Estándares para interconexión de cores en SOC

Diseño Lógico II

Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República

2017

Introducción

Para interconectar 2 diseños se deben definir:

- Interfaces (cantidad de señales, nombres)
- handshake para control de flujo
- protocolo de comunicación
- comportamiento de señales de interfaz

Importancia de los estándares

- Asegura compatibilidad entre diseños
- Acorta tiempos de diseños
- Facilita la reusabilidad de diseños
- Facilita la especificación de un diseño

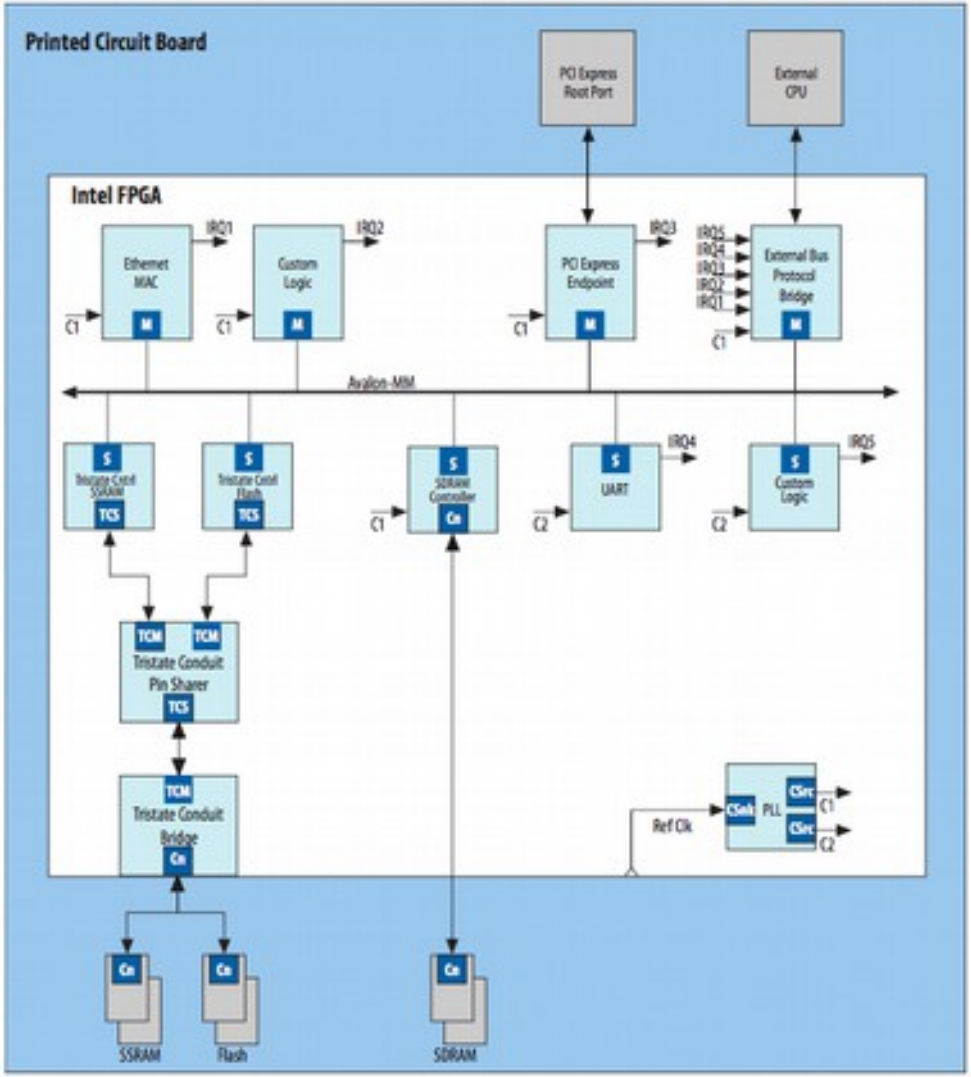
Algunas opciones

- AVALON (Altera)
- WISHBONE (Silicore Corporation)

AVALON

- Desarrollado por ALTERA.
- última revisión: 2017-05-08
- Define señales, su comportamiento, señalización de errores y tipos de transferencias entre periféricos y un "switch de interconexión"
- Multi-master, multi-slave
- Switch de inteconexión generado por wizard de Altera

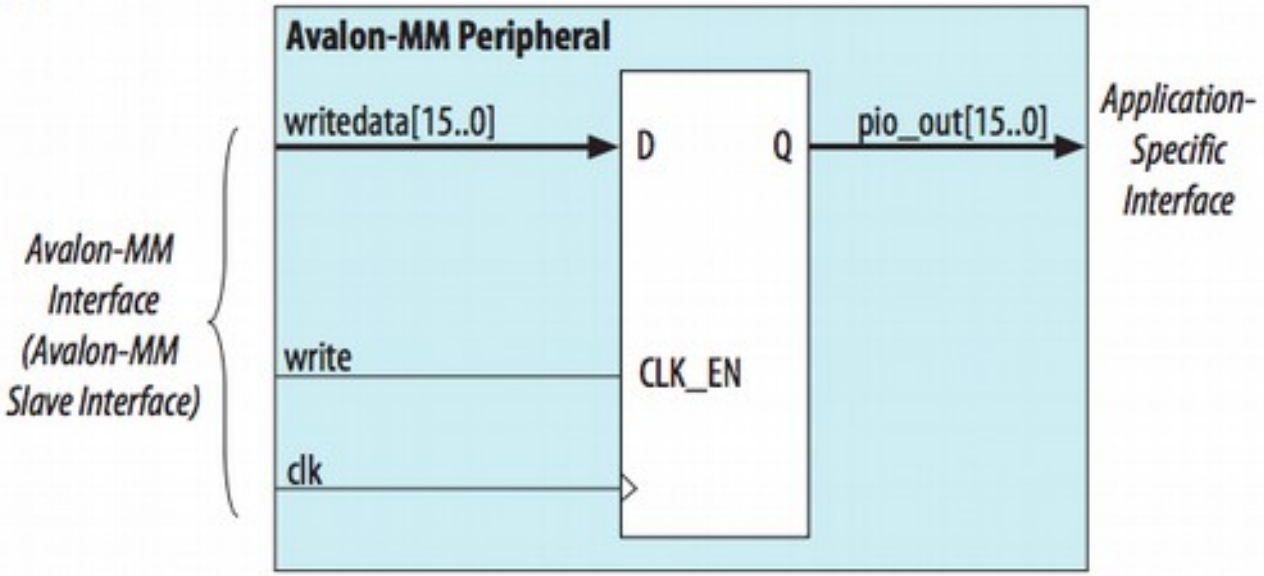
AVALON | sistema típico



AVALON | ejemplo sencillo

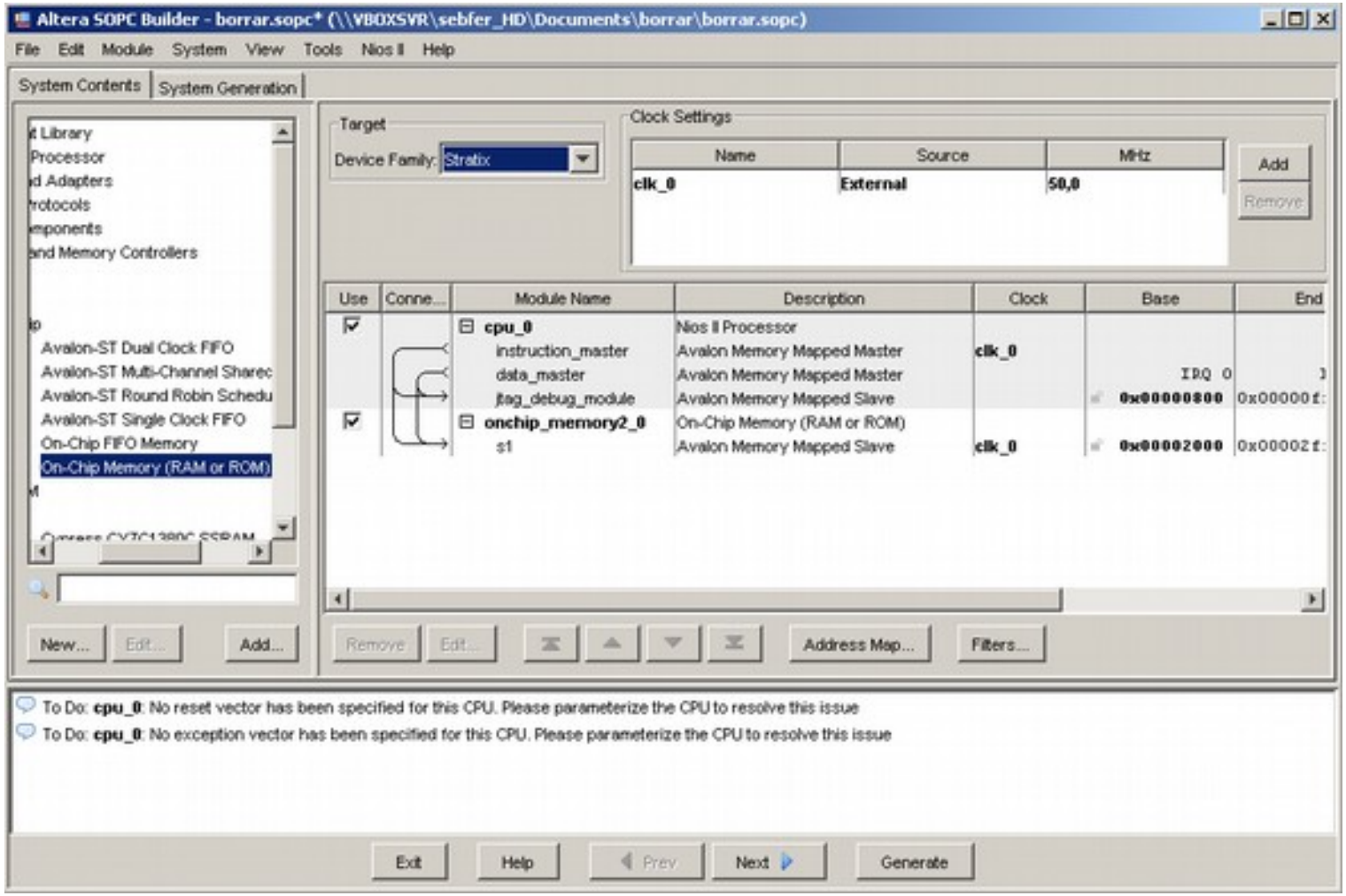
Example Slave Component

The 16-bit general-purpose I/O peripheral shown in the following figure only responds to write requests. This component includes only the slave signals required for write transfers.



Each signal in an Avalon-MM slave corresponds to exactly one Avalon-MM signal role. An Avalon-MM interface can use only one instance of each signal role.

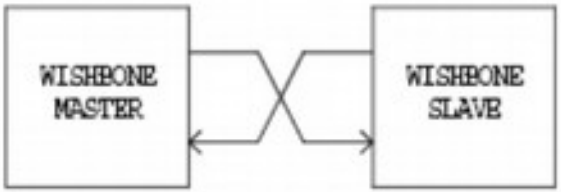
AVALON | Wizard

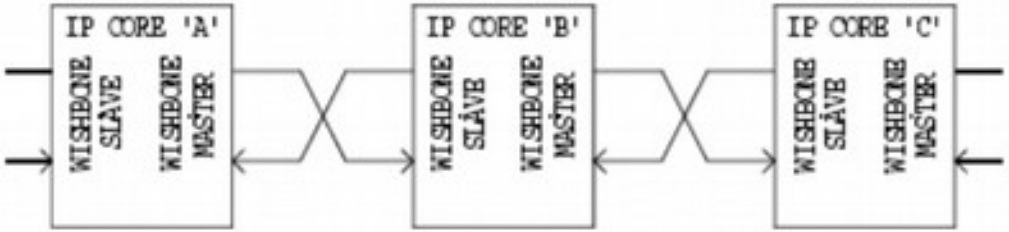


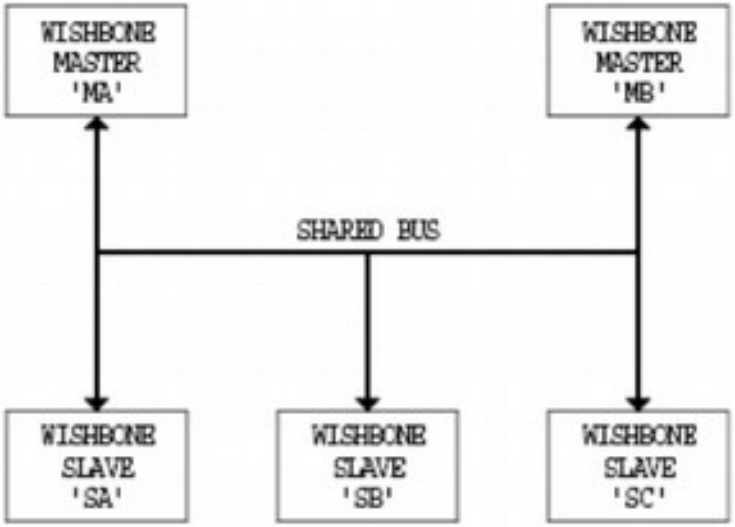
Wishbone

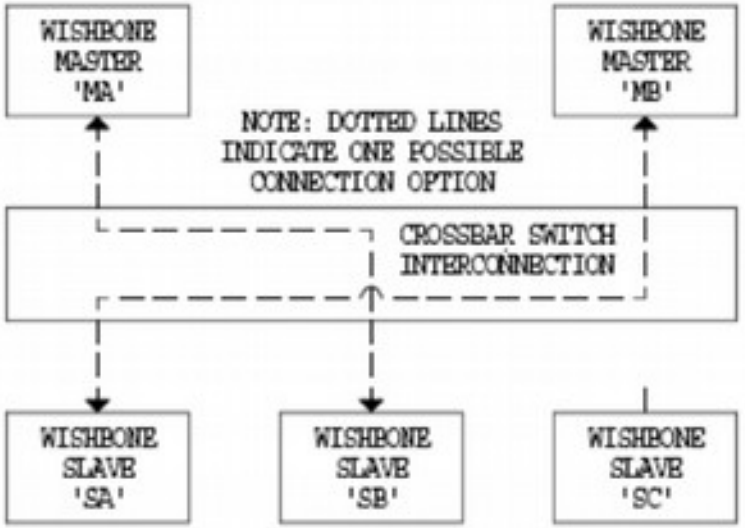
- Desarrollado por Silicore Corporation
- última revisión: B4 (2010)
- Recomendada por Opencores
- Define interfaces, deja libre las formas de interconexión.
- No especifica funcionamiento del Core.
- Orientado a arquitectura Maestro/Esclavo.
- No hay pines bidireccionales.
- Handshake para regular velocidad de la comunicación.

Wishbone | sistemas típicos



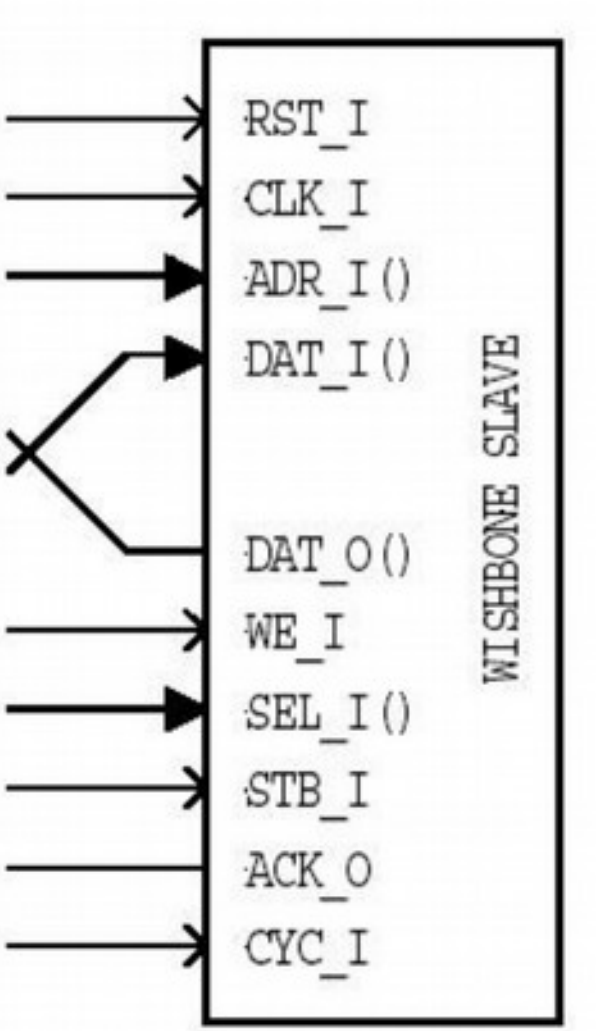






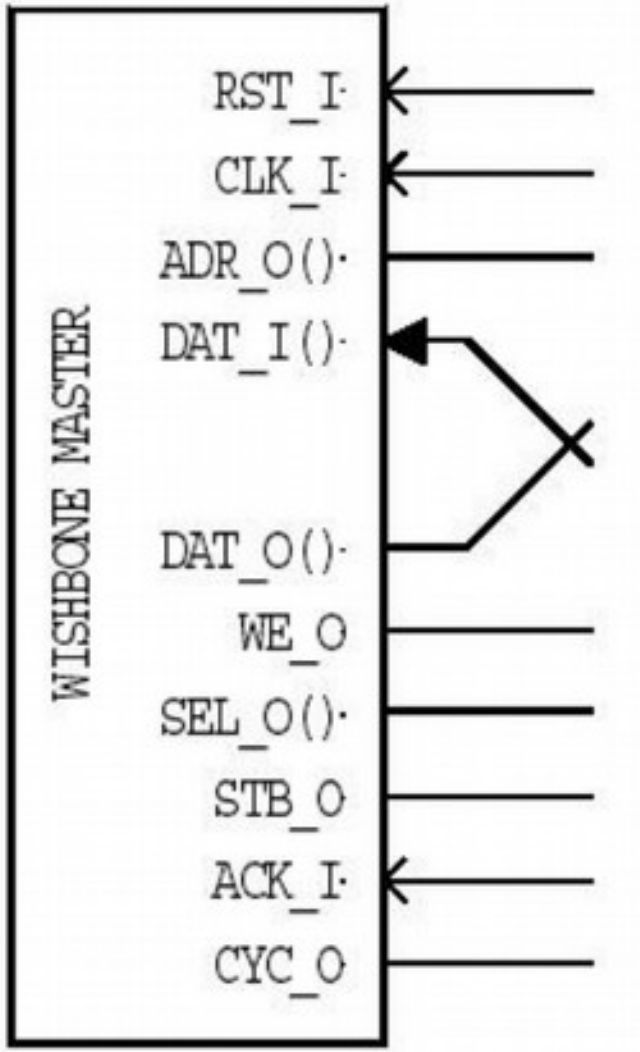
Wishbone | señales interfaz

- Esclavo
  - RST_I y CLK_I
  - DAT_I() y DAT_O()
  - ADR_I() y SEL_I()
  - CYC_I y WE_I
  - ACK_O, ERR_O, RTY_O
  - LOCK_I
  - STB_I
  - TGn_I()



Wishbone | señales interfaz

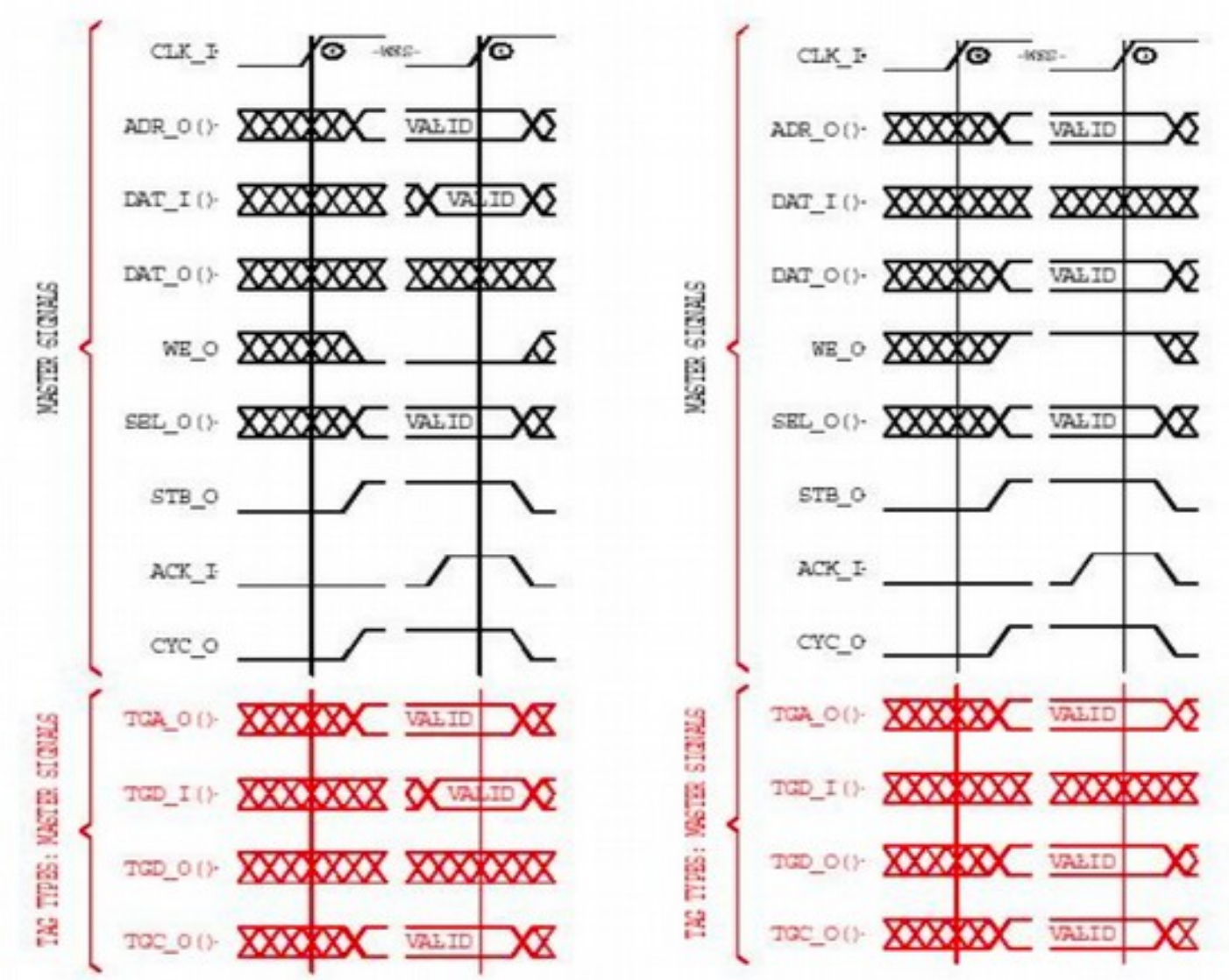
- Maestro
  - RST_I y CLK_I
  - DAT_I() y DAT_O()
  - ADR_O() y SEL_O()
  - CYC_O y WE_O
  - ACK_I, ERR_I, RTY_I
  - LOCK_O
  - STB_O
  - TGn_O()



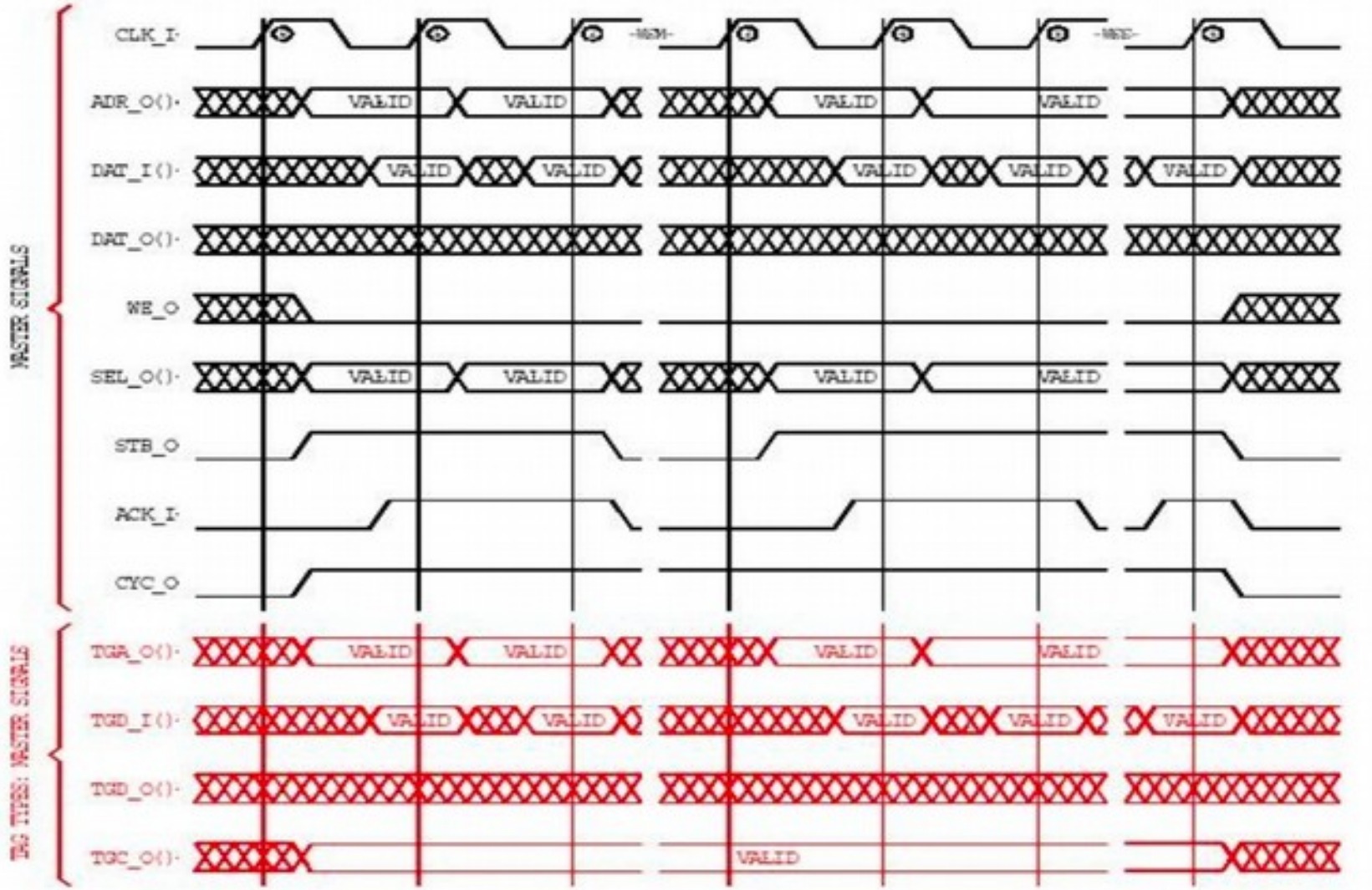
Wishbone | ciclos

- Reset
- READ/WRITE simple
- RMW
- READ/WRITE en bloque

Wishbone | Read/Write simple



Wishbone | Read en bloque



Wishbone Clásico

- Reconocimiento de ciclos como bucle combinatorio

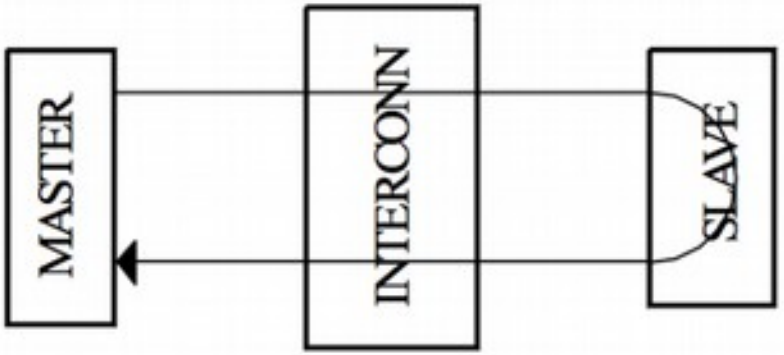


Figure 4-1 Asynchronous cycle termination path

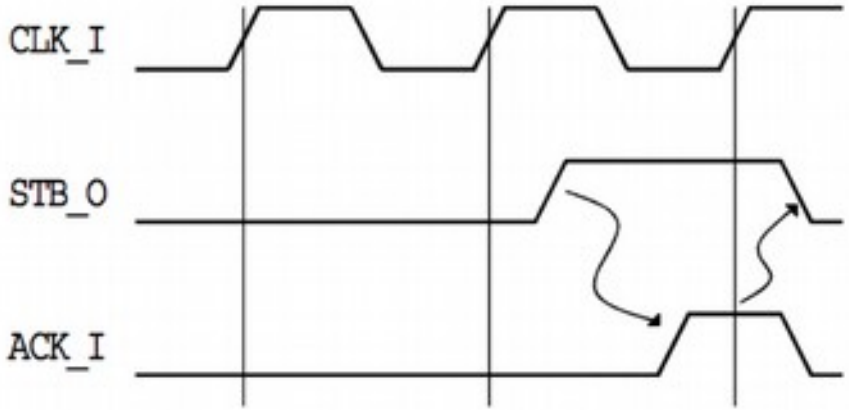
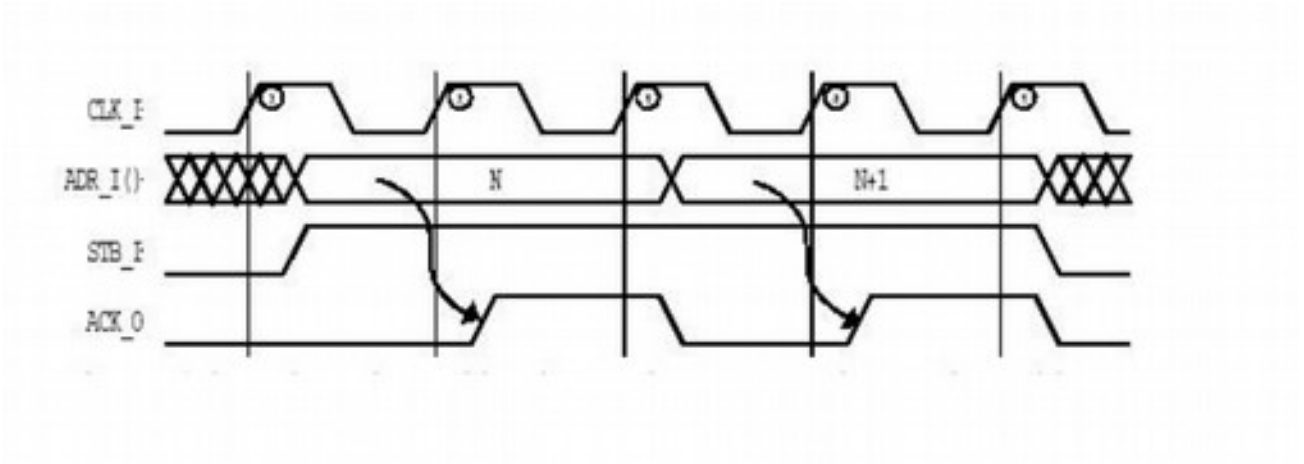


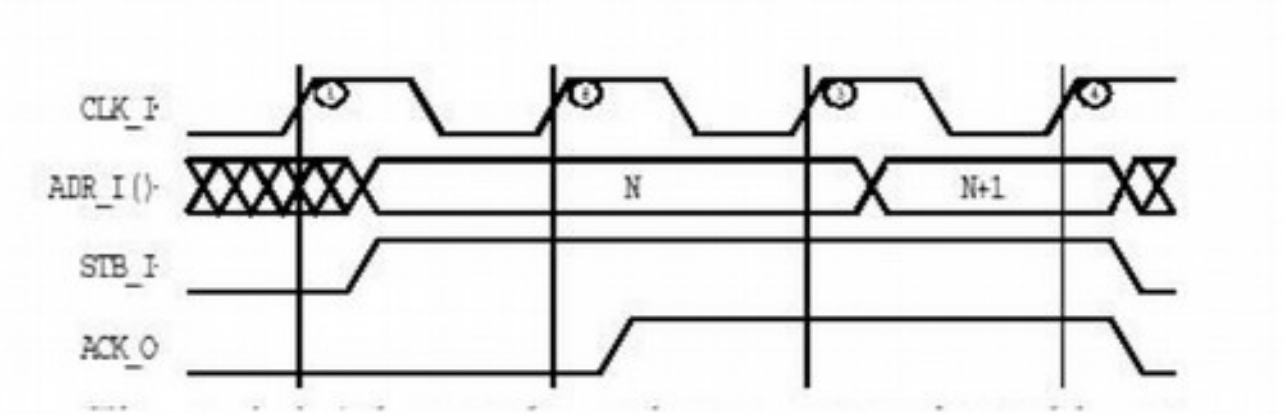
Figure 3-2. Local bus handshaking protocol.

Wishbone | Clásico y con ciclos reg.

- Cortar el loop para aumentar performance



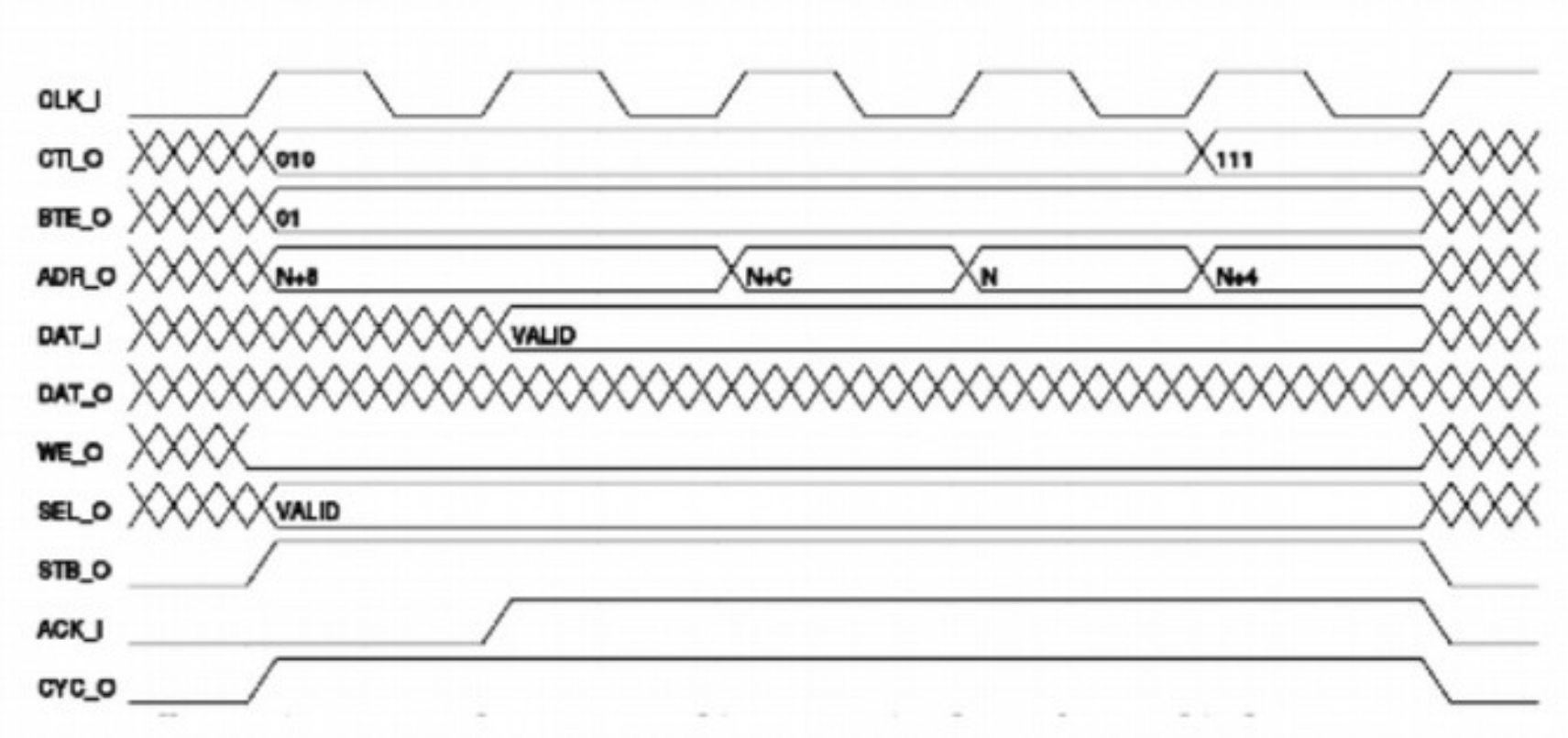
- Reconocimiento de ciclos registrados



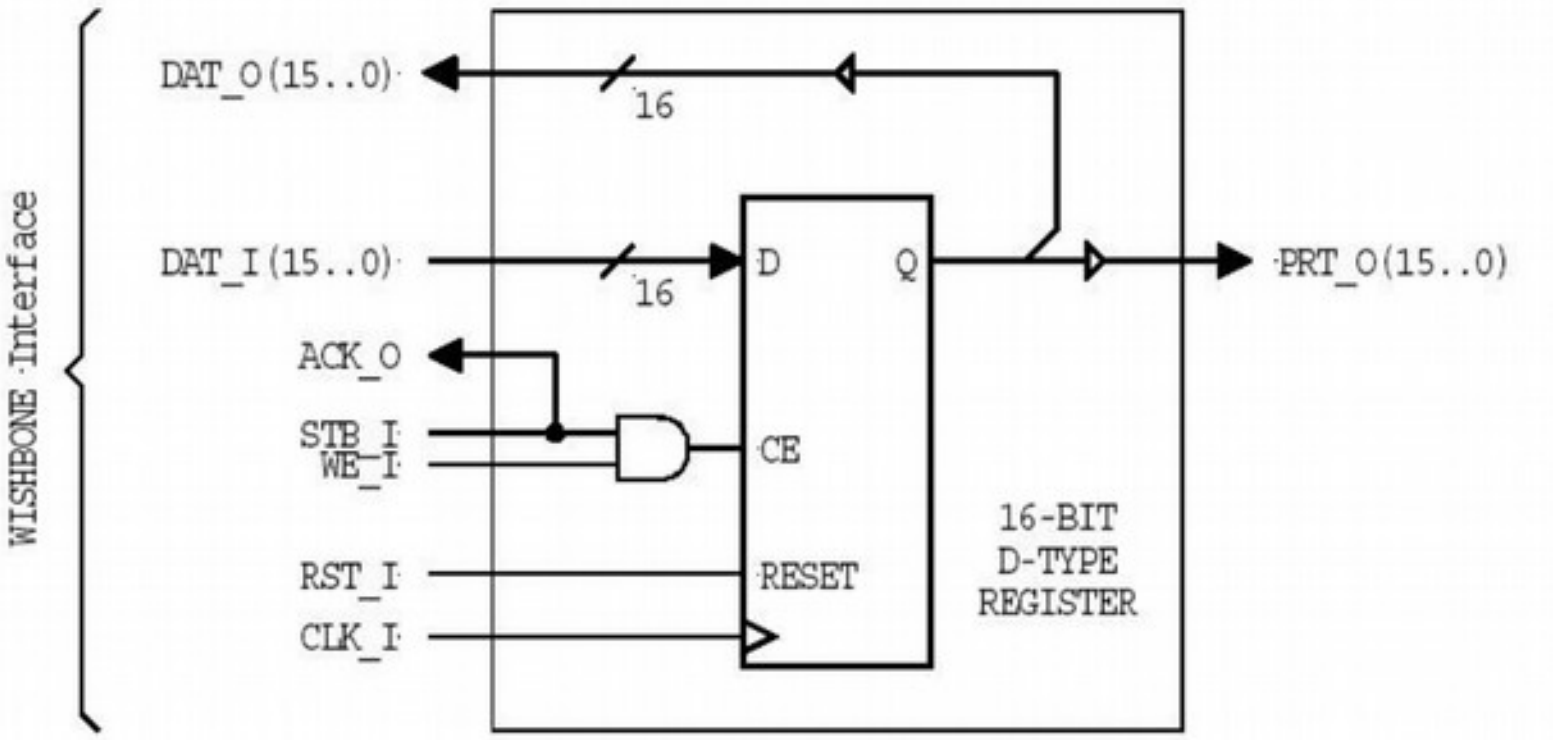
Wishbone | Tags

Table 4-2 Cycle Type Identifiers	
CTI_O(2:0)	Description
'000'	Classic cycle.
'001'	Constant address burst cycle
'010'	Incrementing burst cycle
'011'	*Reserved*
'100'	*Reserved*
'101	*Reserved*
'110'	*Reserved*
'111'	End-of-Burst

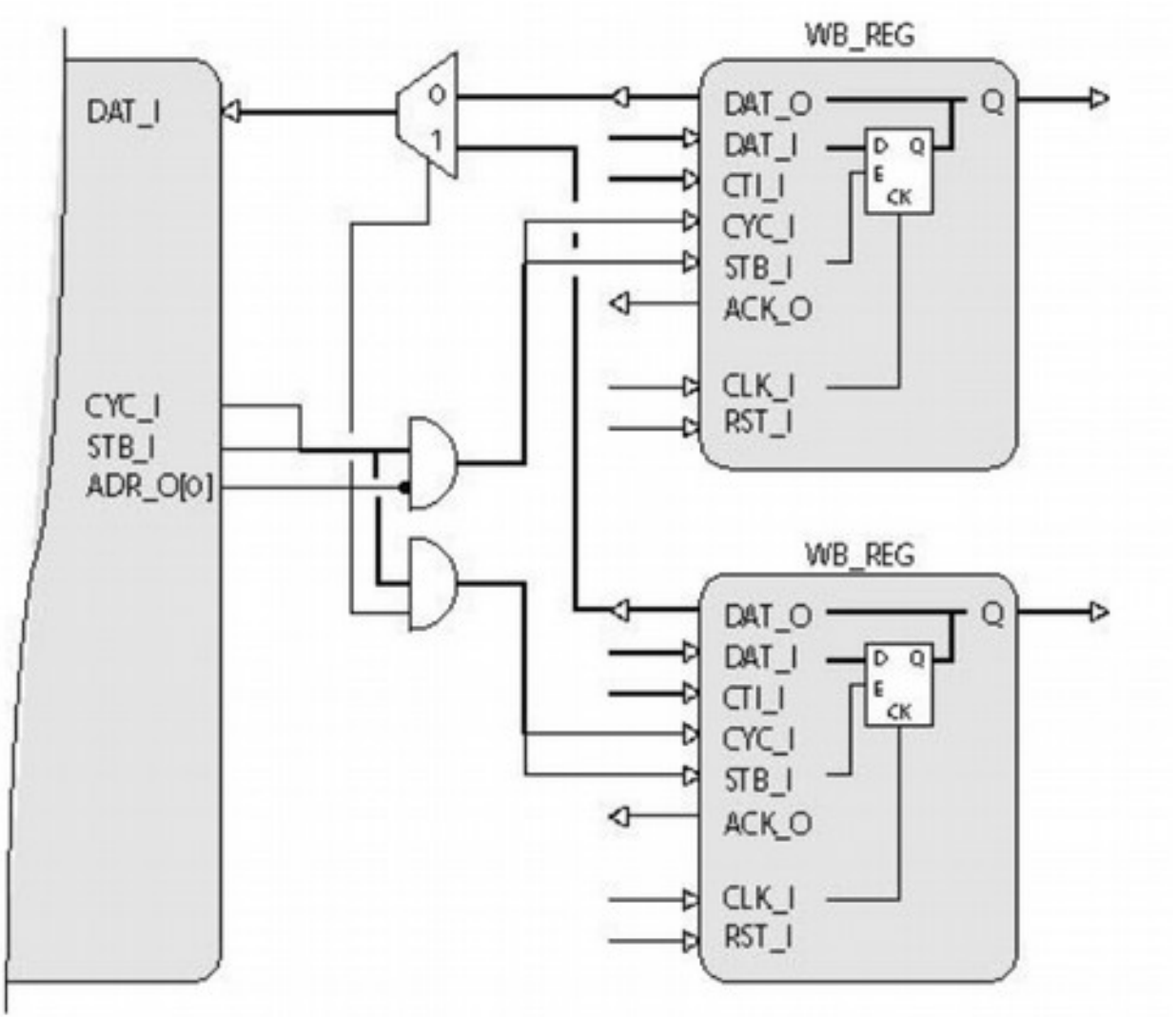
Wishbone | End of cycle



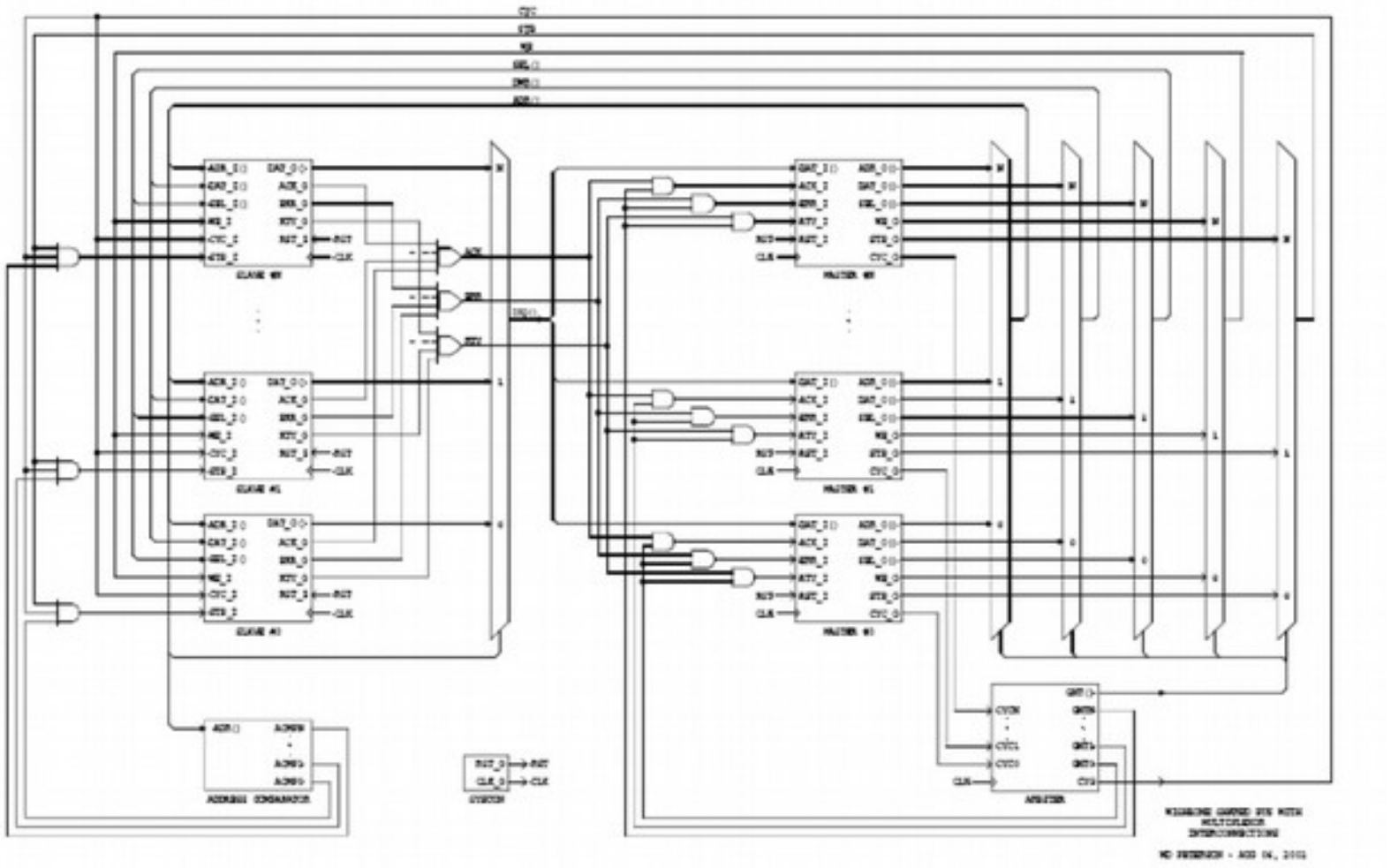
Wishbone | Ejemplo



Wishbone | Multiples esclavos



Wishbone | Bus compartido



Wishbone | Registro ciclo registrado

Modificar registro

https://eva.fing.edu.uy/pluginfile.php/68821/mod_resource/content/2/wb_reg_classic.vhd

Ref: cap.4 de estándar Wishbone