Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE-0523 Circuitos Digitales II
I ciclo 2017

.

Proyecto Final

Controlador de puerto USB tipo C

Profesor: Enrique Coen Alfaro

Luis Diego Fernández, B22492
Lennon Nuñez, B34943
Bernardo Zúñiga, B27445

3 de Junio del 2017

Índice

Índice de figuras

Resumen

Para el presente proyecto se propuso diseñar el controlador de un puerto USB tipo C utilizando el lenguaje de descripción de *Hardware* verilog. El controlador consta de 4 módulos fundamentales que unidos adecuadamente y respetando el debido protocolo logran establecer una comunicación idónea con el *Master* de la comunicación. Estos módulos son un bloque de transmisión Tx, uno de Recepción Rx, módulo de Registros para el dispositivo y el módulo de control de la comunicación al exterior por protocolo I2C. Se crearon los módulos por separado y se les corrieron las respectivas pruebas, las cuales fueron aprobadas con éxito. Se procedió luego a interconectar los módulos y hacerle pruebas, sin embargo, en este caso solo 1 prueba se logró correr de manera exitosa, esta es una transacción completa que verifica el funcionamiento en conjunto del controlador del Hard Reset. Los bloques funcionan y se comunican, sin embargo no se desarrolló pruebas para verificar la operación de Tx y Rx en una transacción completa. El código se compila y ejecuta la simulación por medio de un MakeFile, con el comando *make* en terminal. Para mayor información leer el README del repositorio https://github.com/luisdifc/Proyecto_USB.git.

Palabras clave: TCPC, TCPM, I2C, CRC, Buffer

1. Objetivos

1.1. Objetivo general

Crear el controlador de puerto USB tipo C.

1.2. Objetivo Específico

- Implementar los 4 módulos principales de manera separada creando una descripción conductual.
- Unir los módulos y hacerle pruebas.
- Crear una descripción estructural de los módulos del TCPC.
- Emular el comportamiento del Maestro de USB para realizar las transacciones necesarias

2. Bitácora del Plan de Trabajo

Cuadro 1: Bitácora de trabajo.

	Luis	Lennon	Bernardo
9 Jun, 2017		Empezó I2C	
10 Jun, 2017	Empezó a leer el documento		Empezó Tx y Registers
13 Jun, 2017			Siguió con Tx y Registers
14 Jun, 2017	Volvió a leer el documento	Siguió con I2C	
18 Jun, 2017		Investigó sobre I2C	Siguió con Tx y Registers
20 Jun, 2017	Empezó Rx y Reset		
21 Jun, 2017	Siguió con Rx y Reset	Siguió con I2C	Siguió con Tx,y Registers
22 Jun, 2017	Acabó Rx y Reset	Terminó I2C	Terminó Tx,y Registers
23 Jun, 2017		EXPOSICIÓN	
29 Jun, 2017	Nos reunimos para hablar sobre como unir el proyecto		
4 Jul, 2017			Arregló módulo de Registers
8 Jul, 2017		Trató unir I2C con los registros	
10 Jul, 2017	Trató de empezar una comunicación		
12 Jul, 2017	Nos reunimos a trabajar		
13 Jul, 2017		EXPOSICIÓN FINAL	

3. Estudio de mercado

3.1. ¿Qué bloques funcionales requiere un TCPC?

Un TCPC como se muestra en el presente trabajo requiere 4 bloques funcionales principales. [1]

- Módulo de Recepción.
- Módulo de Transmisión.
- Módulo de Reset.
- Módulo de I2C.

3.2. ¿Qué bloques funcionales requiere un TCPM

El TCPM es el controlador maestro en el protocolo de control del USB tipo C. Este requiere varios módulos entre los que encontramos. [2]

- Módulo de USB PD Phy.
- Módulo de Channel Configuration
- Módulo de Control
- Módulo de VCONN
- Módulo de VBUS

3.3. ¿Cuáles son las diferencias entre la funcionalidad de TCPM y TCPC?

Básicamente como el acrónimo lo dice, uno corresponde al maestro del puerto y otro al esclavo del puerto. Resulta que las comunicaciones seriales asincrónicas como lo es el I2C pueden tener 1 master (a veces más pero se debe configurar de manera especia) y varios esclavos. La interfaz TCPC lo que hace es comunicar a cada uno de los dispositivos esclavos con el master, por otro lado el TCPM lo que hace es administrar la comunicación con todos los esclavos que hayan así como antender sus necesidades. Esto lo podemos observar en la figura

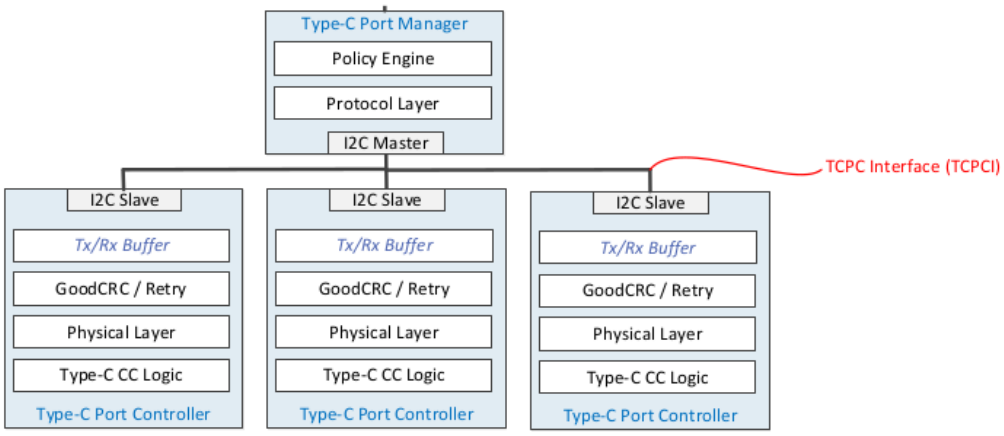

Figura 1: Interfaz del TCPM y el TCPC para USB typo C. [1]

3.4. ¿Cuál debería ser el precio de cada circuito para ser competitivo con los productos que ya están en el mercado?

Si nos basamos en los precios de Texas Instruments para comprar un *USB Type C Port Controller and Power Switch* tenemos precios que van desde los $3.85 en caso de comprar al por menor y precios de $1.56 si se va a comprar al por mayor (1000-9999). Es por esto que si se construye un controlador de estos es deseable que el precio de construcción sea lo más de $1 ya que hay que considerar además de eso costos de diseño, verificación y manufactura.

3.5. ¿Qué factores impactan el costo de los dispositivos comerciales?

En general para dispositivos electrónicos y no solo para controladores de puerto de USB tipo C los precios comerciales del mercado se rigen por la oferta y la demanda así como por la cantidad de unidades que se vayan a comprar por orden. Estos dispositivos rara vez se venden al por menor, y con esto me refiero a 500 unidades o menos, esto por lo general se hace para probar los chips, electrónicos o dispositivos en la integración de un proyecto mayor. Las empresas grandes crean la especificación de un dispositivo, lo diseñan y eligen el hardware que van a utilizar, en este momento tienen que hacer un estudio de mercado para analizar la gama de productos que tienen disponibles y sus especificaciones. Dada esta situación y decantados por cierta unidad en específico, luego de pasar las pruebas de integración, se hacen pedidos de miles y hasta millones de unidades. He aquí uno de los factores, la cantidad de unidades que se compren. Si se compran pocas el precio es más alto, si se compran muchas el precio es más bajo puesto que el vendedor se puede dar e lujo de bajar el porcentaje de ganancia por unidad con el fin de maximizar la ganancia por cantidad vendida.

4. Diagrama Arquitectónico

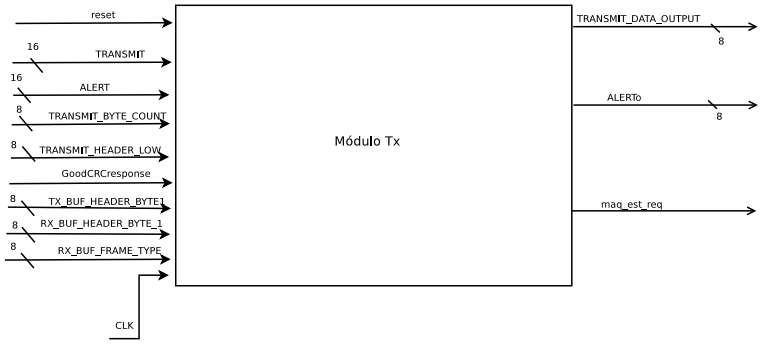

Figura 2: Diagrama arquitectónico máquina de estados Tx

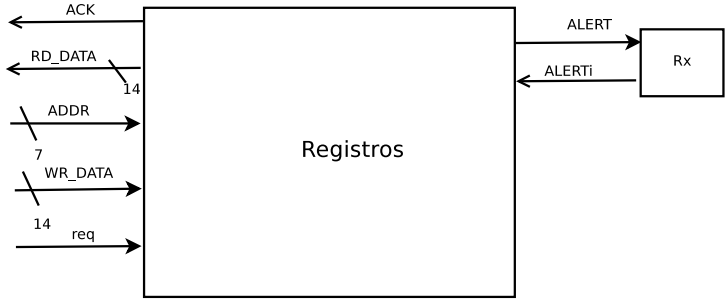

Figura 3: Diagrama arquitectónico de registros, con sólo algunas de las entradas y salidas, y sólo conectado a una máquina de estados

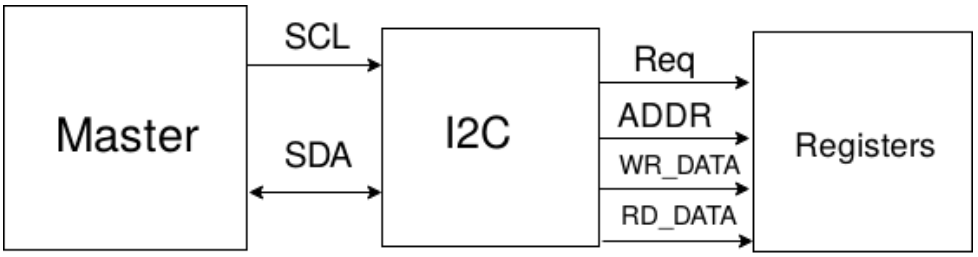

Figura 4: Diagrama arquitectónico del módulo de comunicación I2C

5. Síntesis

La prioridad de este proyecto fue el desarrollo de los módulos de manera conductual, lo que deja sin una idea clara del hardware completo necesario para construir los módulos en físico, para ello se requiere de sintetizar el código, en este caso, con la herramienta de síntesis Yosys para generar la descripción estructural de los módulos individuales a partir de una librería de componentes CMOS llamada cmos_cells.lib.

Al realizar la síntesis se llegó a la generación de pocos, pero existentes, Latches inferidos en algunos de los módulos, los cuales surgen por diferentes motivos de diseño dentro de los que están señales perdidas en el paso de una estado a otro en las transiciones de las máquinas de estados, o dependencias de señales con sigo mismas en diferentes tomas de decisión. Para eliminar estos latches se intentó hacer correcciones en la descripción de cada módulo que les originó, logrando varios módulos sintetizados sin latches.

5.1. Módulo Reset

En cuando a la síntesis de este módulo se logró sintetizar el código conductual con éxito y sin generación de Latches inferidos. Dentro de los entregables hay un carpeta llamada *Reset* que contiene dentro de ella el archivo con el código estructural respectivo, este se llama *Reset_Module_synth.v*. La cantidad de componentes que fueron sintetizados se observan en el cuadro 2.

Cuadro 2: Cantidad de componentes y señales generados por la sintetizacion del Reset.

ABC Results	Compuertas NAND	111
	Compuertas NOR	128
	Compuertas NOT	46
	Señales internas	206
	Señales de entrada	41
	Señales de salida	40

5.2. Módulo Rx

En cuando a la síntesis de este módulo se logró sintetizar el código conductual con éxito y sin generación de Latches inferidos. Dentro de los entregables hay un carpeta llamada *Rx* que contiene dentro de ella el archivo con el código estructural respectivo, este se llama *Rx_Module_synth.v*. La cantidad de componentes que fueron sintetizados se observan en el cuadro 3.

Cuadro 3: Cantidad de componentes y señales generados por la sintetizacion del Rx.

ABC Results	Compuertas NAND	61
	Compuertas NOR	115
	Compuertas NOT	57
	Señales internas	201
	Señales de entrada	74
	Señales de salida	46

5.3. Módulo Tx

Para el caso del módulo de Rx la síntesis se generó con la creación de 6 Latches inferidos relacionados a las señales de los cambios de estado de la máquina, a pesar de los cambios realizados para intentar eliminar estos componentes no se logró una síntesis sin latches inferidos.

Cuadro 4: Cantidad de componentes y señales generados por la sintetización del Tx.

ABC Results	Compuertas NAND	58
	Compuertas NOR	109
	Compuertas NOT	43
	Señales internas	87
	Señales de entrada	69
	Señales de salida	28

5.4. Módulo I2C

Para el caso del módulo I2C no se logró eliminar el latch de la síntesis. Esto debido a que al realizar cambios estructurales para eliminar estos componentes, logrando eliminar la creación del latch en la síntesis, cambiaba de manera significativa el comportamiento de los módulos, lo que hacía que al correr la prueba, estos módulos fallaran. De modo que se prefirió la creación de un latch inferidos por la herramienta de síntesis con una descripción funcional, que eliminar su creación resultando en código con fallos en su operación y sin garantía de éxito en pasar las pruebas propuestas y necesarias.

Sin embargo, se logró sintetizar el código (con 1 latch inferido), en el archivo *I2C_ synth.v*. A continuación se observa en el cuadro 5 la cantidad de componentes y señales generados en la sintetización.

Cuadro 5: Cantidad de componentes y señales generados por la sintetizacion del I2C.

ABC Results	Compuertas NAND	101
	Compuertas NOR	88
	Compuertas NOT	36
	Señales internas	167
	Señales de entrada	35
	Señales de salida	35

6. Plan de Pruebas

El Plan de Pruebas realizado consiste en llevar a cabo una transacción completa para un Hard Reset, esto siguiendo el modelo operacional 4.7.4 propuesto en el documento USB Type-C Controller Interface Specification. En el archivo *test_ bench_ reset.v* se especifica las conexiones entre los módulos involucrados para esta prueba, estos son el módulo de comunicación para I2C (*I2C_ Module.v*),

el módulo de registros (*Registers_ Module.v*). Para la secuencia de solicitudes por parte del Maestro USB se diseña el mensaje de accesos en el archivo *probador_ reset.v*. Para verificar la completitud de la prueba se debe abarcar dos principales puntos de vista, primero observar el correcto funcionamiento del módulo de comunicación I2C, quien se debe encargar de las solicitudes del Maestro USB, es decir, los accesos que solicite (Pasos 1 al 5); segundo se debe observar el comportamiento del módulo de Reset, el cual debe estar atento a los cambios en el registro ALERT (Pasos 6 al 10). Este registro contiene diferentes banderas relacionadas a procesos que controlan el comportamiento de las máquinas Tx y Rx, dentro de estas banderas está el 3er bit que indicará en alto si se ha dado un Hard Reset.

1. Lo primero que se realiza es enviar una señal de Reset (En alto) a todos los módulos y observar que efectivamente las máquinas y registros de cada bloque inicializan con algún valor conocido definido según el diseño. Si no se ve respuesta a la señal de Reset, la máquina no comenzará con su operación correcta y en el mejor de los casos solamente no cambiará de estado. Luego del Reset se debe observar que el módulo de I2C realiza una solicitud al módulo de registros para obtener el valor contenido en el registro DEVICE_ID, este contenido será la etiqueta utilizada por el Maestro para establecer la comunicación con el dispositivo USB Esclavo.

2. Luego, el Maestro envía la señal de START al módulo de I2C, poniendo la línea iSDA en bajo mientras mantiene SCL en alto. Se debe observar un cambio del estado 1: IDLE_ID, a los estados 2: START y 8: WAIT_ID. En este último estado la máquina de I2C lee bit por bit el ID solicitado por el Maestro (8 ciclos de SCL), referente a con cuál dispositivo quiere efectuar la comunicación. Luego de los 8 ciclos de SCL (Controlada por el Maestro) la máquina de I2C debería contener en su buffer de recepción rByte 7 bits con el ID (almacenado en DEVID) y 1 bit con la solicitud de un WRITE (bit en 0, almacenado en RW). Si el ID recibido concuerda DEVID, se debe observar un cambio al estado START_REG, de lo contrario, la máquina rechaza la comunicación y regresa al estado 1: IDLE_ID. En esta prueba se envía la dirección correcta de modo que debería seguir con el estado START_REG.

3. Luego, se recibe del Maestro 1 byte conteniendo la dirección baja del registro al cual se quiere accesar, ALERT, con la dirección base hx10. Después de enviado el mensaje por el Maestro se debe observar en el buffer de recepción el valor hx10, y luego, este mismo valor almacenado en rAdd. No debería haber acceso al módulo de registros por parte del módulo I2C en este punto.

4. Se toma la decisión de si es un READ o un WRITE, según RW. Este caso involucra una escritura. Se procede a recibir 2 B con la información del dato que se quiere escribir en el registro indicado. Como el buffer rByte es de 8 bits, cada vez que se llena se descarga en rRec, quien guarda los 2 B finales.

5. Se debe observar la solicitud de acceso de I2C al módulo de registros, modificando efectivamente el registro ALERT con el dato recibido del Maestro(hx0B).

6. Se procede a verificar operación del módulo de Reset. Se debe observar el inicio del contador de ciclos nanos al recibir el Reset inicial, y los primeros cambios de estado de la máquina del módulo de Reset. Dado que en este punto no se ha recibido la alerta del Hard Reset, el contador debe aumentar indefinidamente hasta llegar a su tope de espera de esta bandera, sin entrar al estado de éxito de la señal de Hard Reset. Estos cambios se deben repetir por un gran período, ya que el reloj SCL es mucho más lento que la base CLK, y esta bandera en el 3er bit de ALERT no se coloca sino hasta casi el final del proceso descrito para el módulo I2C en esta prueba.

7. Al darse un fallo en la recepción de la bandera de Alerta ante un Hard Reset, el módulo de Reset debe accesar al registro ALERT y poner la bandera del mensaje del Fallo, escribiendo hx10.

8. Luego de repetir este proceso en espera de la bandera de Alerta durante varios ciclos de SCL, se debe leer el registro ALERT y la máquina notar la presencia del 3er bit en alto, con lo que debe entrar en los estados de preparación del mensaje de éxito en la recepción de la Alerta, esto en el registro TRANSMIT, se debe observar que la máquina escriba hx05 en el registro correspondiente.

9. Al recibir el mensaje de "Succes"de la Alerta de un Hard Reset se considera que se ha realizado una transacción completa y satisfactoria para el procesamiento de un Hard Reset. Si el registro TRANSMIT no es escrito por la máquina de Reset, la transacción está incompleta

10. En ningún momento de la prueba deben interferir dos o más escrituras simultáneas al mismo registro, el módulo de registros da prioridad a las máquinas de estado internas del dispositivo USB (Tx, Rx y Reset) y la comunicación externa, administrada por el módulo I2C, solo puede acceder a los registros en caso que se le de el acceso por la línea *maq_ req*, el mal funcionamiento de este control de coherencia en los registros puede ser fácilmente detectado por la aparición de estados indefinidos en los registros al realizar las simulaciones.

7. Instrucciones de Utilización de la Simulación

Se incluye un MakeFile para facilitar el inicio de la simulación, la cual se accesa por medio del comando *make* en el directorio general, esto realiza las pruebas descritas y muestra los resultados.

8. Ejemplos de los Resultados

A continuación los resultados de las simulaciones según el orden de los pasos descritos en el Plan de Pruebas.

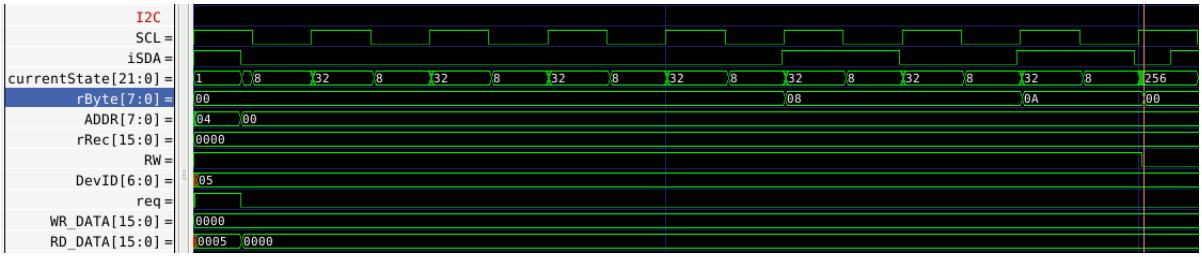

Figura 5: Ejemplo de recepción de un ID correcto y su procesamiento por parte del módulo I2C

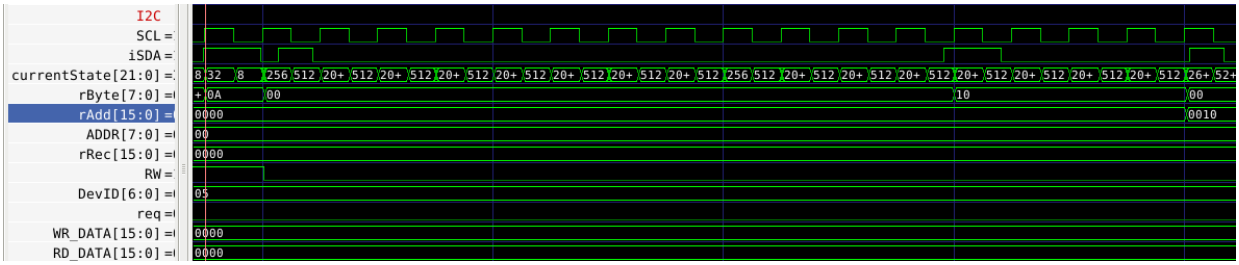

Figura 6: Ejemplo de recepción de la dirección del registro a accesar por parte del Maestro

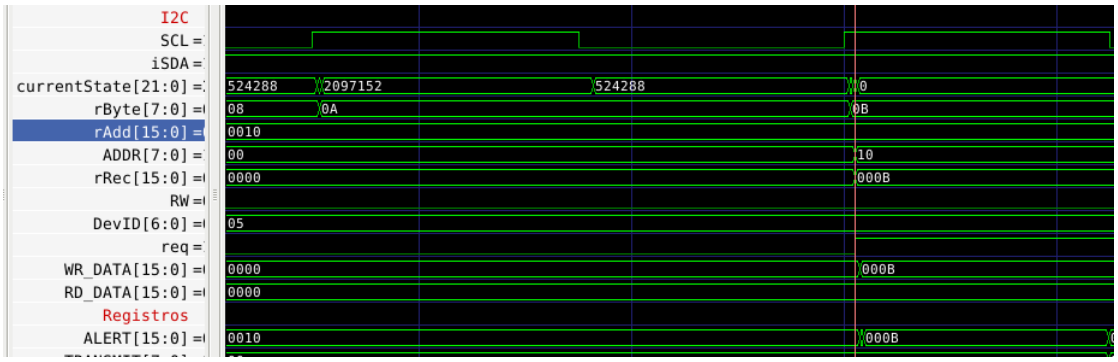

Figura 7: Escritura del dato recibido por I2C en el registro ALERT

Figura 8: Inicio del contador de espera de la bandera en ALERT

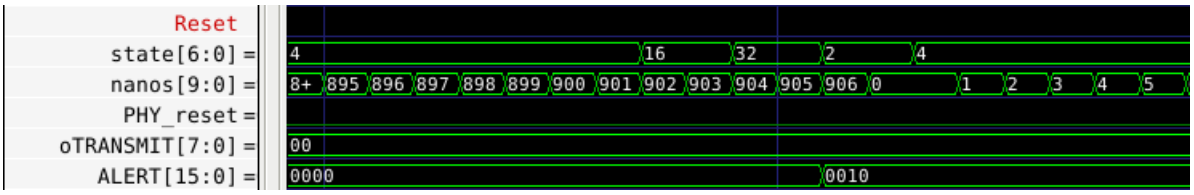

Figura 9: Contador de espera de bandera de Alerta llega a su límite, se entra al estado de fallo de Hard Reset y se vuelve a iniciar la espera de la bandera

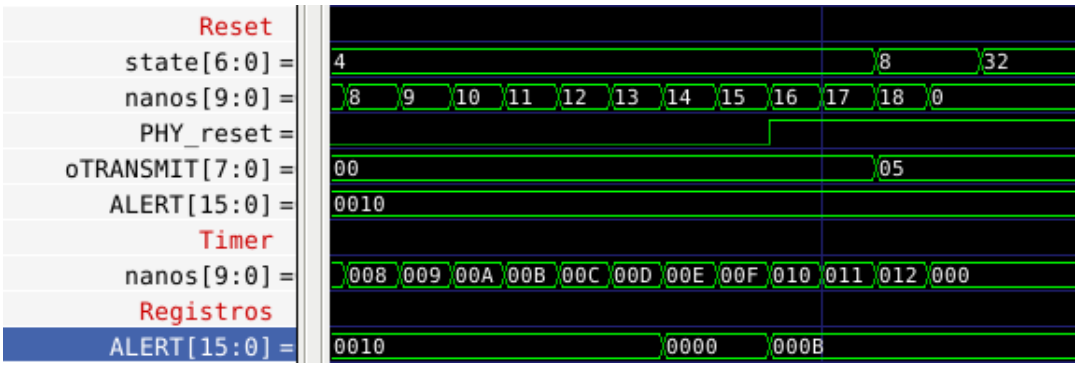

Figura 10: Detección de la bandera de Alerta de Hard Reset, salida del mensaje de éxito en oTRANSMIT

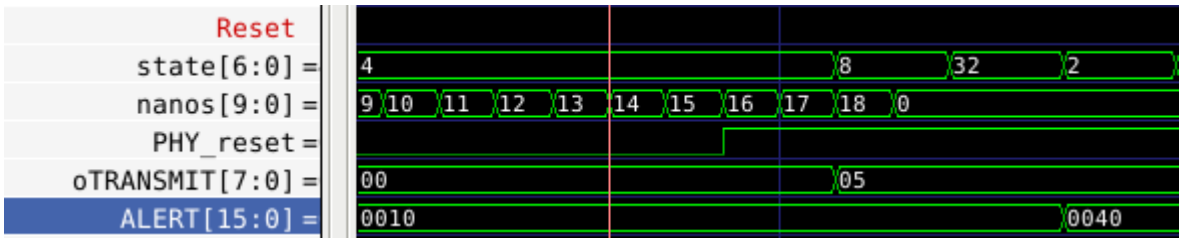

Figura 11: Mensaje de éxito de la recepción de bandera de Alerta escrita en el registro TRANSMIT

9. Conclusiones y Recomendaciones

Se logró cumplir con los siguientes aspectos referentes al desarrollo del proyecto:Conexión entre I2C y los Registros, Conexión de los Registros con las Máquinas, Comunicación del Maestro con los demás módulos por medio del I2C y Transacción del Reset Completa para el caso de un Hard Reset.

Se llegó al entendimiento de diferentes requisitos para el éxito de un trabajo cuya base está en la unificación de diferentes módulos con tareas específicas, como lo es la importancia de una o varias personas con la idea del concepto global del funcionamiento del dispositivo final, con lo cual se asegura una mejor unión de los bloques al saber cómo deben interactuar entre sí. El proceso en desarrollo requirió de analizar el código y desarrollar buenas prácticas de organización para permitir el entendimiento de las operaciones específicas por los diferentes miembros del grupo de trabajo en los bloques en los que no se estaba familiarizado por su poca intervención del grupo, ya que prácticamente a cada miembro le correspondió realizar un bloque funcional del protocolo.

Dentro de las posibilidades de mejora del proyecto están la eliminación por completo de los latches inferidos, donde aplica, sin que se vea afectada la integridad de la operación de los módulos; realizar el diseño de los mensajes de las transacciones pendientes para verificar el funcionamiento en conjunto de las máquina Tx y Rx; simplificar el código descrito.

10. Bibliografía

Referencias

[1] USB 3.0 Promoter Group. 2016. *Universal Serial Bus Type-C^{TM} Port Controller Interface Specification (Revision 1.0, Version 1.2).* (document), 3.1, 1

[2] Texas Instruments. 2017. *TUSB422 USB Type-C^{TM} Port Control with Power Delivery.* Recuperado de: http://www.ti.com/lit/ds/symlink/tusb422.pdf 3.2