Interfaz WISHBONE

Interfaz WISHBONE

Introducción

El presente documento resume los aspectos más importantes de la Revision B.3 de la especificación WISHBONE.

El objetivo de la especificación Wishbone es crear una interfaz común entre IP Cores.

Define una forma estándar de *intercambiar datos* entre módulos IP Core. No pretende especificar el funcionamiento del core, solo la forma de comunicarse con él.

La arquitectura WISHBONE es análoga al bus de un microprocesador.
- ofrece una solución flexible de integración
- ofrece varios tipos de ciclos y anchos de bus para resolver distintas situaciones.
- permite que una misma aplicación sea desarrollada por varios fabricantes.

Características

Las características más sobresalientes de la especificación son:
- Basado en protocolos estándares de transferencia de datos
  - ciclos READ/WRITE
  - ciclos de transferencia en BLOQUE
  - ciclos RMW.
- Soporta varios tipos de interconexionamientos
  - punto a punto
  - bus compartido
  - switch de interconexiones.
- Protocolo de Handshake para regular la velocidad de transferencia de datos.
- Soporta varias terminaciones de ciclos
  - normal
  - con retry
  - por error
- Basado en arquitectura Maestro/Esclavo

Terminología

La especificación WISHBONE define los siguientes términos.

Regla

Todas **DEBEN** ser seguidas para asegurar una compatibilidad entre interfaces.

Recomendaciones

Cuando aparece una recomendación, **SE ACONSEJA** seguirla. El no adoptarla puede implicar una perdida de performance.

Muchas recomendaciones están basadas en la experiencia acumulada de los diseñadores que escribieron la especificación.

Sugerencia

Es **UN CONSEJO** que puede ser considerado por el diseñador. Puede ser útil, pero no vital para el funcionamiento de la interfaz.

Permiso

Cuando se indica que algo **PUEDE** hacerse o no, la decisión de implementarlo puede quedar a cargo del diseñador.

Observación

Usualmente son textos que clarifican alguna situación, pero no aportan nada más que eso.

Convención para el nombre de la señales

Todas las señales tiene "_I" y "_O" para indicar si son salidas o entradas al Core.

Los buses son indicados por nombres seguidos de (). Por ejemplo DAT_I().

Pueden utilizarse señales especiales definidas por el usuario, por ejemplo PAR_O. Estas señales son llamadas *tags*. Los tags tiene asociados un *TAG TYPE* que indican en que momento del ciclo tienen un valor valido.

Especificación de la Interfaz

Documentación de la interfaz

Al especificar la interfaz, debe de incluirse la siguiente información.

1. Nº de revisión WISHBONE con la cual es compatible el Core. Pare este caso **B.3**
2. Tipo de Interfaz: Maestro o Esclavo
3. El nombre de las señales definidas en la interfaz WB.
4. En caso de soportar señalización de Errores
   - Master (ERR_I) debe indicar como reacciona.
   - Slave (ERR_O) debe indicar cuales son las condiciones que activan la señal.
5. En caso de soportar señalización Retry
   - Master (RTY_I) debe indicar como reacciona.
   - Slave (RTY_O) debe indicar cuales son las condiciones que activan la señal.
6. Todas las interfaces que soportan TAGS, deben indicar su nombre, el tipo y su funcionamiento
7. Ancho del bus de datos (8, 16, 32, 64)
8. Granularidad del puerto.
9. Tamaño máximo del operando, en caso de no conocerse, es igual al ancho de bus.
10. BIG ENDIAN o LITTLE ENDIAN
11. Si hay restricciones sobre la señal CLK_I.

Niveles activos

Todas las señales son activas por nivel alto.

Señales en la interfaz

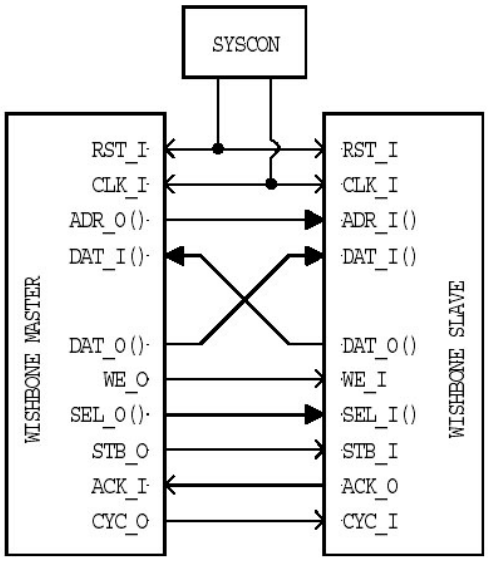
La idea es que las señales permitan las conexiones punto a puto, bus compartido, etc.

Permiten tres ciclos básicos: Read, Write y RMW. Pero no hay obligación de soportarlos a todos.

Permiten un Handshake para adecuar la velocidad de transferencia de datos, e indicar errores y retry.

Las señales no son Bidireccionales, siempre son entradas o salidas. Esto es así pues muchas veces el diseño puede llagar a querer implementarse en hardware que no soporta internamente señales bidireccioneles, por ejemplo los FPGAs de Altera.

Ejemplo de interfaz e interconecciones en una arquitectura punto a punto:



Comunes a Master y Slave

- CLK_I: entrada de reloj
- RST_I: reset del diseño
- DAT_I() y DAT_O(): buses de entrada y salida de datos
- TGD_I() y TGD_O()
  - Opcional
  - Lleva información asociada al bus de datos
  - Validas cuando STB_O activo
  - Ej: paridad

Master

- ADR_O(N..n)
  - N: Limite superior dado por el ancho del bus de direcciones
  - n: limite inferior dado por la granularidad del bus de datos.
- CYC_O: Indica que se esta llevando a cabo un ciclo de bus válido. Se activa al comienzo del ciclo y permanece hasta el final.
- WE_O: Indica si el ciclo es de lectura o escritura.
- STB_O: indica que se esta llevando a cabo un ciclo valido de transferencia de datos.
- ACK_I: recibe la confirmación de una transferencia.
- ERR_I: recibe la indicación de un error en la transferencia.
- RTY_I: recibe pedido de re-transmisión de datos.

- LOCK_O: indica que el ciclo que se esta llevando a cabo no puede ser interrumpido.
- SEL_O(): asociado con la granularidad, indica donde hay o donde espera datos validos en el bus.
- TGA_O
  - Información asociada a las lineas de direcciones
  - Validas cuando STB_O activo
- TGC_O
  - Información asociada con el tipo de ciclo de bus
  - Validas cuando CYC_O activo

Slave

- ADR_I(N..n):
  - N: Limite superior dado por el ancho del bus de direcciones
  - n: limite inferior dado por la granuralidad del bus de datos.
- CYC_I: indica el comienzo de un ciclo.
- WE_I: indica si la transacción es de lectura o escritura.
- STB_I: indica que se esta llevando a cabo un ciclo valido de transferencia de datos.
- ACK_O: indica que la transferencia se ha realizado en forma exitosa.
- ERR_O: utilizado para señalizar un error en la transacción.
- RTY_O: utilizado para pedir una reintento en la transacción.
- LOCK_I: indica que el ciclo que se esta llevando a cabo no puede ser interrumpido
- SEL_I(): asociado con la granularidad, indica donde hay o donde espera datos validos en el bus.
- TGA_I
  - Información asociada a las lineas de direcciones
  - Validas cuando STB_I activo
- TGC_I
  - Información asociada con el tipo de ciclo de bus
  - Validas cuando CYC_I activo

WISHBONE clásico

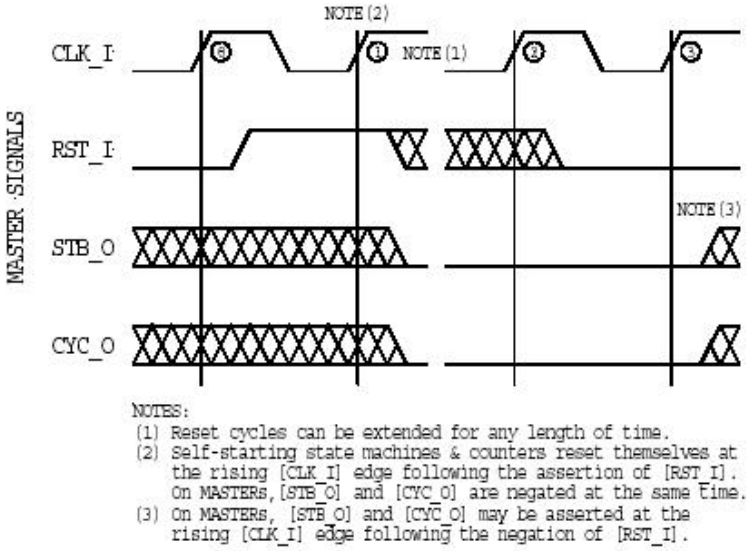
A partir de la revisión B.3, se agregó una nueva modalidad de transacciones y se agregaron TAGs de señalización.
Los cambios permiten una mayor velocidad en la transferencia de datos, pero se mantiene la compatibilidad con las revisiones anteriores del estándar Wishbone.

A la manera de realizar transacciones de las versiones anteriores se les llama "Wishbone clásico"

Funcionamiento general

Reset

EL reset se produce en forma síncrona.



Todas las interfaces WISHBONE deben de inicializarse con el primer flanco de subida en el que RST_I este activo.

Inicialización de ciclo de transacción.

Se indica que hay un ciclo válido si CYC_O esta activo. Cuando CYC_O es 0 ninguna de las otras señales del MASTER tienen sentido.

Protocolo de Handshake

El hanshake para la transferencia de datos es sumamente sencillo, el MASTER activa STB_O y lo mantiene así hasta que el esclavo activa alguna de las señales ACK_I, ERR_I, RTY_I. Al ocurrir esto el Maestro desactiva la señal.

Las señales de respuesta ACK_O, ERR_O, RTY_O deben generarse solo si están activas CYC_I y STB_I.

Las interfaces del esclavo **DEBEN** de ser diseñadas para que ACK_O, ERR_O, RTY_O se activen y desactiven respondiendo a STB_O.

Uso de TAGS

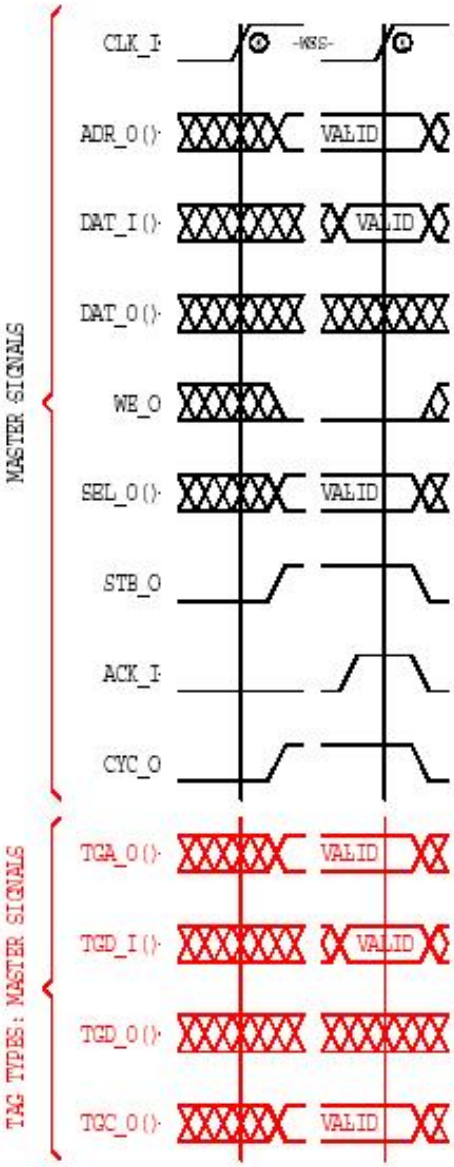
Descripción	TAG TYPE	Asociado a
Address tag	TGA_I/O()	ADR_I/O()
Data tag,input	TGD_I()	DAT_I()
Data tag,output	TGD_O()	DAT_O()
Cycle tag	TGC_O()	Bus Cycle

Ejemplo de definición para señal PAR_O

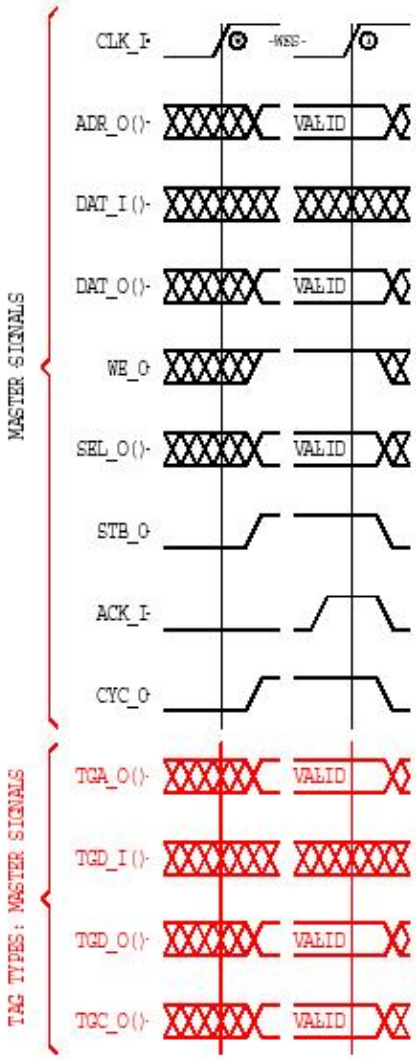
- SIGNAL NAME: PAR_O
- DESCRIPTION: Even parity bit
- MASTER TAG TYPE: TGD_O()

Ciclos READ/WRITE Simple

Read simple



Write simple



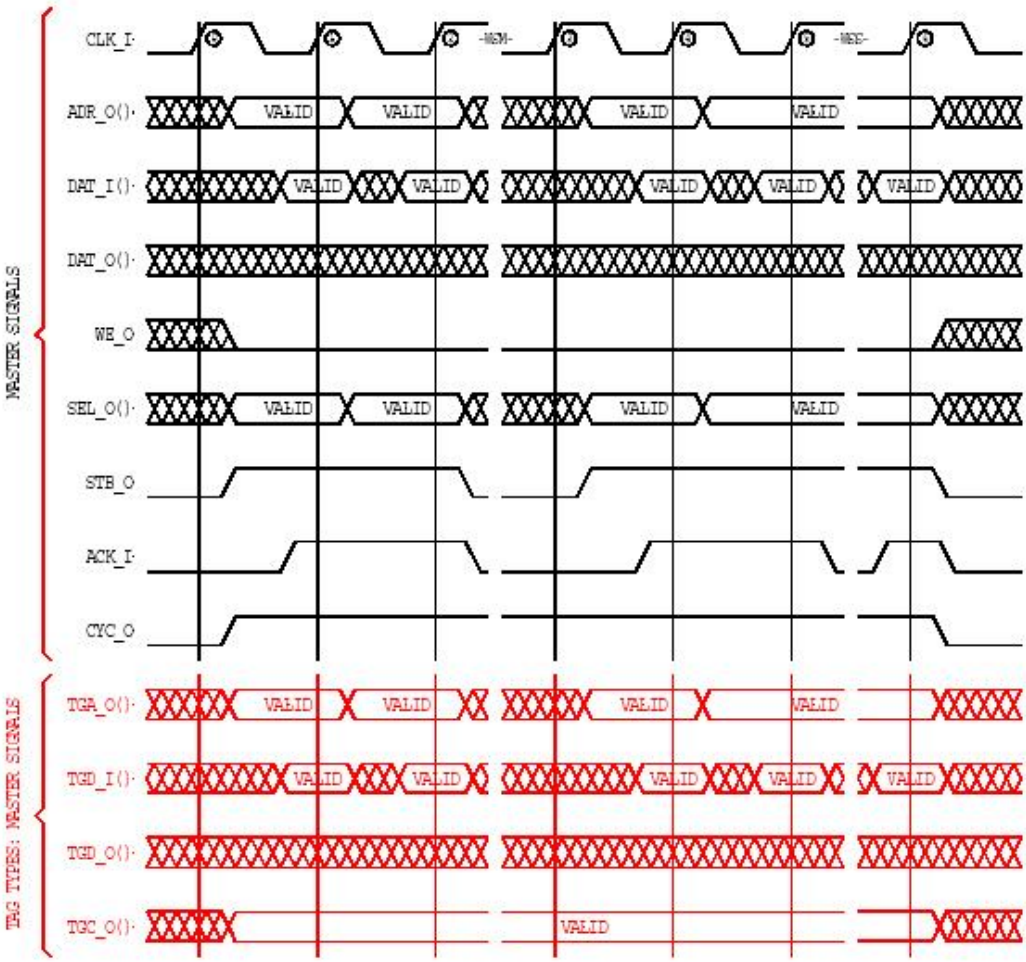
Ciclos READ/WRITE en bloque

Durante las transferencias en bloque, la interfaz básicamente realiza transferencias simples Read/Write.

Este tipo de transferencias es útil, sobre todo en sistemas con varios Master.

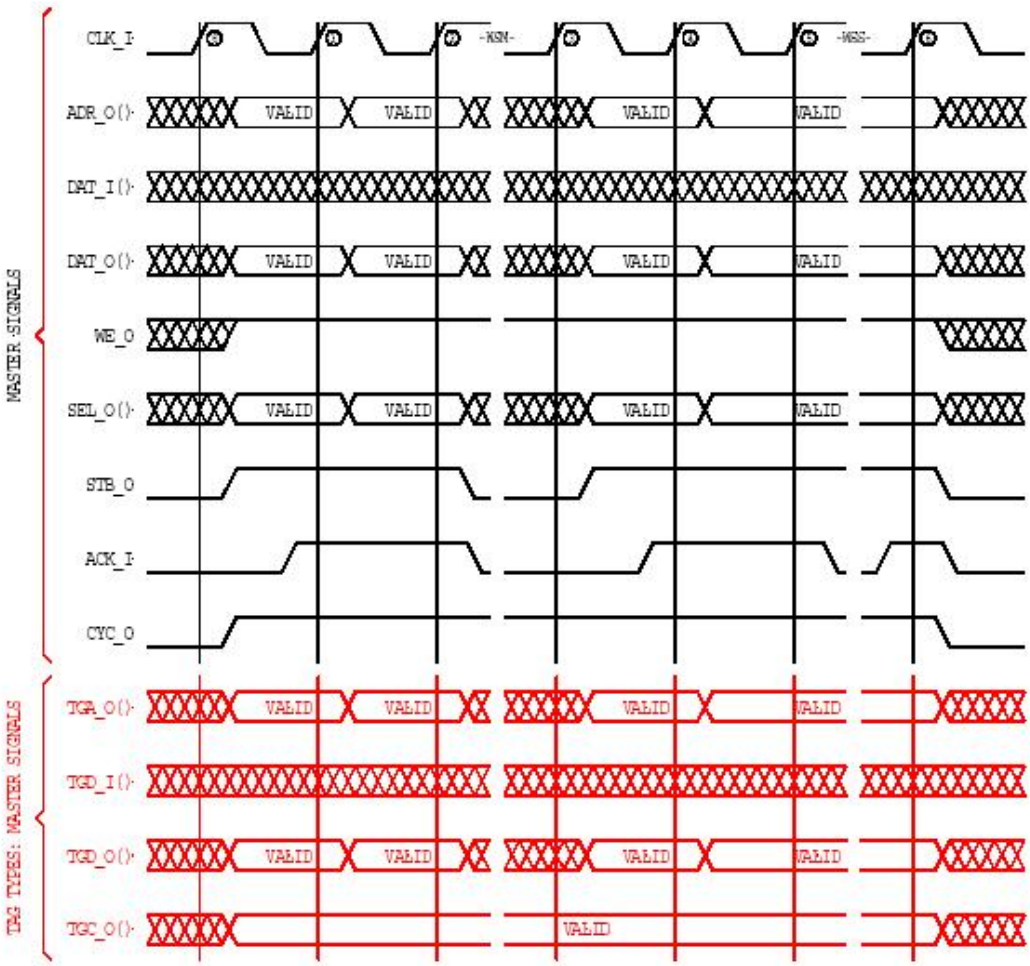
Read en Bloque

Puede realizarse una transferencia en cada ciclo de reloj.

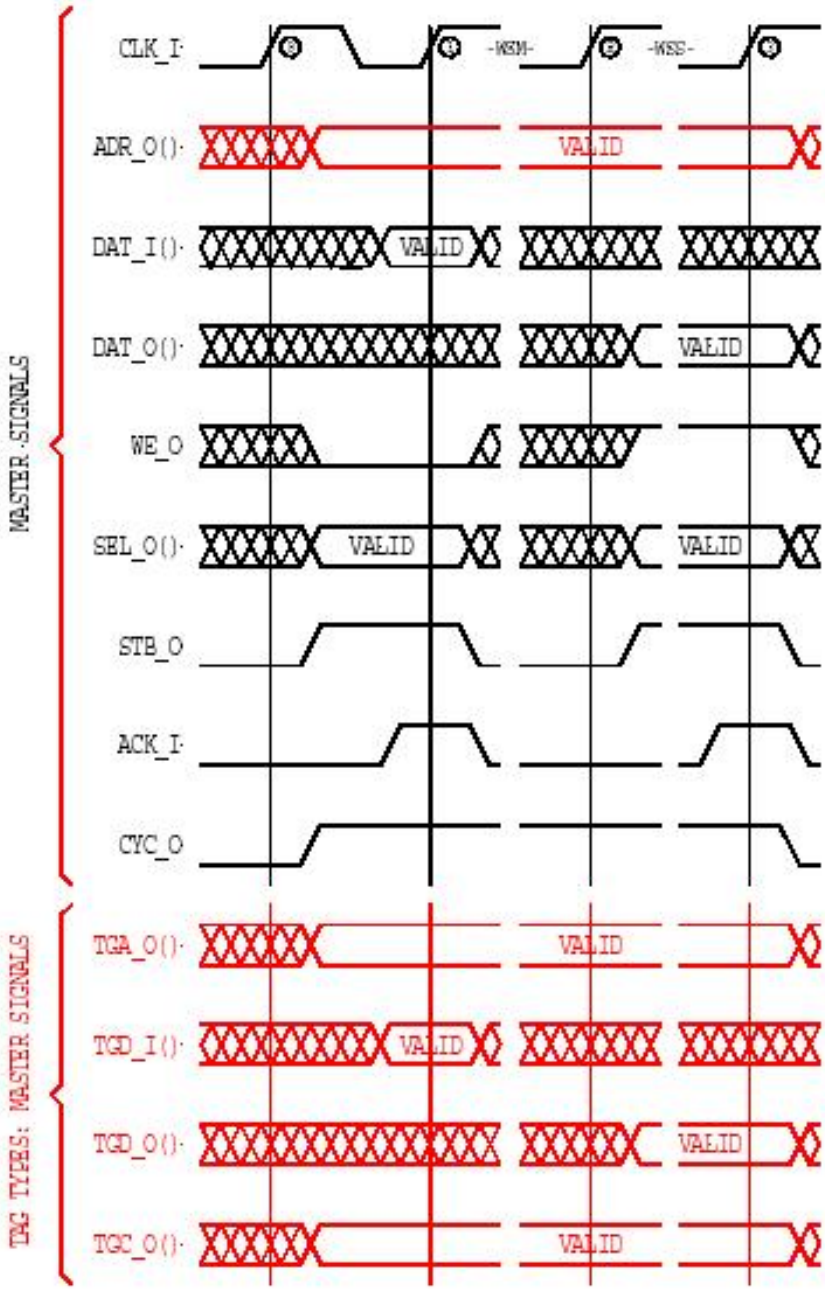


Write en Bloque

Puede realizarse una transferencia en cada ciclo de reloj.



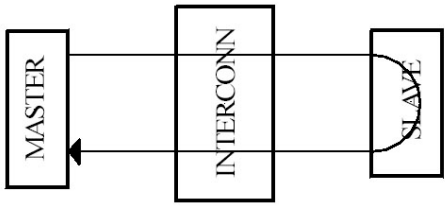
Ciclos RMW


MASTER SIGNALS
TAG TYPES: MASTER SIGNALS
CLK_I
ADR_O()
DAT_I()
DAT_O()
WE_O
SEL_O()
STB_O
ACK_I
CYC_O
TGA_O()
TGD_I()
TGD_O()
TGC_O()
-WSM-
-WSS-
VALID

WISHBONE con Ciclos de bus registrados.

Para lograr la máxima tasa de transferencia, WISHBONE necesita que las señales de terminación de ciclos (ACK_I, ERR_I,RTY_I) sean generadas en forma asíncrona. Por ejemplo ACK_O como el AND de CYC_I y STB_I.

Pero esto puede llevar a tener retardos demasiado grandes en el chip, dado que una señal pasa por varias compuertas.



La solución para esto es registrar las señales de terminación de ciclos.

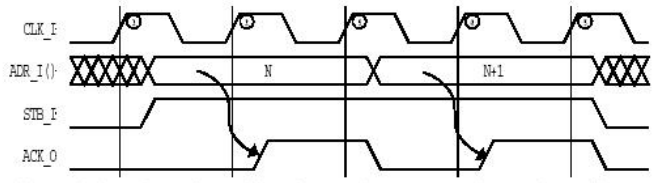


Figure 4-3 WISHBONE Classic synchronous cycle terminated burst

El problema de esto es que genera un WAIT STATE, reduciendo a la mitad la taza de transferencia.

Esto se soluciona con TAGS que identifican el tipo de ciclo que se esta llevando a cabo. Se puede indicar si la transferencia actual es la última o si se siguen transfiriendo datos aún. Con esto se sabe si se debe desactivar ACK_O o mantener activo en el siguiente período de reloj.

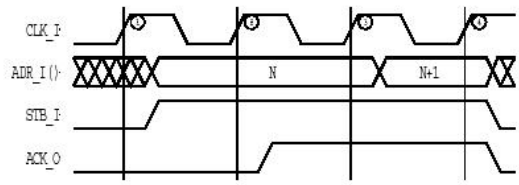


Figure 4-4 Advanced synchronous terminated burst

La interfaces que funcionan como WISHBONE con ciclos registrados, también soportan WB clásico.

TAGS utilizados

CTI_IO()

- Cycle Type Identifier CTI_O() del tipo TGC_O() y CTI_I() del tipo TGC_I()
- El MASTER es el que envía la información. El ESCLAVO puede utilizarlo para preparar su respuesta.
- Para aprovechar al máximo el ancho de banda , tanto la interfaz Master como esclava deben de soportar el modo de trabajo.

CTI_O(2:0)	Descripción
000	Classic cycle.
001	Constant address burst cycle
010	Incrementing burst cycle
011	Reserved
100	Reserved
101	Reserved
110	Reserved
111	End-of-Burst

Las interfaces que soportan Ciclos Registrados, deben soportar al menos el modo clásico, para asegurar compatibilidad.

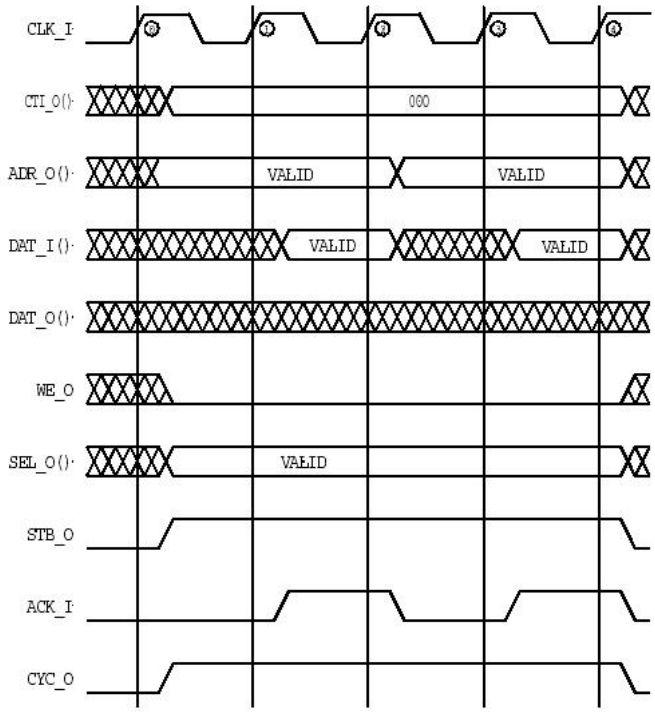
BTE_IO()

- Burst Type Extension BTE_O() y BTE_I() del Tipo TGA_N()
  - Dan información adicional de como se incrementan las direcciones.

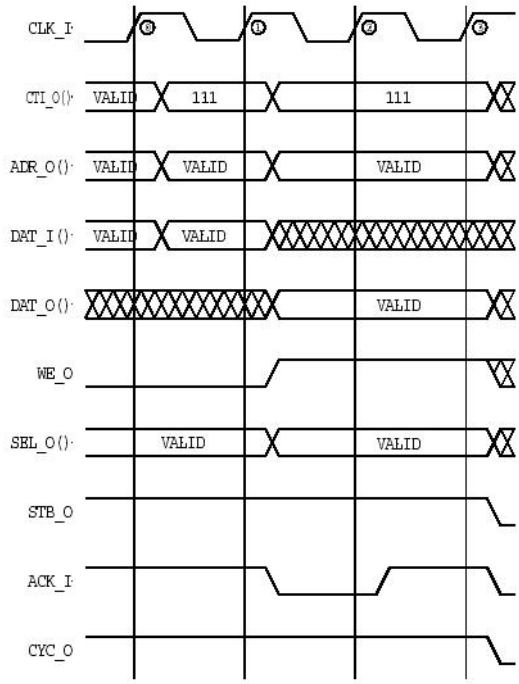
BTE_IO(1:0)	Descripción
00	Linear burst
01	4-beat wrap burst
10	8-beat wrap burst
11	16-beat wrap burst

Las interfaces que soportan BURST incremental, deben de soportar BTE_O() y BTE_I().

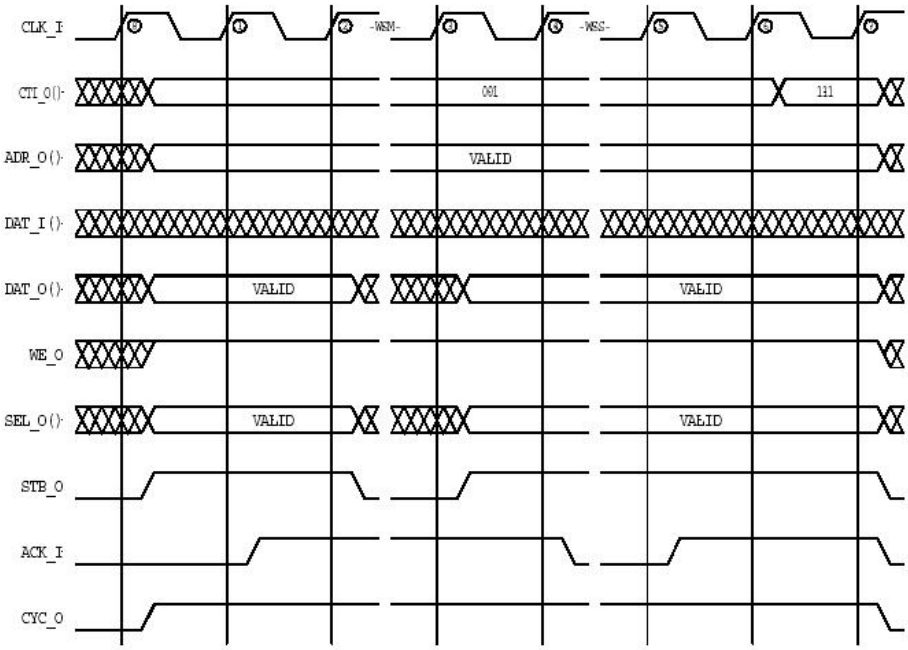
Classic Cycle



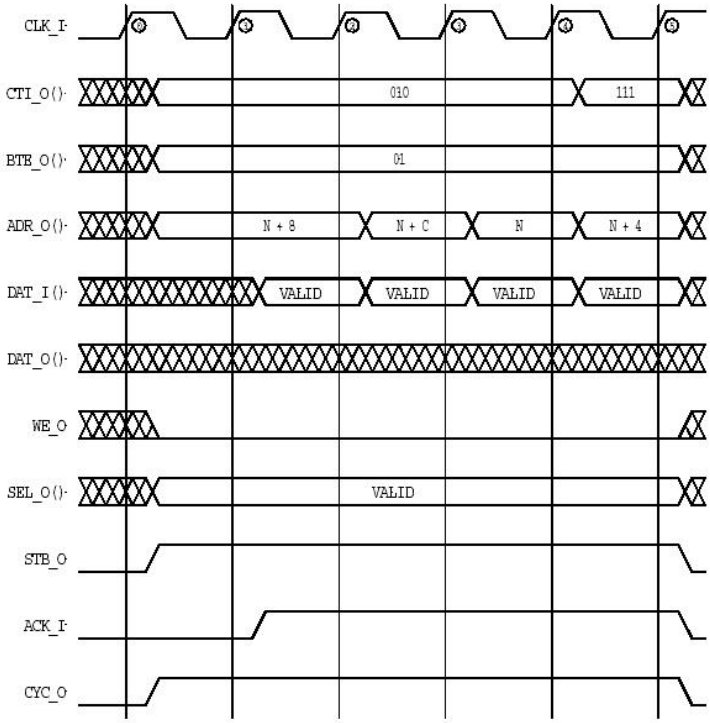
End Of Burst



Constant Address Cycle



Incremental Address Burst



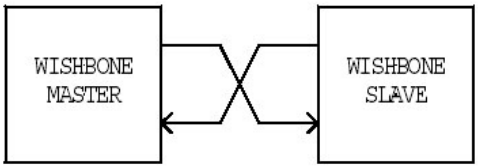
Interconecciones de Módulos

WISHBONE facilita el trabajo a los diseñadores, ya que estandariza la interfaz.

Esta basado en una arquitectura de Maestro/Esclavo y se comunican a través de una interfaz usualmente denominada INTERCON.

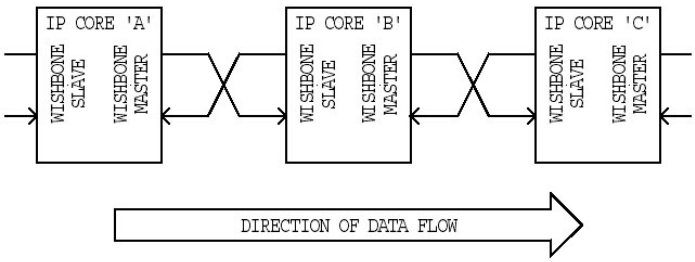
INTERCON no es nada mágico, es también un archivo más escrito en un lenguaje de descripción de HW.

Conexiones punto a punto



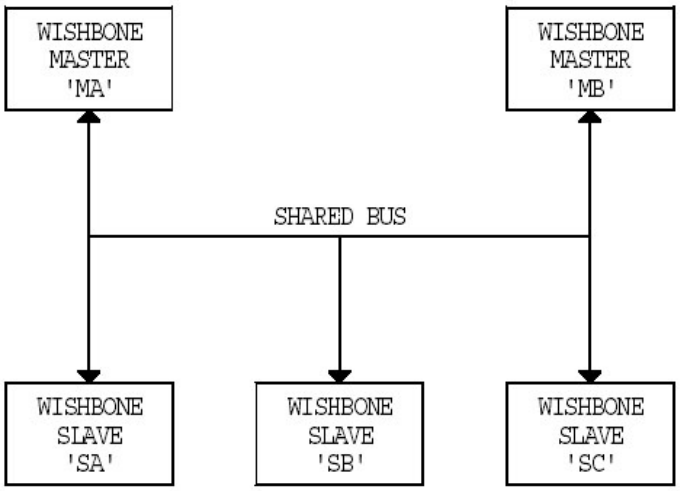
Flujo de Datos

Es una posible arquitectura para implementar diseños que procesen datos en pipeline.



Bus compartido

Todos los dispositivos están conectados sobre un mismo bus y cualquier Master puede iniciar un transacción con cualquier esclavo.



Un arbitro es el que determina cuando le toca el turno a cada uno de los Master.

Switch de interconexiones

Consume más recursos, pero permite varia comunicaciones entre pares Maestro/Esclavo a la misma vez.