UNIVERSIDAD DEL QUINDIO

FACULTAD DE INGENIERÍA

ASIGNATURA

Electiva profesional: Diseño de Circuitos Integrados Digitales

TEMA

Módulo GPIO

PRESENTA

MANUELA ISABELA DAZA ERAZO

JOHAN RODRIGUEZ ROJAS

30/05/2025

Armenia, Quindío

I. INTRODUCCIÓN

El presente documento tiene como objetivo presentar al lector una síntesis de los conocimientos adquiridos en la asignatura, mediante una propuesta de proyecto final enfocada en el diseño del módulo funcional GPIO de 16 pines de un microcontrolador RISC-V de 32 bits. Para ello, se adopta un enfoque de diseño asistido por herramientas (Semi-Custom Design), basado en el uso de celdas estándar correspondientes al PDK Skywater 130, junto con herramientas de síntesis automatizadas provistas por IIC-OSIC-TOOLS (Infraestructura Integrada para Herramientas Colaborativas de Circuitos Integrados de Código Abierto). Esta infraestructura se ejecuta en un contenedor de Docker Desktop y ha sido desarrollada por el Departamento de Circuitos Integrados (DIC) de la Universidad Johannes Kepler (JKU) [1].

II. DESARROLLO

A continuación, se enuncia la metodología empleada para desarrollar el módulo de Entrada/Salida de propósito general (GPIO).

- **Propósito del módulo.**

Se denomina GPIO a los pines presentes en el microcontrolador que pueden ser programados como entradas o salidas de señales digitales, por lo cual, permiten la interacción flexible entre el microcontrolador y una amplia variedad de dispositivos periféricos, desde simples LEDs y botones hasta sensores complejos y módulos de comunicación. La importancia de los GPIO radica en su naturaleza adaptable, ya que no tienen una funcionalidad específica de fábrica, sino que están disponibles para ser configurados y utilizados según las necesidades específicas de cada aplicación [2].

- **Diagrama de caja negra y descripción del módulo.**

Actuando conforme a los requerimientos solicitados para el proyecto, se expone la siguiente figura.

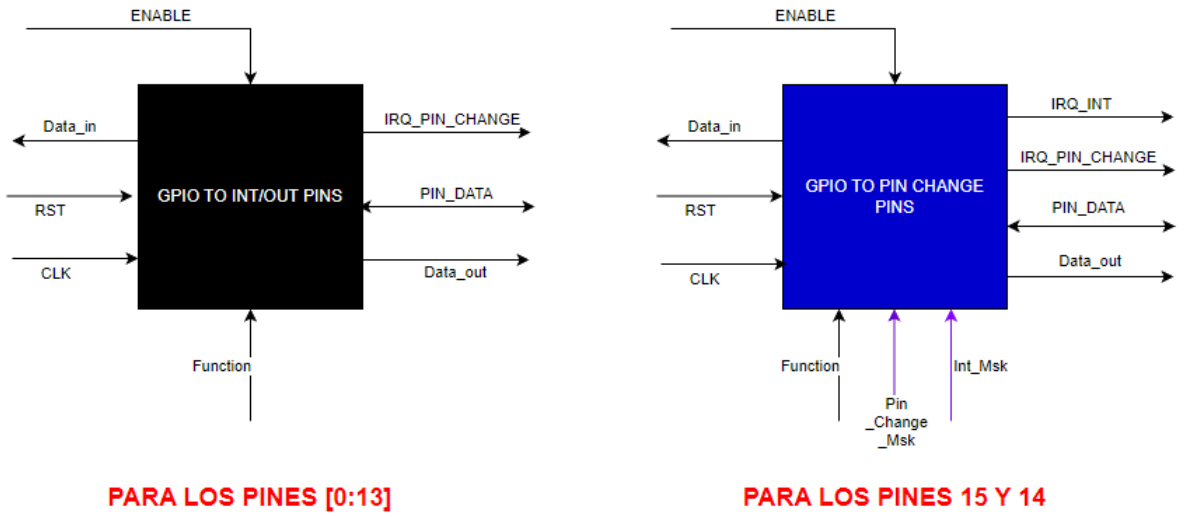

Figura 1. Diagrama de caja negra – GPIO.

La caja negra del costado izquierdo corresponde a la configuración básica que pueden tener los pines, es decir, desempeñar sus funciones como pin de entrada o como pin de salida. A esta función se le denominará “Dual” a lo largo del documento. Además de las funciones básicas que tendrán los 16 pines del GPIO, hay una función adicional que tendrán los pines 14 (*INT0*) y 15 (*INT1*). Dicha función consiste en configurar los pines por cambio de pin por interrupción, es decir, se detectarán flancos de reloj. Por ende, en la figura 2 aparecen 2 cajas negras, la básica al costado izquierdo y la adicional al costado derecho.

Una vez comprendidas las funciones del módulo, es necesario exponer detalladamente la interacción entre entradas y salidas.

Para las funciones básicas:

- El diseño debe ser sincronizado con *clk* y reiniciarse con *RST*.
- La señal *PIN_DATA* debe funcionar como entrada (Leer los datos de un pin) o salida (Escribir los datos hacia un pin) dependiendo de la señal de entrada *Function*.
- El pin se puede configurar como entrada (*Function = 0*) o salida (*Function = 1*).
- Si *Function = 0* (Modo de entrada), el módulo debe capturar el dato de la entrada externa *PIN_DATA* y reflejarlo en *Data_in*.
- Si *Function = 1* (Modo salida), el módulo debe escribir, por medio de *PIN_DATA,* hacia la salida externa el valor de *Data_out*.
- Si enable = 1, el módulo queda inactivo (No transmite ni recibe ningún dato).

Para las funciones adicionales:

- *Pin_Change_Mask* activa la función de interrupción por cambio de señal, si toma el valor de 1, está habilitada. De lo contrario, estará inactiva.
- *Int_Mask [1:0]* define el tipo de interrupción:

    - 2'b00: Deshabilitada.
    - 2'b01: Flanco de subida.
    - 2'b10: Flanco de bajada.
    - 2'b11: Cambio (cualquier flanco).

- *Pin_Change_Mask* e *Int_Mask [1:0]* deben estar habilitadas para activar la interrupción. En ese caso, es necesario que se genere un pulso de 1 ciclo de reloj en las señales *IRQ_INT* e *IRQ_PIN_CHANGE* con el fin de notificar que se está realizando una interrupción por cambio de pin.

Una vez comprendida la interacción de entradas con salidas y las funciones del módulo, se abordará este módulo funcional en dos momentos. El primero realizando un módulo que actúe como un único pin, es decir, todas esas funciones no estarán presentes inicialmente para los 16 pines, sino para uno solo. Posteriormente, cuando se valide el funcionamiento del módulo unitario, se procederá a escalar el diseño para los 16 pines y nuevamente corroborar el funcionamiento de los mismos.

- **Elementos que fueron cumplidos de las especificaciones del diseño**

Para ver qué especificaciones fueron cumplidas, hay que primero enunciar los requerimientos dados:

- 16 pines GPIO.
- Todos los pines pueden ser configurados individualmente como entradas o salidas.
- Algunos de los pines tienen una función especial dependiendo de su configuración.
- Se tienen dos pines que permiten interrupción.
- Todos los pines GPIO pueden generar una señal *IRQ_PIN_CHANGE*.
- La salida *PIN_DATA* puede actuar como salida o como entrada dependiendo de su configuración.

En realidad, el módulo propuesto por el equipo de trabajo cumple con cada uno de estos requerimientos, sin embargo, aprovechando este espacio como oportunidades de mejora para el trabajo realizado por el grupo. Es necesario enunciar que tal vez un plus o una mejora puede ser la de que en lugar de que la señal *enable* habilite o inhabilite el módulo, se puede pensar en que el dato lo deje pasar a una señal de entrada determinada y que, de esta forma, otro módulo de los que compone el microcontrolador pueda disponer de ese dato. Una segunda mejora que se podría incluir al trabajo, sería de emplear de forma adecuada el módulo en el microcontrolador, me refiero a declarar de forma adecuada las señales y los registros que se exponen en los requerimientos del proyecto.

- **Vista RTL del diseño.**

Para el caso, se utilizó la herramienta de síntesis que trae la herramienta mencionada en la introducción con el comando "*yosys -p "prep -top gpio_vector; write_json output.json" GPIO_Vector.v GPIO_Single.v*" donde GPIO_Vector es el módulo que contiene los 16 pines del GPIO y GPIO_Single el pin unitario. Sin embargo, la salida no puede ser presentada en el documento debido a la extensión de dicha salida, por ende, se deja como anexo "*GPIO_Vector_SVG*" a la entrega.

- **Análisis de los diagramas obtenidos en los casos de prueba.**

Para exponer los diagramas de tiempos obtenidos para verificar el funcionamiento del módulo, se exponen los casos de prueba propuestos para dicho fin:

1. **Modo Entrada (*Function = 0*)**
    - Verifica que *Data_in* refleje correctamente los valores de *PIN_DATA.*
    - Prueba varios cambios en la señal externa.
2. **Modo Salida (*Function = 1*)**
    - Verifica que *PIN_DATA* refleje los valores de *Data_out.*
    - Prueba varios valores de salida.

3. **Desactivación del Módulo (*Enable = 1*)**
    - Verifica que el módulo deje de responder cuando *Enable = 1*.
    - Comprueba que *PIN_DATA* quede en alta impedancia.
4. **Reactivación del Módulo (*Enable = 0*)**
    - Verifica que el módulo vuelva a funcionar correctamente. Se tienen valores en *PIN_DATA* de alta impedancia.
5. **Interrupción por Flanco de Subida (*Int_Mask = 2'b01*)**
    - Verifica que se generen las señales *IRQ_PIN_CHANGE e IRQ_INT* en el flanco de subida.
    - Comprueba que no se generen en el flanco de bajada.
6. **Interrupción por Flanco de Bajada (*Int_Mask = 2'b10*)**
    - Verifica que se generen las señales *IRQ_PIN_CHANGE* e *IRQ_INT* en el flanco de bajada.
7. **Interrupción por Cualquier Flanco (*Int_Mask = 2'b11*)**
    - Verifica que se generen las señales de interrupción en ambos flancos.
8. **Deshabilitación de Interrupciones (*Pin_Change_Mask = 0*)**
    - Verifica que no se generen interrupciones cuando están deshabilitadas. No se genera ninguna señal *IRQ.*
9. **Interacción entre Function y Interrupciones**
    - Verifica que las interrupciones solo funcionen en modo entrada. No se genera ninguna señal *IRQ.*
10. **Prueba de Reset**
    - Verifica que la señal *PIN_DATA* se mantenga en alta impedancia.

En la figura 2 se expone el diagrama de tiempos.

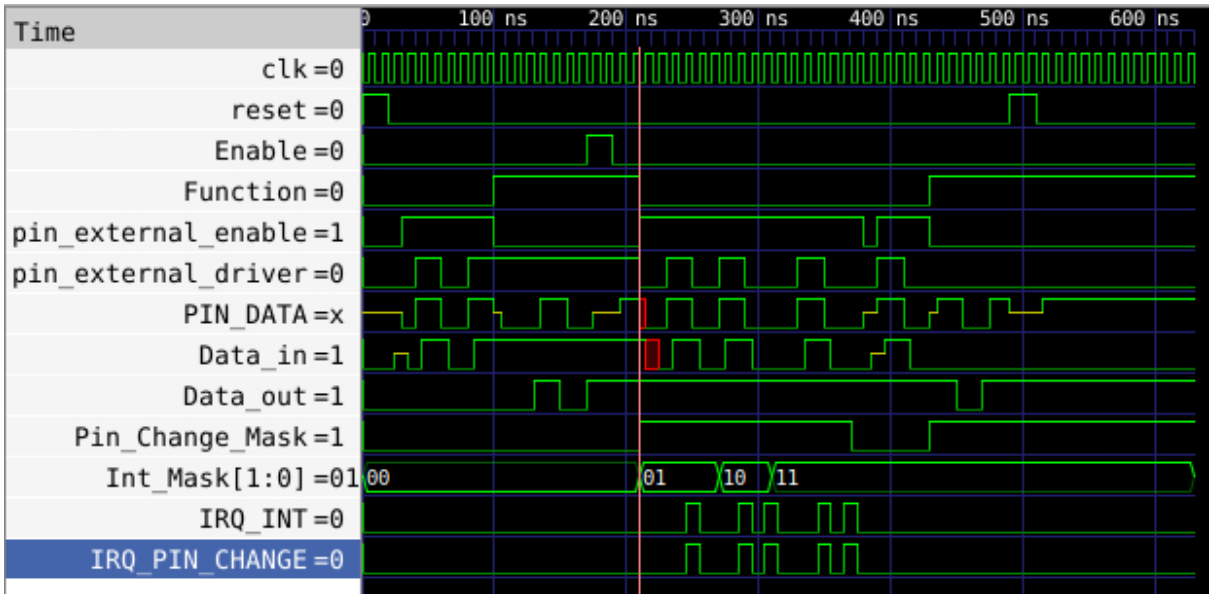

Figura 2. Diagrama de tiempos del módulo GPIO_Single.

La línea roja que está en medio de la gráfica corresponde a una guía que indica que hacia la izquierda se han verificado y cumplido los casos 1, 2, 3,4 y hacia la derecha se cumplen los

demás. También es importante mencionar que los estados en los que *PIN_DATA* está en amarillo significa que hay un estado de alta impedancia y cuando están en rojo, significa que el estado es indefinido. De esta imagen se puede concluir que el funcionamiento del pin individual es correcto y, por ende, se pasa a exponer los casos de prueba y el diagrama de tiempos del módulo integrado:

- **Prueba 1**: Configuración mixta de pines (entradas y salidas) y verificación de valores de salida.
- **Prueba 2**: Simulación de datos entrantes y verificación de su correcta lectura.
- **Prueba 3**: Cambio dinámico de la dirección de los pines y verificación del comportamiento.
- **Prueba 4**: Pruebas específicas para los pines 14 y 15 con interrupciones:
  - Verificación de que se fuerzan a modo entrada cuando se activan las interrupciones.
  - Generación de flancos ascendentes y descendentes para probar *IRQ_PIN_CHANGE* e *IRQ_INT*.
- **Prueba 5**: Aplicación de reset durante la actividad y verificación del estado de reinicio.
- **Prueba 6**: Prueba de la funcionalidad Enable para deshabilitar el módulo.

En la figura 3 se expone el diagrama de tiempos:

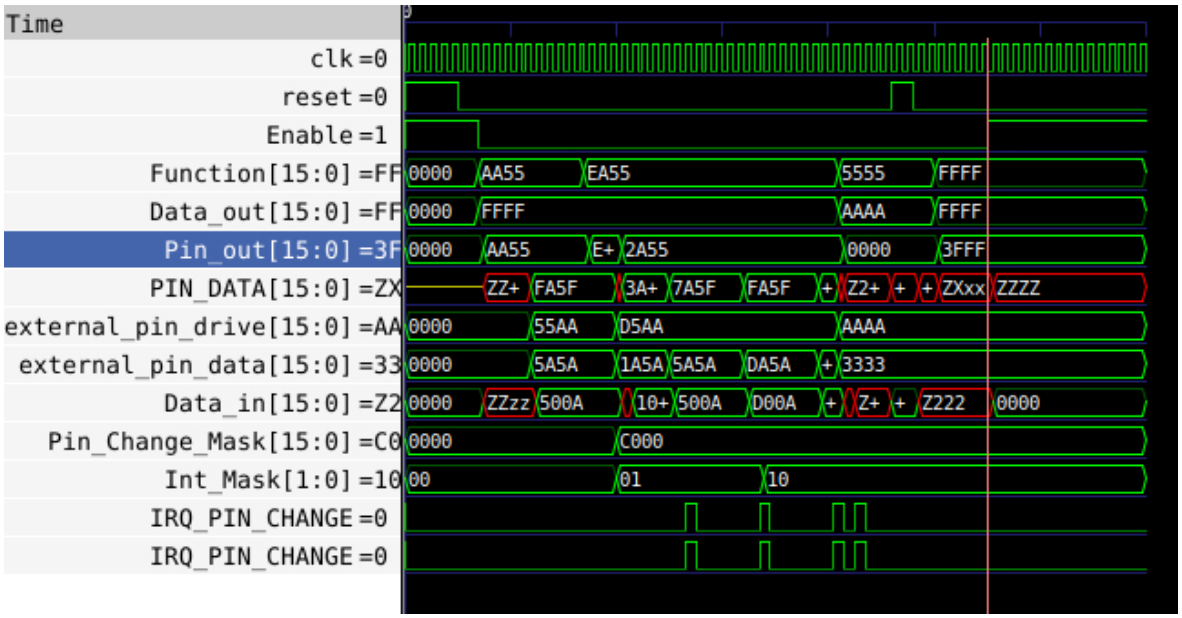

Figura 3. Diagrama de tiempos del módulo GPIO_Vector.

Para el diagrama de tiempos del módulo integrado se tiene una nueva señal *Pin_out* que permite verificar si el dato leído y que irá hacia *DATA_IN* es el mismo que esta señal tiene. Por otro lado, los estados de alta impedancia mantienen su significado, sin embargo, los que están demarcados como rojo denotan un funcionamiento definido (Como es el caso de *DATA_IN* y *PIN_DATA*). En la anterior imagen se tiene 1 error, que corresponde a que cuando la señal de

enable se activa, el módulo sigue funcionando (Y esto es gracias a que la señal de *Pin_out* no toma un valor de cero, sino mantiene su valor anterior y de igual forma *PIN_DATA* lo cual supone un error, ya que en ese caso se estarían leyendo datos del periférico) y se puede observar en la línea roja vertical que está en la figura. Por ende, se corrigen los errores y se tiene el siguiente resultado.

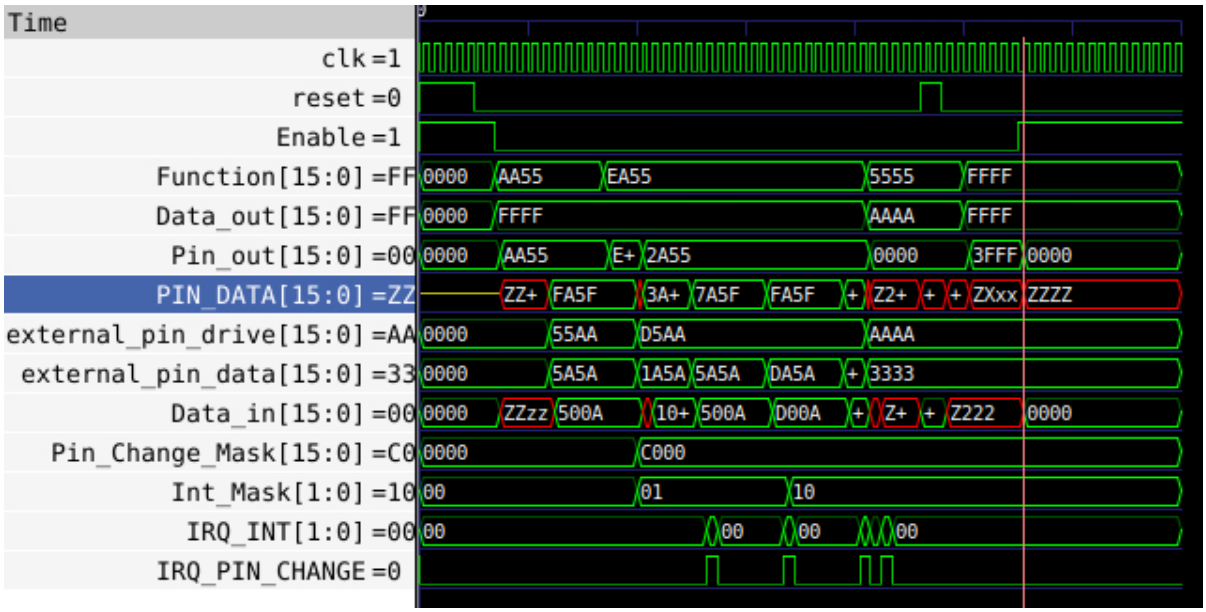

Figura 4. Diagrama de tiempos del módulo GPIO_Vector corregido.

El error se debía a que cuando la señal de *Enable* estaba activa en alto, se había considerado inhabilitar la señal *Data_out* pero no *PIN_DATA ni Data_in.* Realizando ese cambio, se observa que el diagrama de tiempos en la referencia establecida en la figura 3 mantiene su valor en cero en *Pin_out* y, por ende, en *PIN_DATA* también. Con lo cual, se corrige el error y se verifica el correcto funcionamiento del módulo integrado.

- **Caracterización de área y temporización del chip.**

La caracterización de área y temporización del chip se realizó con la herramienta de OpenLane, una herramienta de código abierto que permite pasar de un archivo a nivel de compuertas a un layout físico completo y listo para su fabricación por varias etapas (Synthesis, Floor Planning, Power Planning, Placement, Route y Sign Off). Para el caso, se analizará en primer lugar qué opciones de síntesis la herramienta denota como las mejores. Para esto, se ejecuta el comando "*openlane –flow SynthesisExploration config.json*" con el archivo *config.json* previamente configurado, sobre todo el tiempo del reloj.

SYNTH_STRATEGY	Gates	Area (µm²)	Worst Setup Slack (ns)	Total -ve Setup Slack (ns)
AREA 0	315	3634.736000	6.925787694931576	0.0
AREA 1	315	3634.736000	6.925787694931576	0.0
AREA 2	315	3644.745600	6.925787694931576	0.0
AREA 3	383	4007.593600	7.647359406625744	0.0
DELAY 0	322	3664.764800	7.571254948402939	0.0
DELAY 1	324	3657.257600	7.445646976379671	0.0
DELAY 2	324	3656.006400	7.445646976379671	0.0
DELAY 3	322	3663.513600	7.41028503977762…	0.0
DELAY 4	368	3980.067200	6.09138623449976	0.0

Figura 5. Síntesis de la configuración con 10 ns de reloj.

Esta primera síntesis corresponde al archivo de configuración para un periodo de reloj de 10 ns. Como se observan slacks grandes, se procede a reducir el periodo del reloj hasta encontrar slacks considerables, según el equipo de trabajo lo establezca.

SYNTH_STRATEGY	Gates	Area (µm²)	Worst Setup Slack (ns)	Total -ve Setup Slack (ns)
AREA 0	315	3634.736000	-0.0742121940461…	-0.1484243880922…
AREA 1	315	3634.736000	-0.0742121940461…	-0.1484243880922…
AREA 2	315	3644.745600	-0.0742121940461…	-0.0742121940461…
AREA 3	383	4007.593600	0.647359295603439	0.0
DELAY 0	322	3664.764800	0.57125461533602…	0.0
DELAY 1	324	3657.257600	0.44564642126814…	0.0
DELAY 2	324	3656.006400	0.44564642126814…	0.0
DELAY 3	322	3663.513600	0.41028448466609…	0.0
DELAY 4	368	3980.067200	-0.9086138765225…	-6.2548599181610…

Figura 6. Síntesis de la configuración con 3 ns de reloj.

Este periodo de reloj ya presenta slacks considerables, por lo cual, se puede definir que el periodo máximo del reloj va a ser 3 ns, hasta aquí el módulo o circuito trabaja de forma correcta. Ahora que se halló el periodo de reloj máximo, es necesario elegir una estrategia de síntesis para proceder al análisis de las dimensiones y temporización.

Para elegir una estrategia de síntesis hay que tener 2 cosas en cuenta: Primero, que la estrategia permita reducir el área y segundo, que permita tener slacks aceptables. De acuerdo a estos 2 criterios, la estrategia elegida es "*DELAY 0*", ya que es la que mejor se adapta a los criterios tenidos en cuenta. Ahora, lo que procede es configurar el archivo de configuración para que la síntesis que se realice, se haga con esta estrategia.

Hasta aquí la caracterización de área y temporización, sin embargo, si se quiere ir un poco más allá, se puede ver la etapa de STA (Análisis estático después de hacer el layout) que es la etapa que contiene las pruebas que hizo la herramienta al layout. Debido a que la herramienta hace una considerable cantidad de iteraciones para cumplir con slacks esperados, dopaje, niveles de entrada, etc. Se centra el análisis en el archivo "*summary.rpt*". cuya salida se expone a continuación.

Corner/Group	Hold Worst Slack	Reg to Reg Paths	Hold TNS	Hold Vio Count	of which reg to reg
Overall	0.2976	0.2976	0.0000	0	0
nom_tt_025C_1v80	0.4718	0.4718	0.0000	0	0
nom_ss_100C_1v60	0.9611	0.9611	0.0000	0	0
nom_ff_n40C_1v95	0.2986	0.2986	0.0000	0	0
min_tt_025C_1v80	0.4704	0.4704	0.0000	0	0
min_ss_100C_1v60	0.9588	0.9588	0.0000	0	0
min_ff_n40C_1v95	0.2976	0.2976	0.0000	0	0
max_tt_025C_1v80	0.4734	0.4734	0.0000	0	0
max_ss_100C_1v60	0.9638	0.9638	0.0000	0	0
max_ff_n40C_1v95	0.2997	0.2997	0.0000	0	0

Figura 7. Archivo *summary.rpt* para los tiempos de hold del módulo.

La primera columna corresponde al grupo que se está analizando. Este tendrá 4 variantes separadas por "__".

- El primer factor indica el rango de interconexión, este puede variar entre Nominal, Máximo o Mínimo.
- El segundo factor es el proceso de fabricación, este puede variar entre típico (tt), lento-lento (ss) o rápido-rápido (ff). Esto va orientado al dopaje de los transistores.
- El tercer factor alude a la temperatura de operación dada en °C. Donde el prefijo n corresponde a un voltaje negativo y el sufijo corresponde a la temperatura a prueba.
- El último parámetro corresponde al voltaje de operación donde 1vn corresponde a n volts de operación.

Una vez comprendido qué se va a analizar, se hace énfasis en las 3 últimas columnas de la figura 7, que corresponden a irregularidades en la síntesis, como violaciones a los tiempos de hold o total de slack negativos. La segunda columna corresponde a los peores slacks en el tiempo de hold de todas las rutas posibles, como es positivo y no se tienen restricciones en cuanto a dichos tiempos, indica una temporización adecuada. Por otro lado, la tercera columna corresponde a las rutas de retención entre registros, es decir, cuánto slack hay en las rutas entre registros. Una conclusión del análisis de estos tiempos de hold puede ser que, a cualquier temperatura y voltaje de alimentación, el circuito podrá funcionar.

Este análisis aplica también para los tiempos de setup del módulo, que se presentan a continuación.

Setup Worst Slack	Reg to Reg Paths	Setup TNS	Setup Vio Count	of which reg to reg	Max Cap Violatio…	Max Slew Violati…
0.1791	0.1791	0.0000	0	0	0	0
1.6468	1.6468	0.0000	0	0	0	0
0.2014	0.2014	0.0000	0	0	0	0
2.1718	2.1718	0.0000	0	0	0	0
1.6630	1.6630	0.0000	0	0	0	0
0.2226	0.2226	0.0000	0	0	0	0
2.1841	2.1841	0.0000	0	0	0	0
1.6285	1.6285	0.0000	0	0	0	0
0.1791	0.1791	0.0000	0	0	0	0
2.1590	2.1590	0.0000	0	0	0	0

Figura 8. Archivo *summary.rpt* para los tiempos de setup del módulo.

- **Imagen del layout final**

Para generar una imagen del layout final, es necesario ejecutar en la terminal del docker el siguiente comando "*openlane – last-run –flow openinklayout config.json*". La salida del comando es el layout que corresponde al módulo GPIO_Vector. Por limitaciones en las dimensiones del documento no es posible exponer ante el lector la imagen del layout, sin embargo, este se anexa a la entrega bajo el nombre "*KLAYOU_GPIO*".

III. **REFERENCIAS**

[1]. hpretl. “IIC-OSIC-TOOLS”. Github. [En línea]. Disponible: https://github.com/iic-jku/IIC-OSIC-TOOLS

[2]. gosrios. “GPIO, pines de propósito general”. wordpress. [En línea]. Disponible: https://descubriendolaorangepi.wordpress.com/2017/01/11/gpio-pines-de-proposito-general/