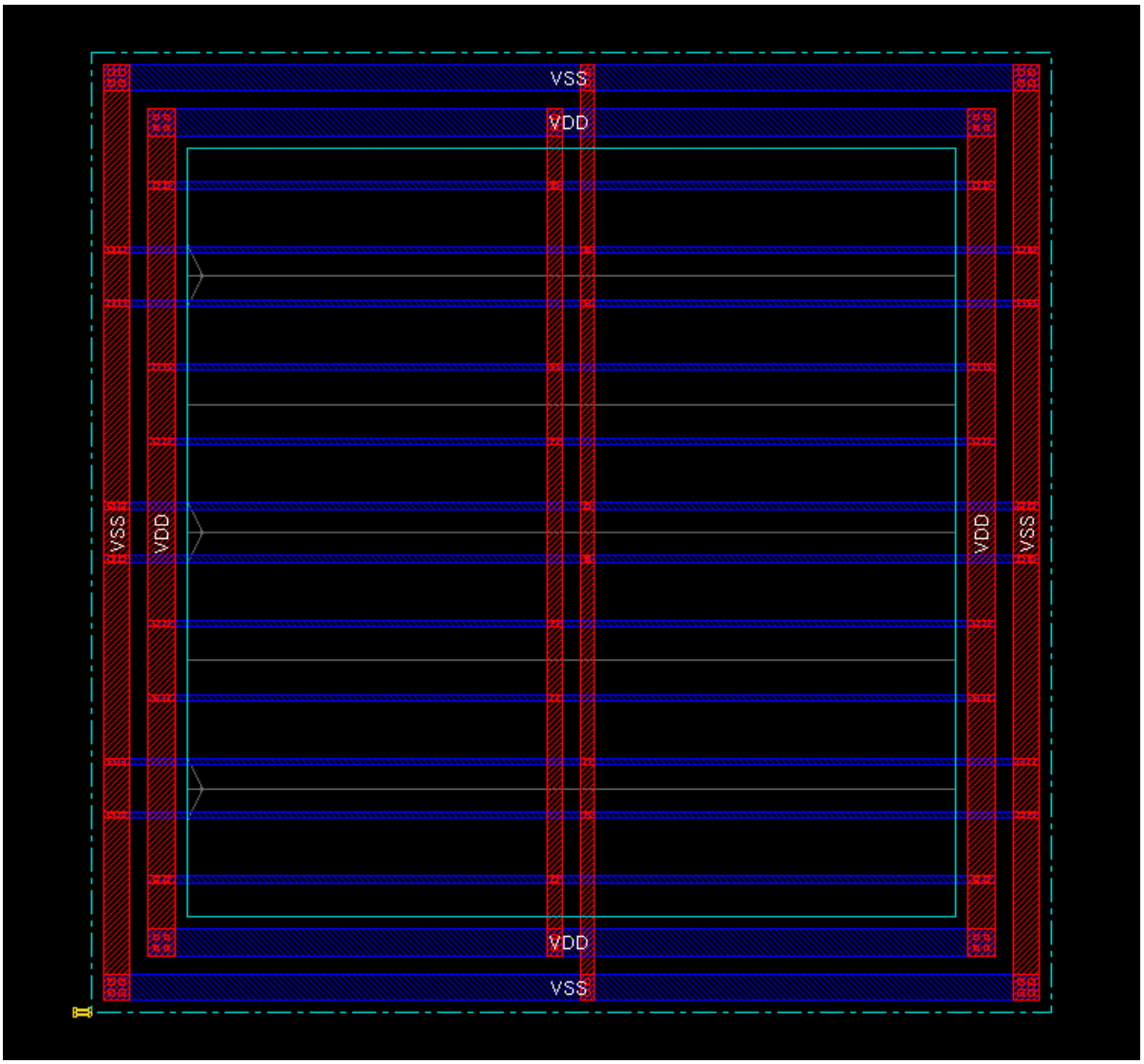


Figura 6-2. Resultado del proceso de creación del power grid.

6.4. Placement

Después de importar el diseño, definir el floorplan y crear el power grid, comenzó el proceso del *Placement*, se posicionaron las celdas estándar y bloques que no fueron pre-posicionados durante el proceso de floorplanning tomando en cuenta la conectividad y jerarquía del diseño. Para la automatización del proceso se especificaron los pasos necesarios dentro del archivo create_cts.tcl, disponible para su consulta en el Apéndice G.

La Figura 6-3 muestra el resultado del Placement.

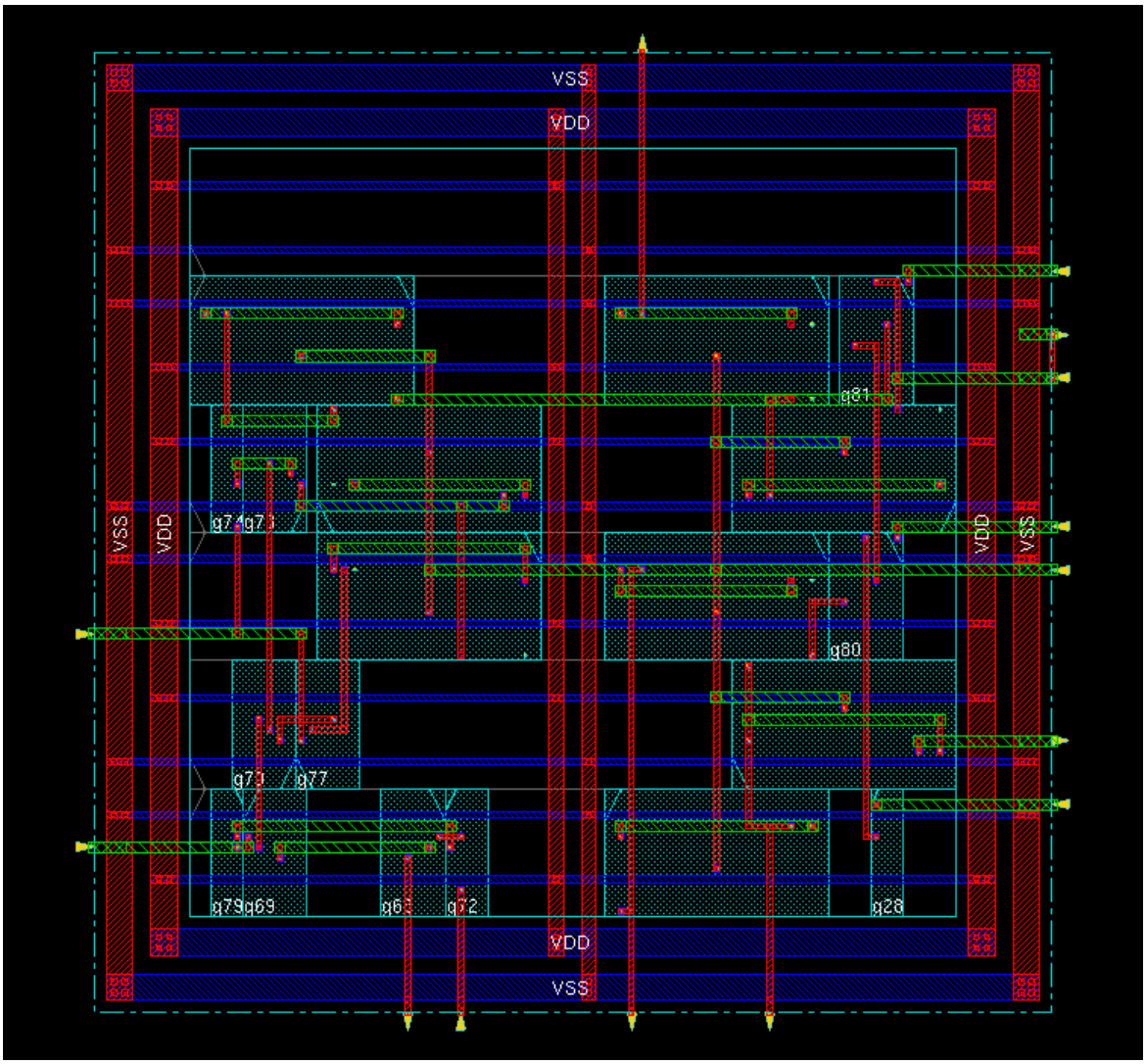


Figura 6-3. Resultado del proceso de placement.

6.5. Síntesis del árbol de reloj

Las señales del reloj se mueven a lo largo de todo el diseño para llegar a los flip-flops y latches; para mantener la integridad de las señales de reloj, se usan búferes y/o inversores en las estructuras conocidas como árboles del reloj. La síntesis del árbol de reloj o CTS (*Clock Tree Synthesis*, por sus siglas en inglés) es la etapa en la que los árboles de reloj se agregan al diseño.

CTS analizó todos los relojes en un diseño e insertó buffers e inversores para reducir o eliminar el *skew* del reloj. Este apartado describe cómo usaron los comandos de CTS para construir un árbol de búfer y equilibrar los retrasos desde la fuente de reloj a todos los flip-flops. Para la descripción del proceso se especifican los comandos necesarios dentro del archivo `create_cts.tcl`, disponible para su consulta en el Apéndice G.

La Figura 6-4 muestra el resultado del CTS.

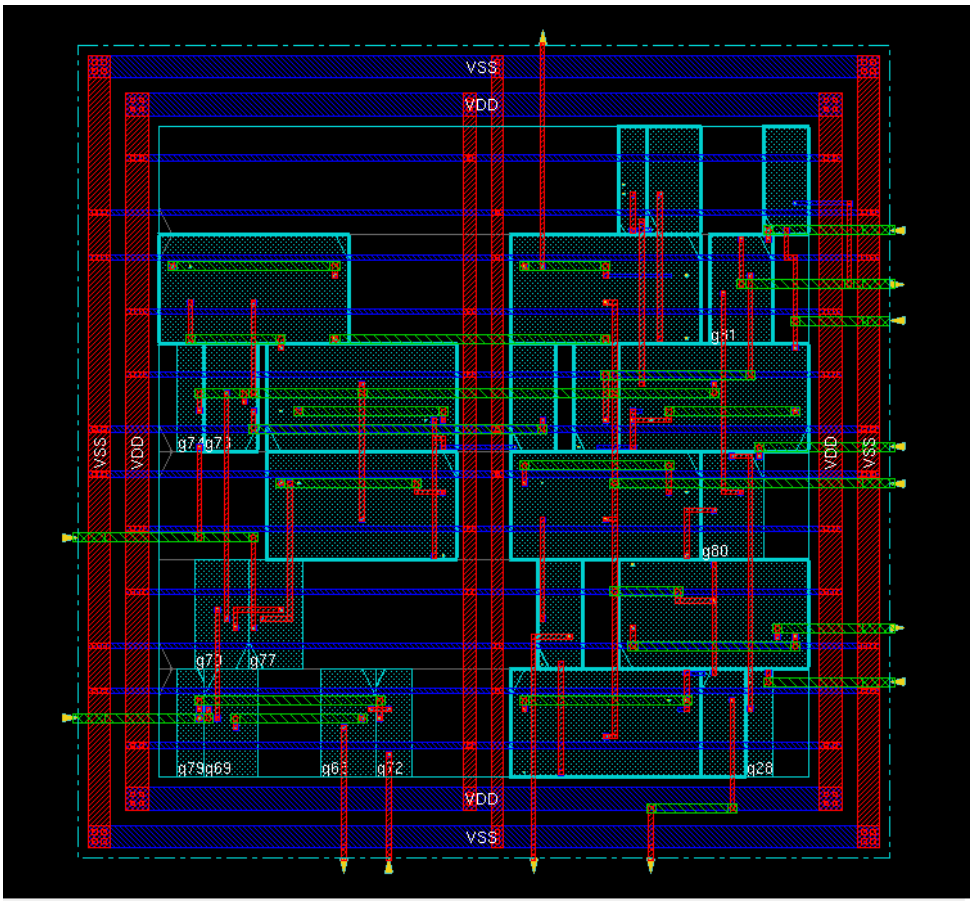


Figura 6-4. Resultado del proceso del CTS.

6.6. Enrutamiento

Para concluir la síntesis física, se realizó una serie de optimizaciones para corregir las violaciones de las reglas de diseño, reducir el *negative slack*, optimizar el tiempo de *setup* al trabajar en los *worst paths*, corrige las violaciones del *hold*, optimizar el skew y optimizar el *leakage power*. Las optimizaciones fueron; *Pre-CTS*, que tomo lugar antes de la síntesis del árbol de reloj, *Post-CTS*, que tomo lugar después de la síntesis del árbol de reloj y *Post-Route*, optimización que se realizó después del enrutamiento del diseño.

Para realizar el enrutamiento del diseño se hizo uso de NanoRoute™ de EDI, optimizada para dicho propósito. Para las optimizaciones y el uso del NanoRoute se usó el archivo opt_post_cts.tcl, disponible para su consulta en el Apéndice H.

El resultado final de la síntesis física es ilustrado en la Figura 6-5.

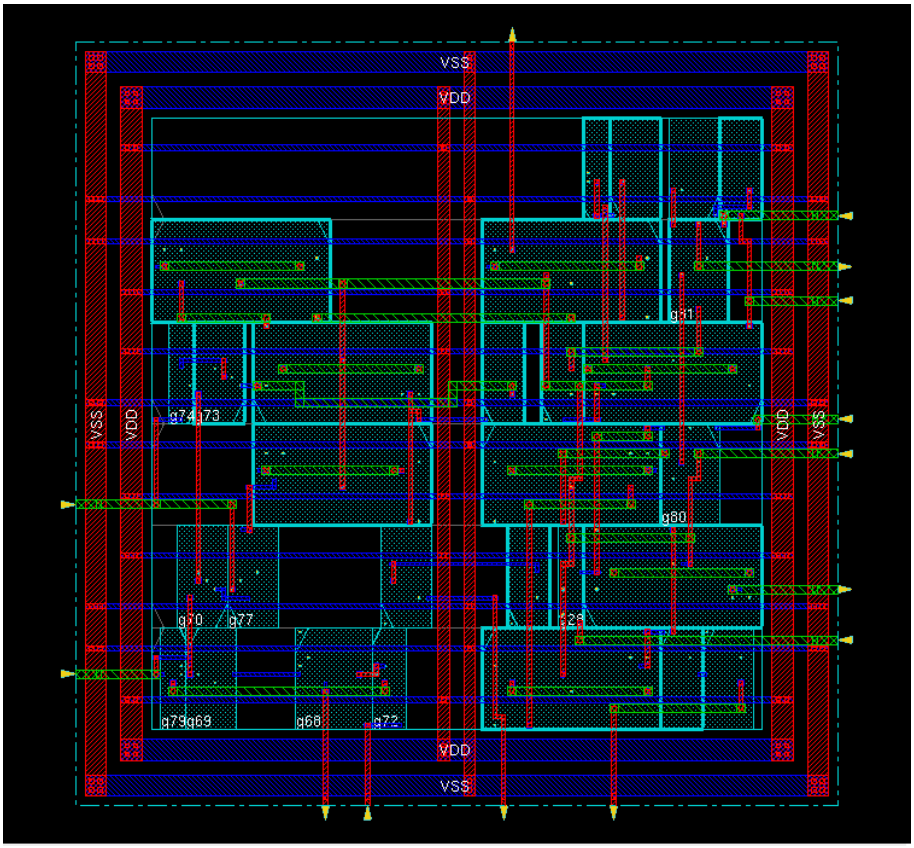


Figura 6-5. Resultado final de la síntesis física.

6.7. Verificación del diseño

Se verificó la conectividad del diseño para detectar e informar de *open nets*, problemas de antenas, *loops* y enrutamiento parcial de todos los *nets*. Se generó un reporte de conectividad, se muestra a continuación en la Figura 6-6 y se puede ver que no se presentaron violaciones de conectividad.

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Wed Jul 25 19:25:47 2018

Design Name: Div_and_Mux
Database Units: 1000
Design Boundary: (0.0000, 0.0000) (36.0000, 36.0000)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
Found no problems or warnings.
End Summary

End Time: Wed Jul 25 19:25:47 2018
Time Elapsed: 0:00:00.0

 ******** End: VERIFY CONNECTIVITY ********
Verification Complete : 0 Viols.  0 Wrngs.
(CPU Time: 0:00:00.0  MEM: 0.000M)
```

Figura 6-6. Reporte de la verificación de conectividad.

La geometría del diseño también se verificó, esto permite comprobar el diseño, como el ancho, el espaciado y la geometría interna de los objetos. Se generó un reporte de geometría, se muestra a continuación en la Figura 6-7 y se puede ver que no se presentaron violaciones de geometría.

```
*** Starting Verify Geometry (MEM: 609.0) ***

VERIFY GEOMETRY ..... Starting Verification
VERIFY GEOMETRY ..... Initializing
VERIFY GEOMETRY ..... Deleting Existing Violations
VERIFY GEOMETRY ..... Creating Sub-Areas
                      ..... bin size: 2560
VERIFY GEOMETRY ..... SubArea : 1 of 1
VERIFY GEOMETRY ..... Cells          :  0 Viols.
VERIFY GEOMETRY ..... SameNet        :  0 Viols.
VERIFY GEOMETRY ..... Wiring         :  0 Viols.
VERIFY GEOMETRY ..... Antenna        :  0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 1 complete 0 Viols. 0 Wrngs.
VG: elapsed time: 0.00
Begin Summary ...
Cells          : 0
SameNet        : 0
Wiring         : 0
Antenna        : 0
Short          : 0
Overlap        : 0
End Summary

Verification Complete : 0 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
*** verify geometry (CPU: 0:00:00.1  MEM: 65.5M)
```

Figura 6-7. Reporte de la verificación de geometría.

El siguiente paso en este proceso fue realizar la verificación de reglas de diseño, más comúnmente conocida como DRC (*Design Rule Check*, por sus siglas en inglés), en el diseño. La violación de cualquier regla de diseño podría dar lugar a que el chip fabricado no funcione como se desea. El reporte de DRC se generó y se muestra a continuación en la Figura 6-8, dicho reporte no presenta violaciones de DRC.

```
*** Starting Verify DRC (MEM: 674.4) ***

### nrEnv::init -minimal called
#WARNING (NRDB-728) PIN PAD in CELL_VIEW VSS_VDD150_PM_A does not
have antenna diff area.
#WARNING (NRDB-728) PIN VSS150 in CELL_VIEW VSSBR_VDD150_PM_A does
not have antenna diff area.
#WARNING (NRDB-728) PIN PAD in CELL_VIEW AINSD_VDD150_PM_A does
not have antenna diff area.
#WARNING (NRDB-728) PIN PT in CELL_VIEW AINSD_VDD150_PM_A does not
have antenna diff area.

### nrEnv::init completed with status success.
VERIFY DRC ..... Starting Verification
VERIFY DRC ..... Initializing
VERIFY DRC ..... Deleting Existing Violations
VERIFY DRC ..... Creating Sub-Areas
VERIFY DRC ..... Using new threading
VERIFY DRC ..... Sub-Area : 1 of 1
VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.

Verification Complete : 0 Viols.

*** End Verify DRC (CPU: 0:00:00.1  ELAPSED TIME: 0.00  MEM: 63.6M) ***
```

Figura 6-8. Reporte de la verificación de reglas de diseño (DRC).

7. Integración del SoC CDR adaptativo a jitter con LFSR para pruebas y PLL interno

En este capítulo se presentan los pasos necesarios para lograr la integración del SoC CDR adaptativo a jitter con LFSR para pruebas y PLL interno. A lo largo de este apartado serán descritos de manera genérica todos los pasos que se llevaron a cabo para lograr el objetivo de integrar todos los módulos que componen al SoC. Si se desea conocer a profundidad como se realiza cada parte del proceso, se puede acudir a un capítulo en específico de este trabajo, en cada uno de los pasos que se realicen a continuación serán referidos los capítulos o secciones donde puede ser consultado a detalle el proceso mencionado.

7.1. Proceso de síntesis lógica

7.1.1. Conversión de modelo RTL a modelo de lógica genérica.

Primer paso de la síntesis, donde se recibió el modelo RTL y tradujo este archivo HDL de nivel medio de abstracción a lógica combinatoria y elementos de memoria.

Para la elaboración fue necesario generar un nuevo *top-level*, en un nuevo archivo se incluyeron todos los módulos digitales correspondientes al CDR y LFSR, además, se elaboraron módulos en HDL que describen las entradas y salidas del módulo del PLL. EL *top-level* final también contiene los módulos que hacen alusión a los pads de entrada y salida, así como las alimentaciones y esquinas necesarias para completar el *Pad-Ring*.

Al final de la elaboración, el compilador RTL muestra cualquier referencia no resuelta, los resultados se muestran en la Figura 7-1.

```
============================================
The RUNTIME after Elaboration is 15 secs
and the MEMORY_USAGE after Elaboration is 267.00 MB
============================================
Checking the design.

Check Design Report
-------------------

Unresolved References & Empty Modules
-------------------------------------
design 'PLL_based_CDR' has the following unresolved references
/designs/PLL_based_CDR/instances_hier/mBIAS_CIRCUIT_1
/designs/PLL_based_CDR/instances_hier/mBIAS_CIRCUIT_2
/designs/PLL_based_CDR/instances_hier/mdifferentialPFD_chargepump_filterloop_LEF
/designs/PLL_based_CDR/instances_hier/pad_VSS
/designs/PLL_based_CDR/instances_hier/pad_VSS_A
/designs/PLL_based_CDR/instances_hier/pad_VSS_D
Total number of unresolved references in design 'PLL_based_CDR' : 6

design 'PLL_based_CDR' has the following empty module(s)
CORNER_VDD150_PM_A
Total number of empty modules in design 'PLL_based_CDR' : 1

 Done Checking the design.
```

Figura 7-1. Resultados de la elaboración del diseño SoC.

En la Figura 7-2 se muestra como después de la elaboración, el compilador RTL tiene una estructura de datos creada internamente para todo el diseño para que pueda aplicar restricciones y realizar otras operaciones.

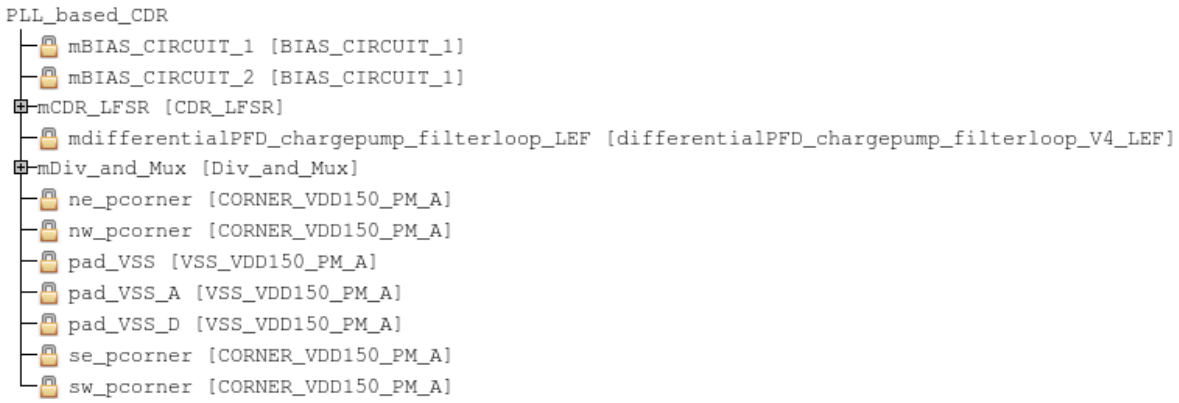


Figura 7-2. Estructura de datos del SoC.

7.1.2. Optimización lógica

En el siguiente paso de Optimización Lógica, las ecuaciones que representan los circuitos lógicos se minimizaron, aplanaron y factorizaron. Después de cargar y elaborar el diseño, se especificaron restricciones. Estas restricciones se aplicaron mediante la inclusión del archivo de restricciones PLL_based_CDR.sdc, este archivo puede ser consultado en el Apéndice I.
Además de aplicar restricciones de diseño, se usaron estrategias de optimización adicional para obtener los objetivos de rendimiento deseados a partir de la síntesis. Estas estrategias se pueden ver más a detalle consultando la sección **5.2. Optimización lógica RTL**.

7.1.3. Conversión de modelo de lógica optimizada a tecnología de celdas estándar

En este paso se tradujo la descripción a nivel de lógica optimizada a una descripción a nivel de compuertas lógicas, utilizando celdas estándar de una librería de tecnología específica, la librería .lib proporcionó información de definición funcional, tiempo, potencia y ruido para cada celda. Las librerías y su ubicación se exponen a detalle en la sección **5.3. Conversión de modelo de lógica optimizada a tecnología de celdas estándar**.

7.1.4. Optimización de tiempo y área

El paso Optimización de tiempo y área optimizo la descripción a nivel de compuertas, utilizando la sustitución de compuertas para cumplir con el área específica y las restricciones de tiempo. Para saber qué tipo de optimización realiza el compilador RTL Compiler diríjase a la sección **5.4. Optimización de tiempo y área**.

El resultado de las optimizaciones es ilustrado en la Figura 7-3.

Figura 7-3. Resultados de la optimización de tiempo y área para el SoC.

7.1.5. Archivos de salida

En este apartado se especifican los archivos de salida generados y los resultados del proceso de síntesis lógica, además se describen algunos archivos que se utilizaron durante el proceso.

Como archivo de salida generados tenemos, el archivo PLL_based_CDR_m.v resultado del proceso de síntesis lógica. Para el proceso de síntesis física se requieren restricciones de tiempo en formato .sdc, cada uno de estos archivos corresponde a una esquina proceso-voltaje-temperatura, estos archivos son PLL_based_CDR_m_Typ.sdc y PLL_based_CDR_m_WC.sdc. El proceso de síntesis lógica se llevó a cabo con un *script*, este archivo es PLL_based_CDR.tcl y puede ser consultado en el Apéndice J. La generación y uso de estos archivos se encuentra en la sección **5.6. Archivos de salida**. Como resultado de los procesos de optimización durante la síntesis lógica, mostrado en la Figura 7-4, se presenta el *timing* de cada una de las fases.

```
================================================================
Metric                          generic     map         incremental
Slack (ps):                     -639.7      -640.4      -640.4
  R2R (ps):                     215.6       14.8        14.8
  I2R (ps):                     263.8       14.1        14.1
  R2O (ps):                     no_value    no_value    no_value
  I2O (ps):                     no_value    no_value    no_value
  CG  (ps):                     -639.7      -640.4      -640.4
TNS (ps):                       894         895         895
  R2R (ps):                     0.0         0.0         0.0
  I2R (ps):                     0.0         0.0         0.0
  R2O (ps):                     no_value    no_value    no_value
  I2O (ps):                     no_value    no_value    no_value
  CG  (ps):                     894.0       895.0       895.0
Failing Paths:                  2           2           2
Area:                           33756       18054       17960
Instances:                      1275        776         771
Utilization (%):                0.00        0.00        0.00
Tot. Net Length (um):           no_value    no_value    no_value
Avg. Net Length (um):           no_value    no_value    no_value
Total Overflow H:               0           0           0
Total Overflow V:               0           0           0
Route Overflow H (%):           no_value    no_value    no_value
Route Overflow V (%):           no_value    no_value    no_value
```

Figura 7-4. Resultado de los procesos de optimización durante la síntesis lógica para el SoC.

En la Figura 7-5, se detallan las celdas que se describen en el modelo estructural a nivel de compuerta, así como el área total de las mismas.

```
================================================================================
                  Library                                  Instances    Area       Instances %
--------------------------------------------------------------------------------
IBM_CMOS8HP_BASE_WB_IO_SLOW_V108_V140_T125                 37           0.000      4.8
PwcV1p08T125_STD_CELL_8HP_12T                             734          17959.680  95.2

        Type                Instances       Area            Area %
        ----------------------------------------
        timing_model        37              0.000           0.0
        sequential          355             14330.880       79.8
        inverter            108             654.720         3.6
        unresolved          6               0.000           0.0
        logic_abstract      4               0.000           0.0
        logic               271             2974.080        16.6
        ----------------------------------------
total                       781             17959.680       100.0
```

Figura 7-5. Reporte de las celdas que se describen en el modelo estructural a nivel de compuerta para el SoC.

7.2 Proceso de síntesis física

7.2.1. Importado de diseño

Para realizar la importación del diseño son necesarios los siguientes archivos, mismos que son necesarios para comenzar con el proceso de la síntesis física.

Las librerías de *Timing*:

PnomV1p50T025_STD_CELL_8HP_12T.lib

IBM_CMOS8HP_BASE_WB_IO_TYP_V150_V150_T25.lib

PwcV1p08T125_STD_CELL_8HP_12T.lib

IBM_CMOS8HP_BASE_WB_IO_SLOW_V108_V140_T125.lib

Los archivos de tecnología:

bicmos8hp_5AM_21_tech.lef

BICMOS8HP_SC_1P2V_12T_RVT.lef

CMOS8HP_BASE_WB_IO_5LM.lef

Las rutas y uso específico de estos archivos se encuentran en la sección **6.1. Importado de diseño**.

Los archivos de tecnología de los módulos analógicos full-custom, que proporciona al software EDI las reglas de diseño para ubicación, enrutamiento y que también contienen información de proceso para la interconexión de capas de metal. Estos archivos fueron proporcionados por cada uno de los diseñadores de dichos módulos.

La ruta a las librerías .lef de los módulos analógicos que se usaron son:

/home/fnunez/Cadence_Desings/EDI/bicmos8hp/PLL_based_CDR_2v0/BIAS_CIRCUIT_1.lef

/home/fnunez/Cadence_Desings/EDI/bicmos8hp/PLL_based_CDR_2v0/differentialPFD_charge pump_filterloop_V4_LEF.lef

El archivo PLL_based_CDR_m.v resultado de la síntesis lógica, que es necesario en el proceso de la síntesis del árbol de reloj.

Y por último los archivos de restricciones de tiempo en formato. sdc, PLL_based_CDR_m_Typ.sdc y PLL_based_CDR_m_WC.sdc.

Todo el proceso de importación de diseño se guardó en los archivos PLL_based_CDR_Typ_WC.globals y PLL_based_CDR_Typ_WC_analysis.view, ambos pueden ser consultados en el Apéndice K.

7.2.2. Definición del floorplan

Se realizó un *pre-placement* de los bloques, módulos y submódulos. Se definió la tecnología del proceso, en este caso 130 nm y se declararon las medidas tanto del *Core*, los márgenes con el *I/O Boundary*, dimensiones del *Die* y el espaciamiento entre columnas. Adicional a todo lo anterior se le asigno un lugar a cada uno de los pads dentro del pad-ring, esto mediante el archivo PLL_based_CDR.ioc. Todo esto se encuentra especificado en el archivo create_power_grid4_SoC.tcl, archivo en el cual se automatizo el proceso, el cual se encuentra disponible en el Apéndice L.

El resultado de esta definición es ilustrado en la Figura 7-6.

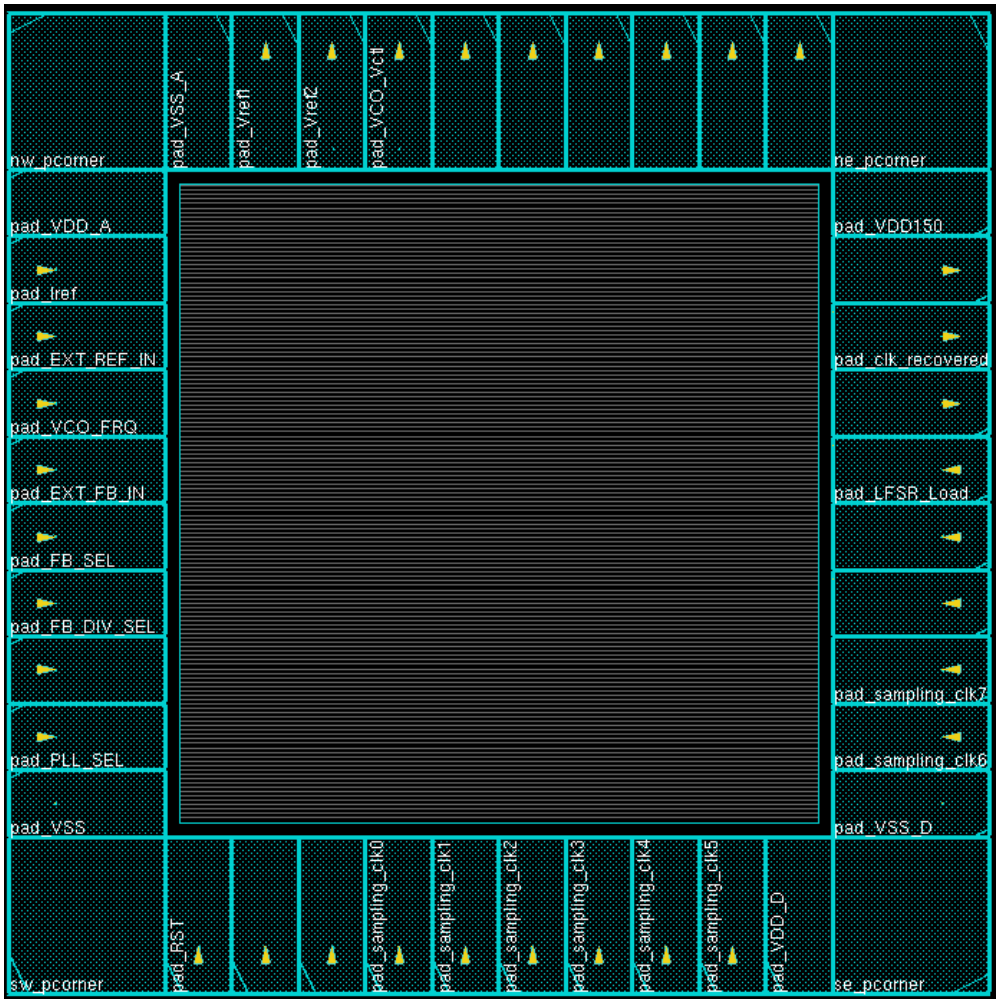


Figura 7-6. Floorplan del SoC después de la definición.

7.2.3. Creación power grid

Para la creación del power grid se definieron los *global nets*, la definición del *core ring* de acuerdo a los requerimientos de potencia y para concluir el grid se insertaron los *stripes* horizontales y verticales, todo esto de acuerdo a las dimensiones de las celdas estándar. Todo especificado para consulta en el archivo create_power_grid4_SoC.tcl, disponible en el Apéndice L.

El resultado de esta definición es ilustrado en la Figura 7-7.

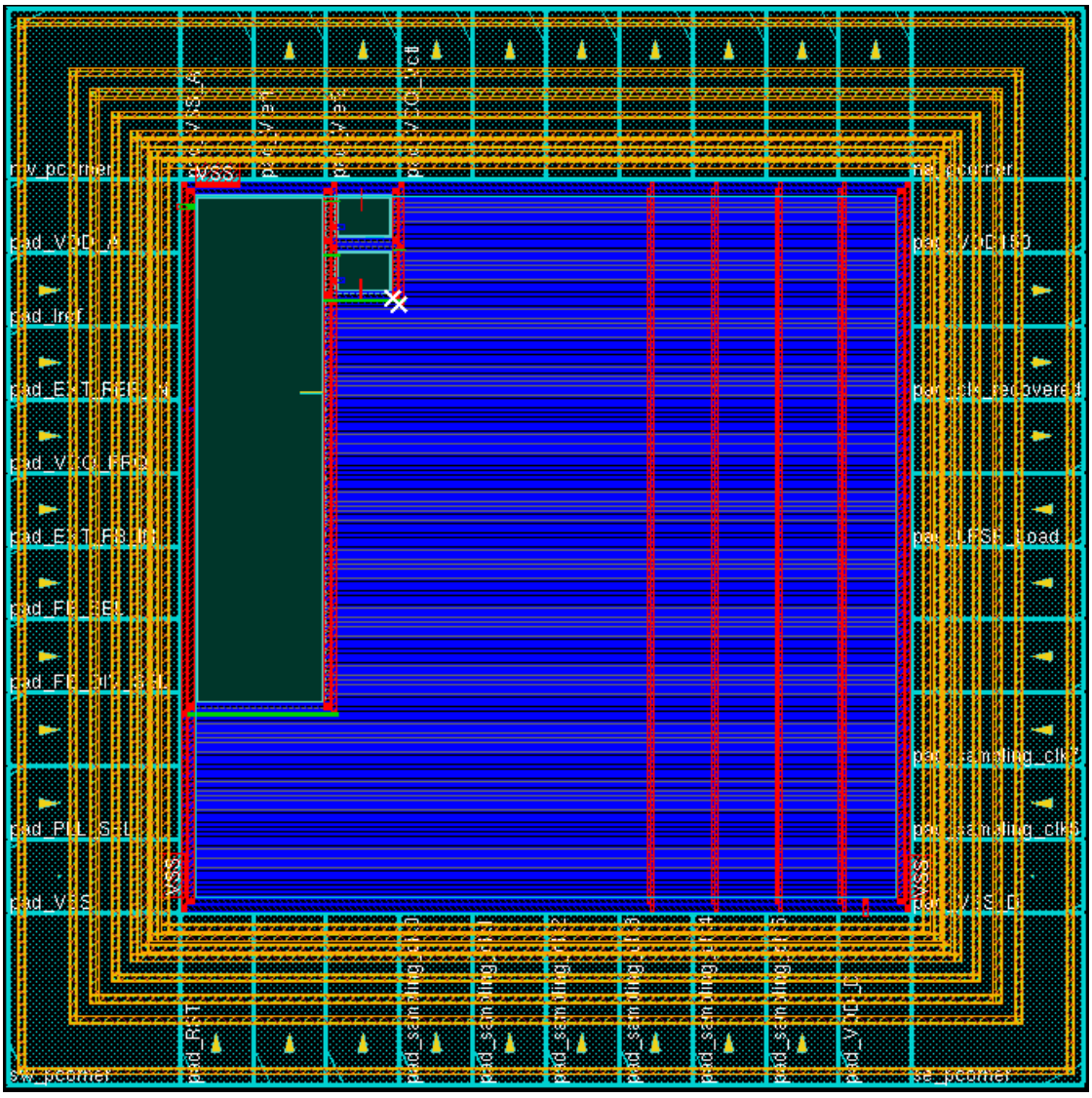


Figura 7-7. Resultado del proceso de creación del power grid para el SoC.

7.2.4. Placement y síntesis del árbol de reloj

Comenzando con el proceso del *Placement,* se posicionaron las celdas estándar y bloques que no fueron pre-posicionados durante el proceso de floorplanning tomando en cuenta la conectividad y jerarquía del diseño. La síntesis del árbol de reloj ó CTS es la etapa en la que los árboles de reloj se agregan al diseño. Para la descripción del proceso se especifican los comandos necesarios dentro del archivo `create_cts_SoC.tcl`, disponible para su consulta en el Apéndice M.

La Figura 7-8 muestra el resultado del CTS.

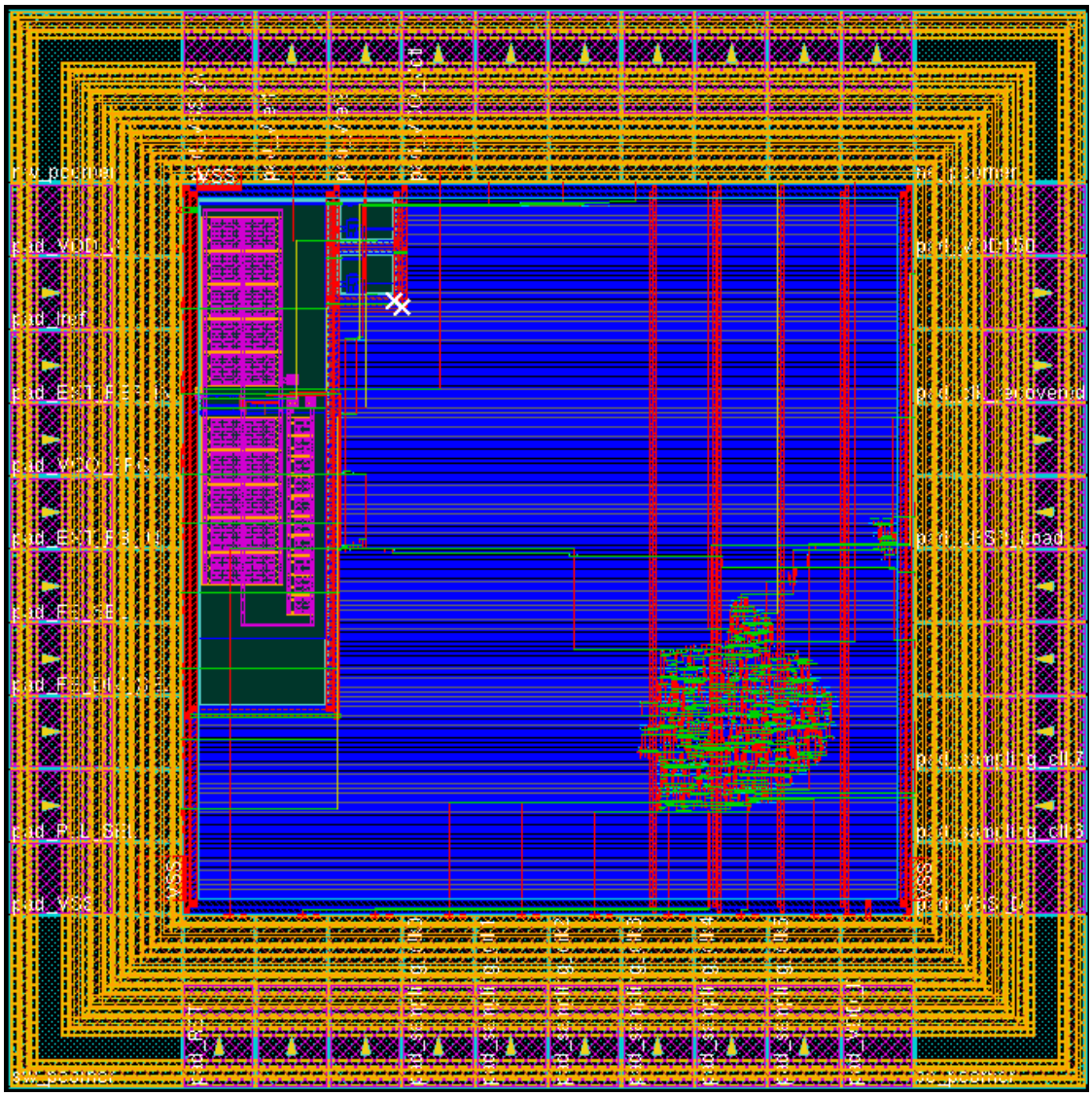


Figura 7-8. Resultado del proceso de placement y CTS para el SoC.

7.2.5. Enrutamiento

Para concluir la síntesis física, se realizaron optimizaciones *Pre-CTS*, *Post-CTS* y *Post-Route* para corregir las violaciones de las reglas de diseño, reducir el *negative slack*, optimizar el tiempo de *setup*, corregir las violaciones del *hold*, optimizar el skew y optimizar el *leakage power*. Para las optimizaciones y el uso del NanoRoute se usó el archivo `opt_post_cts_SoC.tcl`, disponible para su consulta en el Apéndice N.

El resultado final de la síntesis física es ilustrado en la Figura 7-9.

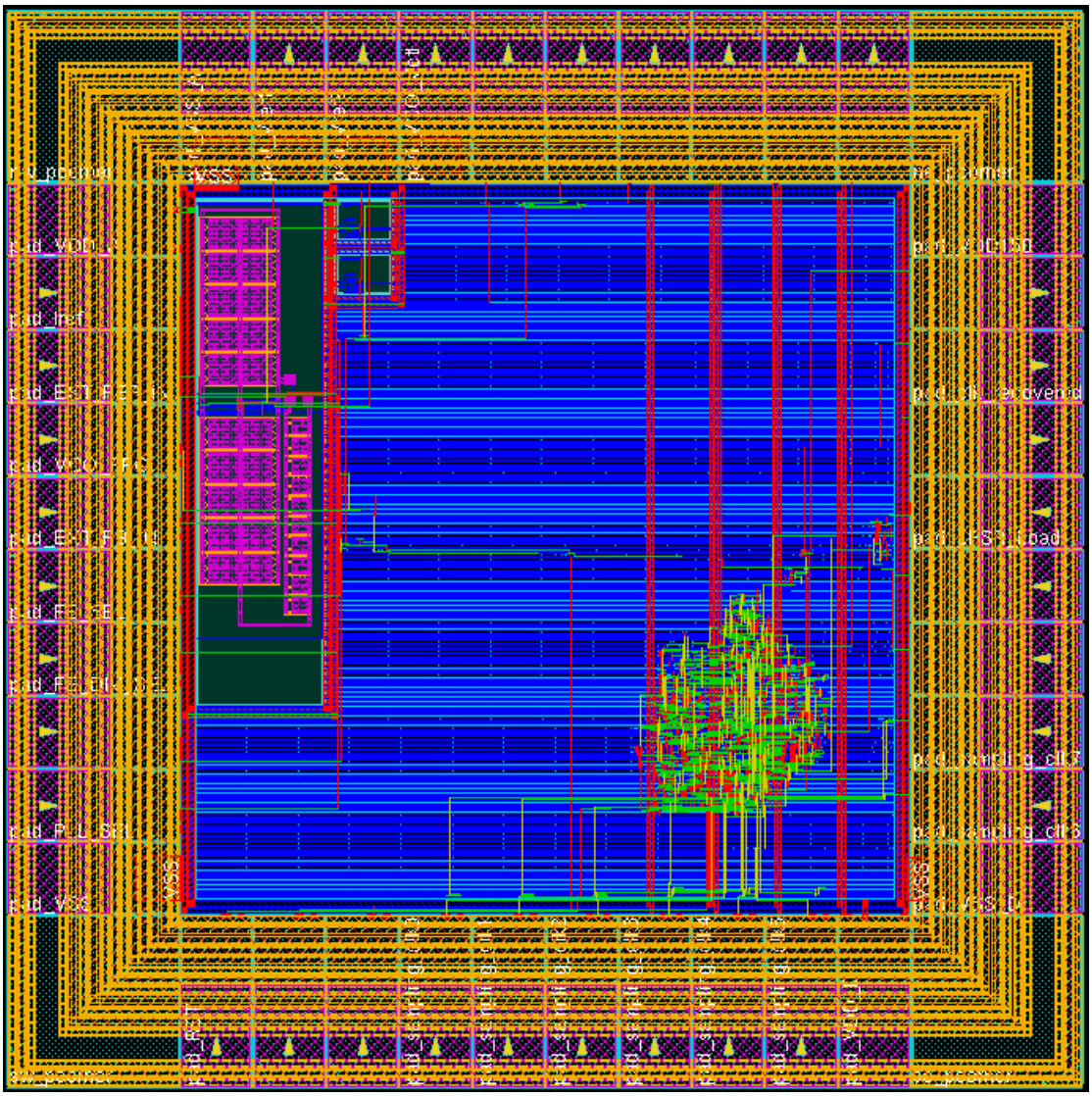


Figura 7-9. Resultado final de la síntesis física para el SoC.

7.2.6. Verificación del diseño

Se verificó la conectividad del diseño para detectar e informar de *open nets*, problemas de antenas, *loops* y enrutamiento parcial de todos los *nets*. Se generó un reporte de conectividad, se muestra a continuación en la Figura 7-10 y se puede ver las violaciones de conectividad presentadas.

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Tue Dec 11 20:23:11 2018

Design Name: PLL_based_CDR
Database Units: 1000
Design Boundary: (0.0000, 0.0000) (1372.0000, 1372.0000)
Error Limit = 1000; Warning Limit = 50
Check all nets
VDD: dangling Wire.
Net VSS: dangling Wire.

Begin Summary
2 Problem(s) (ENCVFC-94): The net has dangling wire(s).
2 total info(s) created.
End Summary

End Time: Tue Dec 11 20:23:11 2018
Time Elapsed: 0:00:00.0

******** End: VERIFY CONNECTIVITY ********
Verification Complete : 2 Viols.  0 Wrngs.
 (CPU Time: 0:00:00.1  MEM: 0.000M)
```

Figura 7-10. Reporte de la verificación de conectividad del SoC.

La geometría del diseño también se verificó, esto permite comprobar el diseño, como el ancho, el espaciado y la geometría interna de los objetos Se generó un reporte de geometría, se muestra a continuación en la Figura 7-11 y se puede ver las violaciones de conectividad presentadas.

```
*** Starting Verify Geometry (MEM: 887.3) ***

VERIFY GEOMETRY ..... Starting Verification
VERIFY GEOMETRY ..... Initializing
VERIFY GEOMETRY ..... Deleting Existing Violations
VERIFY GEOMETRY ..... Creating Sub-Areas
                      ..... bin size: 2560
VERIFY GEOMETRY ..... SubArea : 1 of 9
VERIFY GEOMETRY ..... Cells          : 0 Viols.
VERIFY GEOMETRY ..... SameNet        : 0 Viols.
VERIFY GEOMETRY ..... Wiring         : 0 Viols.
VERIFY GEOMETRY ..... Antenna        : 0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 1 complete 0 Viols. 0 Wrngs.
VERIFY GEOMETRY ..... SubArea : 2 of 9
VERIFY GEOMETRY ..... Cells          : 0 Viols.
VERIFY GEOMETRY ..... SameNet        : 0 Viols.
VERIFY GEOMETRY ..... Wiring         : 0 Viols.
VERIFY GEOMETRY ..... Antenna        : 0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 2 complete 0 Viols. 0 Wrngs.
VERIFY GEOMETRY ..... SubArea : 3 of 9
VERIFY GEOMETRY ..... Cells          : 0 Viols.
VERIFY GEOMETRY ..... SameNet        : 0 Viols.
VERIFY GEOMETRY ..... Wiring         : 0 Viols.
VERIFY GEOMETRY ..... Antenna        : 0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 3 complete 0 Viols. 0 Wrngs.
VERIFY GEOMETRY ..... SubArea : 4 of 9
VERIFY GEOMETRY ..... Cells          : 0 Viols.
VERIFY GEOMETRY ..... SameNet        : 0 Viols.
VERIFY GEOMETRY ..... Wiring         : 0 Viols.
VERIFY GEOMETRY ..... Antenna        : 0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 4 complete 0 Viols. 0 Wrngs.
VERIFY GEOMETRY ..... SubArea : 5 of 9
VERIFY GEOMETRY ..... Cells          : 0 Viols.
VERIFY GEOMETRY ..... SameNet        : 0 Viols.
VERIFY GEOMETRY ..... Wiring         : 0 Viols.
VERIFY GEOMETRY ..... Antenna        : 0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 5 complete 0 Viols. 0 Wrngs.
VERIFY GEOMETRY ..... SubArea : 6 of 9
VERIFY GEOMETRY ..... Cells          : 0 Viols.
VERIFY GEOMETRY ..... SameNet        : 0 Viols.
VERIFY GEOMETRY ..... Wiring         : 0 Viols.
VERIFY GEOMETRY ..... Antenna        : 0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 6 complete 0 Viols. 0 Wrngs.
VERIFY GEOMETRY ..... SubArea : 7 of 9
VERIFY GEOMETRY ..... Cells          : 0 Viols.
VERIFY GEOMETRY ..... SameNet        : 0 Viols.
VERIFY GEOMETRY ..... Wiring         : 0 Viols.
VERIFY GEOMETRY ..... Antenna        : 0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 7 complete 0 Viols. 0 Wrngs.
VERIFY GEOMETRY ..... SubArea : 8 of 9
VERIFY GEOMETRY ..... Cells          : 0 Viols.
VERIFY GEOMETRY ..... SameNet        : 1 Viols.
VERIFY GEOMETRY ..... Wiring         : 0 Viols.
VERIFY GEOMETRY ..... Antenna        : 0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 8 complete 1 Viols. 0 Wrngs.
VERIFY GEOMETRY ..... SubArea : 9 of 9
VERIFY GEOMETRY ..... Cells          : 0 Viols.
VERIFY GEOMETRY ..... SameNet        : 0 Viols.
VERIFY GEOMETRY ..... Wiring         : 0 Viols.
VERIFY GEOMETRY ..... Antenna        : 0 Viols.
VERIFY GEOMETRY ..... Sub-Area : 9 complete 0 Viols. 0 Wrngs.

VG: elapsed time: 2.00
Begin Summary ...
    Cells       : 0
    SameNet     : 1
    Wiring      : 0
    Antenna     : 0
    Short       : 0
    Overlap     : 0
End Summary

Verification Complete : 1 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
*** verify geometry (CPU: 0:00:01.8  MEM: 78.7M)
```

Figura 7-11. Reporte de la verificación de geometría del SoC.

El siguiente paso en este proceso fue realizar la verificación de DRC en el diseño. El reporte de DRC se generó y se muestra a continuación en la Figura 7-12, se pueden ver las violaciones de DRC presentadas.

```
*** Starting Verify DRC (MEM: 966.0) ***

### nrEnv::init -minimal called
### nrEnv::init completed with status success.
VERIFY DRC ..... Starting Verification
VERIFY DRC ..... Initializing
VERIFY DRC ..... Deleting Existing Violations
VERIFY DRC ..... Creating Sub-Areas
VERIFY DRC ..... Using new threading
VERIFY DRC ..... Sub-Area : 1 of 9
VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.
VERIFY DRC ..... Sub-Area : 2 of 9
VERIFY DRC ..... Sub-Area : 2 complete 0 Viols.
VERIFY DRC ..... Sub-Area : 3 of 9
VERIFY DRC ..... Sub-Area : 3 complete 0 Viols.
VERIFY DRC ..... Sub-Area : 4 of 9
VERIFY DRC ..... Sub-Area : 4 complete 0 Viols.
VERIFY DRC ..... Sub-Area : 5 of 9
VERIFY DRC ..... Sub-Area : 5 complete 0 Viols.
VERIFY DRC ..... Sub-Area : 6 of 9
VERIFY DRC ..... Sub-Area : 6 complete 0 Viols.
VERIFY DRC ..... Sub-Area : 7 of 9
VERIFY DRC ..... Sub-Area : 7 complete 0 Viols.
VERIFY DRC ..... Sub-Area : 8 of 9
VERIFY DRC ..... Sub-Area : 8 complete 2 Viols.
VERIFY DRC ..... Sub-Area : 9 of 9
VERIFY DRC ..... Sub-Area : 9 complete 0 Viols.

Verification Complete : 2 Viols.

*** End Verify DRC (CPU: 0:00:01.0  ELAPSED TIME: 1.00  MEM: 0.0M) ***
```

Figura 7-12. Reporte de la verificación de reglas de diseño (DRC) del SoC.

7.3. Proceso de importación de EDI a Virtuoso

Todo este proceso comenzó con la generación de un archivo .gds, GDS II (Graphic Database System) es un formato de archivo de base de datos que es el estándar para la industria para el intercambio de datos de circuitos integrados o diseños de SoC. Es un formato de archivo binario que representa formas geométricas planas, etiquetas de texto y otra información sobre el diseño en forma jerárquica. Los datos se pueden utilizar para reconstruir la totalidad o parte del diseño o transferirlos entre diferentes herramientas, en este caso se usó para este último propósito.

El archivo PLL_based_CDR_bicmos8hp.gds fue generado, pero, no sin antes pasar todas las verificaciones pertinentes realizadas en EDI, para generar el archivo se implementaron las siguientes instrucciones:

```
set nameGDS "PLL_based_CDR_bicmos8hp.gds";
streamOut       ../GDS/${nameGDS}
-mapFile        /media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/bicmos8hp_soce2gds.map
-libName        DesignLib
-merge          {/media/Ext/libs/8HP_IP_CELL_AND_IO_Libs/BiCMOS8HP_Digital_Kit/Ibm_cmos
8hp/sc_1p2v_12t_rvt/v.20171220/gds2/BICMOS8HP_SC_1P2V_12T_RVT.gds/opt/libs/IBM_PD
K/bicmos8hp/v.20160727/gds2/CMOS8HP_BASE_WB_IO_7LM.gds }
-outputMacros -units 1000 -mode ALL
```

Se usó el archivo opt_post_cts_SoC.tcl para la ejecución de las instrucciones anteriores, disponible para su consulta en el Apéndice N.

Para dar inicio a la importación y teniendo disponible el archivo GDS, se ejecutó el script PLL_based_CDR_VirtuosoScript1.il (Apéndice O), específicamente diseñado para generar una nueva librería en Virtuoso donde se pudiera trabajar en la integración e importar el recientemente creado archivo GDS. A continuación, en la Figura 7-13 se muestra el SoC en el ambiente de Virtuoso listo para continuar integrando el diseño.

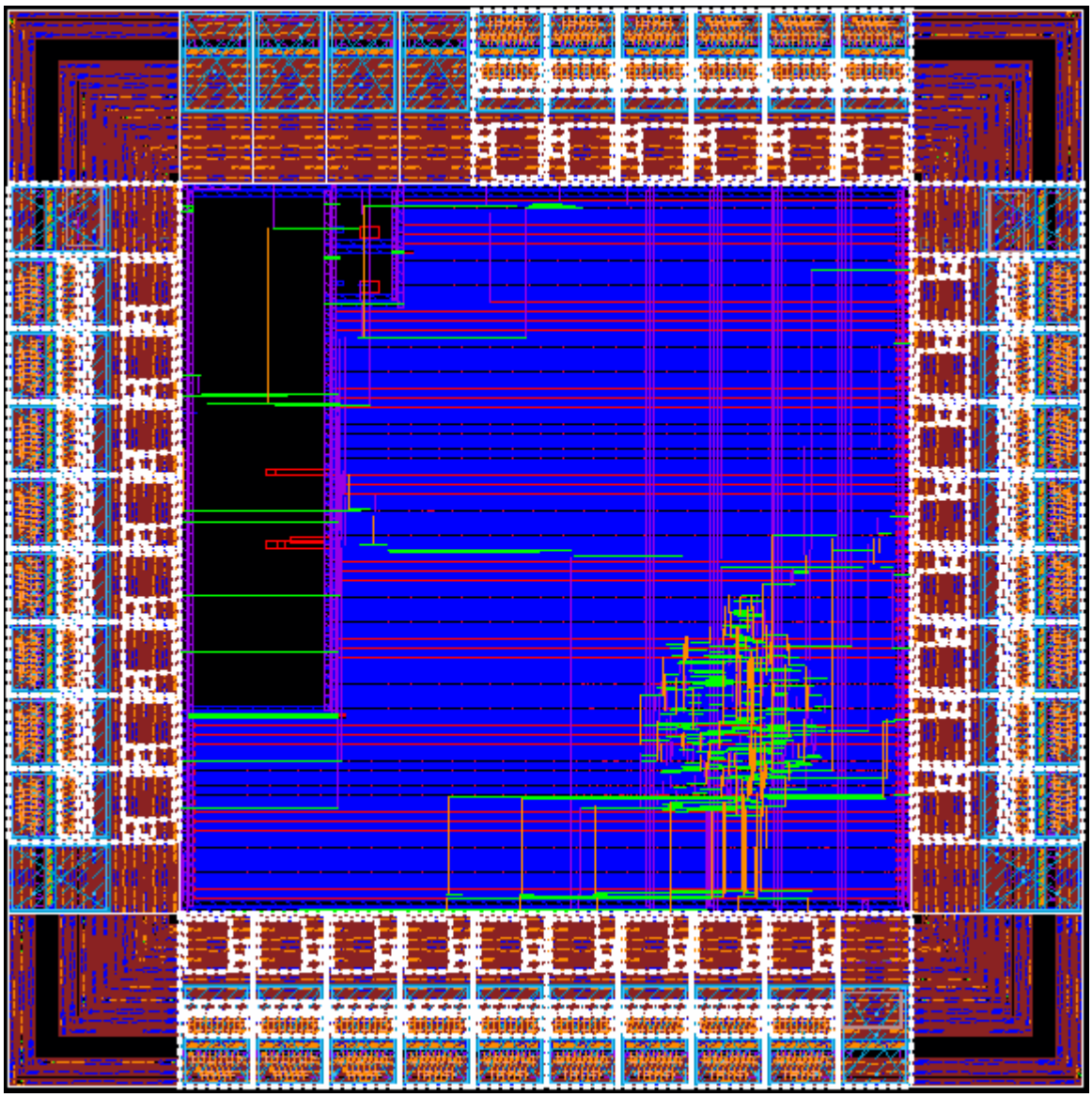


Figura 7-13. SoC en ambiente de Virtuoso listo para continuar integrando el diseño.

7.4. Placement de los módulos analógicos

Para lograr la integración final entre los módulos digitales y los analógicos contenidos en el diseño, fue necesario para estos últimos posicionarlos de manera manual, se les solicitaron a los diseñadores analógicos sus librerías de trabajo, para poder tomar el layout de los módulos faltantes y copiarlos dentro del diseño donde se estaba integrando todo. El resultado final de este proceso manual es ilustrado en la Figura 7-14.

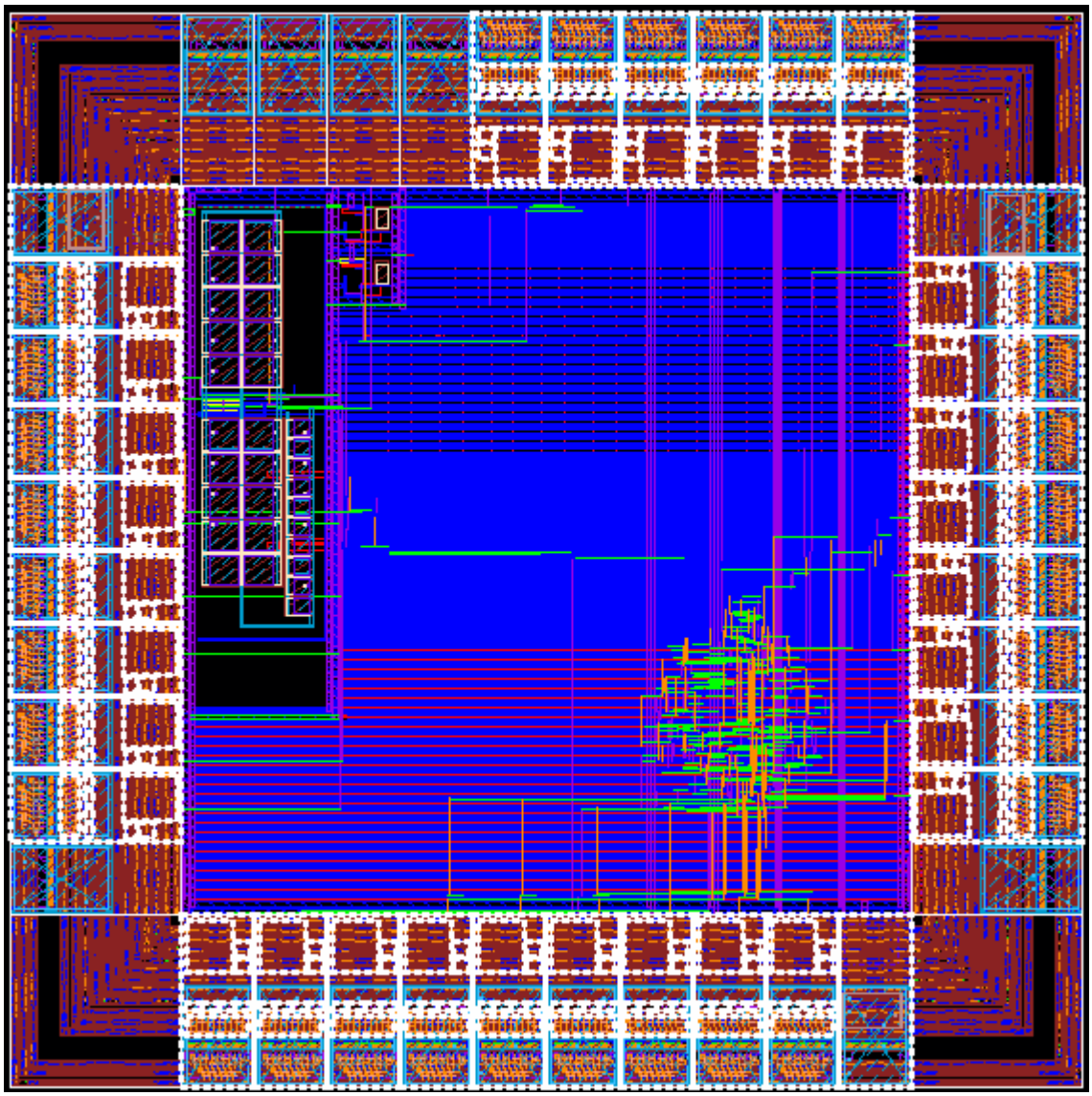


Figura 7-14. SoC en ambiente de Virtuoso después del placement de los módulos analógicos.

7.5. Verificación final del SoC

Para concluir la integración se verificaron las reglas de diseño, esto con el fin de obtener un diseño libre de errores y que pudiera ser fabricado sin ninguna clase de problemas, pero sobre todo obtener un SoC completamente funcional. De nuevo se echó mano de un script que pudiera automatizar el proceso, se ejecutó un el script PLL_based_CDR_skill_script2.il (Apéndice P), en este se indican las reglas de diseño que se verificaran, la ubicación del archivo de reglas necesario para la herramienta y además ayuda creando una carpeta donde se depositan todos los archivos generados después de la verificación, todo esto con la intención de poder acudir a ellos de manera simple.

Una vez terminado el proceso y analizados los resultados, no se encontraron violaciones en las reglas de diseño que pudieran comprometer el funcionamiento, la única advertencia que arrogaron los resultados fue la ausencia de un *chip-edge*, este es una parte fundamental para los fabricantes, ya que una vez terminado el proceso de fabricación podrecen al corte de los chips, y es cuando el chip-edge es necesario ya que indica por donde se debe de cortar el chip, de otra manera podrían cortarse los chips de manera incorrecta. Ya agregado el chip-edge es ilustrado en la Figura 7-15, ahora se cuenta con un SoC listo para la fabricación.

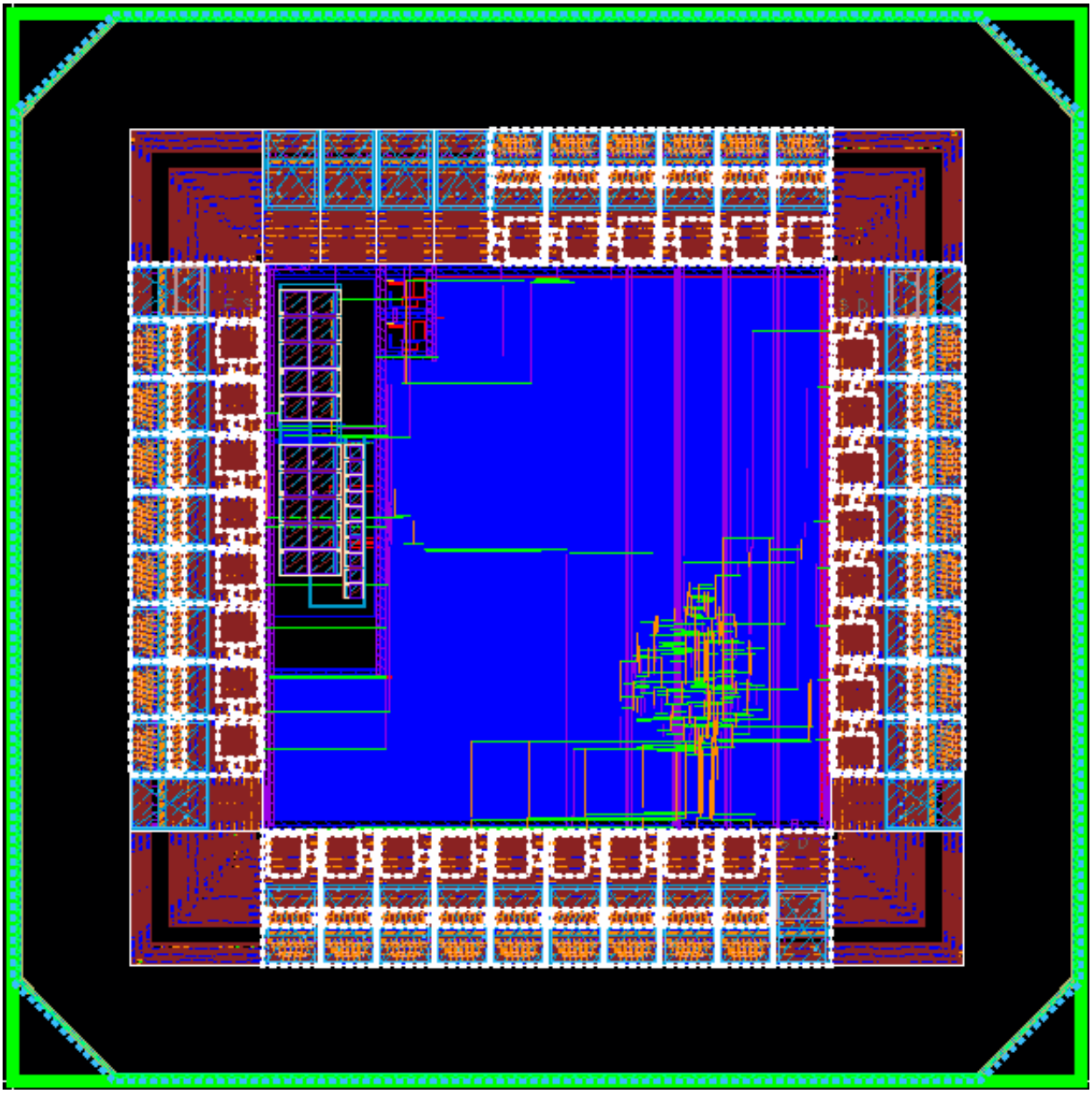


Figura 7-15. SoC verificado y listo para fabricación después de agregar el chip-edge.

7.6. Archivos para la fabricación del SoC

Como se mencionó en **7.3. Proceso de importación de EDI a Virtuoso,** GDS II es un formato de archivo de base de datos para el intercambio de diseños de SoC y es precisamente un archivo como este el que necesita GLOBALFOUNDRIES para realizar la fabricación.

Por medio del script PLL_based_CDR_Skill_StreamOut.il (Apéndice Q) se generó el archivo final, PLL_based_CDR_3v0.gds podrá ser enviado a GLOBALFOUNDRIES para que realicen el proceso de litografía de 130nm para la fabricación del SoC.

Conclusiones

En este trabajo se describió la metodología seguida en el diseño, implementación y verificación del módulo Divisor de Frecuencia e integración del SoC CDR adaptativo a jitter con LFSR para pruebas y PLL interno en la tecnología 130 nm BiCMOS8HP proporcionada por GLOBALFOUNDRIES.

El divisor de frecuencia diseñado cumplió las especificaciones, pudiendo dividir una frecuencia de entrada de hasta 1.3 GHz. La fuente de señal entrada puede ser seleccionada entre la proveniente del VCO externo o frecuencia externa y retroalimentarla al PFD del PLL. Se generó una frecuencia de 400 MHz para la operación del LFSR. Además, es capaz de otorgar una serie de frecuencias para uso externo, útiles para probar el funcionamiento interno de los diversos divisores y multiplexores de frecuencias.

Se realizaron simulaciones de verificación funcional del módulo, simulaciones de las distintas etapas del diseño, como son el RTL y los modelos estructurales a nivel de compuerta de la síntesis lógica y física (gate-level netlist). Se realizaron pruebas con distintas esquinas de proceso-voltaje-temperatura, una de esas esquinas fue para un proceso típico a 1.2 V y 25° C y otra con un proceso lento a 1.08 V y 125° C, para cubrir las posibles variaciones en el proceso de fabricación, voltaje y temperatura. Con esto se aseguró que el SoC funcione dentro de dicho rango de variaciones.

Durante el proceso de integración del SOC, se logró dominar la generación de archivos LEF de los módulos analógicos, así como su uso en las herramientas CAD empleadas en esta tesina. El layout obtenido se optimizó para lograr el mejor desempeño, y se pasaron satisfactoriamente las verificaciones de DRC, conectividad y geometría. Finalmente se logró, aunque de manera manual, posicionar los módulos analógicos dentro del SoC y cumplir con las reglas de diseño. Se generaron los archivos necesarios para su exportación y posterior fabricación.

Apéndices

A. MODELOS HDL DE LOS MODULOS DEL DIVISOR

```
module Div_and_Mux(
        input PLL_FRQ_IN,
        input EXT_REF_IN,
        input EXT_FB_IN,
        input FB_SEL,
        input FB_DIV_SEL,
        input PLL_OUT_DIV_SEL,
        input PLL_SEL,
        input RST,
        output PLL_FRQ_OUT,
        output PLL_FRQ_OR_EXT_REF_2,
        output PLL_FRQ_OR_EXT_REF_4,
        output PLL_FRQ_OR_EXT_REF_8,
        output LFSR_CLK_OUT,
        output FDBK_LP
);

        wire wOut_Clk_by_8_single;
        wire wOut_Clk_by_16_single;
        wire wOut_Clk_by_2_single;
        wire wDiv_by_1_or_2;
        wire wEXT_REF_IN_or_DIV;
        wire wDiv_by_8_or_16;

        assign PLL_FRQ_OUT = PLL_FRQ_IN;
        ////////////////////////////////////////////
        Divider_by_4 mDivider_by_4 (
           .In_Clk(PLL_FRQ_IN),
           .Rst(RST),
           .Out_Clk_by_4(LFSR_CLK_OUT)
        );

        ////////////////////////////////////////////
        Divider_by_2 mDivider_by_2 (
           .In_Clk(PLL_FRQ_IN),
           .Rst(RST),
           .Out_Clk_by_2(wOut_Clk_by_2_single)
        );

        Mux_2_to_1 mDiv_by_1_or_2 (
           .In_A(PLL_FRQ_IN),
           .In_B(wOut_Clk_by_2_single),
```

```
    .Sel(PLL_OUT_DIV_SEL),
    .Out_Z(wDiv_by_1_or_2)
);

Mux_2_to_1 mEXT_REF_IN_or_DIV (
    .In_A(EXT_REF_IN),
    .In_B(wDiv_by_1_or_2),
    .Sel(PLL_SEL),
    .Out_Z(wEXT_REF_IN_or_DIV)
);
////////////////////////////////////////////

Divider_by_8 mDivider_by_8 (
    .In_Clk(PLL_FRQ_IN),
    .Rst(RST),
    .Out_Clk_by_8(wOut_Clk_by_8_single)
);

Divider_by_2 mDivider_by_16 (
    .In_Clk(wOut_Clk_by_8_single),
    .Rst(RST),
    .Out_Clk_by_2(wOut_Clk_by_16_single)
);

Mux_2_to_1 mDiv_by_8_or_16 (
    .In_A(wOut_Clk_by_8_single),
    .In_B(wOut_Clk_by_16_single),
    .Sel(FB_DIV_SEL),
    .Out_Z(wDiv_by_8_or_16)
);

Mux_2_to_1 mEXT_FB_IN_or_DIV (
    .In_A(EXT_FB_IN),
    .In_B(wDiv_by_8_or_16),
    .Sel(FB_SEL),
    .Out_Z(FDBK_LP)
);

////////////////////////////////////////////

Divider_by_2_4_8 mDivider_by_2_4_8 (
    .In_Clk(wEXT_REF_IN_or_DIV),
    .Rst(RST),
    .Out_Clk_by_2(PLL_FRQ_OR_EXT_REF_2),
    .Out_Clk_by_4(PLL_FRQ_OR_EXT_REF_4),
```

```
            .Out_Clk_by_8(PLL_FRQ_OR_EXT_REF_8)
            );

        ////////////////////////////////////////////

        endmodule
```

```
        module Divider_by_2(
                input In_Clk,
                input Rst,
                output Out_Clk_by_2
            );

        T_FF instance_T_FF (
            .In_T(In_Clk),
            .Rst(Rst),
            .Out_Q(Out_Clk_by_2)
        );

endmodule
```

```
module Divider_by_4(
        input In_Clk,
        input Rst,
        output Out_Clk_by_4
);

        wire wOut_2_to_In_4;

        T_FF instance_T_FF_by_2 (
            .In_T(In_Clk),
            .Rst(Rst),
            .Out_Q(wOut_2_to_In_4)
        );

        T_FF instance_T_FF_by_4 (
            .In_T(wOut_2_to_In_4),
            .Rst(Rst),
            .Out_Q(Out_Clk_by_4)
        );

endmodule
```

```
module Divider_by_8(
	input In_Clk,
	input Rst,
	output Out_Clk_by_8
);

	wire wOut_2_to_In_4;
	wire wOut_4_to_In_8;

	T_FF instance_T_FF_by_2 (
	   .In_T(In_Clk),
	   .Rst(Rst),
	   .Out_Q(wOut_2_to_In_4)
	);

	T_FF instance_T_FF_by_4 (
	   .In_T(wOut_2_to_In_4),
	   .Rst(Rst),
	   .Out_Q(wOut_4_to_In_8)
	);

	T_FF instance_T_FF_by_8 (
	   .In_T(wOut_4_to_In_8),
	   .Rst(Rst),
	   .Out_Q(Out_Clk_by_8)
	);

endmodule
```

```
module Divider_by_16(
	input In_Clk,
	output Out_Clk_by_16
);

	wire wOut_2_to_In_4;
	wire wOut_4_to_In_8;
	wire wOut_8_to_In_16;

	T_FF instance_T_FF_by_2 (
	    .In_T(In_Clk),
	    .Out_Q(wOut_2_to_In_4)
	);

	T_FF instance_T_FF_by_4 (
	    .In_T(wOut_2_to_In_4),
```

```
            .Out_Q(wOut_4_to_In_8)
        );

        T_FF instance_T_FF_by_8 (
            .In_T(wOut_4_to_In_8),
            .Out_Q(wOut_8_to_In_16)
        );

        T_FF instance_T_FF_by_16 (
            .In_T(wOut_8_to_In_16),
            .Out_Q(Out_Clk_by_16)
        );

endmodule
```

```
module Divider_by_2_4_8(
        input In_Clk,
        input Rst,
        output Out_Clk_by_2,
        output Out_Clk_by_4,
        output Out_Clk_by_8
);

        T_FF instance_T_FF_by_2 (
            .In_T(In_Clk),
            .Rst(Rst),
            .Out_Q(Out_Clk_by_2)
        );

        T_FF instance_T_FF_by_4 (
            .In_T(Out_Clk_by_2),
            .Rst(Rst),
            .Out_Q(Out_Clk_by_4)
        );

        T_FF instance_T_FF_by_8 (
            .In_T(Out_Clk_by_4),
            .Rst(Rst),
            .Out_Q(Out_Clk_by_8)
        );

endmodule
```

```
module T_FF(
        input In_T,
        input Rst,
        output Out_Q
   );

        reg rT_FF ;

        always @(posedge In_T or negedge Rst)
        begin
                if (!Rst) begin
                        rT_FF <= 1'b0;
                end else begin
                        rT_FF <= ~rT_FF;
                end
        end

        assign Out_Q = rT_FF;

endmodule
```

```
module Mux_2_to_1(
        input In_A,
        input In_B,
        input Sel,
        output Out_Z
   );

        assign Out_Z = Sel ? In_B : In_A;

endmodule
```

B. TESTBENCH

```
module Div_and_Mux_TB;

	// Inputs
	reg PLL_FRQ_IN;
	reg EXT_REF_IN;
	reg EXT_FB_IN;
	reg FB_SEL;
	reg FB_DIV_SEL;
	reg PLL_OUT_DIV_SEL;
	reg PLL_SEL;
	reg RST;

	// Outputs
	wire PLL_FRQ_OUT;
	wire PLL_FRQ_OR_EXT_REF_2;
	wire PLL_FRQ_OR_EXT_REF_4;
	wire PLL_FRQ_OR_EXT_REF_8;
	wire LFSR_CLK_OUT;
	wire FDBK_LP;

	// Instantiate the Unit Under Test (UUT)
	Div_and_Mux uut (
		.PLL_FRQ_IN(PLL_FRQ_IN),
		.EXT_REF_IN(EXT_REF_IN),
		.EXT_FB_IN(EXT_FB_IN),
		.FB_SEL(FB_SEL),
		.FB_DIV_SEL(FB_DIV_SEL),
		.PLL_OUT_DIV_SEL(PLL_OUT_DIV_SEL),
		.PLL_SEL(PLL_SEL),
		.RST(RST),
		.PLL_FRQ_OUT(PLL_FRQ_OUT),
		.PLL_FRQ_OR_EXT_REF_2(PLL_FRQ_OR_EXT_REF_2),
		.PLL_FRQ_OR_EXT_REF_4(PLL_FRQ_OR_EXT_REF_4),
		.PLL_FRQ_OR_EXT_REF_8(PLL_FRQ_OR_EXT_REF_8),
		.LFSR_CLK_OUT(LFSR_CLK_OUT),
		.FDBK_LP(FDBK_LP)
	);

	integer seed = 1;

	initial begin
		// Initialize Inputs
```

```
                PLL_FRQ_IN = 0;
                EXT_REF_IN = 0;
                EXT_FB_IN = 0;
                FB_SEL = 1;
                FB_DIV_SEL = 0;
                PLL_OUT_DIV_SEL = 0;
                PLL_SEL = 1;
                RST = 0;

                // Wait 100 ns for global reset to finish
                #20;
                RST                     = 1;
                #130;
                FB_SEL                  = 1;
                FB_DIV_SEL              = 1;
                PLL_OUT_DIV_SEL         = 1;
                PLL_SEL                 = 1;
                #130;
                FB_SEL                  = 0;
                FB_DIV_SEL              = 1;
                PLL_OUT_DIV_SEL         = 1;
                PLL_SEL                 = 0;
        end

                always #7 EXT_REF_IN = ~EXT_REF_IN;
                always #2 PLL_FRQ_IN = ~PLL_FRQ_IN;
                always #3 EXT_FB_IN = ~EXT_FB_IN;

endmodule
```

C. ARCHIVO DE RESTRICCIONES

```
# ITESO University
# Francisco Nuñez
# User Constraint File

set_time_unit -picoseconds
set_load_unit -femtofarads

# Clock definition at 1300Mhz
define_clock -name 1300MHz_CLK -period 770 Div_and_Mux/PLL_FRQ_IN

define_clock -name by_2_CLK -period 770
Div_and_Mux/instances_hier/mDivider_by_2/instances_hier/instance_T_FF/instances_seq/
rT_FF_reg/pins_in/clk

define_clock -name by_4_CLK -period 1540
Div_and_Mux/instances_hier/mDivider_by_4/instances_hier/instance_T_FF_by_4/instance
s_seq/rT_FF_reg/pins_in/clk

define_clock -name by_8_2_CLK -period 770
Div_and_Mux/instances_hier/mDivider_by_8/instances_hier/instance_T_FF_by_2/instance
s_seq/rT_FF_reg/pins_in/clk

define_clock -name by_8_4_CLK -period 1540
Div_and_Mux/instances_hier/mDivider_by_8/instances_hier/instance_T_FF_by_4/instance
s_seq/rT_FF_reg/pins_in/clk

define_clock -name by_8_8_CLK -period 3080
Div_and_Mux/instances_hier/mDivider_by_8/instances_hier/instance_T_FF_by_8/instance
s_seq/rT_FF_reg/pins_in/clk

define_clock -name by_16_CLK -period 6160
Div_and_Mux/instances_hier/mDivider_by_16/instances_hier/instance_T_FF/instances_se
q/rT_FF_reg/pins_in/clk

define_clock -name by_2_4_8_2_CLK -period 770
Div_and_Mux/instances_hier/mDivider_by_2_4_8/instances_hier/instance_T_FF_by_2/ins
tances_seq/rT_FF_reg/pins_in/clk

define_clock -name by_2_4_8_4_CLK -period 1540
Div_and_Mux/instances_hier/mDivider_by_2_4_8/instances_hier/instance_T_FF_by_4/ins
tances_seq/rT_FF_reg/pins_in/clk
```

```
define_clock -name by_2_4_8_8_CLK -period 3080
Div_and_Mux/instances_hier/mDivider_by_2_4_8/instances_hier/instance_T_FF_by_8/ins
tances_seq/rT_FF_reg/pins_in/clk

# Input delay definition: This is the delay coming from outside the design. Here, it's defined
at 10% of clock period.

#external_delay -clock [find / -clock 1300MHz_CLK] -input 77 -name IDelay [find /des* -
port ports_in/*]

external_delay -clock [find / -clock 1300MHz_CLK] -input 81 -name IDelay_0 [find /des*
-port ports_in/PLL_FRQ_IN]

external_delay -clock [find / -clock 1300MHz_CLK] -input 382 -name IDelay_1 [find
/des* -port ports_in/EXT_REF_IN]

external_delay -clock [find / -clock 1300MHz_CLK] -input 382 -name IDelay_2 [find
/des* -port ports_in/EXT_FB_IN]

external_delay -clock [find / -clock 1300MHz_CLK] -input 382 -name IDelay_3 [find
/des* -port ports_in/FB_SEL]

external_delay -clock [find / -clock 1300MHz_CLK] -input 382 -name IDelay_4 [find
/des* -port ports_in/FB_DIV_SEL]

external_delay -clock [find / -clock 1300MHz_CLK] -input 382 -name IDelay_5 [find
/des* -port ports_in/PLL_OUT_DIV_SEL]

external_delay -clock [find / -clock 1300MHz_CLK] -input 382 -name IDelay_6 [find
/des* -port ports_in/PLL_SEL]

external_delay -clock [find / -clock 1300MHz_CLK] -input 382 -name IDelay_7 [find
/des* -port ports_in/RST]

# Output delay definition: This is the delay going outside the design.  Here, it's defined at
10% of clock period.

external_delay -clock [find / -clock 1300MHz_CLK] -output 77 -name ODelay [find /des*
-port ports_out/*]

# Driving cell definition

set_attribute external_driver [find [find / -libcell BUFFER_O] -libpin Z] {
/designs/Div_and_Mux/ports_in/PLL_FRQ_IN }
```

```
set_attribute external_driver [find [find / -libcell BC1520_PM_A] -libpin Z] {
/designs/Div_and_Mux/ports_in/EXT_REF_IN }

set_attribute external_driver [find [find / -libcell BC1520_PM_A] -libpin Z] {
/designs/Div_and_Mux/ports_in/EXT_FB_IN }

set_attribute external_driver [find [find / -libcell BC1520_PM_A] -libpin Z] {
/designs/Div_and_Mux/ports_in/FB_SEL}

set_attribute external_driver [find [find / -libcell BC1520_PM_A] -libpin Z] {
/designs/Div_and_Mux/ports_in/FB_DIV_SEL}

set_attribute external_driver [find [find / -libcell BC1520_PM_A] -libpin Z] {
/designs/Div_and_Mux/ports_in/PLL_OUT_DIV_SEL}

set_attribute external_driver [find [find / -libcell BC1520_PM_A] -libpin Z] {
/designs/Div_and_Mux/ports_in/PLL_SEL}

set_attribute external_driver [find [find / -libcell BC1520_PM_A] -libpin Z] {
/designs/Div_and_Mux/ports_in/RST}

# The input capacitance
#set_attribute max_capacitance 24 /designs/Div_and_Mux/ports_in/*

# Considering a pad output buffer load
set_attribute external_pin_cap 31 /designs/Div_and_Mux/ports_out/*

# Setting maximum value of fanout
set_attribute max_fanout 10 /designs/*

set_attribute lp_power_unit {uW}
set_attribute lp_power_optimization_weight 0.4 [current_design]

## Setting leakage_power_effort
set_attribute leakage_power_effort high
set_attribute max_leakage_power 1 [current_design]
set_attribute max_dynamic_power 700 [current_design]
```

D. *SCRIPT* AUTOMATIZACON PARA LA SÍNTESIS LÓGICA

```
## Script for RTL->Gate-Level Flow (generated from RC v12.10-s012_1)
## Div_and_Mux
## ITESO University
## Francisco Nuñez

if {[file exists /proc/cpuinfo]} {
  sh grep "model name" /proc/cpuinfo
  sh grep "cpu MHz"    /proc/cpuinfo
}

puts "Hostname : [info hostname]"

##############################################################################
## Preset global variables and attributes
##############################################################################

set DESIGN Div_and_Mux
set SYN_EFF high
set MAP_EFF high
set DATE [clock format [clock seconds] -format "%b%d-%T"]
set _OUTPUTS_PATH outputs_${DATE}
set _REPORTS_PATH reports_${DATE}
set _LOG_PATH logs_${DATE}

# Variable to specify the technology Typ .lib file name
#set timing_library
{v.20171220/synopsys/typ_v150_t025/PnomV1p50T025_STD_CELL_8HP_12T.lib
v.20160727/synopsys/typ_v150_v150_t25/IBM_CMOS8HP_BASE_WB_IO_TYP_V150_
V150_T25.lib}

# Variable to specify the technology WC .lib file name
set timing_library
{v.20171220/synopsys/slow_v108_t125/PwcV1p08T125_STD_CELL_8HP_12T.lib
v.20160727/synopsys/slow_v108_v140_t125/IBM_CMOS8HP_BASE_WB_IO_SLOW_V
108_V140_T125.lib}

# Variable to specify the LEF library
set my_lef_library {
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/BICMOS8HP_SC_1P2V_12T_RVT
.lef /media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/bicmos8hp_5AM_21_tech.lef}
```

```
set_attribute script_search_path {. <path>} /
set_attribute hdl_search_path {. ./rtl} /
set_attribute information_level 9 /

################################################################################
## Definitions for timing-libraries serch path
################################################################################

set_attribute lib_search_path {/media/Ext/libs/IBM_PDK/bicmos8hp} /

################################################################################
## Setting the Target Technology Library (Library setup)
################################################################################

set_attribute library $timing_library
set_attribute auto_ungroup both /

################################################################################
## Load Design (HDL files)
################################################################################

read_hdl -v2001 {
../Div_and_Mux.v
../Divider_by_2.v
../Divider_by_4.v
../Mux_2_to_1.v
../Divider_by_8.v
../Divider_by_2_4_8.v
../T_FF.v
}

################################################################################
## Performing Elaboration
################################################################################

elaborate $DESIGN
puts "Runtime & Memory after 'read_hdl'"
timestat Elaboration
check_design -unresolved

################################################################################
## Constraints Setup
################################################################################

read_sdc {../Div_and_Mux.sdc}
```

```
puts "The number of exceptions is [llength [find /designs/$DESIGN -exception *]]"

if {![file exists ${_LOG_PATH}]} {
  file mkdir ${_LOG_PATH}
  puts "Creating directory ${_LOG_PATH}"
}

if {![file exists ${_OUTPUTS_PATH}]} {
  file mkdir ${_OUTPUTS_PATH}
  puts "Creating directory ${_OUTPUTS_PATH}"
}

if {![file exists ${_REPORTS_PATH}]} {
  file mkdir ${_REPORTS_PATH}
  puts "Creating directory ${_REPORTS_PATH}"
}

puts "####################"
puts "Timing -lint Report"
puts "####################"

# The following report shows any synthesis timing problems
report timing -lint -verbose

################################################################################
## Define cost groups (clock-clock, clock-output, input-clock, input-output)
################################################################################

if {[llength [all::all_seqs]] > 0} {
  define_cost_group -name I2C -design Div_and_Mux
  define_cost_group -name C2O -design Div_and_Mux
  define_cost_group -name C2C -design Div_and_Mux
  path_group -from [all::all_seqs] -to [all::all_seqs] -group C2C -name C2C
  path_group -from [all::all_seqs] -to [all::all_outs] -group C2O -name C2O
  path_group -from [all::all_inps]  -to [all::all_seqs] -group I2C -name I2C
}

define_cost_group -name I2O -design Div_and_Mux
path_group -from [all::all_inps]  -to [all::all_outs] -group I2O -name I2O
foreach cg [find / -cost_group *] {
  report timing -cost_group [list $cg] >> Div_and_Mux_pretim.rpt
}
```

```
################################################################################
## Performing Synthesis
## Synthesizing to generic
################################################################################

synthesize -to_generic -eff $SYN_EFF
puts "Runtime & Memory after 'synthesize -to_generic'"
timestat GENERIC
write_hdl       > Div_and_Mux_generic.v
report datapath > $_REPORTS_PATH/${DESIGN}_datapath_generic.rpt
report gates    > $_REPORTS_PATH/${DESIGN}_gates_generic.rpt
report area     > $_REPORTS_PATH/${DESIGN}_area_generic.rpt
report qor      > $_REPORTS_PATH/${DESIGN}_qor_generic.rpt
report messages > $_REPORTS_PATH/${DESIGN}_messages_generic.rpt

generate_reports -outdir $_REPORTS_PATH/resume_rpts -tag generic
summary_table -outdir $_REPORTS_PATH

################################################################################
## Performing Synthesis
## Synthesizing to gates
################################################################################

synthesize -to_mapped -eff $MAP_EFF -no_incr
puts "Runtime & Memory after 'synthesize -to_map -no_incr'"
timestat MAPPED
write_hdl       > Div_and_Mux_m_noincr.v
report datapath > $_REPORTS_PATH/${DESIGN}_datapath_m_noincr.rpt
report gates    > $_REPORTS_PATH/${DESIGN}_gates_m_noincr.rpt
report area     > $_REPORTS_PATH/${DESIGN}_area_m_noincr.rpt
report qor      > $_REPORTS_PATH/${DESIGN}_qor_m_noincr.rpt
report messages > $_REPORTS_PATH/${DESIGN}_messages_m_noincr.rpt
report power -depth 0 > $_REPORTS_PATH/${DESIGN}_power_m_noincr.rpt

foreach cg [find / -cost_group *] {
  report timing -cost_group [list $cg] > $_REPORTS_PATH/${DESIGN}_[basename
$cg]_post_map.rpt
}

generate_reports -outdir $_REPORTS_PATH/resume_rpts -tag map
summary_table -outdir $_REPORTS_PATH

##Intermediate netlist for LEC verification..
write_hdl -lec > ${_OUTPUTS_PATH}/${DESIGN}_intermediate.v
```

```
write_do_lec -verbose -no_exit -revised_design
${_OUTPUTS_PATH}/${DESIGN}_intermediate.v -logfile
${_LOG_PATH}/rtl2intermediate.lec.log > ${_OUTPUTS_PATH}/rtl2intermediate.lec.do

################################################################################
## Performing Synthesis
## Incremental Synthesis
################################################################################

synthesize -to_mapped -eff $MAP_EFF -incr
puts "Runtime & Memory after incremental synthesis"
timestat INCREMENTAL
write_hdl       > Div_and_Mux_m.v
report gates    > $_REPORTS_PATH/${DESIGN}_gates_m.rpt
report area     > $_REPORTS_PATH/${DESIGN}_area_m.rpt
report timing   > $_REPORTS_PATH/${DESIGN}_timing_m.rpt
report qor      > $_REPORTS_PATH/${DESIGN}_qor_m.rpt
report datapath > $_REPORTS_PATH/${DESIGN}_datapath_m.rpt
report messages > $_REPORTS_PATH/${DESIGN}_messages_m.rpt
report power -depth 0 > $_REPORTS_PATH/${DESIGN}_power_m.rpt

foreach cg [find / -cost_group *] {
  report timing -cost_group [list $cg] > $_REPORTS_PATH/${DESIGN}_[basename
$cg]_post_incr.rpt
}

generate_reports -outdir $_REPORTS_PATH/resume_rpts -tag incremental
summary_table -outdir $_REPORTS_PATH

#write_sdc > ${_OUTPUTS_PATH}/${DESIGN}_m_Typ.sdc
write_sdc > ${_OUTPUTS_PATH}/${DESIGN}_m_WC.sdc

###################################
### write_do_lec
###################################

write_do_lec -verbose -no_exit -golden_design
${_OUTPUTS_PATH}/${DESIGN}_intermediate.v -revised_design
${_OUTPUTS_PATH}/${DESIGN}_m.v -logfile
${_LOG_PATH}/intermediate2final.lec.log >
${_OUTPUTS_PATH}/intermediate2final.lec.do

#Uncomment if the RTL is to be compared with the final netlist..
write_do_lec -verbose -no_exit -revised_design ${_OUTPUTS_PATH}/${DESIGN}_m.v
-logfile ${_LOG_PATH}/rtl2final.lec.log > ${_OUTPUTS_PATH}/rtl2final.lec.do
```

```
puts "Final Runtime & Memory."
timestat FINAL
puts "============================"
puts "Synthesis Finished ....."
puts "============================"
report power
report design_rules
report clocks
```

E. *SCRIPT* PROCESO DE IMPORTACIÓN DE DISEÑO SÍNTESIS FÍSICA

```
#############################################################################
#  Generated by:      Cadence Encounter 14.27-s035_1
#  OS:                Linux x86_64(Host ID fv00)
#  Generated on:      Wed Mar 21 20:18:39 2018
#  Design:
#  Command:           save_global ../Default.globals
#############################################################################
#
# Version 1.1
#

set ::TimeLib::tsgMarkCellLatchConstructFlag 1
set conf_qxconf_file {NULL}
set conf_qxlib_file {NULL}
set defHierChar {/}
set distributed_client_message_echo {1}
set gpsPrivate::dpgNewAddBufsDBUpdate 1
set gpsPrivate::lsgEnableNewDbApiInRestruct 1
set init_gnd_net {VSS}
set init_lef_file
{/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/bicmos8hp_5AM_21_tech.lef
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/BICMOS8HP_SC_1P2V_12T_RVT
.lef
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/lef/CMOS8HP_BASE_WB_IO_5LM.l
ef}
set init_mmmc_file {../Div_and_Mux_Typ_WC_analysis.view}
set init_pwr_net {VDD}
set init_top_cell {Div_and_Mux}
set init_verilog {../Div_and_Mux_m.v}
set lsgOCPGainMult 1.000000
set pegDefaultResScaleFactor 1.000000
set pegDetailResScaleFactor 1.000000
set timing_library_float_precision_tol 0.000010
set timing_library_load_pin_cap_indices {}
set tso_post_client_restore_command {update_timing ; write_eco_opt_db ;}
```

```
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_rc_corner -name Typ_RC_Corner -T {25} -preRoute_res {1.0} -preRoute_cap {1.0}
-preRoute_clkres {0.0} -preRoute_clkcap {0.0} -postRoute_res {1.0} -postRoute_cap
{1.0} -postRoute_xcap {1.0} -postRoute_clkres {0.0} -postRoute_clkcap {0.0}
```

```
create_rc_corner -name WC_RC_Corner -T {125} -preRoute_res {1.0} -preRoute_cap
{1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -postRoute_res {1.0} -
postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres {0.0} -postRoute_clkcap
{0.0}

create_library_set -name Typ_timing_lib -timing
{/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/synopsys/typ_v150_t025/PnomV1p50
T025_STD_CELL_8HP_12T.lib
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/synopsys/typ_v150_v150_t25/IBM_C
MOS8HP_BASE_WB_IO_TYP_V150_V150_T25.lib}

create_library_set -name WC_timing_lib -timing
{/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/synopsys/slow_v108_t125/PwcV1p08
T125_STD_CELL_8HP_12T.lib
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/synopsys/slow_v108_v140_t125/IBM_
CMOS8HP_BASE_WB_IO_SLOW_V108_V140_T125.lib}

create_constraint_mode -name Typ_Constraint_Mode -sdc_files
{../Div_and_Mux_m_Typ.sdc}

create_constraint_mode -name WC_Constraint_Mode -sdc_files
{../Div_and_Mux_m_WC.sdc}

create_delay_corner -name Typ_Delay_Corner -library_set {Typ_timing_lib} -rc_corner
{Typ_RC_Corner}

create_delay_corner -name WC_Delay_Corner -library_set {WC_timing_lib} -rc_corner
{WC_RC_Corner}

create_analysis_view -name Typ_Analysis_View -constraint_mode
{Typ_Constraint_Mode} -delay_corner {Typ_Delay_Corner}

create_analysis_view -name WC_Analysis_View -constraint_mode
{WC_Constraint_Mode} -delay_corner {WC_Delay_Corner}

set_analysis_view -setup {WC_Analysis_View} -hold {Typ_Analysis_View}
```

F. *SCRIPT* DEFINICIÓN FLOORPLAN Y CREACIÓN POWER GRID

```
# ITESO University
# Francisco Nuñez

# Defining process mode
setDesignMode -process 130

# Defining floorplan
getIoFlowFlag
setFPlanRowSpacingAndType 0 2
setIoFlowFlag 0
floorPlan -dieSizeByIoHeight max -site CORE -s 28.8 28.8 3.6 3.6 3.6 3.6

# Defining global nets
clearGlobalNets
globalNetConnect VDD -type pgpin -pin VDD -inst * -module {} -verbose
globalNetConnect VSS -type pgpin -pin VSS -inst * -module {} -verbose
globalNetConnect VDD -type tiehi -pin VDD -inst * -module {} -verbose
globalNetConnect VSS -type tielo -pin VSS -inst * -module {} -verbose

# Creating power rins
set sprCreateIeRingNets {}
set sprCreateIeRingLayers {}
set sprCreateIeRingWidth 1.0
set sprCreateIeRingSpacing 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0

addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell -center 1 -
stacked_via_top_layer AM -type core_rings -jog_distance 0.28 -threshold 0.28 -nets {VDD
VSS} -follow core -stacked_via_bottom_layer M1 -layer {bottom M1 top M1 right M2 left
M2} -width 1 -spacing 0.66 -offset 0.28

# Adding horizontal stripes
sroute -connect { blockPin padPin padRing corePin floatingStripe } -layerChangeRange {
M1 AM } -blockPinTarget { nearestTarget } -padPinPortConnect { allPort oneGeom } -
padPinTarget { nearestTarget } -corePinTarget { firstAfterRowEnd } -floatingStripeTarget
{ blockring padring ring stripe ringpin blockpin followpin } -allowJogging 1 -
crossoverViaLayerRange { M1 AM } -allowLayerChange 1 -blockPin useLef -
targetViaLayerRange { M1 AM }
```

```
# Adding vertical stripes
addStripe -skip_via_on_wire_shape Noshape -block_ring_top_layer_limit M3 -
max_same_layer_jog_length 1 -padcore_ring_bottom_layer_limit M1 -number_of_sets 1 -
skip_via_on_pin Standardcell -stacked_via_top_layer AM -padcore_ring_top_layer_limit
M3 -spacing 0.66 -xleft_offset 13.5  -xright_offset 0 -merge_stripes_value 0.28 -layer M2 -
block_ring_bottom_layer_limit M1 -width .56 -nets {VDD VSS} -
stacked_via_bottom_layer M1
```

G. *SCRIPT* PLACEMENT Y CTS

```
# ITESO University
# Francisco Nuñez
#

Puts "Starting to do Design Placement..."
setPlaceMode -congEffort high
placeDesign
Puts "... finished Design Placement"

# Use the FE-CTS
setCTSMode -engine ck

# Create clock tree using the clock buffers list:
createClockTreeSpec -bufferList {BUFFER_C BUFFER_D BUFFER_E BUFFER_F
BUFFER_H BUFFER_I BUFFER_J BUFFER_K BUFFER_L BUFFER_M BUFFER_N
BUFFER_O CLKI_C CLKI_D CLKI_E CLKI_F CLKI_H CLKI_I CLKI_K CLKI_M
CLKI_O CLKI_Q CLK_C CLK_D CLK_E CLK_F CLK_H CLK_I CLK_K CLK_M
CLK_O CLK_Q DELAY4_C DELAY4_F DELAY4_J DELAY6_C DELAY6_F
DELAY6_J DELAY6_M INVERTBAL_C INVERTBAL_D INVERTBAL_E
INVERTBAL_F INVERTBAL_H INVERTBAL_J INVERTBAL_L INVERT_A
INVERT_B INVERT_C INVERT_D INVERT_E INVERT_F INVERT_H INVERT_I
INVERT_J INVERT_K INVERT_L INVERT_M INVERT_N INVERT_O} -file
../Div_and_Mux_tutor.ctstch
```

H. *SCRIPT* OPTIMIZACIONES SÍNTESIS FÍSICA

```
# ITESO University
# Francisco Nuñez

Puts "Timing the design before CTS"

setAnalysisMode -analysisType onChipVariation

timeDesign -preCTS -prefix preCTS_setup
timeDesign -preCTS -prefix preCTS_hold  -hold

Puts "Running CTS"
dbDeleteTrialRoute
clockDesign -specFile ../Div_and_Mux_tutor.ctstch -outDir clock_report -
fixedInstBeforeCTS
Puts "Finished running CTS"

Puts "Timing the design after CTS"
timeDesign -postCTS -prefix postCTS_setup
timeDesign -postCTS -prefix postCTS_hold  -hold

Puts "Setting Optimizaiton Mode Options for DRV fixes"
setOptMode -fixFanoutLoad true
setOptMode -addInstancePrefix postCTSdrv

Puts "Optimizing for DRV"
optDesign -postCTS -drv

Puts "Timing the design after DRV fixes"
timeDesign -postCTS -prefix postCTS_setup_DRVfix
timeDesign -postCTS -prefix postCTS_hold_DRVfix -hold

Puts "Setting Optimization Mode Options for Setup fixes"
setOptMode -addInstancePrefix postCTSsetup

Puts "Optimizing for Setup"
optDesign -postCTS

Puts  "Timing the design after Setup fixes"
timeDesign -postCTS -prefix postCTS_setup_Setupfix
timeDesign -postCTS -prefix postCTS_hold_Setupfix  -hold

setOptMode -addInstancePrefix postCTShold
```

```
optDesign -postCTS -hold
Puts  "Timing the design after Hold fixes"
timeDesign -postCTS -prefix postCTS_setup_Holdfix
timeDesign -postCTS -prefix postCTS_hold_Holdfix  -hold

Puts "Routing the Design"
setNanoRouteMode -quiet -timingEngine {}
setNanoRouteMode -quiet -routeWithSiPostRouteFix 0
setNanoRouteMode -quiet -routeTopRoutingLayer default
setNanoRouteMode -quiet -routeBottomRoutingLayer default
setNanoRouteMode -quiet -drouteEndIteration default
setNanoRouteMode -quiet -routeWithTimingDriven false
setNanoRouteMode -quiet -routeWithSiDriven false
routeDesign -globalDetail

Puts "Timing the design after Route"
timeDesign -postRoute -prefix postRoute_setup
timeDesign -postRoute -prefix postRoute_hold  -hold
```

I. ARCHIVO DE RESTRICCIONES DEL SOC

```
# ITESO University
# Francisco Nuñez
# User Constraint File


set_time_unit -picoseconds
set_load_unit -femtofarads

# Clock definition at 1300Mhz
define_clock -domain D_and_M -name 1300MHz_CLK -period 770
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/PLL_FRQ_IN

define_clock -domain D_and_M -name by_2_CLK -period 770
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/instances_hier/mDivider_by_2/i
nstances_hier/instance_T_FF/instances_seq/rT_FF_reg/pins_in/clk

define_clock -domain D_and_M -name by_4_CLK -period 1540
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/instances_hier/mDivider_by_4/i
nstances_hier/instance_T_FF_by_4/instances_seq/rT_FF_reg/pins_in/clk

define_clock -domain D_and_M -name by_8_2_CLK -period 770
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/instances_hier/mDivider_by_8/i
nstances_hier/instance_T_FF_by_2/instances_seq/rT_FF_reg/pins_in/clk
define_clock -domain D_and_M -name by_8_4_CLK -period 1540
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/instances_hier/mDivider_by_8/i
nstances_hier/instance_T_FF_by_4/instances_seq/rT_FF_reg/pins_in/clk
define_clock -domain D_and_M -name by_8_8_CLK -period 3080
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/instances_hier/mDivider_by_8/i
nstances_hier/instance_T_FF_by_8/instances_seq/rT_FF_reg/pins_in/clk

define_clock -domain D_and_M -name by_16_CLK -period 6160
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/instances_hier/mDivider_by_16
/instances_hier/instance_T_FF/instances_seq/rT_FF_reg/pins_in/clk

define_clock -domain D_and_M -name by_2_4_8_2_CLK -period 770
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/instances_hier/mDivider_by_2_
4_8/instances_hier/instance_T_FF_by_2/instances_seq/rT_FF_reg/pins_in/clk
define_clock -domain D_and_M -name by_2_4_8_4_CLK -period 1540
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/instances_hier/mDivider_by_2_
4_8/instances_hier/instance_T_FF_by_4/instances_seq/rT_FF_reg/pins_in/clk
```

```
define_clock -domain D_and_M -name by_2_4_8_8_CLK -period 3080
/designs/PLL_based_CDR/instances_hier/mDiv_and_Mux/instances_hier/mDivider_by_2_
4_8/instances_hier/instance_T_FF_by_8/instances_seq/rT_FF_reg/pins_in/clk

external_delay -clock [find / -clock 1300MHz_CLK] -input 546 -name IDelay_1 [find
/des* -port ports_in/EXT_REF_IN]
external_delay -clock [find / -clock 1300MHz_CLK] -input 546 -name IDelay_2 [find
/des* -port ports_in/EXT_FB_IN]
external_delay -clock [find / -clock 1300MHz_CLK] -input 546 -name IDelay_3 [find
/des* -port ports_in/FB_SEL]
external_delay -clock [find / -clock 1300MHz_CLK] -input 546 -name IDelay_4 [find
/des* -port ports_in/FB_DIV_SEL]
external_delay -clock [find / -clock 1300MHz_CLK] -input 546 -name IDelay_5 [find
/des* -port ports_in/PLL_OUT_DIV_SEL]
external_delay -clock [find / -clock 1300MHz_CLK] -input 546 -name IDelay_6 [find
/des* -port ports_in/PLL_SEL]
external_delay -clock [find / -clock 1300MHz_CLK] -input 546 -name IDelay_7 [find
/des* -port ports_in/RST]
external_delay -clock [find / -clock 1300MHz_CLK] -input 546 -name IDelay_8 [find
/des* -port ports_in/Iref]

# Output delay definition: This is the delay going outside the design.  Here, it's defined at
10% of clock period.
external_delay -clock [find / -clock 1300MHz_CLK] -output 77 -name ODelay [find /des*
-port ports_out/*]

# Driving cell definition
## Available buffers BUFX2TS BUFX3TS BUFX4TS BUFX6TS BUFX8TS BUFX12TS
BUFX16TS BUFX20TS

set_attribute external_driver [find [find / -libcell BC1520_PM_A] -libpin Z] {
/designs/PLL_based_CDR/ports_in/*}

# The input capacitance for a VOC cell is ?fF,  considering  the wires caps:
#set_attribute max_capacitance 24 /designs/PLL_based_CDR/ports_in/*

# Considering a pad output buffer POC2A load
set_attribute external_pin_cap 31 /designs/PLL_based_CDR/ports_out/*

# Setting maximum value of fanout
set_attribute max_fanout 10 /designs/*

set_attribute lp_power_unit {uW}
set_attribute lp_power_optimization_weight 0.2 [current_design]
```

```
## To enable the recommended leakage power optimization flow,  use the root
## attribute  leakage_power_effort  set to low, medium or high-
## with an optional specification of max_leakage_power attribute for a specific power
budget.
## Setting leakage_power_effort to 'none' will enable the backward compatible mode.
set_attribute leakage_power_effort medium
set_attribute max_leakage_power 1 [current_design]
set_attribute max_dynamic_power 700 [current_design]

## CDR Timing Constraints
## ITESO University
## Nestor Garcia Hdez.

####  Clock definition for the input clks from PLL (800MHz)  ###

define_clock -name 800MHz_CLK0 -period 1250  -rise 0 -fall 50 -domain 1 -divide_period
1
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/pins_i
n/sampling_clk0;  #0
define_clock -name 800MHz_CLK45 -period 1250 -rise 12 -fall 62 -domain 2 -
divide_period 1
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/pins_i
n/sampling_clk1;   #45
define_clock -name 800MHz_CLK90 -period 1250 -rise 25 -fall 75  -domain 3 -
divide_period 1
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/pins_i
n/sampling_clk2;  #90
define_clock -name 800MHz_CLK135 -period 1250 -rise 37 -fall 87 -domain 4 -
divide_period 1
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/pins_i
n/sampling_clk3 ; #135
define_clock -name 800MHz_CLK180 -period 1250 -rise 50 -fall 0 -domain 5 -
divide_period 1
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/pins_i
n/sampling_clk4  ; #180
define_clock -name 800MHz_CLK225 -period 1250 -rise 62 -fall 12 -domain 6 -
divide_period 1
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/pins_i
n/sampling_clk5 ; #225
define_clock -name 800MHz_CLK270 -period 1250 -rise 75 -fall 25 -domain 7 -
divide_period 1
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/pins_i
n/sampling_clk6 ; #270
define_clock -name 800MHz_CLK315 -period 1250 -rise 87 -fall 37 -domain 8 -
divide_period 1
```

```
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/pins_in/sampling_clk7 ; #315


############### 1st stage of FF ##############################

define_clock -name FF_1st_clk_p0   -period 1250   -domain 1 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_p0/pins_in/clk]
define_clock -name FF_1st_clk_p45   -period 1250  -domain 2  [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_p45/pins_in/clk]
define_clock -name FF_1st_clk_p90   -period 1250   -domain 3 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_p90/pins_in/clk]
define_clock -name FF_1st_clk_p135   -period 1250  -domain 4 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_p135/pins_in/clk]
define_clock -name FF_1st_clk_p180   -period 1250  -domain 5 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_p180/pins_in/clk]
define_clock -name FF_1st_clk_p225   -period 1250  -domain 6 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_p225/pins_in/clk]
define_clock -name FF_1st_clk_p270   -period 1250   -domain 7 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_p270/pins_in/clk]
define_clock -name FF_1st_clk_p315   -period 1250  -domain 8  [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_p315/pins_in/clk]

############### 2nd stage of FF ##############################
#Clock domain is 315phase

define_clock -name FF_2nd_clk_0 -period 1250 -rise 0 -fall 50 -domain 8 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_0_2nd/pins_in/clk]
define_clock -name FF_2nd_clk_45 -period 1250 -rise 0 -fall 50 -domain 8 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_45_2nd/pins_in/clk]
define_clock -name FF_2nd_clk_90 -period 1250 -rise 0 -fall 50 -domain 8 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/FF_90_2nd/pins_in/clk]
```

```
define_clock -name FF_2nd_clk_135 -period 1250 -rise 0 -fall 50 -domain 8 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/FF_135_2nd/pins_in/clk]
define_clock -name FF_2nd_clk_180 -period 1250 -rise 0 -fall 50 -domain 8 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/FF_180_2nd/pins_in/clk]
define_clock -name FF_2nd_clk_225 -period 1250 -rise 0 -fall 50 -domain 8 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/FF_225_2nd/pins_in/clk]
define_clock -name FF_2nd_clk_270 -period 1250 -rise 0 -fall 50 -domain 8 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/FF_270_2nd/pins_in/clk]



################## Xor stage-Neg 225° domain 6 (1 cycle to launch and
capture)#################################################
#Create a negated clock from phase 6 clock

define_clock -name xorclk1  -period 1250 -rise 12 -fall 62 -domain 2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/xor_2/instances_seq/X1_2_reg_reg/pins_in/CLK]
define_clock -name xorclk2  -period 1250 -rise 12 -fall 62 -domain 2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/xor_2/instances_seq/X2_3_reg_reg/pins_in/CLK]
define_clock -name xorclk3  -period 1250 -rise 12 -fall 62 -domain 2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/xor_2/instances_seq/X3_4_reg_reg/pins_in/CLK]
define_clock -name xorclk4  -period 1250 -rise 12 -fall 62 -domain 2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/xor_2/instances_seq/X4_5_reg_reg/pins_in/CLK]
define_clock -name xorclk5  -period 1250 -rise 12 -fall 62 -domain 2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/xor_2/instances_seq/X5_6_reg_reg/pins_in/CLK]
define_clock -name xorclk6  -period 1250 -rise 12 -fall 62 -domain 2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/xor_2/instances_seq/X6_7_reg_reg/pins_in/CLK]
define_clock -name xorclk7  -period 1250 -rise 12 -fall 62 -domain 2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/xor_2/instances_seq/X7_8_reg_reg/pins_in/CLK]
define_clock -name xorclk8  -period 1250 -rise 12 -fall 62 -domain 2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/xor_2/instances_seq/X8ant_1_reg_reg/pins_in/CLK
]
```

```
##########  Neg 180 clock domain 5, 180deg
#################################################################

#/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/div8mod1/instances_seq/counter_reg_reg[0]/pins_in/CLK
#/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/div8mod1/instances_seq/counter_reg_reg[1]/pins_in/CLK

define_clock -name Clkin_Detectedge -period 1250 -rise 0 -fall 50 -domain 8 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/div8mod1/pins_in/clk]



create_generated_clock \
        -name not_180clk \
        -domain 8 \
        -source [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/div8mod1/pins_in/clk] \
        -divide_by 4 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/div8mod1/pins_out/flag ]

create_generated_clock \
        -name clkdiv8mod2 \
        -domain 10 \
        -source [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/div8mod1/pins_out/flag] \
        -divide_by 2 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/div8mod2/pins_out/flag ]

create_generated_clock \
        -name clk_div3 \
        -domain 11 \
        -source [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/div8mod2/pins_out/flag] \
        -divide_by 3 [get_pins /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instances_hier/Edge_detector/instances_hier/div3mod/pins_out/flag ]
```

```
create_generated_clock \
        -name clk_div5 \
        -domain 12 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/div3mod/pins_out/flag] \
        -divide_by 5 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/div5mod/pins_out/flag ]


create_generated_clock \
        -name clk_oneshot \
        -domain 10 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/div8mod2/pins_out/flag] \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/onesht/pins_in/clk]


####for second stage FF only 315 Flip flop
define_clock -name FF_2nd_clk_315 -period 1250 -rise 0 -fall 50 -domain 1  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/FF_315_2nd/pins_in/clk]

####for save previous Xor 8 Flip flop (probably this can use the 315 clock )

define_clock -name prevXor8clk -period 1250 -rise 0 -fall 50 -domain 1  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/save_previous_Xor8/pins_in/clk]

####for counting stage . It uses 180phase

define_clock -name CountingStageclk12 -period 1250 -rise 0 -fall 50 -domain 5  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/counter_1_2/pins_in/clk]
define_clock -name CountingStageclk23 -period 1250 -rise 0 -fall 50 -domain 5  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/counter_2_3/pins_in/clk]
define_clock -name CountingStageclk34 -period 1250 -rise 0 -fall 50 -domain 5  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/counter_3_4/pins_in/clk]
```

```
define_clock -name CountingStageclk45 -period 1250 -rise 0 -fall 50 -domain 5  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/counter_4_5/pins_in/clk]
define_clock -name CountingStageclk56 -period 1250 -rise 0 -fall 50 -domain 5  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/counter_5_6/pins_in/clk]
define_clock -name CountingStageclk67 -period 1250 -rise 0 -fall 50 -domain 5  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/counter_6_7/pins_in/clk]
define_clock -name CountingStageclk78 -period 1250 -rise 0 -fall 50 -domain 5  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/counter_7_8/pins_in/clk]
define_clock -name CountingStageclk81 -period 1250 -rise 0 -fall 50 -domain 5  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/Edge_detector/instances_hier/counter_8_1/pins_in/clk]

####for flip flops stage between detect_edge module and phase selector mode.

define_clock -name FF_edgePhaseSelclk12 -period 1250 -rise 0 -fall 50 -domain 1
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/FF_edgePhaseSel12/pins_in/clk]
define_clock -name FF_edgePhaseSelclk23 -period 1250 -rise 0 -fall 50 -domain 1
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/FF_edgePhaseSel23/pins_in/clk]
define_clock -name FF_edgePhaseSelclk34 -period 1250 -rise 0 -fall 50 -domain 1
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/FF_edgePhaseSel34/pins_in/clk]
define_clock -name FF_edgePhaseSelclk45 -period 1250 -rise 0 -fall 50 -domain 1
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/FF_edgePhaseSel45/pins_in/clk]
define_clock -name FF_edgePhaseSelclk56 -period 1250 -rise 0 -fall 50 -domain 1
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/FF_edgePhaseSel56/pins_in/clk]
define_clock -name FF_edgePhaseSelclk67 -period 1250 -rise 0 -fall 50 -domain 1
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/FF_edgePhaseSel67/pins_in/clk]
define_clock -name FF_edgePhaseSelclk78 -period 1250 -rise 0 -fall 50 -domain 1
[get_pins
```

```
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/FF_edgePhaseSel78/pins_in/clk]
define_clock -name FF_edgePhaseSelclk81 -period 1250 -rise 0 -fall 50 -domain 1
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/FF_edgePhaseSel81/pins_in/clk]
#-------------------------------------------------------------------------------------
## *************   for Phase selector mode start
**********************************************


define_clock -name not180clkFdelay1 -period 1250 -rise 0 -fall 50  -domain 1  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_delay1/pins_in/clk]


#Clocks for C1 module

define_clock -name FFdelay1clk1 -period 1250 -rise 0 -fall 50  -domain 1  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/c1/pins_in/clk]

define_clock -name FFdelay1clk2 -period 2500 -rise 0 -fall 50  -domain 1  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_delay2/pins_in/clk]

define_clock -name FFdelay1clk3 -period 3750 -rise 0 -fall 50  -domain 1  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_delay3/pins_in/clk]

create_generated_clock \
        -name delay3Clk1 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/c1/pins_in/clk ] \
        -divide by 3 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/c1/pins_out/clk]

##########################################################################
####################
########################################## Comp1
##########################################
```

```
#define_clock -name delay1_comp1 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/pins_in/clk]

define_clock -name delay2_comp1 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_delay2/pins_in/clk]

create_generated_clock \
        -name delay3_comp1 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/pins_in/clk] \
        -invert \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_delay3/pins_in/clk]


#Signal 1

define_clock -name DataValue1_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Value1_input/pins_in/
clk]

define_clock -name signal1_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name signal1_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name signal1_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name signal1_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]
```

```
#signal 2

define_clock -name DataValue2_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name signal2_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name signal2_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name signal2_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name signal2_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

#Value4phase

define_clock -name clkVal4ph -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[0]/pins_in/CLK]

#Datamux

define_clock -name C1clkDatamux_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Datamux/instances_se
q/output_data_reg[0]/pins_in/CLK]

define_clock -name C1clkDatamux_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
```

```
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Datamux/instances_se
q/output_data_reg[1]/pins_in/CLK]

define_clock -name C1clkDatamux_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Datamux/instances_se
q/output_data_reg[2]/pins_in/CLK]

define_clock -name C1clkDatamux_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp1/instances_hier/FF_Datamux/instances_se
q/output_data_reg[3]/pins_in/CLK]

############################################ Comp1
############################################
##########################################################################
####################


##########################################################################
####################
############################################ Comp2
############################################

define_clock -name C2not180clkFdelay1 -period 1250 -rise 0 -fall 50  -domain 13
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_delay1/pins_in/clk]

define_clock -name C2FFdelay1clk1 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/c1/pins_in/clk]

define_clock -name C2FFdelay1clk2 -period 2500 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_delay2/pins_in/clk]

define_clock -name C2FFdelay1clk3 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_delay3/pins_in/clk]


define_clock -name delay2_comp2 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_delay2/pins_in/clk]
```

```
create_generated_clock \
        -name delay3_comp2 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/pins_in/clk] \
        -invert \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_delay3/pins_in/clk]


#Signal 1

define_clock -name C2DataValue1_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Value1_input/pins_in/
clk]

define_clock -name C2signal1_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C2signal1_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C2signal1_2 -period 3750 -rise 0 -fall 50 -domain 13 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C2signal1_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

#signal 2

define_clock -name C2DataValue2_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[0]/pins_in/CLK]
```

```
define_clock -name C2signal2_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C2signal2_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C2signal2_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C2signal2_3 -period 3750 -rise 0 -fall 50 -domain 13 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

define_clock -name C2clkVal4ph_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[0]/pins_in/CLK]

define_clock -name C2clkVal4ph_1 -period 3750 -rise 0 -fall 50 -domain 13 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[1]/pins_in/CLK]

define_clock -name C2clkVal4ph_2 -period 3750 -rise 0 -fall 50 -domain 13 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[2]/pins_in/CLK]


#Datamux

define_clock -name C2clkDatamux_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Datamux/instances_se
q/output_data_reg[0]/pins_in/CLK]
```

```
define_clock -name C2clkDatamux_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Datamux/instances_se
q/output_data_reg[1]/pins_in/CLK]

define_clock -name C2clkDatamux_2 -period 3750 -rise 0 -fall 50 -domain 13 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Datamux/instances_se
q/output_data_reg[2]/pins_in/CLK]

define_clock -name C2clkDatamux_3 -period 3750 -rise 0 -fall 50 -domain 13 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp2/instances_hier/FF_Datamux/instances_se
q/output_data_reg[3]/pins_in/CLK]
############################################ Comp2
############################################
##########################################################################
####################

##########################################################################
####################
############################################ Comp3
############################################

define_clock -name C3not180clkFdelay1 -period 1250 -rise 0 -fall 50  -domain 13
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_delay1/pins_in/clk]

define_clock -name C3FFdelay1clk1 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/c1/pins_in/clk]

define_clock -name C3FFdelay1clk2 -period 2500 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_delay2/pins_in/clk]

define_clock -name C3FFdelay1clk3 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_delay3/pins_in/clk]


define_clock -name delay2_comp3 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_delay2/pins_in/clk]
```

```
create_generated_clock \
        -name delay3_comp3 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/pins_in/clk] \
        -invert \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_delay3/pins_in/clk]


#Signal 1

define_clock -name C3DataValue1_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Value1_input/pins_in/
clk]

define_clock -name C3signal1_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C3signal1_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C3signal1_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C3signal1_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

#signal 2

define_clock -name C3DataValue2_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[0]/pins_in/CLK]
```

```
define_clock -name C3signal2_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C3signal2_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C3signal2_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C3signal2_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

define_clock -name C3clkVal4ph_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[0]/pins_in/CLK]

define_clock -name C3clkVal4ph_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[1]/pins_in/CLK]

define_clock -name C3clkVal4ph_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[2]/pins_in/CLK]


#Datamux

define_clock -name C3clkDatamux_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Datamux/instances_se
q/output_data_reg[0]/pins_in/CLK]
```

```
define_clock -name C3clkDatamux_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Datamux/instances_se
q/output_data_reg[1]/pins_in/CLK]

define_clock -name C3clkDatamux_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Datamux/instances_se
q/output_data_reg[2]/pins_in/CLK]

define_clock -name C3clkDatamux_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp3/instances_hier/FF_Datamux/instances_se
q/output_data_reg[3]/pins_in/CLK]
############################################ Comp3
############################################
##########################################################################
####################


##########################################################################
####################
############################################ Comp4
############################################

define_clock -name C4not180clkFdelay1 -period 1250 -rise 0 -fall 50  -domain 13
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_delay1/pins_in/clk]


define_clock -name C4FFdelay1clk1 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/c1/pins_in/clk]

define_clock -name C4FFdelay1clk2 -period 2500 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_delay2/pins_in/clk]

define_clock -name C4FFdelay1clk3 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_delay3/pins_in/clk]
```

```
define_clock -name delay2_comp4 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_delay2/pins_in/clk]

create_generated_clock \
        -name delay3_comp4 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/pins_in/clk] \
        -invert \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_delay3/pins_in/clk]


#Signal 1

define_clock -name C4DataValue1_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Value1_input/pins_in/
clk]

define_clock -name C4signal1_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C4signal1_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C4signal1_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C4signal1_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

#signal 2
```

```
define_clock -name C4DataValue2_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C4signal2_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C4signal2_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C4signal2_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C4signal2_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

define_clock -name C4clkVal4ph_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[0]/pins_in/CLK]

define_clock -name C4clkVal4ph_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[1]/pins_in/CLK]

define_clock -name C4clkVal4ph_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[2]/pins_in/CLK]


#Datamux

define_clock -name C4clkDatamux_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
```

```
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Datamux/instances_se
q/output_data_reg[0]/pins_in/CLK]

define_clock -name C4clkDatamux_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Datamux/instances_se
q/output_data_reg[1]/pins_in/CLK]

define_clock -name C4clkDatamux_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Datamux/instances_se
q/output_data_reg[2]/pins_in/CLK]

define_clock -name C4clkDatamux_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp4/instances_hier/FF_Datamux/instances_se
q/output_data_reg[3]/pins_in/CLK]
########################################### Comp4
#########################################
##########################################################################
###################

##########################################################################
###################
########################################### Comp5
#########################################

define_clock -name C5not180clkFdelay1 -period 1250 -rise 0 -fall 50  -domain 13
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_delay1/pins_in/clk]

define_clock -name C5FFdelay1clk1 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/c1/pins_in/clk]

define_clock -name C5FFdelay1clk2 -period 2500 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_delay2/pins_in/clk]

define_clock -name C5FFdelay1clk3 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_delay3/pins_in/clk]
```

```
define_clock -name delay2_comp5 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_delay2/pins_in/clk]

create_generated_clock \
        -name delay3_comp5 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/pins_in/clk] \
        -invert \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_delay3/pins_in/clk]


#Signal 1

define_clock -name C5DataValue1_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Value1_input/pins_in/
clk]

define_clock -name C5signal1_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C5signal1_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C5signal1_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C5signal1_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

#signal 2
```

```
define_clock -name C5DataValue2_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[0]/pins_in/CLK]


define_clock -name C5DataValue2_clk1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C5DataValue2_clk2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[2]/pins_in/CLK]


define_clock -name C5signal2_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C5signal2_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C5signal2_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C5signal2_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

define_clock -name C5clkVal4ph_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[0]/pins_in/CLK]

define_clock -name C5clkVal4ph_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
```

```
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[1]/pins_in/CLK]

define_clock -name C5clkVal4ph_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[2]/pins_in/CLK]


#Datamux

define_clock -name C5clkDatamux_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Datamux/instances_se
q/output_data_reg[0]/pins_in/CLK]

define_clock -name C5clkDatamux_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Datamux/instances_se
q/output_data_reg[1]/pins_in/CLK]

define_clock -name C5clkDatamux_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Datamux/instances_se
q/output_data_reg[2]/pins_in/CLK]

define_clock -name C5clkDatamux_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp5/instances_hier/FF_Datamux/instances_se
q/output_data_reg[3]/pins_in/CLK]
########################################### Comp5
########################################
##########################################################################
###################

##########################################################################
###################
########################################### Comp6
########################################

define_clock -name C6not180clkFdelay1 -period 1250 -rise 0 -fall 50  -domain 13
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_delay1/pins_in/clk]
```

```
define_clock -name C6FFdelay1clk1 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/c1/pins_in/clk]

define_clock -name C6FFdelay1clk2 -period 2500 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_delay2/pins_in/clk]

define_clock -name C6FFdelay1clk3 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_delay3/pins_in/clk]

define_clock -name delay2_comp6 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_delay2/pins_in/clk]



define_clock -name C6_inclk -period 1250 -rise 0 -fall 50  -domain 13
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/pins_in/clk

define_clock -name C6FFdelayc1 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/c1/pins_in/clk]

define_clock -name C6FFdelay1clk1 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_delay1/pins_in/clk]

define_clock -name C6FFdelay1clk2 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_delay2/pins_in/clk]

define_clock -name C6FFdelay1clk3 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_delay3/pins_in/clk]



create_generated_clock \
        -name delay3_comp6 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/pins_in/clk] \
        -invert \
```

```
        [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_delay3/pins_in/clk]



#Signal 1

define_clock -name C6DataValue1_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Value1_input/pins_in/
clk]


define_clock -name C6signal1_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C6signal1_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C6signal1_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C6signal1_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

#signal 2

define_clock -name C6DataValue2_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C6DataValue2_clk1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[1]/pins_in/CLK]
```

```
define_clock -name C6signal2_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C6signal2_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C6signal2_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C6signal2_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

define_clock -name C6clkVal4ph_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[0]/pins_in/CLK]

define_clock -name C6clkVal4ph_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[1]/pins_in/CLK]

define_clock -name C6clkVal4ph_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[2]/pins_in/CLK]


#Datamux

define_clock -name C6clkDatamux_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Datamux/instances_se
q/output_data_reg[0]/pins_in/CLK]
```

```
define_clock -name C6clkDatamux_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Datamux/instances_se
q/output_data_reg[1]/pins_in/CLK]

define_clock -name C6clkDatamux_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Datamux/instances_se
q/output_data_reg[2]/pins_in/CLK]

define_clock -name C6clkDatamux_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp6/instances_hier/FF_Datamux/instances_se
q/output_data_reg[3]/pins_in/CLK]
############################################ Comp6
############################################
##########################################################################
####################

##########################################################################
####################
############################################ Comp7
############################################

define_clock -name C7not180clkFdelay1 -period 1250 -rise 0 -fall 50  -domain 13
[get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_delay1/pins_in/clk]

define_clock -name C7_inclk -period 1250 -rise 0 -fall 50  -domain 13
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/pins_in/clk

define_clock -name C7FFdelayc1 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/c1/pins_in/clk]

define_clock -name C7FFdelay1clk1 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_delay1/pins_in/clk]

define_clock -name C7FFdelay1clk2 -period 1250 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_delay2/pins_in/clk]
```

```
define_clock -name C7FFdelay1clk3 -period 3750 -rise 0 -fall 50  -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_delay3/pins_in/clk]



create_generated_clock \
        -name delay3_comp7 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/pins_in/clk] \
        -invert \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_delay3/pins_in/clk]



#Signal 1

define_clock -name C7DataValue1_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Value1_input/pins_in/
clk]



define_clock -name C7signal1_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C7signal1_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C7signal1_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C7signal1_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Signal1_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]
```

```
#signal 2


define_clock -name C7DataValue2_clk1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C7DataValue2_clk2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C7DataValue2_clk0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Value2_input/instance
s_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C7signal2_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[0]/pins_in/CLK]

define_clock -name C7signal2_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[1]/pins_in/CLK]

define_clock -name C7signal2_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[2]/pins_in/CLK]

define_clock -name C7signal2_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Signal2_input/instanc
es_seq/output_data_reg[3]/pins_in/CLK]

define_clock -name C7clkVal4ph_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[0]/pins_in/CLK]

define_clock -name C7clkVal4ph_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
```

```
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[1]/pins_in/CLK]

define_clock -name C7clkVal4ph_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Value4ph/instances_s
eq/output_data_reg[2]/pins_in/CLK]


#Datamux

define_clock -name C7clkDatamux_0 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Datamux/instances_se
q/output_data_reg[0]/pins_in/CLK]

define_clock -name C7clkDatamux_1 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Datamux/instances_se
q/output_data_reg[1]/pins_in/CLK]

define_clock -name C7clkDatamux_2 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Datamux/instances_se
q/output_data_reg[2]/pins_in/CLK]

define_clock -name C7clkDatamux_3 -period 3750 -rise 0 -fall 50 -domain 13  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/phase_select_mod/instances_hier/comp7/instances_hier/FF_Datamux/instances_se
q/output_data_reg[3]/pins_in/CLK]




########################################### Comp7
###########################################
##########################################################################
####################
#---------------------------------------


############### for adaptive update phase module
define_clock -name allowupdclk0 -period 5000 -rise 0 -fall 50 -domain 14  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/allow_updateCount/instances_seq/selector_final_reg_reg[0]/pins_in/CLK]
```

```
define_clock -name allowupdclk1 -period 5000 -rise 0 -fall 50 -domain 14  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/allow_updateCount/instances_seq/selector_final_reg_reg[1]/pins_in/CLK]

define_clock -name allowupdclk2 -period 5000 -rise 0 -fall 50 -domain 14  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/allow_updateCount/instances_seq/selector_final_reg_reg[2]/pins_in/CLK]



#create_generated_clock \
        -name allowupdclk0 \
        -domain 1 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/allow_updateCount/instances_seq/selector_final_reg_reg[0]/pins_in/CLK] \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/allow_updateCount/pins_out/selector_final[0]]

#create_generated_clock \
        -name allowupdclk1 \
        -domain 1 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/allow_updateCount/instances_seq/selector_final_reg_reg[0]/pins_in/CLK] \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/allow_updateCount/pins_out/selector_final[1]]

#create_generated_clock \
        -name allowupdclk2 \
        -domain 1 \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/allow_updateCount/instances_seq/selector_final_reg_reg[0]/pins_in/CLK] \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/allow_updateCount/pins_out/selector_final[2]]

##################################### end
#########################################################



################## Other Flip flops domain 11
###########################################
```

```
## This will use as clock the signal "flag_to_reset_counting" but negated
define_clock -name save_selecClk1 -period 311250 -rise 3 -fall 6 -domain 15
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/save_previousSel/clk

#This will use as clock the signal "flag_to_startcounting_decision". It is a pulse generated
after 12 data
define_clock -name save_selecClk3 -period 150000 -rise 0 -fall 3 -domain 16
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/save_Newselection/clk

##################################### end
#################################################



################## Mux-phase domain 15
########################################
#Define the multiple clocks which can come from the input of the multiplexer

create_clock -name muxin1 -period 1250  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected]
create_clock -name muxin2 -period 1250  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] -add
create_clock -name muxin3 -period 1250  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] -add
create_clock -name muxin4 -period 1250  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] -add
create_clock -name muxin5 -period 1250  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] -add
create_clock -name muxin6 -period 1250  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] -add
create_clock -name muxin7 -period 1250  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] -add
create_clock -name muxin8 -period 1250  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] -add
```

```
create_generated_clock -name clkgen1 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] \
        -master_clock muxin1 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_out/clk_recovered]

create_generated_clock -name clkgen2 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] \
        -master_clock muxin2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_out/clk_recovered] -add

create_generated_clock -name clkgen3 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] \
        -master_clock muxin3 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_out/clk_recovered] -add

create_generated_clock -name clkgen4 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] \
        -master_clock muxin4 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_out/clk_recovered] -add

create_generated_clock -name clkgen5 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] \
        -master_clock muxin5 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_out/clk_recovered] -add

create_generated_clock -name clkgen6 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] \
```

```
        -master_clock muxin6 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_out/clk_recovered] -add

create_generated_clock -name clkgen7 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] \
        -master_clock muxin7 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_out/clk_recovered] -add

create_generated_clock -name clkgen8 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_in/clk_selected] \
        -master_clock muxin8 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_out/clk_recovered] -add


################### Sampling original data domain 14
#########################################

#Negated clock for the sampling flip flop

create_generated_clock \
        -name divClk_FFrecovered \
        -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/clkgenT/pins_out/clk_recovered] \
        -invert \
         [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/instan
ces_hier/FF_recovered/pins_in/clk]


##############################################################################
#################
################################### LFSR constraints
#######################################

# Clock definition
#Use 200MHz
```

```
#Define the multiple clocks which can come from the input of the multiplexer


create_clock -name LSFRmuxin1 -period 5000  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/modlsfr/instances_
hier/LFSR/pins_in/clk]
create_clock -name LSFRmuxin2 -period 5000  [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/modlsfr/instances_
hier/LFSR/pins_in/clk] -add

create_generated_clock -name clk_LFSRgen1 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/modlsfr/instances_
hier/LFSR/pins_in/clk] \
        -master_clock LSFRmuxin1 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/modlsfr/instances_
hier/LFSR/pins_out/LFSR_Data_Out]

create_generated_clock -name clk_LFSRgen2 \
        -divide_by 1 -source [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/modlsfr/instances_
hier/LFSR/pins_in/clk] \
        -master_clock LSFRmuxin2 [get_pins
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/modlsfr/instances_
hier/LFSR/pins_out/LFSR_Data_Out] -add


define_clock -name 300Mhz_CLK -period 5000 -rise 90 -fall 90 -domain 12
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/LFSR_TOP/pins_i
n/clk

# slew rate definitions (min rise, min fall, max rise, max fall).
# The values coming from IBM typical specification.
#set_attribute slew { 56 56 56 56 } 300Mhz_CLK

# network clock latency
#set_attribute clock_network_late_latency 100 300Mhz_CLK
#set_attribute clock_network_early_latency 90 300Mhz_CLK
# source clock latency
#set_attribute clock_source_late_latency 50 300Mhz_CLK
#set_attribute clock_source_early_latency 40 300Mhz_CLK

# clock skew
#set_attribute clock_setup_uncertainty {17 10} 300Mhz_CLK
#set_attribute clock_hold_uncertainty {13 5} 300Mhz_CLK
```

```
# Driving cell definition
#set_attribute external_driver [find [find / -libcell BUFX2TS] -libpin Y]
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/ports_in/*


# We are considering around six times the clock slew rate.
#set_attribute max_transition 150
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/LFSR_TOP/

# The input capacitance for a NOR4X8 cell is 24.9fF  considering fanout of 5  and the
wires caps:
#set_attribute max_capacitance 130
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/LFSR/

# Considering a pad output buffer POC2A load
#set_attribute external_pin_cap 31
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/LFSR_TOP/ports_
out/*

# Setting maximum value of fanout
#set_attribute max_fanout 10 /designs/*

#set_attribute lp_power_unit {uW}
#set_attribute lp_power_optimization_weight 0.2 [current_design]

##########################################################################
#################


# Input delay definition: This is the delay coming from outside the design. Here, it's defined
at 10% of clock period.

external_delay -clock [find / -clock 800MHz_CLK0] -input 125 -name IDelay [find /des* -
port ports_in/*]
external_delay -clock [find / -clock 800MHz_CLK0] -input 125 -name IDelay [find -port
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/ports_i
n/original_data]
external_delay -clock [find / -clock 800MHz_CLK0] -input 125 -name IDelay [find -port
/designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/reset]
```

```
# Output delay definition: This is the delay going outside the design.  Here, it's defined at 10% of clock period.

external_delay -clock [find / -clock 800MHz_CLK0] -output 125 -name ODelay [find /des* -port ports_out/*]


# Driving cell definition


set_attribute external_driver [find [find / -libcell BUFFER_D] -libpin Z] { /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/ports_in/* }


# Considering a pad output buffer POC2A load

#set_attribute external_pin_cap 31 /designs/PLL_based_CDR/instances_hier/mCDR_LFSR/instances_hier/cdr_module/ports_out/*

set_attr preserve true pad_VSS
set_attr preserve true pad_VSS_D
set_attr preserve true pad_VSS_A

set_attr preserve true ne_pcorner
set_attr preserve true nw_pcorner
set_attr preserve true se_pcorner
set_attr preserve true sw_pcorner
```

J. SCRIPT AUTOMATIZACON PARA LA SÍNTESIS LÓGICA DEL SOC

```
#### Script for RTL->Gate-Level Flow (generated from RC v12.10-s012_1)
## PLL_Project
## ITESO University
## Francisco Nuñez

if {[file exists /proc/cpuinfo]} {
  sh grep "model name" /proc/cpuinfo
  sh grep "cpu MHz"    /proc/cpuinfo
}

puts "Hostname : [info hostname]"

##############################################################################
#####
## Preset global variables and attributes
##############################################################################
#####

set DESIGN PLL_based_CDR
set SYN_EFF high
###set MAP_EFF medium
set MAP_EFF high
set DATE [clock format [clock seconds] -format "%b%d-%T"]
set _OUTPUTS_PATH outputs_${DATE}
set _REPORTS_PATH reports_${DATE}
set _LOG_PATH logs_${DATE}

# Variable to specify the technology .lib file name
#set timing_library
{v.20171220/synopsys/typ_v120_t025/PnomV1p20T025_STD_CELL_8HP_12T.lib
v.20160727/synopsys/typ_v120_v150_t25/IBM_CMOS8HP_BASE_WB_IO_TYP_V120_
V150_T25.lib}

set timing_library
{v.20171220/synopsys/slow_v108_t125/PwcV1p08T125_STD_CELL_8HP_12T.lib
v.20160727/synopsys/slow_v108_v140_t125/IBM_CMOS8HP_BASE_WB_IO_SLOW_V
108_V140_T125.lib}

# Variable to specify the LEF library
```

```
set my_lef_library {
/opt/libs/IBM_PDK/bicmos8hp/v.20171220/lef/BICMOS8HP_SC_1P2V_12T_RVT.lef
/opt/libs/IBM_PDK/bicmos8hp/v.20171220/lef/bicmos8hp_7AM_41_tech.lef}

set_attribute script_search_path {. <path>} /
set_attribute hdl_search_path {. ./rtl} /

set_attribute information_level 9 /

#############################################################
## Definitions for timing-libraries serch path
#############################################################

set_attribute lib_search_path {/opt/libs/IBM_PDK/bicmos8hp} /

#####################################################################
## Setting the Target Technology Library (Library setup)
#####################################################################
set_attribute library $timing_library

# Variable that points to the technology's .lef file
## PLE
set_attribute lef_library $my_lef_library /

set_attribute auto_ungroup none /
#set_attribute auto_ungroup both /

#########################################################################
## Load Design (HDL files)
#########################################################################

read_hdl -v2001 {
../PLL_based_CDR.v
../VCO.v

../CORNER_VDD150_PM_A.v

../DIV_MUX_5v0/Div_and_Mux.v
../DIV_MUX_5v0/Divider_by_2.v
../DIV_MUX_5v0/Divider_by_4.v
../DIV_MUX_5v0/Mux_2_to_1.v
../DIV_MUX_5v0/Divider_by_8.v
../DIV_MUX_5v0/Divider_by_2_4_8.v
../DIV_MUX_5v0/T_FF.v
```

```
../CDRLFSRxRTLCompiler/CDR_LFSR.v
../CDRLFSRxRTLCompiler/CDR.v
../CDRLFSRxRTLCompiler/Phase_selector.v
../CDRLFSRxRTLCompiler/mux2to1.v
../CDRLFSRxRTLCompiler/detect_edge.v
../CDRLFSRxRTLCompiler/not_module.v
../CDRLFSRxRTLCompiler/Flipflop_data.v
../CDRLFSRxRTLCompiler/CounterEvents.v
../CDRLFSRxRTLCompiler/Allow_updateSignalSelector.v
../CDRLFSRxRTLCompiler/Mux8_to_1.v
../CDRLFSRxRTLCompiler/Flipflop_D.v
../CDRLFSRxRTLCompiler/One_Shot.v
../CDRLFSRxRTLCompiler/Comparador_maj.v
../CDRLFSRxRTLCompiler/Xor_2ndStage2.v
../CDRLFSRxRTLCompiler/xor_3.v
../CDRLFSRxRTLCompiler/clockGenerator_FFneg.v
../CDRLFSRxRTLCompiler/Top_comp4bit.v
../CDRLFSRxRTLCompiler/comp4bit.v
../CDRLFSRxRTLCompiler/Div5_clk.v
../CDRLFSRxRTLCompiler/Div8_clk.v
../CDRLFSRxRTLCompiler/Div3_clk.v
../CDRLFSRxRTLCompiler/xor_3.v
../CDRLFSRxRTLCompiler/LFSR_TOP.v
../CDRLFSRxRTLCompiler/LFSR_10_3_1.v
../CDRLFSRxRTLCompiler/Mux2to1_lfsr.v
../CDRLFSRxRTLCompiler/Register.v
../CDRLFSRxRTLCompiler/xor_lfsr.v
}

##########################################################################
## Performing Elaboration
##########################################################################

elaborate $DESIGN
puts "Runtime & Memory after 'read_hdl'"
timestat Elaboration

check_design -unresolved

##########################################################################
## Constraints Setup
##########################################################################

read_sdc {../PLL_based_CDR.sdc}
puts "The number of exceptions is [llength [find /designs/$DESIGN -exception *]]"
```

```
#set_attribute force_wireload <wireload name> "/designs/$DESIGN"

if {![file exists ${_LOG_PATH}]} {
  file mkdir ${_LOG_PATH}
  puts "Creating directory ${_LOG_PATH}"
}

if {![file exists ${_OUTPUTS_PATH}]} {
  file mkdir ${_OUTPUTS_PATH}
  puts "Creating directory ${_OUTPUTS_PATH}"
}

if {![file exists ${_REPORTS_PATH}]} {
  file mkdir ${_REPORTS_PATH}
  puts "Creating directory ${_REPORTS_PATH}"
}

puts "####################"
puts "Timing -lint Report"
puts "####################"
# The following report shows any synthesis timing problems
report timing -lint -verbose

################################################################################
##########
## Define cost groups (clock-clock, clock-output, input-clock, input-output)
################################################################################
##########

## Uncomment to remove already existing costgroups before creating new ones.
## rm [find /designs/* -cost_group *]

if {[llength [all::all_seqs]] > 0} {
  define_cost_group -name I2C -design $DESIGN
  define_cost_group -name C2O -design $DESIGN
  define_cost_group -name C2C -design $DESIGN
  path_group -from [all::all_seqs] -to [all::all_seqs] -group C2C -name C2C
  path_group -from [all::all_seqs] -to [all::all_outs] -group C2O -name C2O
  path_group -from [all::all_inps]  -to [all::all_seqs] -group I2C -name I2C
}

define_cost_group -name I2O -design $DESIGN
#path_group -from [all::all_inps]  -to [all::all_outs] -group I2O -name I2O
foreach cg [find / -cost_group *] {
  report timing -cost_group [list $cg] >> ${DESIGN}_pretim.rpt
```

```
}


################################################################################
############################
## Performing Synthesis
## Synthesizing to generic
################################################################################
############################

synthesize -to_generic -eff $SYN_EFF
puts "Runtime & Memory after 'synthesize -to_generic'"
timestat GENERIC
write_hdl       > ${DESIGN}_generic.v
report datapath > $_REPORTS_PATH/${DESIGN}_datapath_generic.rpt
report gates    > $_REPORTS_PATH/${DESIGN}_gates_generic.rpt
report area     > $_REPORTS_PATH/${DESIGN}_area_generic.rpt
report qor      > $_REPORTS_PATH/${DESIGN}_qor_generic.rpt
report messages > $_REPORTS_PATH/${DESIGN}_messages_generic.rpt

generate_reports -outdir $_REPORTS_PATH/resume_rpts -tag generic
summary_table -outdir $_REPORTS_PATH

################################################################################
############################
## Performing Synthesis
## Synthesizing to gates
################################################################################
############################

synthesize -to_mapped -eff $MAP_EFF -no_incr
puts "Runtime & Memory after 'synthesize -to_map -no_incr'"
timestat MAPPED
write_hdl       > ${DESIGN}_m_noincr.v
report datapath > $_REPORTS_PATH/${DESIGN}_datapath_m_noincr.rpt
report gates    > $_REPORTS_PATH/${DESIGN}_gates_m_noincr.rpt
report area     > $_REPORTS_PATH/${DESIGN}_area_m_noincr.rpt
report qor      > $_REPORTS_PATH/${DESIGN}_qor_m_noincr.rpt
report messages > $_REPORTS_PATH/${DESIGN}_messages_m_noincr.rpt
report power -depth 0 > $_REPORTS_PATH/${DESIGN}_power_m_noincr.rpt

foreach cg [find / -cost_group *] {
  report timing -cost_group [list $cg] > $_REPORTS_PATH/${DESIGN}_[basename
$cg]_post_map.rpt
}
```

```
generate_reports -outdir $_REPORTS_PATH/resume_rpts -tag map
summary_table -outdir $_REPORTS_PATH

##Intermediate netlist for LEC verification..
write_hdl -lec > ${_OUTPUTS_PATH}/${DESIGN}_intermediate.v
write_do_lec -verbose -no_exit -revised_design
${_OUTPUTS_PATH}/${DESIGN}_intermediate.v -logfile
${_LOG_PATH}/rtl2intermediate.lec.log > ${_OUTPUTS_PATH}/rtl2intermediate.lec.do

################################################################################
##############################
## Performing Synthesis
## Incremental Synthesis
################################################################################
##############################

synthesize -to_mapped -eff $MAP_EFF -incr
puts "Runtime & Memory after incremental synthesis"
timestat INCREMENTAL
write_hdl       > ${DESIGN}_m.v
report gates    > $_REPORTS_PATH/${DESIGN}_gates_m.rpt
report area     > $_REPORTS_PATH/${DESIGN}_area_m.rpt
report timing   > $_REPORTS_PATH/${DESIGN}_timing_m.rpt
report qor      > $_REPORTS_PATH/${DESIGN}_qor_m.rpt
report datapath > $_REPORTS_PATH/${DESIGN}_datapath_m.rpt
report messages > $_REPORTS_PATH/${DESIGN}_messages_m.rpt
report power -depth 0 > $_REPORTS_PATH/${DESIGN}_power_m.rpt

foreach cg [find / -cost_group *] {
  report timing -cost_group [list $cg] > $_REPORTS_PATH/${DESIGN}_[basename
$cg]_post_incr.rpt
}

generate_reports -outdir $_REPORTS_PATH/resume_rpts -tag incremental
summary_table -outdir $_REPORTS_PATH

#write_sdc > ${_OUTPUTS_PATH}/${DESIGN}_m_Typ.sdc
write_sdc > ${_OUTPUTS_PATH}/${DESIGN}_m_WC.sdc

################################
### write_do_lec
################################
```

```
write_do_lec -verbose -no_exit -golden_design
${_OUTPUTS_PATH}/${DESIGN}_intermediate.v -revised_design
${_OUTPUTS_PATH}/${DESIGN}_m.v -logfile
${_LOG_PATH}/intermediate2final.lec.log >
${_OUTPUTS_PATH}/intermediate2final.lec.do
##Uncomment if the RTL is to be compared with the final netlist..
write_do_lec -verbose -no_exit -revised_design ${_OUTPUTS_PATH}/${DESIGN}_m.v
-logfile ${_LOG_PATH}/rtl2final.lec.log > ${_OUTPUTS_PATH}/rtl2final.lec.do

puts "Final Runtime & Memory."
timestat FINAL
puts "============================"
puts "Synthesis Finished ....."
puts "============================"
report power
report design_rules
report clocks
```

K. SCRIPT PROCESO DE IMPORTACIÓN DE DISEÑO SÍNTESIS FÍSICA DEL SOC

```
#################################################################
#  Generated by:      Cadence Encounter 14.27-s035_1
#  OS:                Linux x86_64(Host ID fv00)
#  Generated on:      Wed Mar 21 20:18:39 2018
#  Design:
#  Command:           save_global ../Default.globals
#################################################################
#
# Version 1.1
#

set ::TimeLib::tsgMarkCellLatchConstructFlag 1
set conf_qxconf_file {NULL}
set conf_qxlib_file {NULL}
set defHierChar {/}
set distributed_client_message_echo {1}
set gpsPrivate::dpgNewAddBufsDBUpdate 1
set gpsPrivate::lsgEnableNewDbApiInRestruct 1
set init_gnd_net {VSS}
set init_io_file {../PLL_based_CDR.ioc}
set init_lef_file
{/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/bicmos8hp_7AM_41_tech.lef
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/BICMOS8HP_SC_1P2V_12T_RVT
.lef
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/lef/CMOS8HP_BASE_WB_IO_7LM.l
ef
/home/fnunez/Cadence_Desings/EDI/bicmos8hp/PLL_based_CDR_2v0/BIAS_CIRCUIT_
1.lef
/home/fnunez/Cadence_Desings/EDI/bicmos8hp/PLL_based_CDR_2v0/differentialPFD_c
hargepump_filterloop_V4_LEF.lef}
set init_mmmc_file {../PLL_based_CDR_Typ_WC_analysis.view}
set init_pwr_net {VDD}
set init_top_cell {PLL_based_CDR}
set init_verilog {../PLL_based_CDR_m.v}
set lsgOCPGainMult 1.000000
set pegDefaultResScaleFactor 1.000000
set pegDetailResScaleFactor 1.000000
set timing_library_float_precision_tol 0.000010
set timing_library_load_pin_cap_indices {}
set tso_post_client_restore_command {update_timing ; write_eco_opt_db ;}
# Version:1.0 MMMC View Definition File
```

```
# Do Not Remove Above Line
create_rc_corner -name Typ_RC_Corner -T {25} -preRoute_res {1.0} -preRoute_cap {1.0}
-preRoute_clkres {0.0} -preRoute_clkcap {0.0} -postRoute_res {1.0} -postRoute_cap
{1.0} -postRoute_xcap {1.0} -postRoute_clkres {0.0} -postRoute_clkcap {0.0}
create_rc_corner -name WC_RC_Corner -T {125} -preRoute_res {1.0} -preRoute_cap
{1.0} -preRoute_clkres {0.0} -preRoute_clkcap {0.0} -postRoute_res {1.0} -
postRoute_cap {1.0} -postRoute_xcap {1.0} -postRoute_clkres {0.0} -postRoute_clkcap
{0.0}
create_library_set -name Typ_timing_lib -timing
{/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/synopsys/typ_v150_t025/PnomV1p50
T025_STD_CELL_8HP_12T.lib
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/synopsys/typ_v150_v150_t25/IBM_C
MOS8HP_BASE_WB_IO_TYP_V150_V150_T25.lib}
create_library_set -name WC_timing_lib -timing
{/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/synopsys/slow_v108_t125/PwcV1p08
T125_STD_CELL_8HP_12T.lib
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20160727/synopsys/slow_v108_v140_t125/IBM_
CMOS8HP_BASE_WB_IO_SLOW_V108_V140_T125.lib}
create_constraint_mode -name Typ_Constraint_Mode -sdc_files
{../PLL_based_CDR_m_Typ.sdc}
create_constraint_mode -name WC_Constraint_Mode -sdc_files
{../PLL_based_CDR_m_WC.sdc}
create_delay_corner -name Typ_Delay_Corner -library_set {Typ_timing_lib} -rc_corner
{Typ_RC_Corner}
create_delay_corner -name WC_Delay_Corner -library_set {WC_timing_lib} -rc_corner
{WC_RC_Corner}
create_analysis_view -name Typ_Analysis_View -constraint_mode
{Typ_Constraint_Mode} -delay_corner {Typ_Delay_Corner}
create_analysis_view -name WC_Analysis_View -constraint_mode
{WC_Constraint_Mode} -delay_corner {WC_Delay_Corner}
set_analysis_view -setup {WC_Analysis_View} -hold {Typ_Analysis_View}
```

L. SCRIPT DEFINICIÓN FLOORPLAN Y CREACIÓN POWER GRID DEL SOC

```
# ITESO University
# Francisco Nuñez


# Defining process mode
setDesignMode -process 130

# Defining floorplan
getIoFlowFlag
#setFPlanRowSpacingAndType 0 2
setIoFlowFlag 0
floorPlan -dieSizeByIoHeight max -site CORE -s 643.2 643.2 16 16 16 16

# Defining global nets
clearGlobalNets
globalNetConnect VDD -type pgpin -pin VDD -inst * -module {} -verbose
globalNetConnect VSS -type pgpin -pin VSS -inst * -module {} -verbose
globalNetConnect VDD -type tiehi -pin VDD -inst * -module {} -verbose
globalNetConnect VSS -type tielo -pin VSS -inst * -module {} -verbose

# Creating power rins
set sprCreateIeRingNets {}
set sprCreateIeRingLayers {}
set sprCreateIeRingWidth 1.0
set sprCreateIeRingSpacing 1.0
set sprCreateIeRingOffset 1.0
set sprCreateIeRingThreshold 1.0
set sprCreateIeRingJogDistance 1.0
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell -
stacked_via_top_layer AM -type core_rings -jog_distance 0.2 -threshold 0.2 -nets {VDD
VSS} -follow core -stacked_via_bottom_layer M1 -layer {bottom M1 top M1 right M2 left
M2} -width 5 -spacing 2 -offset {bottom 2 left 2 right 2 top 4}

#addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell -center 1 -
stacked_via_top_layer AM -type core_rings -jog_distance 0.28 -threshold 0.28 -nets {VDD
VSS} -follow core -stacked_via_bottom_layer M1 -layer {bottom M1 top M1 right M2 left
M2} -width 4.5 -spacing {top 2 bottom 2 left 2 right 2} -offset 0.28

placeInstance mdifferentialPFD_chargepump_filterloop_LEF 337.5 529.61
selectInst mdifferentialPFD_chargepump_filterloop_LEF
addHaloToBlock {.5 .5 .5 .5} mdifferentialPFD_chargepump_filterloop_LEF
```

```
placeInstance mBIAS_CIRCUIT_1 478.04 993.41
addHaloToBlock {.5 .5 .5 .5} mBIAS_CIRCUIT_1

placeInstance mBIAS_CIRCUIT_2 478.04 939.32
addHaloToBlock {.5 .5 .5 .5} mBIAS_CIRCUIT_2

deselectAll

selectInst mdifferentialPFD_chargepump_filterloop_LEF
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell -
stacked_via_top_layer AM -around selected -jog_distance 0.2 -threshold 0.2 -type
block_rings -follow core -stacked_via_bottom_layer M1 -layer {bottom M1 top M1 right
M2 left M2} -width 5 -spacing 2 -offset .5 -skip_side {left top} -nets {VDD VSS}

deselectAll

selectInst mBIAS_CIRCUIT_1
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell -
stacked_via_top_layer AM -around selected -jog_distance 0.2 -threshold 0.2 -type
block_rings -follow core -stacked_via_bottom_layer M1 -layer {bottom M1 top M1 right
M2 left M2} -width 5 -spacing 2 -offset .5 -skip_side {left top} -nets {VDD VSS}

deselectAll

selectInst mBIAS_CIRCUIT_2
addRing -skip_via_on_wire_shape Noshape -skip_via_on_pin Standardcell -
stacked_via_top_layer AM -around selected -jog_distance 0.2 -threshold 0.2 -type
block_rings -follow core -stacked_via_bottom_layer M1 -layer {bottom M1 top M1 right
M2 left M2} -width 5 -spacing 2 -offset .5 -skip_side {left top} -nets {VDD VSS}

deselectAll

sroute -connect { blockPin padPin padRing } -layerChangeRange { M1 AM } -
blockPinTarget { nearestTarget } -padPinPortConnect { allPort oneGeom } -padPinTarget
{ nearestTarget } -allowJogging 1 -crossoverViaLayerRange { M1 AM } -nets { VDD VSS
} -allowLayerChange 1 -blockPin useLef -targetViaLayerRange { M1 AM }

createRouteBlk -box 321.163 522.110 472.540 1050.673 -layer {M1}
createRouteBlk -box 453.858 922.62 539.570 1050.569 -layer {M1}

sroute -connect { corePin floatingStripe } -layerChangeRange { M1 AM } -blockPinTarget
{ nearestTarget } -corePinTarget { firstAfterRowEnd } -floatingStripeTarget { blockring
padring ring stripe ringpin blockpin followpin } -allowJogging 1 -crossoverViaLayerRange
{ M1 AM } -allowLayerChange 1 -nets { VSS } -targetViaLayerRange { M1 AM }
```

```
deleteFPObject LayerShape (453860,922620,539570,1050570)
deleteFPObject LayerShape (321160,522110,472540,1050670)

createRouteBlk -box 321.233 529.110 465.540 1050.698 -layer {M1}
createRouteBlk -box 442.732 932.2 532.570 1050.700 -layer {M1}

sroute -connect { corePin floatingStripe } -layerChangeRange { M1 AM } -blockPinTarget
{ nearestTarget } -corePinTarget { firstAfterRowEnd } -floatingStripeTarget { blockring
padring ring stripe ringpin blockpin followpin } -allowJogging 1 -crossoverViaLayerRange
{ M1 AM } -allowLayerChange 1 -nets { VDD } -targetViaLayerRange { M1 AM }

deleteFPObject LayerShape (442730,932200,532570,1050700)
deleteFPObject LayerShape (321230,529110,465540,1050700)

# Adding vertical stripes
addStripe -skip_via_on_wire_shape Noshape -block_ring_top_layer_limit M3 -
max_same_layer_jog_length 3 -padcore_ring_bottom_layer_limit M1 -number_of_sets 4 -
skip_via_on_pin Standardcell -stacked_via_top_layer AM -padcore_ring_top_layer_limit
M3 -spacing 1 -xleft_offset 450 -xright_offset 50 -merge_stripes_value 0.28 -layer M2 -
block_ring_bottom_layer_limit M1 -width 3 -nets {VDD VSS} -stacked_via_bottom_layer
M1 -break_stripes_at_block_rings 1

createPlaceBlockage -box 337.122 515.291 477.724 1034.971
createPlaceBlockage -box 477.813 922.608 544.788 1034.606
```

M. SCRIPT PLACEMENT Y CTS DEL SOC

```
# ITESO University
# Francisco Nuñez
# This script shoud be sourced affter power grid has been defined and
# before CTS was created
#
Puts "Starting to do Design Placement..."
# Flooplan Mode for prototyping, runs quickly
setPlaceMode -fp false
#setPlaceMode -congEffort high
setPlaceMode -congEffort auto -timingDriven 1 -modulePlan 1 -clkGateAware 1 -
powerDriven 0 -ignoreScan 0 -reorderScan 0 -ignoreSpare 0 -placeIOPins 1 -
moduleAwareSpare 0 -preserveRouting 0 -rmAffectedRouting 0 -checkRoute 0 -swapEEQ
0
placeDesign
Puts "... finished Design Placement"

# Use the FE-CTS
setCTSMode -engine ck

# Create clock tree using the clock buffers list:
createClockTreeSpec -bufferList {BUFFER_C BUFFER_D BUFFER_E BUFFER_F
BUFFER_H BUFFER_I BUFFER_J BUFFER_K BUFFER_L BUFFER_M BUFFER_N
BUFFER_O CLKI_C CLKI_D CLKI_E CLKI_F CLKI_H CLKI_I CLKI_K CLKI_M
CLKI_O CLKI_Q CLK_C CLK_D CLK_E CLK_F CLK_H CLK_I CLK_K CLK_M
CLK_O CLK_Q DELAY4_C DELAY4_F DELAY4_J DELAY6_C DELAY6_F
DELAY6_J DELAY6_M INVERTBAL_C INVERTBAL_D INVERTBAL_E
INVERTBAL_F INVERTBAL_H INVERTBAL_J INVERTBAL_L INVERT_A
INVERT_B INVERT_C INVERT_D INVERT_E INVERT_F INVERT_H INVERT_I
INVERT_J INVERT_K INVERT_L INVERT_M INVERT_N INVERT_O} -file
../PLL_based_CDR_tutor.ctstch

deleteFPObject LayerShape (327800,807800,1044200,1043000)

# Edit the .ctstch that was created to complete constraints...
```

N. SCRIPT OPTIMIZACIONES SÍNTESIS FÍSICA DEL SOC

```
# ITESO University
# Francisco Nuñez
# This script shoud be sourced affter the
# .ctstch file has been modified for completing constraints

Puts "Timing the design before CTS"

# Calculates the delays for paths based on max. opertating conditions (op) and min. op.
setAnalysisMode -analysisType onChipVariation

timeDesign -preCTS -prefix preCTS_setup
timeDesign -preCTS -prefix preCTS_hold  -hold

Puts "Running CTS"
dbDeleteTrialRoute
clockDesign -specFile ../PLL_based_CDR_tutor.ctstch -outDir clock_report -
fixedInstBeforeCTS
Puts "Finished running CTS"

Puts "Timing the design after CTS"
timeDesign -postCTS -prefix postCTS_setup
timeDesign -postCTS -prefix postCTS_hold  -hold

Puts "Setting Optimizaiton Mode Options for DRV fixes"
setOptMode -fixFanoutLoad true
setOptMode -addInstancePrefix postCTSdrv

Puts "Optimizing for DRV"
optDesign -postCTS -drv

Puts "Timing the design after DRV fixes"
timeDesign -postCTS -prefix postCTS_setup_DRVfix
timeDesign -postCTS -prefix postCTS_hold_DRVfix -hold

Puts "Setting Optimization Mode Options for Setup fixes"
setOptMode -addInstancePrefix postCTSsetup

Puts "Optimizing for Setup"
optDesign -postCTS

Puts  "Timing the design after Setup fixes"
timeDesign -postCTS -prefix postCTS_setup_Setupfix
```

```
timeDesign -postCTS -prefix postCTS_hold_Setupfix  -hold

setOptMode -addInstancePrefix postCTShold
optDesign -postCTS -hold
Puts  "Timing the design after Hold fixes"
timeDesign -postCTS -prefix postCTS_setup_Holdfix
timeDesign -postCTS -prefix postCTS_hold_Holdfix  -hold

Puts "Routing the Design"
setNanoRouteMode -quiet -timingEngine {}
setNanoRouteMode -quiet -routeWithSiPostRouteFix 0
setNanoRouteMode -quiet -routeTopRoutingLayer default
setNanoRouteMode -quiet -routeBottomRoutingLayer default
setNanoRouteMode -quiet -drouteEndIteration default
setNanoRouteMode -quiet -routeWithTimingDriven false
setNanoRouteMode -quiet -routeWithSiDriven false
routeDesign -globalDetail

Puts "Timing the design after Route"
timeDesign -postRoute -prefix postRoute_setup
timeDesign -postRoute -prefix postRoute_hold  -hold

###### Add fillers #######
getFillerMode -quiet
addFiller -cell NWSX FILL8 FILL64 FILL4 FILL32 FILL2 FILL16 FILL128 FILL1 -
prefix FILLER

#### Exportar hacia Virtuoso
set nameGDS "PLL_based_CDR_bicmos8hp.gds";  #Editar nombre de ser necesario

streamOut ../GDS/${nameGDS} -mapFile
/media/Ext/libs/IBM_PDK/bicmos8hp/v.20171220/lef/bicmos8hp_soce2gds.map -libName
DesignLib -merge {
/media/Ext/libs/8HP_IP_CELL_AND_IO_Libs/BiCMOS8HP_Digital_Kit/ibm_cmos8hp/s
c_1p2v_12t_rvt/v.20171220/gds2/BICMOS8HP_SC_1P2V_12T_RVT.gds
/opt/libs/IBM_PDK/bicmos8hp/v.20160727/gds2/CMOS8HP_BASE_WB_IO_7LM.gds }
-outputMacros -units 1000 -mode ALL
```

O. SCRIPT PROCESO DE IMPORTACIÓN DE EDI A VIRTUOSO

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;          Brief: This scripts imports a design from Encounter to Virtuoso
;;;  --> To load this script do: load("PLL_based_CDR_VirtuosoScript1.il")

; Prerequisite: The following library needs to be added
;/media/Ext/libs/IBM_PDK/bicmos8hp/v.20170531/cdslib61/BICMOS8HP_SC_1P2V_12T_RVT_CDSLIB


;Look on Virtuoso Layout Suite->Virtuoso Layout Suite SKILL Reference

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;; Change the values of the following variables ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;Create a new library

Libname="PLL_based_CDR_no_VCO_lib_2v0"               ;This will be the name of the library to be created [To edit]
Topcell="PLL_based_CDR"            ;Name of the top cell of the design    [To edit]
Logdesignname="PLL_based_CDR_lib.log"          ;Name of the log file
           [To edit]

                                                ;Path of the GDS file                      [To edit]

GDSpath="/home/fnunez/Cadence_Desings/EDI/bicmos8hp/PLL_based_CDR_2v0/GDS/PLL_based_CDR_bicmos8hp.gds"


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
MapFilePath="../../../media/Ext/libs/IBM_PDK/bicmos8hp/V1.7_0.6HP/cdslib/bicmos8hp/bicmos8hp.layermap"

editor="gedit"
ddsOpenLibManager()
ddsHiCreateLibrary() ;opens the gui to create the library
hiiSetCurrentForm('ddsCreateLibForm)
ddsCreateLibForm->LibName->value=Libname
ddsCreateLibForm->Option->value= "Attach to an existing technology library"
hiFormDone(ddsCreateLibForm)
hiiSetCurrentForm('tcNewLibAttachTechForm)
```

```
tcNewLibAttachTechForm->attachTechLibList->value= '( "bicmos8hp" )
hiiSetCurrentForm('ddsCreateLibForm)
hiiSetCurrentForm('tcNewLibAttachTechForm)
hiFormDone(tcNewLibAttachTechForm)    ;ok click

;;; Commands for the GDS stream importing  ;;;;;

pipoDisplay(transStreamInForm) ;open XStream In window
_xst_strminuiMainDialog_setStrmFileName( GDSpath )
_xst_strminuiMainDialog_setLibName( Libname )
_xst_strminuiMainDialog_setCellsName( Topcell )
_xst_strminuiMainDialog_setLogFile( Logdesignname )
_xst_strminuiMainDialog_setAttachTechLib( "bicmos8hp" )
_xst_strminuiMainDialog_loadLayerMapFile(
"../../../media/Ext/libs/IBM_PDK/bicmos8hp/V1.7_0.6HP/cdslib/bicmos8hp/bicmos8hp.lay
ermap" )

;;;;;;; Select the libraries

_xst_strminuiMainDialog_toggleOptionsWindow(  )
_xst_strminuiMainDialog_setTab( "4" )
_xst_strminuiMainDialog_setTab( "5" )
_xst_strminuiMainDialog_setSelectedItemsInAvailableLibs(
"BICMOS8HP_SC_1P2V_12T_RVT" )
_xst_strminuiMainDialog_setSelectedItemsInAvailableLibs(
"BICMOS8HP_SC_1P2V_12T_RVT BICMOS8HP_SC_1P2V_12T_RVT_CDSLIB" )
_xst_strminuiMainDialog_setSelectedItemsInAvailableLibs(
"BICMOS8HP_SC_1P2V_12T_RVT BICMOS8HP_SC_1P2V_12T_RVT_CDSLIB
IO_pads_8hp" )
_xst_strminuiMainDialog_addItemsInApplyList(  )
_xst_strminuiMainDialog_setSelectedItemsInAvailableLibs(
"BICMOS8HP_SC_1P2V_12T_RVT_CDSLIB IO_pads_8hp" )
_xst_strminuiMainDialog_setSelectedItemsInAvailableLibs( "IO_pads_8hp" )

;Do translation

_xst_strminuiMainDialog_doTranslate(  )
```

P. SCRIPT VERIFICACIÓN FINAL DEL SOC

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;; Part 2 of the script ;;;;;;;;;;;;;;;;;;;;

;;;;; Run DRC analysis  ;;;;

; To run this script,do: load("PLL_based_CDR_skill_script2.il")

;; change the following variables according to the design

Topcell="PLL_based_CDR"
Libname="PLL_based_CDR_no_VCO_lib_2v0"          ;This will be the name of the
library to be created [To edit]
Runname="PLL_based_CDR_Run_2v0"
Dir4DRC="./PLL_based_CDR_RunDir_2v0"
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



vuiDRCRun()

;;Change DRC settings

vuiDRCForm->library->value=Libname
vuiDRCForm->cell->value=Topcell
vuiDRCForm->view->value="layout"
vuiDRCForm->runName->value=Runname
vuiDRCForm->runDir->value=Dir4DRC


;Select the rules file

vuiDRCForm->rules-
>value="/opt/libs/IBM_PDK/bicmos8hp/V1.7_0.6HP/Assura/DRC/drc.rul"

;Select the switch names

_vuiSwitchesListBox("DRC")
vuiSwitchesListBox->value = '( "BEOL_STACK_211" "Cell")
hiListBoxDone(vuiSwitchesListBox)

;Select the RSF include file
```

```
hiiSetCurrentForm('vuiRcxFileSelectForm)
vuiRcxFileSelectForm->fileList->value= '( "DRCinclude.rsf" )
hiFormDone(vuiRcxFileSelectForm)


;; Apply the settings

_hiFormApplyCB(vuiDRCForm)
```

Q. SCRIPT ARCHIVOS PARA LA FABRICACIÓN DEL SOC

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Brief: This scripts streams out the design to GDS format
; Prerequisite: The following library needs to be added
;/media/Ext/libs/IBM_PDK/bicmos8hp/v.20170531/cdslib61/BICMOS8HP_SC_1P2V_12T_RVT_CDSLIB


;;;  --> To load this script do: load("PLL_based_CDR_Skill_StreamOut.il")


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;; Change the values of the following variables ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Edit the name of the output file
Name_of_Output_GDSFile="PLL_based_CDR_3v0.gds"

;This will be the name of the library to export
Libname="PLL_based_CDR_no_VCO_lib_2v0"

;Edit the name of the top cell of the design to export
Topcell = "PLL_based_CDR"

;Edit the name of the log file
Logdesignname="PLL_based_CDR_3v0.log"



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
layermapFile="../../../media/Ext/libs/IBM_PDK/bicmos8hp/V1.7_0.6HP/cdslib/bicmos8hp/bicmos8hp.layermap"

; Open The Stream out wizard window
pipoDisplay(transStreamOutForm)

; Change the name of the output file
_xst_strmoutuiMainDialog_setStrmFileName( Name_of_Output_GDSFile )

;Set the technology library
_xst_strmoutuiMainDialog_setTechLib( "bicmos8hp" )

;Set the name of the library to create in which the imported design will be placed
```

```
_xst_strmoutuiMainDialog_setLibName( Libname )

;Set the name of the Top Cell
_xst_strmoutuiMainDialog_setCellsName( Topcell )

_xst_strmoutuiMainDialog_setViewsName( "layout" )

;Set the name of the log file
_xst_strmoutuiMainDialog_setLogFile( Logdesignname )

;Specify the layer map file
_xst_strmoutuiMainDialog_setTab( "4" )
_xst_strmoutuiMainDialog_loadLayerMapFile( layermapFile )

;Specify the libraries to use
_xst_strmoutuiMainDialog_setTab( "5" )
_xst_strmoutuiMainDialog_setSelectedItemsInAvailableLibs(
"BICMOS8HP_SC_1P2V_12T_RVT_CDSLIB" )
_xst_strmoutuiMainDialog_addItemsInApplyList(  )

; Last steps
_xst_saveDlg_toggleLib( "2" )
_xst_saveDlg_setLibListFile( ".cadence/xstream/OUT/reflib.list" )
_xst_saveDlg_toggleLib( "0" )
_xst_saveDlg_setLibListFile(  )
_xst_saveDlg_closeDlgAndUpdate(  )

;Do translation
_xst_strmoutuiMainDialog_doTranslate(  )
```

Bibliografía

[1] T. Ananthapadmanabha y e. al., «Programmable Frequency Divider for PLL,» *Int. J. of Recent Trends in Engineering and Technology, , No. 4,,* vol. 4, nº 4, pp. 9-13, Noviembre 2010.

[2] D. Clein, «What is Full Custom Layout Design,» UBM Electronics, 08 06 2001. [En línea]. Available: https://www.eetimes.com/document.asp?doc_id=1277368. [Último acceso: 06 06 2018].

[3] G. Jervan, «Hybrid Built-In Self-Test and Test Generation Techniques for Digital Systems,» Department of Computer and Information Science Linköpings universitet, Linköping, 2005.

[4] P. S. Corp., «Optimization of phase-locked loop performance in data recovery systems,» *IEEE Journal of Solid-State Circuits,* vol. 29, nº 9, pp. 1022 - 1034, 1994.

[5] F. Barale, «Frequency Dividers Design for Multi-GHz,» Georgia Institute of Technology, Georgia, 2008.

[6] I. Analog Devices, *MT-086 Tutorial,* Massachusetts: Analog Devices, Inc., 2009.

[7] C. Barrett, *Fractional/Integer-N PLL Basics,* Dallas, Texas: Texas Instruments, 2010.

[8] F. Barale y e. al., «Programmable Frequency-Divider for Millimeter-Wave PLL Frequency Synthesizers,» de *Proceedings of the 38th European Microwave Conference*, Amsterdam, 2008.

[9] B. Razavi, Monolithic Phase-Locked Loops and Clock Recovery Circuits: Theory and Design, New York: Wiley-IEEE Press, 1996, p. 508.

[10] C. B. G. &. A. I. Quemada, Design Methodology for RF CMOS Phase Locked Loops., Boston: Artech House, Inc., 2009.

[11] S. J. Goldman, Phase-locked Loop Engineering Handbook for Integrated Circuits, Boston: Artech House, Inc., 2007.