Diseño Automático de Sistemas Fiables

Interfaces Digitales

UNIVERSIDAD NEBRIJA

Referencias

- Guia 3, subida en la plataforma

Objetivos Clase

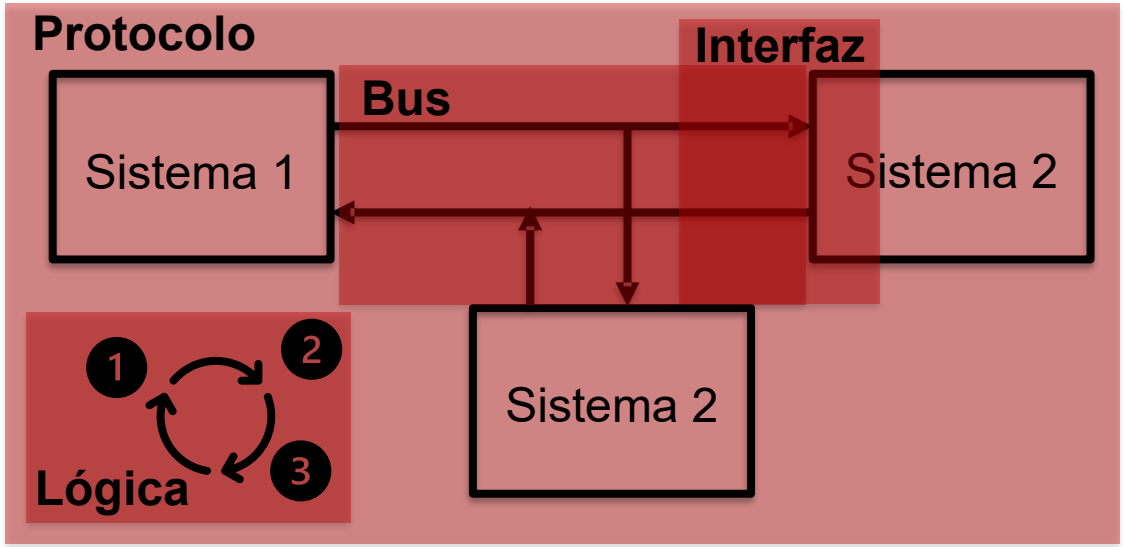
- Conocer qué es un bus, una interfaz y un protocolo y sus diferencias.
- Clasificar las diferentes interfaces según uso/función, sincronismo, ancho, direccionalidad y tipo de datos.
- Entender su uso interno para la FPGA y externo para los periféricos.

Contenidos

1. Bus, Protocolo e Interfaz
2. Clasificación según la finalidad
3. Clasificación según el reloj
4. Clasificación según el número de líneas
5. Clasificación según la direccionalidad
6. Modos de funcionamiento: Ráfagas vs bit a bit
7. Modos de funcionamiento: Paquetes
8. Métricas de una interfaz
9. Ejemplos de uso

Bus, Protocolo e Interfaz

- Bus: Sistema de comunicación entre componentes. (HW)
- Protocolo: Conjunto de reglas describen el proceso de comunicación entre dos sistemas (HW y Lógico).
- Interfaz: Elemento o bloque conceptual que realiza la conexión de un sistema con otro.



Protocolo: Necesidad

Los protocolos son necesarios en las comunicaciones para poder tener comunicaciones complejas (multipunto diferentes maestros/esclavos), fiables (corrección/detección de errores), seguras (cifrado/determinismo), responsivos(baja latencia), alto ancho de banda y cualquier otra métrica aplicable a una comunicación o a la implementación de la misma.

Cada protocolo suele estar orientado a un uso determinado y por tanto tendrá una serie de ventajas o desventajas en cada una de las áreas mencionadas.

Protocolo: No solo datos

A la hora de transferir datos es necesario acompañar los datos de una serie de señales o de bits extra (pueden ser en los propios datos añadiendo una cabecera o cola) o mediante otras líneas de conexión que indiquen información necesaria como puede ser la dirección de destino o el tamaño del paquete.

Clasificación según finalidad

La clasificación según la finalidad cubre el objetivo para el que fue diseñado el protocolo/interfaz.

- Video: VGA, HDMI, DVI, SDI
- Interconexión de periféricos: SPI, I2C, CAN, PCIe, SATA
- Interconexión interna de sistemas: Wishbone, AXI
- Audio: S/PDIF
- Comunicaciones cableadas ethernet, RS-232, RS-485
- Comunicaciones inalámbricas WIFI, BT, Lora, Zigbee

Clasificación según reloj

Los protocolos pueden ser síncronos o asíncronos. Si es síncrona se comparte la señal de reloj entre los sistemas.

Las interfaces síncronas tienen al menos una línea de reloj y son más rápidas que sus equivalentes asíncronas.

Ejemplos de protocolos síncronos: SPI, USB, USART

En el caso de los protocolos asíncronos los relojes se pueden sincronizar/recuperar de los propios datos (Ej protocolos: Ethernet, PCIe; Ej codificación: P2.4 Código manchester o 8b/10b).

Clasificación según número de lineas

Los protocolos pueden ser paralelos o serie, dentro de los serie puede haber diversas líneas serie como por ejemplo los PCIe(x1, x4, x16).

Los puertos paralelo cada vez son menos usados en interfaces externas debido a la dificultad de sincronizar los diferentes bit de una misma palabra y a los problemas derivados de las interferencias.

Sin embargo, es muy habitual usar interfaces paralelas en la conexión entre bloques interna de una FPGA.

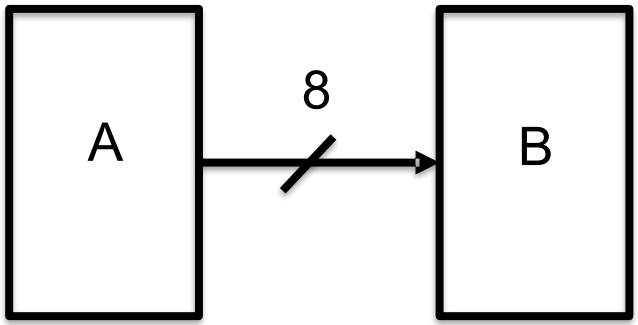
Clasificación según la direccionalidad

Según la dirección de la comunicación tenemos: simplex, half dúplex y full dúplex.

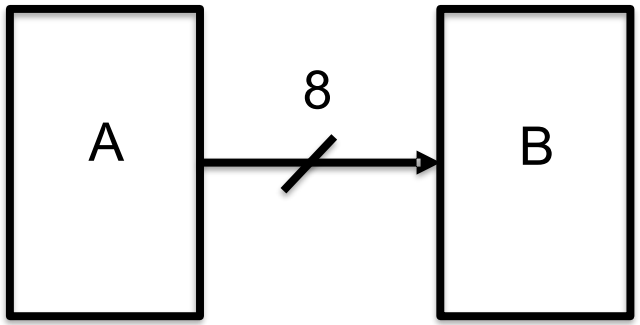
- Simplex: Unidireccional, Ej: Mando a distancia, S/PDIF
- Half Duplex: Bidireccional pero solamente una dirección en cada instante de tiempo. Ej: Walkie Talkie, I2C, WiFi
- Full Duplex: Bidereccional Ej: Teléfono, SPI, Ethernet

Ejemplo: Diseño de un protocolo desde 0

Deseamos enviar una palabra de 8 bits entre dos sistemas (A y B).
Para ello la aproximación más sencilla es una interfaz paralelo:



Ejemplo: Diseño de un protocolo desde 0

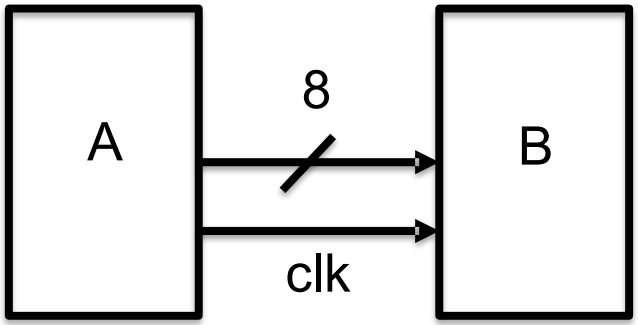



```
1  entity basic_protocol is
2  end basic_protocol;
3
4  architecture parallel of basic_protocol is
5      signal bus_line : std_logic_vector(7 downto 0);
6      component A is
7      port(
8          TX out std_logic_vector(7 downto 0)
9      );
10     end A;
11     component B is
12     port(
13         RX in std_logic_vector(7 downto 0)
14     );
15     end B;
16 begin
17     component A
18     port map(
19         TX => bus_line
20     );
21     component B
22     port map(
23         RX => bus_line
24     );
25
26 end parallel;
```

Ejemplo: Diseño de un protocolo desde 0

Problemas -> Solución:

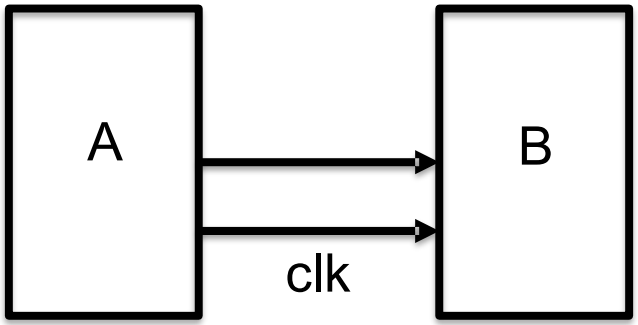
- Ambos sistemas deben estar sincronizados -> Añadir una linea de reloj



Ejemplo: Diseño de un protocolo desde 0

Problemas -> Solución:

- Sincronismo de la señal en paralelo-> Usar una interfaz serie



Ejemplo: Diseño de un protocolo desde 0

Problemas -> Solución:

- Comienzo de la trama -> Usar una cabecera

Cabecera:

Subir el valor de 0 a 1 (Lo mismo que hace el protocolo RS232).

Ejercicio: Diseñar la maquina de estados para este receptor.

Ejemplo: Diseño de un protocolo desde 0

Problemas -> Solución:

- Integridad de la señal -> Añadir un bit de paridad.

Bit de paridad:

Puede ser par o impar (even or odd) y consiste en comprobar si el numero de 1s en la palabra es par o impar

Ejercicio: Diseñar el calculo del bit de paridad

Modo de funcionamiento: Ráfaga vs bit a bit

En la mayoría de protocolos es posible una vez establecida la conexión comunicar una ráfaga de datos (burst), de esta manera se reduce el overhead (sobrecoste) introducido por el protocolo.

Para poder implementar esta característica es necesario tener un buffer (habitualmente una FIFO), en ambos extremos para poder gestionar la ráfaga.

Modo de funcionamiento: Paquetes

Otra estrategia habitual en los protocolos es añadir una cabecera (header) y una cola (tail), que de funcionalidades extra como puede ser dirección de destino del paquete de datos, longitud del paquete, bits de paridad o de crc.

Esta técnica permite mejorar ciertas características del protocolo como puede ser seguridad, integridad de los datos, flexibilidad, sencillez de la capa física o velocidad.

Ejemplos de uso

Vamos a revisar una serie de protocolos de comunicación que se utilizan habitualmente en sistemas electrónicos.

Para la comunicación interna entre diferentes bloques los más habituales son AXI y Wishbone.

Métricas de una interfaz

Cuando usamos una interfaz interna los objetivos son:

- Ancho de banda: El ancho de banda máximo teórico esta limitado por la frecuencia del sistema y el ancho del bus (n bits paralelo).
- Latencia: En un sistema por ráfagas la latencia se mide como la latencia de la primera transferencia, el objetivo es tener latencia 0

Protocolo AXI (I)

AXI es una interfaz definida por ARM y que es la base de sus procesadores y de la flexibilidad que ofrecen para añadir diferentes periféricos y coprocesadores dedicados. El estándar es público y cada vez se utiliza más en el entorno de las FPGAs para conectar los subsistemas.

Protocolo AXI (II)

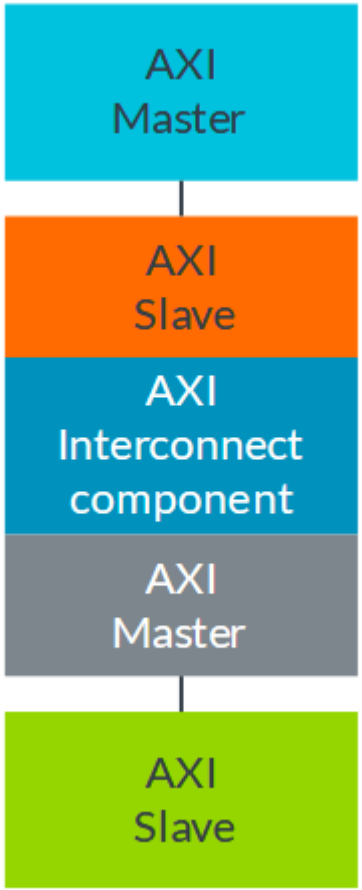
Beneficios: Reusabilidad de IP, Flexibilidad, Compatibilidad, Soporte oficial

Características clave de rendimiento en un bus:

- Ancho de banda: El ancho de banda máximo teórico esta limitado por la frecuencia del sistema y el ancho del bus (n bits paralelo). El bus AXI puede trabajar muy cerca de ese limite.
- Latencia: En un sistema por ráfagas la latencia se mide como la latencia de la primera transferencia, el objetivo es tener latencia 0. AXI esta diseñado para minimizar la latencia

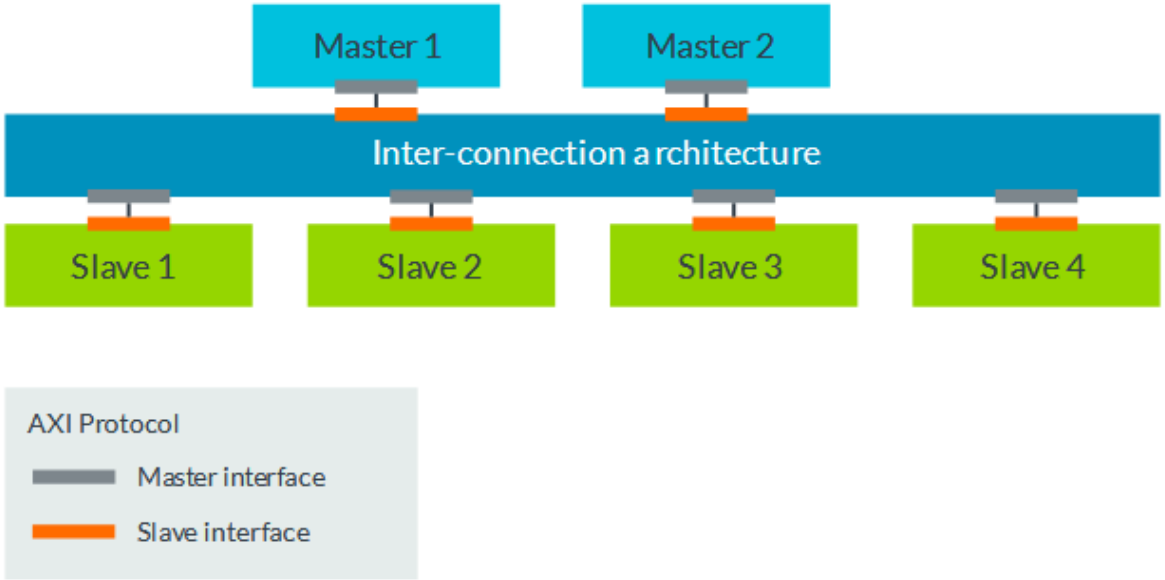
Protocolo AXI (III)

Es un protocolo punto a punto que tiene dos tipos de interfaz esclavo y maestro



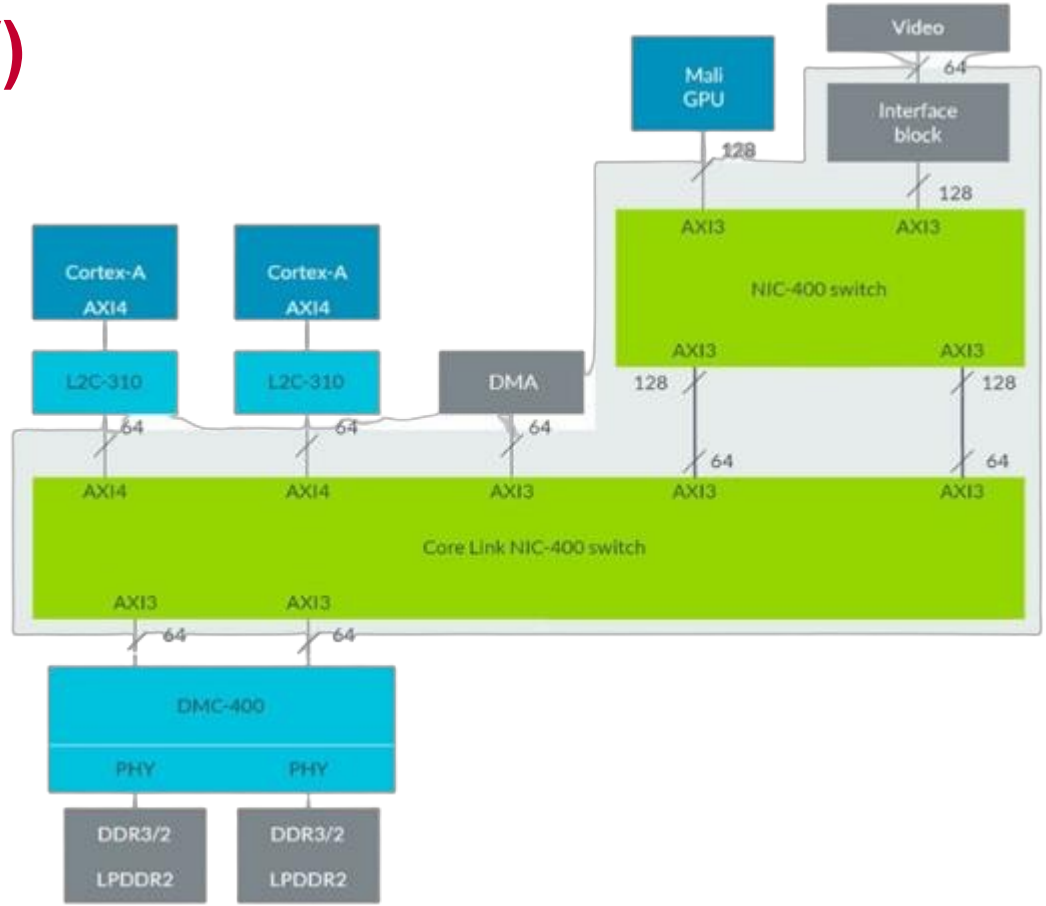
Protocolo AXI (IV)

Para poder usarlo como bus es necesario usar una arquitectura de interconexiones entre los esclavos y los maestros



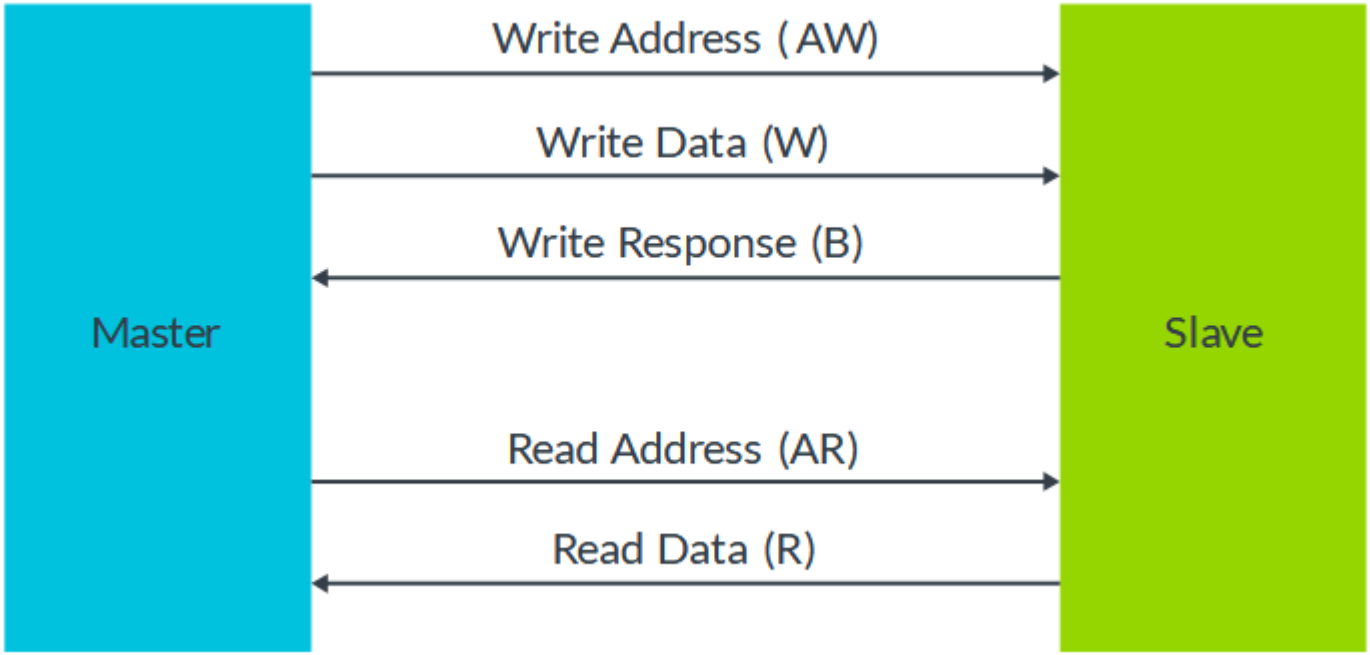
Protocolo AXI (V)

Aplicado a un SoC (System on Chip), usado en smartphones.

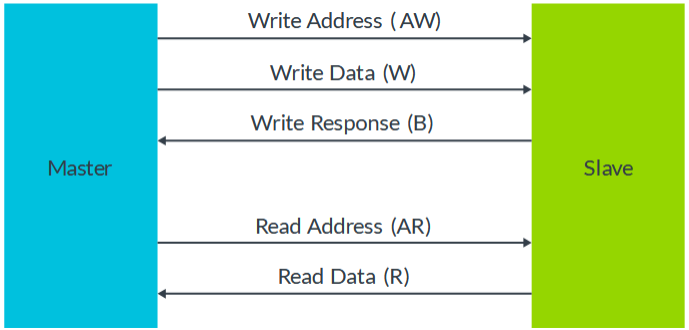


Protocolo AXI (VI)

Tiene 5 canales principales:



Protocolo AXI (VII)



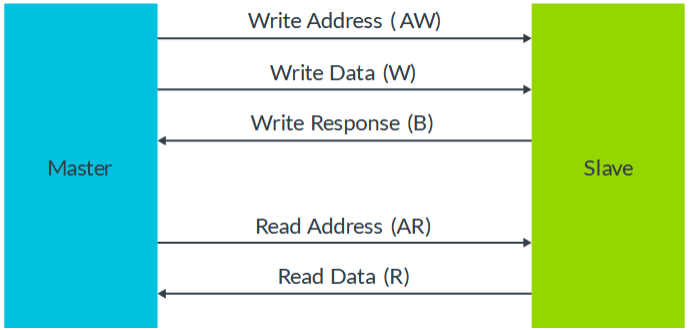
Proceso de escritura

- El maestro escribe la dirección en Write Address (AW) y escribe el dato en Write Data
- El esclavo cuando ha leído el dato escribe en Write Response (B)

Proceso de lectura

- El maestro escribe la dirección en Read Address (AR)
- El esclavo devuelve el dato en Read Data (R)

Protocolo AXI (VI)



En esta explicación se ha simplificado mucho el protocolo AXI, habitualmente se utilizan IPs para implementarlo.
El protocolo AXI permite bloquear accesos durante una transacción determinada.

Es necesario iniciar la transferencia de datos primero mediante un handshake en el que se comprueba que la dirección solicitada para lectura/escritura es valida.