Verificación Funcional de Circuitos Integrados
Tecnológico de Costa Rica
Proyecto I - TestPlan

Hernández Zárate Amanda Pérez Ramírez Gabriel

1 Funcionamiento

A continuación se trata de resumir el funcionamiento general que tiene el DUT.

- El DUT tiene como objetivo general el recibir paquetes de datos sin un orden definido, y dar como salida estos mismos paquetes pero ordenados. La distribución que se va a seguir para el proceso de ordenamiento está definida en el registro de control. Los bits de "SIZE" corresponden a la cantidad de paquetes que hay por columna, siendo `SIZE = 1`, cuatro columnas de un solo paquete, `SIZE = 2`, dos columnas con dos paquetes y `SIZE = 4`, una única columna con los 4 paquetes recibidos. Mencionar que solo estos tres datos son válidos, en caso de ingresar algún otro, resultará en un error por parte del DUT. Ahora, los bits de "OFFSET" definen dicho parámetro para los datos que se envían. Para un `SIZE = 1` son válidos valores de `OFFSET = {0, 1, 2, 3}`, para un `SIZE = 2` son válidos valores de `OFFSET = {0, 2}` y con un `SIZE = 4` es válido el valor de `OFFSET = {0}`.

Ahora se van a mostrar algunas funcionalidades concretas que tiene el dispositivo en cuestión.

- Se pueden parametrizar los módulos, siendo `ALGN_DATA_WIDTH` el ancho de los datos y `FIFO_DEPTH` la profundidad de las dos FIFOs.
- Con la señal `md_rx_valid`, el usuario indica que quiere iniciar una transacción para enviar paquetes a ordenar. Este bit debe mantenerse en alto hasta que la señal `md_rx_ready` se ponga también en alto.
- Cuando se quiere enviar un dato desde la RX FIFO para que sea ordenado, si este es válido la señal `md_rx_ready` se pone en alto, indicando que se va a proceder con la transacción. De mantenerse en bajo, se detiene el tráfico de entrada. Para que una señal se considere válida, debe cumplir con `((ALGN_DATA_WIDTH / 8) + offset) % size == 0`.
- Con el registro de control se pueden definir ciertos criterios como el tamaño, en bytes, de los datos a alinear, `SIZE`. El offset, en bytes, de los datos a alinear, `OFFSET`. Y también un bit, `CLR`, para limpiar el contador de solicitudes inválidas `CNT_DROP`.
- Con el registro de estatus se pueden observar ciertos valores de interés. En primera instancia, el contador de solicitudes inválidas, `CNT_DROP`. Una vez este llegue a su valor máximo, va a mantenerse en este, hasta que se reinicie por medio de escribir un 1 en `CTRL.CRL`. Cada vez que este contador alcanza su valor máximo, se dispara una interrupción. En segunda instancia, el estado de capacidad en la que se encuentra la RX FIFO, `RX_LVL`. En tercera instancia, el estado de capacidad en la que se encuentra la TX FIFO, `TX_LVL`.
- Se tiene también el registro habilitador de solicitudes de interrupción. En este se pueden habilitar las solicitudes de interrupción que se muestran en el registro de solicitudes de interrupción. Este registro está compuesto por: `IRQEN.RX_FIFO_EMPTY`, `IRQEN.RX_FIFO_FULL`, `IRQEN.TX_FIFO_EMPTY`, `IRQEN.TX_FIFO_FULL` y `IRQEN.MAX_DROP`.
- Además, está el registro de solicitudes de interrupción. Estos son: `IRQ.RX_FIFO_EMPTY`, `IRQ.RX_FIFO_FULL`, `IRQ.TX_FIFO_EMPTY`, `IRQ.TX_FIFO_FULL` y `IRQ.MAX_DROP`. Mencionar que estos funcionan con una lógica "sticky", la cual indica que una vez el bit se ponga en alto, la única forma de bajarlo es manualmente. Esto por medio de escribir en el respectivo campo, por ejemplo `IRQ.TX_FIFO_EMPTY`, un uno.
- El "Controller" es el responsable de realizar el alineamiento en sí, usando como parámetros `CTRL.SIZE` y `CTRL.OFFSET`. Cada vez que el RX FIFO tiene datos disponibles, este módulo los toma y los envía al TX FIFO.

- En el acceso de registro, dado requisitos de sincronización, se pueden dar estados de espera antes de brindar los datos consultados. Sin embargo, una transferencia en el APB no puede tener más de 5 estados de espera, de lo contrario se considera como ilegal.
- Detección y contador de datos inválidos y errores.
- Acceso al registro de archivo, para leer información sobre las FIFO, escribir parámetros de control e interrupciones.
- Hay interfases para leer y escribir datos.

1.1 Features

A continuación se muestra una lista de capacidades que tiene el dispositivo bajo prueba.

- Leer y escribir datos de entrada con diferentes size y offset.
- Errores en la entrada indicados (se dispara una flag) con error.
- Leer y escribir registros.
- Errores en la lectura y escritura de registros.
- Interrupciones, flagged, enmascarado.
- Contador de errores, el cual se puede resetear.
- Contador del estado de parámetros de las FIFO, los cuales se pueden resetear.
- Alineaciones de salida.

2 TestPlan

A continuación se muestra el testplan planteado, este dividido en los casos de uso general y los casos de esquina.

- Casos de uso común:
  - Aleatorizar atributos de RX.
    - `md_rx_valid`. Este corresponde a un bit que indica cuándo se quiere iniciar una transferencia de un dato para que sea alineado.
    - `md_rx_data`. Corresponde al dato que se va a enviar para ordenar. Tiene 32 bits.
    - `md_rx_offset`. Indica el offset, en bytes, en el bus del `md_rx_data`, a partir de dónde los datos válidos empiezan.
    - `md_rx_size`. Representa el tamaño, en bytes, de los datos válidos en el bus del `md_rx_data`.
  - Aleatorizar atributos de TX.
    - `md_tx_ready`. Bit que le indica al transmisor cuando ya se quiere iniciar a recibir datos.
  - Acceder a información de los distintos módulos del DUT a través del "STATUS register". Por ejemplo, `RX_FIFO_EMPTY` que indica cuando este FIFO está vacío, `RX_FIFO_FULL` que indica cuando este FIFO se llena, entre otros.
  - Overflow y Underflow. Se trata de ingresar un nuevo dato mientras la FIFO (RX o TX) está llena. No se considera caso de esquina, dado que el DUT tiene previsiones para que no se pierdan datos. Esto es, que el `md_rx_ready` se mantiene en 0, para que una transacción aunque el usuario la quiera iniciar, el DUT no la acepta.
- Casos de esquina:
  - Realizar transferencias ilegales. Por ejemplo, poner un `SIZE` y o `OFFSET` que no sean válidos, como un $SIZE = 4 \wedge OFFSET = 2$. Ver que esto va a resultar en un aumento del contador de transacciones inválidas, `CNT_DROP`.

- Forzar que indicadores que usan una lógica "sticky" se pongan en uno. Por ejemplo, los indicadores de cuando cualquiera de las dos FIFO estén ya sea vacías o llenas o el contador de transacciones ilegales se llene, se van a poner en alto y aunque su estado real cambie, este indicador no se va a actualiza hasta que el usuario lo fuerce así. Puede ser el caso en el que la RX FIFO se llene, entonces el indicador `IRQ.RX_FIFO_FULL` se va a poner en uno. Aunque después se saque un dato de esta, dejando esta de estar llena, este indicador se va a mantener en alto hasta que se baje de manera manual, por medio de escribir en `IRQ.RX_FIFO_FULL` un 1.
- Resetear indicadores con una lógica "sticky" mientras la condición sigue siendo cierta. Por ejemplo, poner el `IRQ.RX_FIFO_FULL` en 1 mientras la RX FIFO está llena.
- Forzar error en la interfaz APB, esto por medio de alguna de estas tres acciones:
  1. Acceso a una dirección de memoria inválida (no mapeada).
  2. Tratar de escribir el registro "STATUS".
  3. Tratar de escribir en "CTRL" con una combinación ilegal de valores en `CTRL.OFFSET` y `CTRL.SIZE`.
- Forzar el tener un número ilegal de estados de espera en una transferencia de APB, es decir, tener más de 5.
- Forzar a que ambos FIFOS (RX Y TX) tomen el estado de full simultáneamente.
- Desactivar `md_rx_valid` antes de que `md_rx_ready` se ponga en 1.
- Aplicar reset por ciertos ciclos, mientras que `md_rx_valid=1`, y las FIFOS se encuentran con datos y liberar el reset luego.

En seguida se muestra otro planteamiento del Test Plan, el cual fue elaborado de manera conjunta con el profesor.

Caso de prueba general: Se van a aleatorizar los siguientes datos / funcionalidades.

- **Datos de entrada:** size, entre $[1:4]$, offset, entre $[0:3]$, dato, lac combinaciones que pueda generar con 32 bits, el tiempo de acceso que hay entre las transacciones, que va a ser entre [0-8] ciclos y el hecho de que trabaje con o sin ventana. Con respecto al size y el offset, hay 7 de las 16 posibles combinaciones que son válidas, por lo que hay que generar constraints que permitan definir si se quiere trabajar con más o menos combinaciones válidas.
- **Configurar el registro de control:** Se va a especificar el size, valor entre $[1:4]$ y el del offset, valor entre $[0:3]$. De igual forma, se deben implementar constraints que permitan definir qué tipo de transacciones se quieren probar, solo válidas, solo inválidas o una combinación de ambas. Además de esto, estos cambios en la configuración se deben de realizar cada $[2:32]$ datos de entrada.
- **Contador de errores:** Aleatorizo cada cuánto voy a resetear este indicador, que es entre $[1:32]$ datos de entrada.
- **Registro de interrupciones:** Cada cuánto voy a resetear uno de los 5 bits, valores entre $[0:4]$, y cáda cuánto voy a configurar este registro, cada $[1:32]$ transacciones.
- **Reseteo del flag de las interrupciones:** Cuando se da una interrupción, se decide si resetearlo o no. Si sí se resetea, cuánto ciclos duro en resetearlo, esto entre $[1:5]$. Además, se realiza un resetero aleatorio cada $[3:32]$ ciclos. Cuando se da este reseteo, se aleatoriza cuál de los 5 bits se resetea, $[0:4]$.
- Para manejar las interrupciones, dado que usan la misma interfaz, se debe implementar una FIFO por si ambas transacciones se dan en el mismo tiempo. Además de una lógica, que puede ser aleatoria, para decidir cuál va primero.

3 Estructura del ambiente

3.1 Diagrama del ambiente

En la Fig. 1 se observan los distintos transactores conexiones entre estos, para formar el ambiente de pruebas.

3.2 Paquetes de transacción

A continuación se muestran los tipos de paquetes que se van a utilizar en el ambiente de pruebas. Además, se muestran los atributos y qué transactores van a conectar.

- **trans_apb_in:** Paquete que se usa para realizar la comunicación de entrada con la interfaz APB. Este sale del sequencer, pasa por el driver y se conecta con la interfaz del APB.

1. psel: APB select, permite selección del registros .
2. penable: APB enable, señal para habilitar el bus del APB.
3. pwrite: señal que indica una operación de escritura.
4. paddr: Dirección de 16 bits que selecciona el registro específico a acceder. Los bits [1:0] se ignoran, forzando alineamiento a palabras de 4 bytes.
5. pwdata: Datos de 32 bits que se escribirán en el registro seleccionado

- **trans_apb_out:** Paquete que se usa para realizar la comunicación de salida con la interfaz APB. Este sale de la interfaz del APB, pasa por el monitor y llega al scoreboard.

1. pready: Señal que indica que se completo la transferencia.
2. prdata: Datos de lectura de 32 bits que se reciben de los registros.
3. pslverr: Señal que indica si ha ocurrido un error en la transferencia

- **trans_rx_in:** Paquete que se usa para realizar la comunicación de entrada con la interfaz MD del FIFO RX. Este sale del sequencer, pasa por el driver y se conecta con la interfaz del MD del FIFO RX.

1. md_rx_valid: Es un solo bit, el cual se usa para indicarle al DUT que se quiere ingresar un dato al FIFO RX. Este se debe mantener en alto hasta que el md_rx_ready esté en alto.
2. md_rx_data: Dato, de 32 bits, que se va a ingresar a la FIFO RX. Este debe mantenerse constante hasta que md_rx_ready se ponga en alto.
3. md_rx_offset: Valor entre $[0:3]$ que indica el offset, en bytes, del dato de entrada a la FIFO RX.
4. md_rx_size: Valor entre $[1:4]$ que indica el size, en bytes, del dato de entrada a la FIFO RX.

- **trans_rx_out:** Paquete que se usa para realizar la comunicación de salida con la interfaz MD del FIFO RX. Este sale de la interfaz del MD del FIFO RX, pasa por el monitor y llega al scoreboard.

1. md_rx_ready: Señal de un bit que el DUT usa para indicar que los datos que se quieren ingresar son válidos y se pueden recibir.
2. md_rx_err: Señal de un bit que el DUT levanta cuando se incumplen las posibles combinaciones de offset con size.

- **trans_tx_in:** Paquete que se usa para realizar la comunicación de entrada con la interfaz MD del FIFO TX. Este sale del sequencer, pasa por el driver y se conecta con la interfaz del MD del FIFO TX.

1. md_tx_ready: Señal para indicar que el receptor esta listo para aceptar datos.
2. md_tx_err: Senal que indica si hubo una condición de error en el receptor

- **trans_tx_out:** Paquete que se usa para realizar la comunicación de salida con la interfaz MD del FIFO TX. Este sale de la interfaz del MD del FIFO TX, pasa por el monitor y llega al scoreboard.

1. md_tx_valid: Señal de un bit que el DUT usa para indicar que sí va a poder recibir en la FIFO el dato ordenado.
2. md_tx_data: Señal de 32 bits que contiene el dato de entrada del FIFO RX, pero ya alineado por el DUT.
3. md_tx_offset: Valor entre $[0:3]$ que indica el offset, en bytes, del dato que sale del alineador.
4. md_tx_size: Valor entre $[1:4]$ que indica el size, en bytes, del dato que sale del alineador.

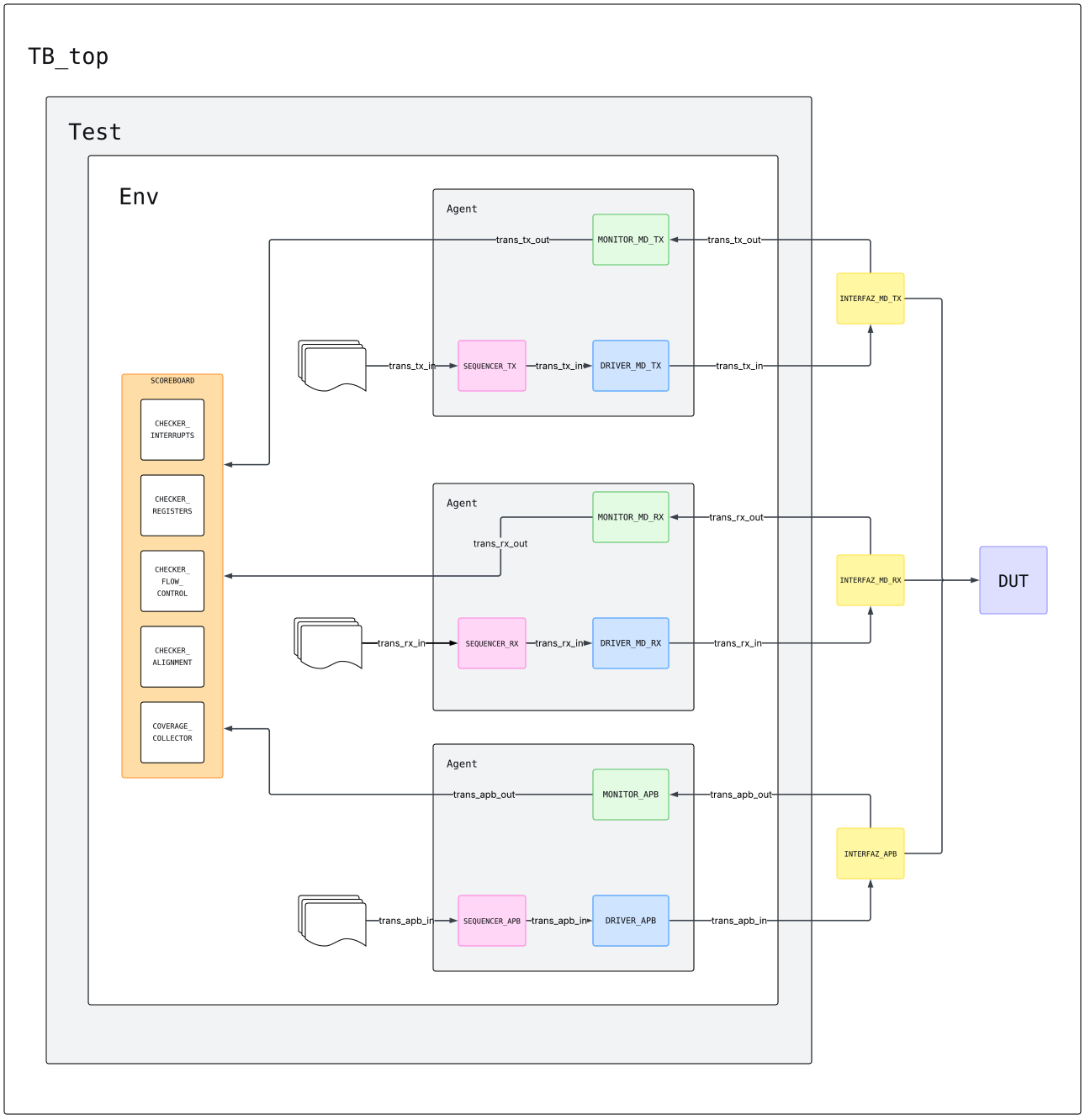

Fig. 1: Diagrama de módulos (transactores), conexiones entre estos y paquetes que se usan para comunicarse entre estos.