UiO : Department of Informatics
University of Oslo

System on Chip (SoC)

Agenda

- System on Chip (SoC) Introduction
- Xilinx Zynq 7000 SoC and Zynq Ultrascale+ MPSoC
- AXI4 Interface
- AXI4-Lite to internal shared bus bridge
- Serial Peripheral Interconnect (SPI)

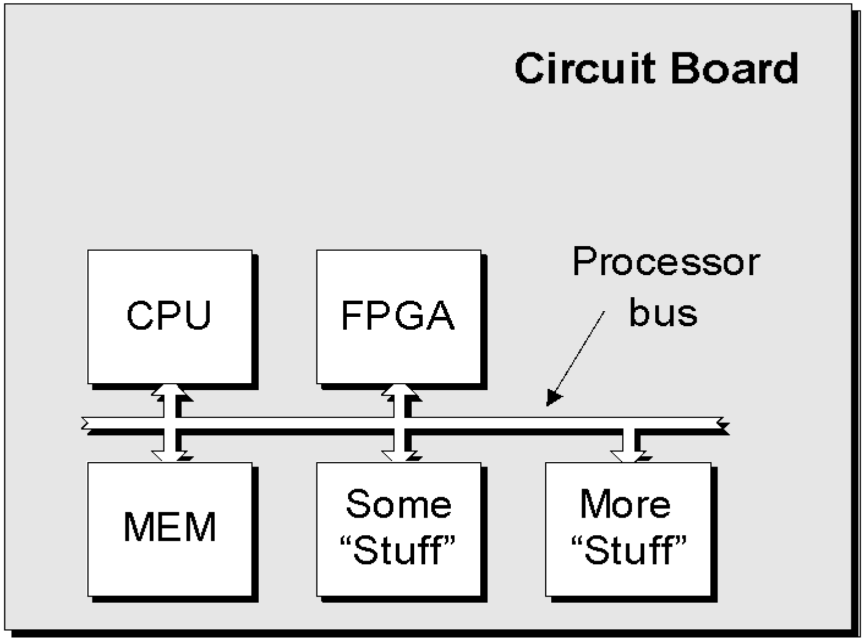
System on Chip (SoC)

Complete computer systems implemented in a single chip: CPU, memory controller and peripheral devices

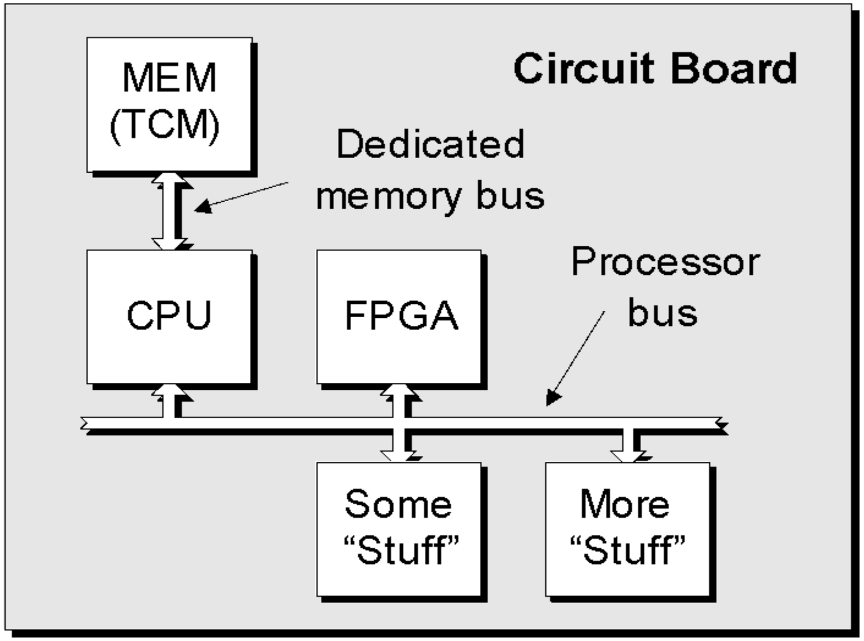
Memory is typically external

Can be digital or mixed signal (typical RF SoCs)

PCB Architecture



(a) Memory connected to CPU via general-purpose processor bus



(b) Tightly-coupled memory (TCM) connected to CPU via dedicated bus

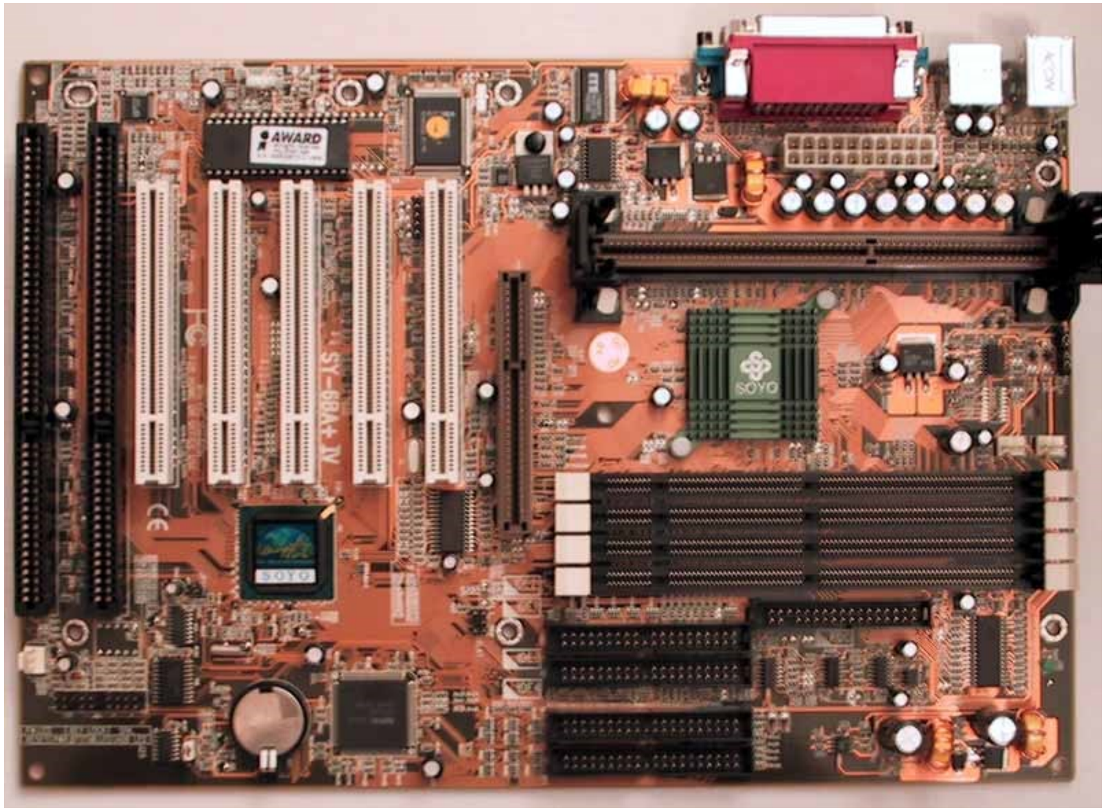
PCB Architecture

Ca 2000

External memory controller

Discrete ICs

Parallel busses



PCB Architecture

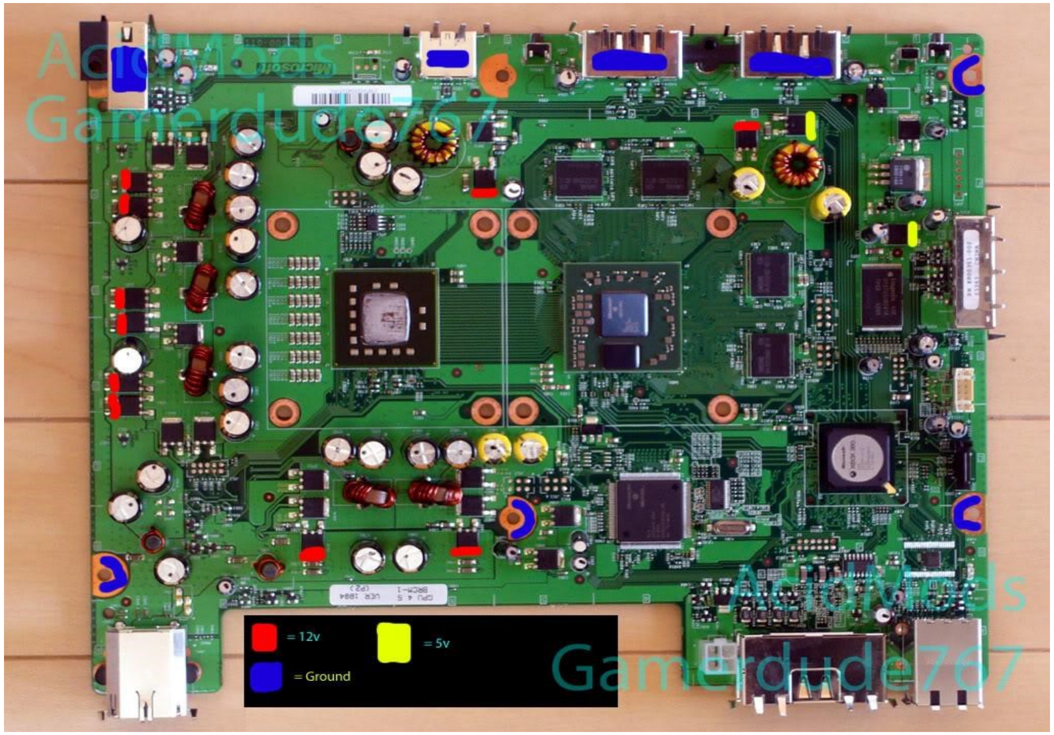
2005

Integrated memory controller

Fewer discrete ICs

High speed serial busses

Discrete GPU



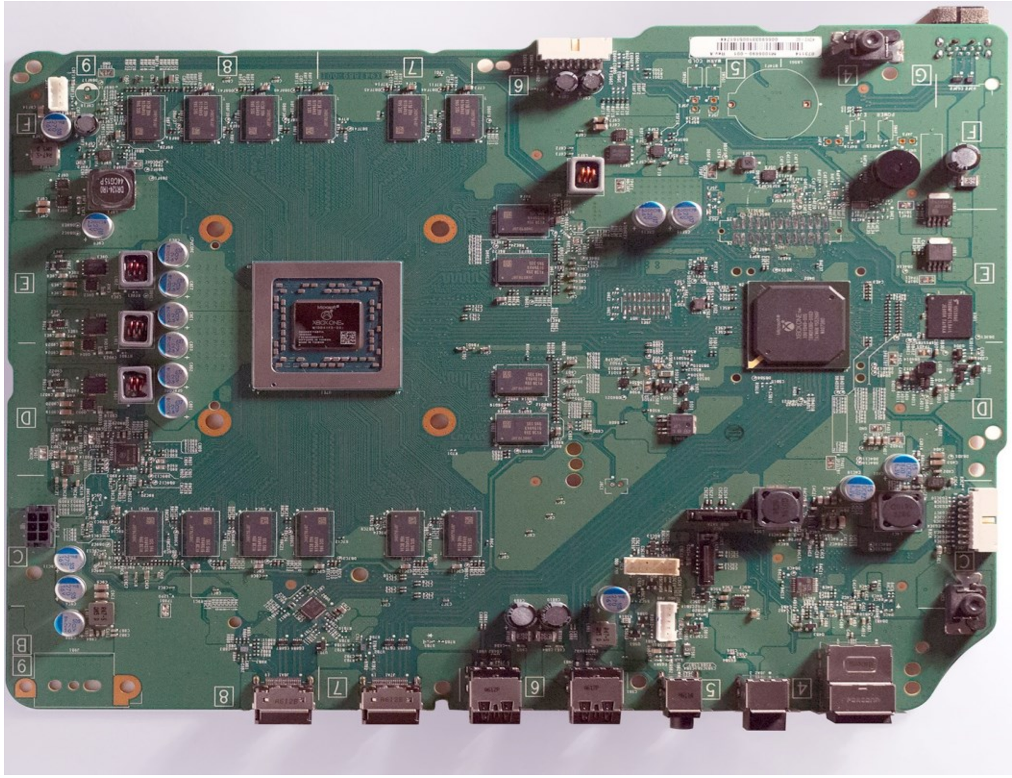
[redacted]

PCB Architecture

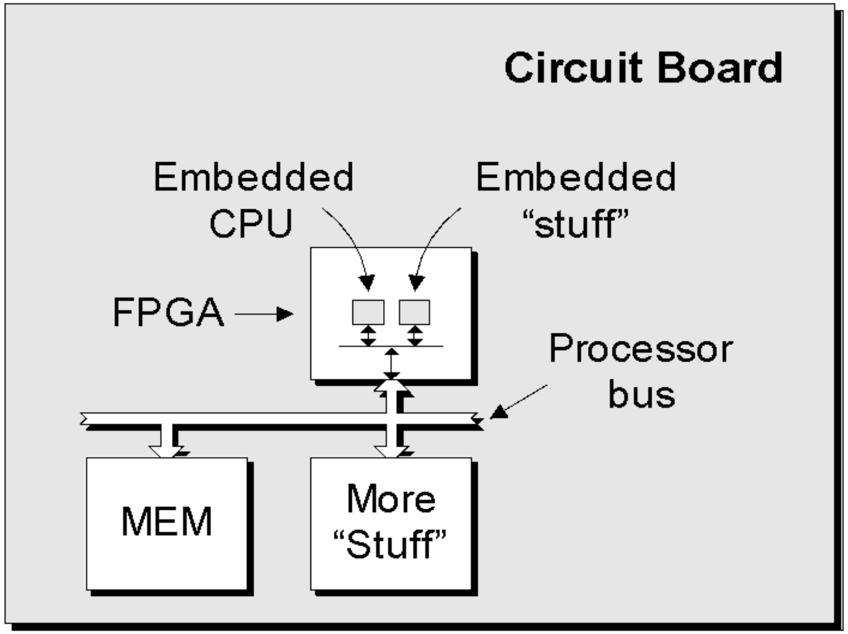
2013

Few discrete ICs

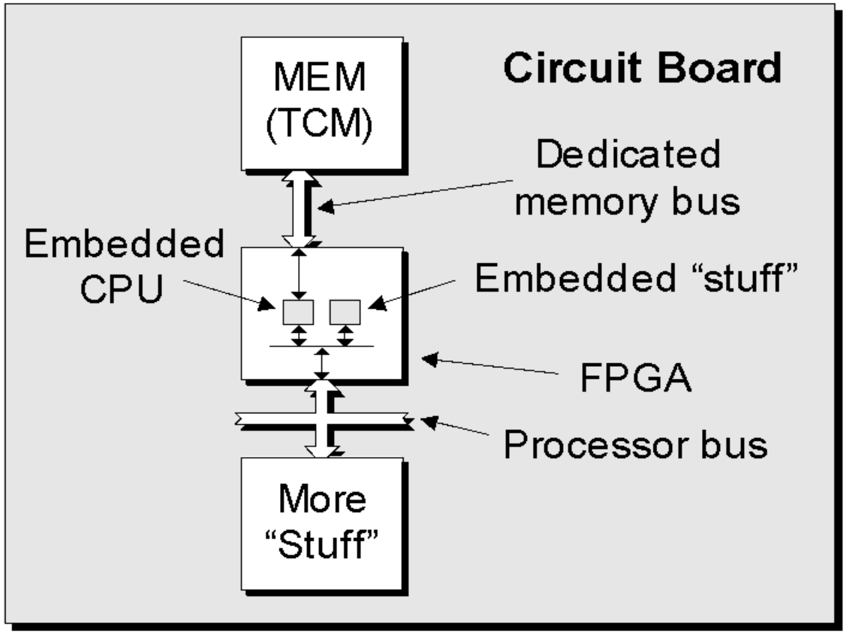
Integrated CPU/GPU



FPGA Architecture



(a) Memory connected to CPU via general-purpose processor bus



(b) Tightly-coupled memory (TCM) connected to CPU via dedicated bus

System on Chip benefits

Designs with SoC have:

- Fewer discrete ICs (simpler logistics)
- Higher performance internal busses
- Lower power
- Simpler PCB design

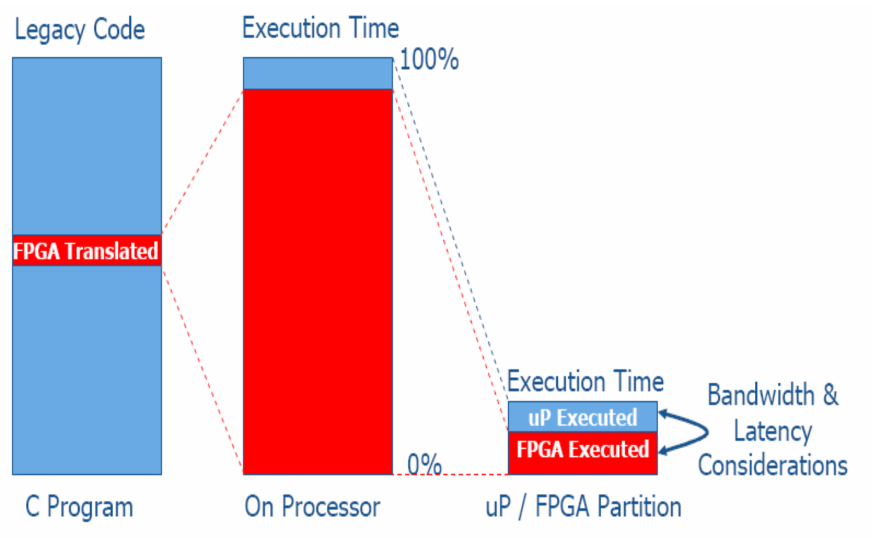
--> Reduced time to market

[redacted]

SoC with FPGA, software/hardware codesign

Profile applications to detect hotspots

Implement the hotspot as an IP module (VHDL or C with HLS tool from Xilinx)



Reduced latency and increased bandwidth compared to a two-chip solution (i.e. FPGA+SoC).

[redacted]

SoC with FPGA from Xilinx

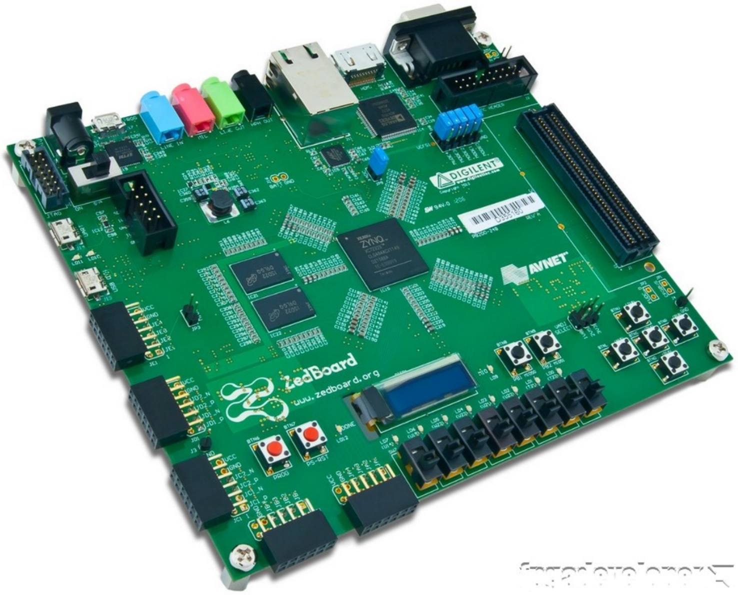
Xilinx Zynq 7000 announced in 2011; a SoC with a dual core CPU

Many peripherals implemented; Memory controller, USB, Ethernet

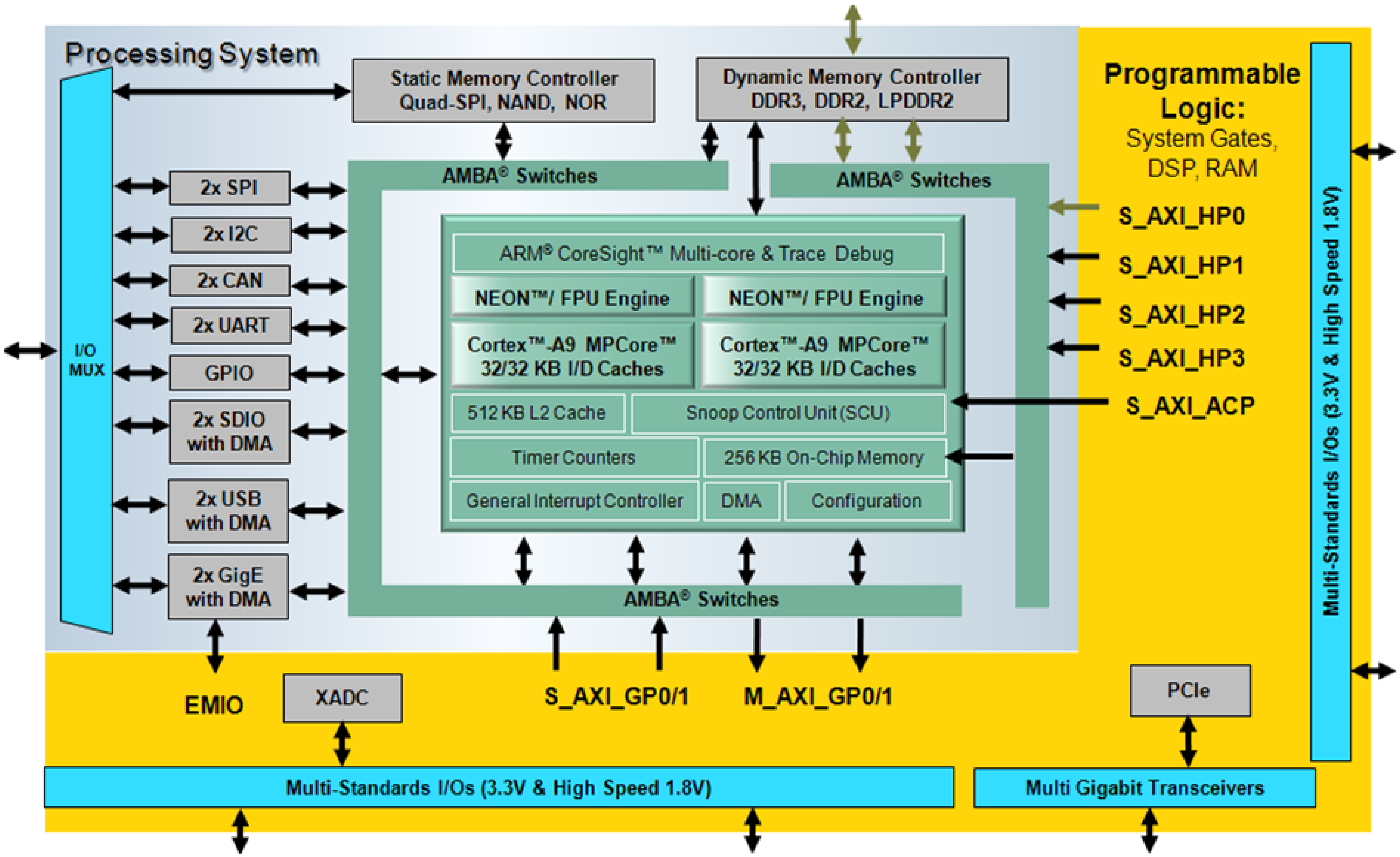
Improved software (Vivado, SDK / VITIS) Allows us to add custom modules (IP cores).

The custom IP modules can be developed by ourselves or bought.

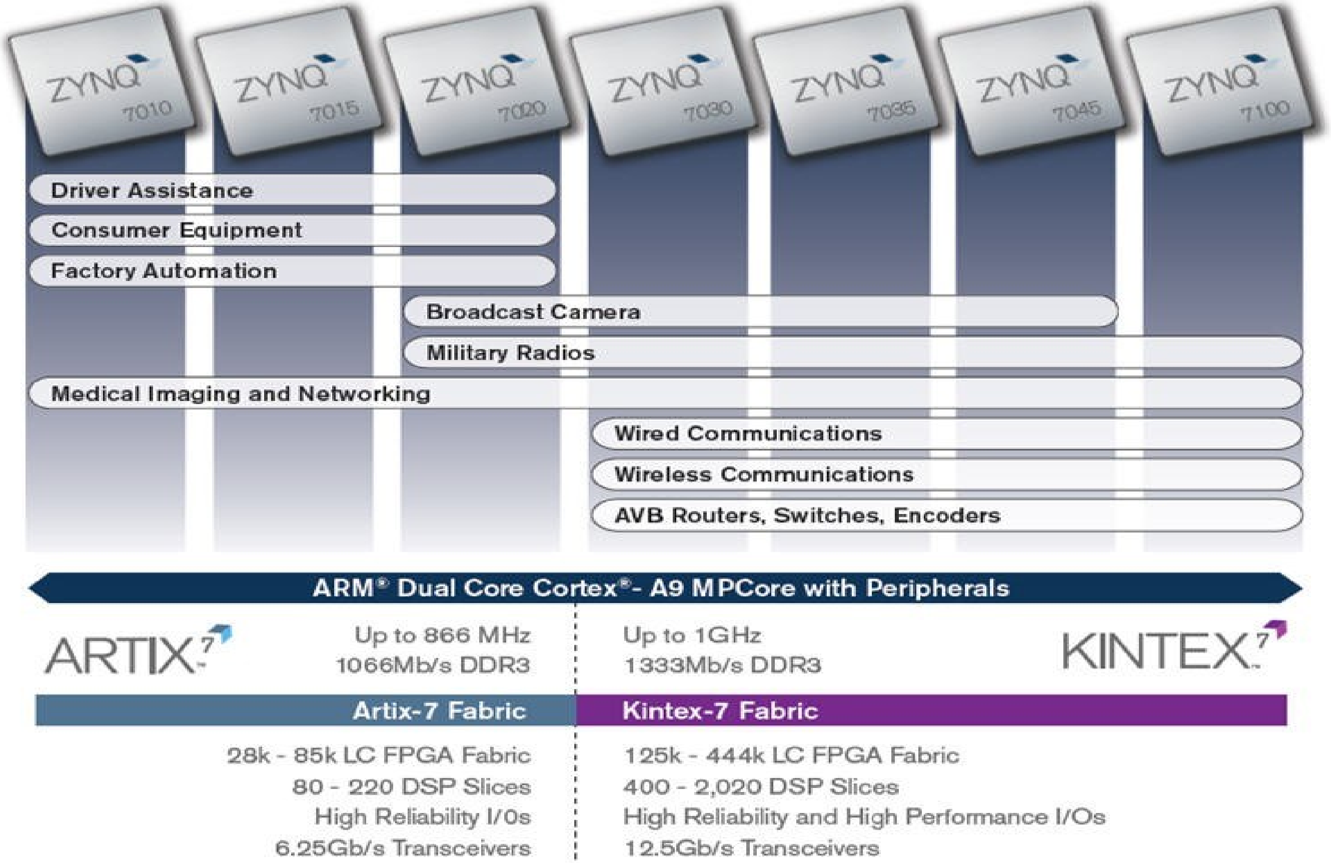
Connect the custom IP modules to the internal SoC bus interconnect.



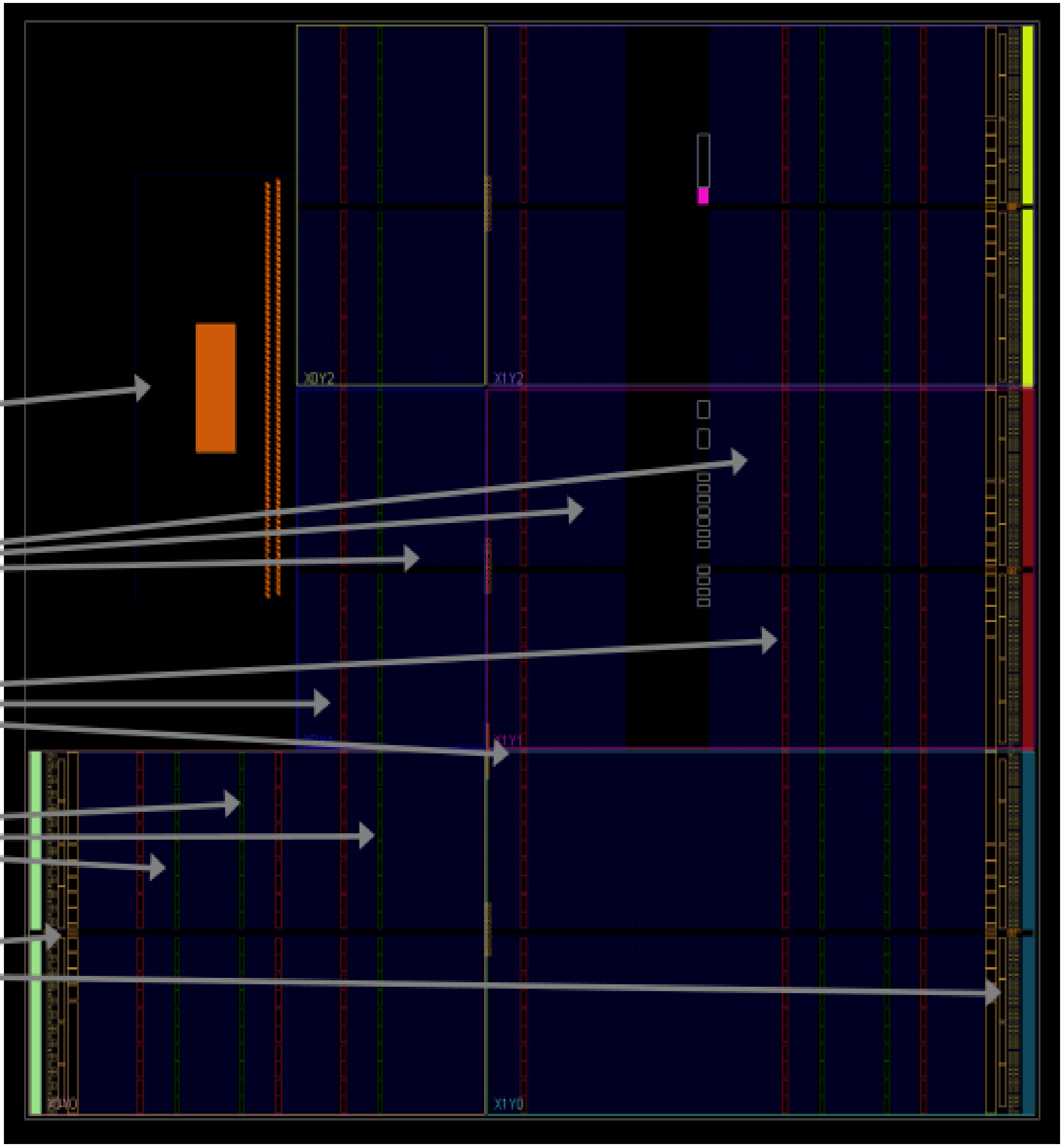
Zynq 7000 SoC



Zedboard uses Zynq 7020



[redacted]



ARM Cortex-A9

Logic slices

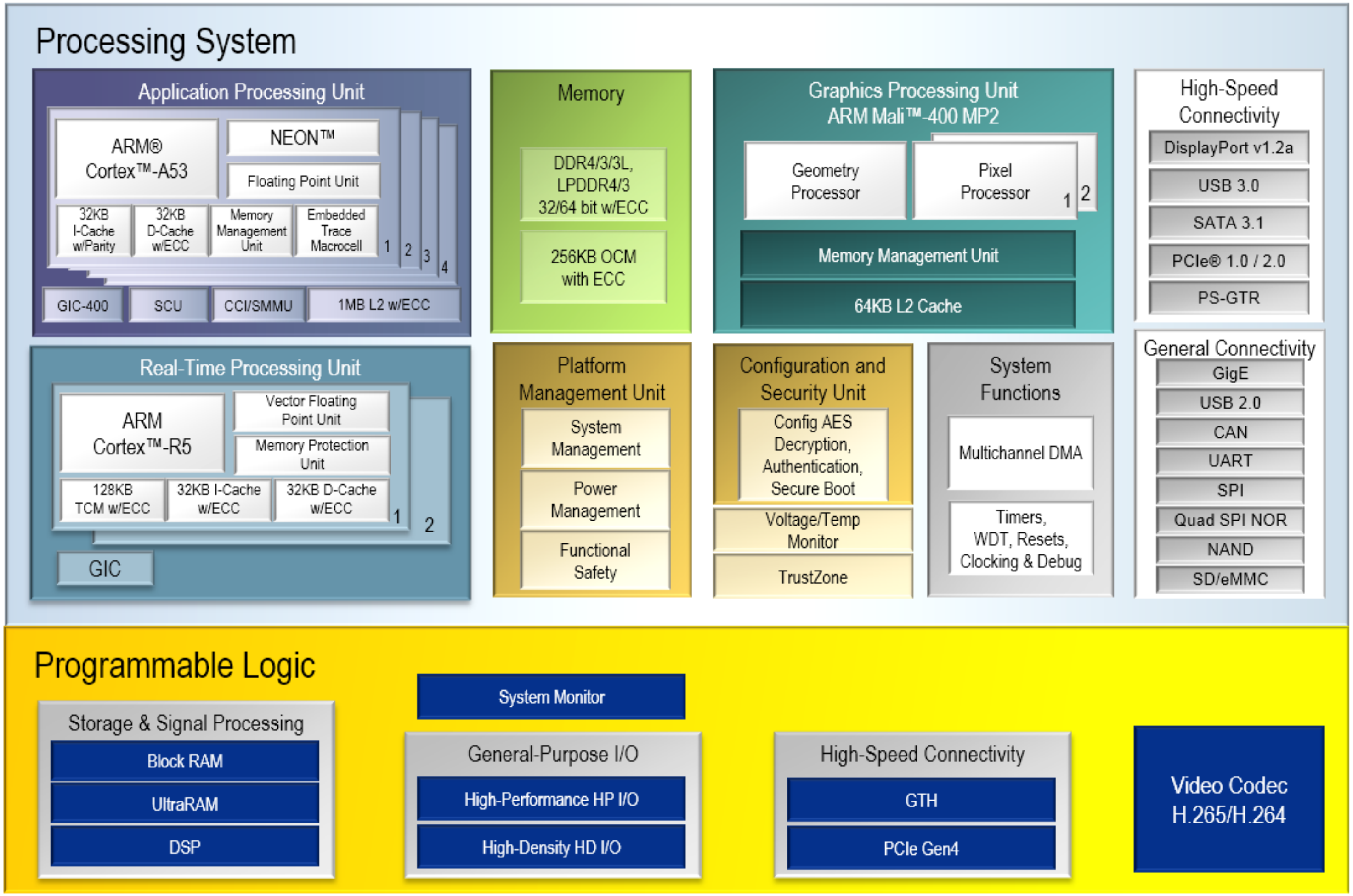
BRAM

DSP

IO

[redacted]

Zynq Ultrascale+ Multiprocessor SoC (MPSoC)



Processing System

Application Processing Unit

ARM® Cortex™-A53

NEON™

Floating Point Unit

32KB I-Cache w/Parity

32KB D-Cache w/ECC

Memory Management Unit

Embedded Trace Macrocell

1 2 3 4

GIC-400

SCU

CCI/SMMU

1MB L2 w/ECC

Memory

DDR4/3/3L, LPDDR4/3 32/64 bit w/ECC

256KB OCM with ECC

Graphics Processing Unit ARM Mali™-400 MP2

Geometry Processor

Pixel Processor

1 2

Memory Management Unit

64KB L2 Cache

High-Speed Connectivity

DisplayPort v1.2a

USB 3.0

SATA 3.1

PCIe® 1.0 / 2.0

PS-GTR

Real-Time Processing Unit

ARM Cortex™-R5

Vector Floating Point Unit

Memory Protection Unit

128KB TCM w/ECC

32KB I-Cache w/ECC

32KB D-Cache w/ECC

1 2

GIC

Platform Management Unit

System Management

Power Management

Functional Safety

Configuration and Security Unit

Config AES Decryption, Authentication, Secure Boot

Voltage/Temp Monitor

TrustZone

System Functions

Multichannel DMA

Timers, WDT, Resets, Clocking & Debug

General Connectivity

GigE

USB 2.0

CAN

UART

SPI

Quad SPI NOR

NAND

SD/eMMC

Programmable Logic

Storage & Signal Processing

Block RAM

UltraRAM

DSP

System Monitor

General-Purpose I/O

High-Performance HP I/O

High-Density HD I/O

High-Speed Connectivity

GTH

PCIe Gen4

Video Codec H.265/H.264

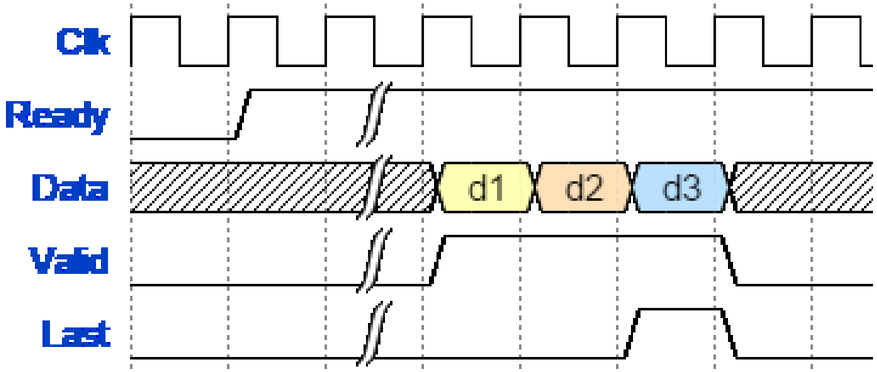
AXI4 Interface

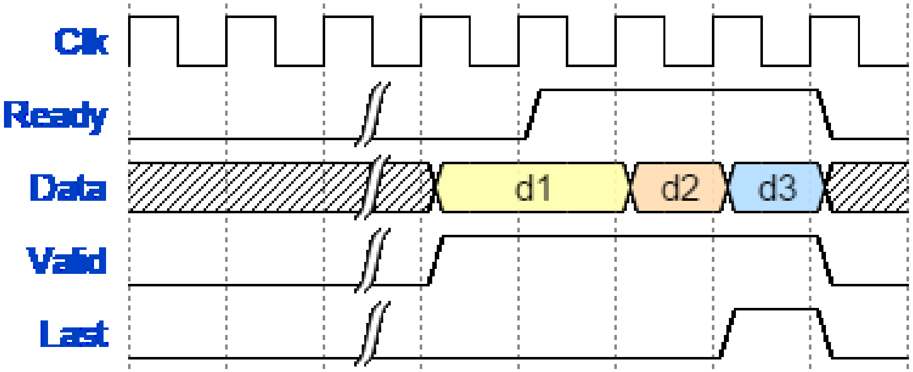
- Defined in the AMBA3 specification from ARM
- Targeted at high performance, high clock frequency systems
- Not a bus, but a point-to-point interface

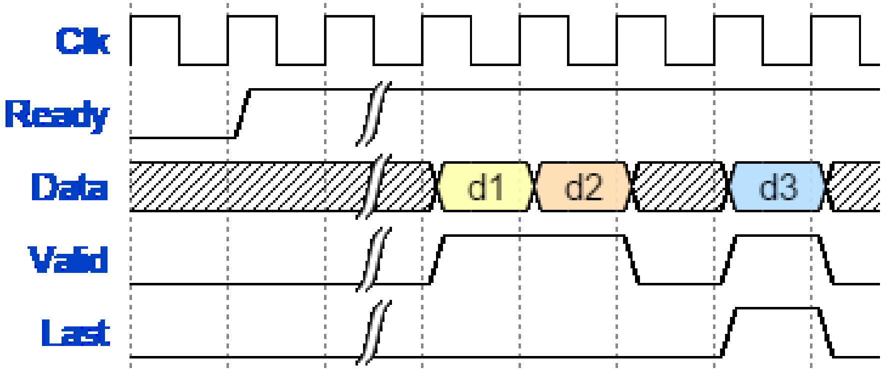
Types

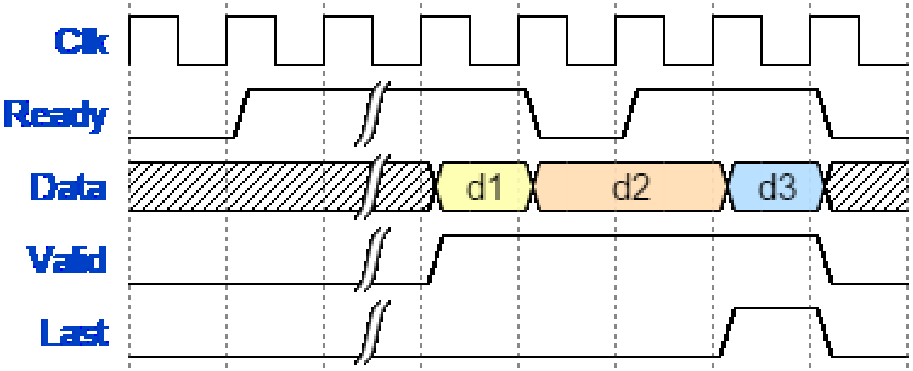
- **AXI4-stream**
  - Supports single or multiple streams on same wires
  - Supports multiple data widths within same interconnect
  - Only contains a data channel:
    - Typically *Ready*, *Valid*, *Data*, *Last* (+ *clk*)
- AXI4-lite
  - Transaction length of one
  - 32/64 bit data access widths
- AXI4
  - Supports burst length up to 256 beats

AXI4-stream examples





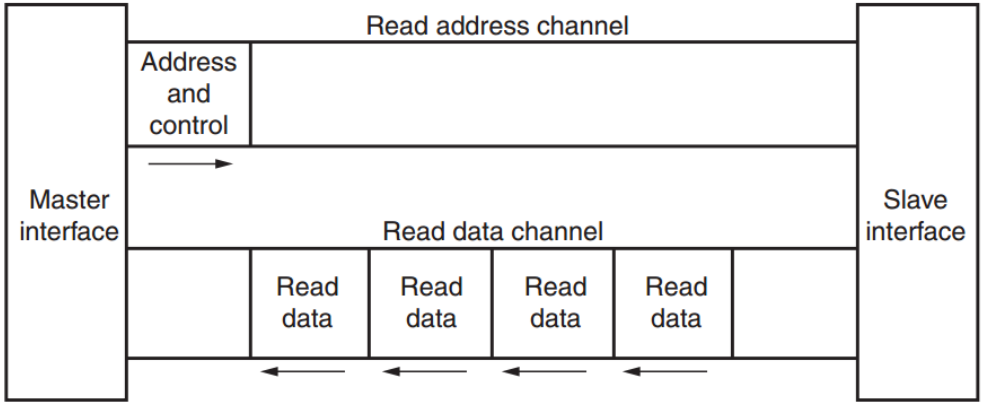




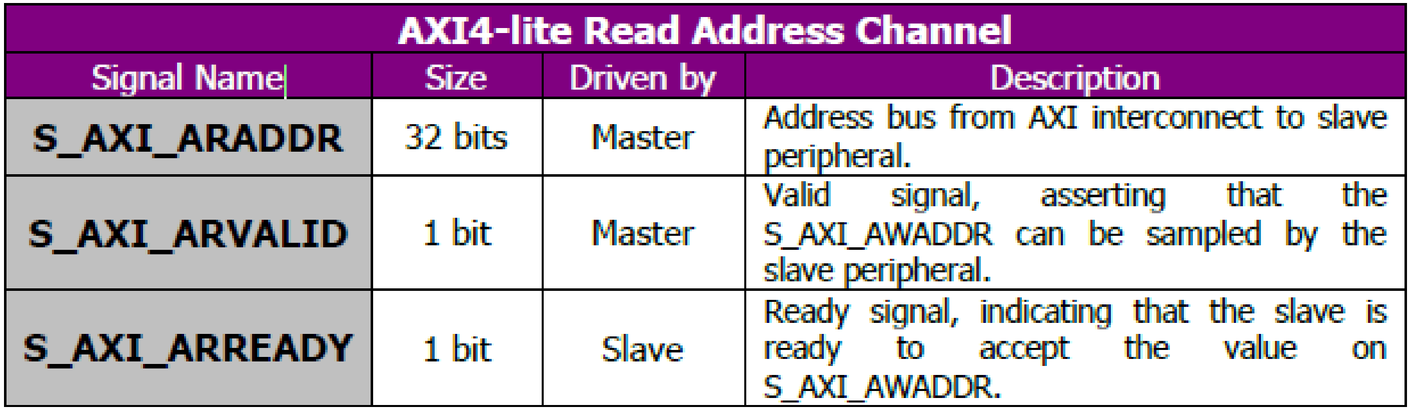
Types

- AXI4-stream
  - Supports single or multiple streams on same wires
  - Supports multiple data widths within same interconnect
  - Only contains a data channel:
    - Typically *Ready*, *Valid*, *Data*, *Last* (+ *clk*)
- **AXI4-lite**
  - Transaction length of one
  - 32/64 bit data access widths
- AXI4
  - Supports burst length up to 256 beats

AXI4-lite Read Address Channel



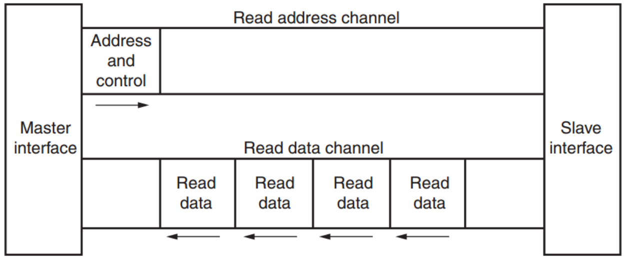
- Two independent channels: Read and Write
- Response is always generated



AXI4-lite Read Address Channel			
Signal Name	Size	Driven by	Description
S_AXI_ARADDR	32 bits	Master	Address bus from AXI interconnect to slave peripheral.
S_AXI_ARVALID	1 bit	Master	Valid signal, asserting that the S_AXI_AWADDR can be sampled by the slave peripheral.
S_AXI_ARREADY	1 bit	Slave	Ready signal, indicating that the slave is ready to accept the value on S_AXI_AWADDR.

[redacted]

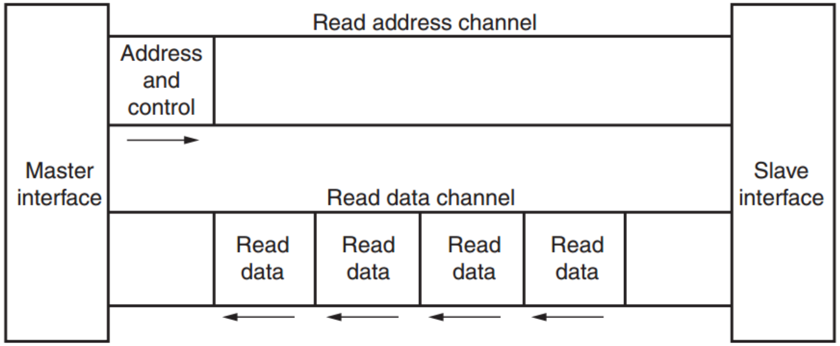
AXI4-lite Read Data Channel

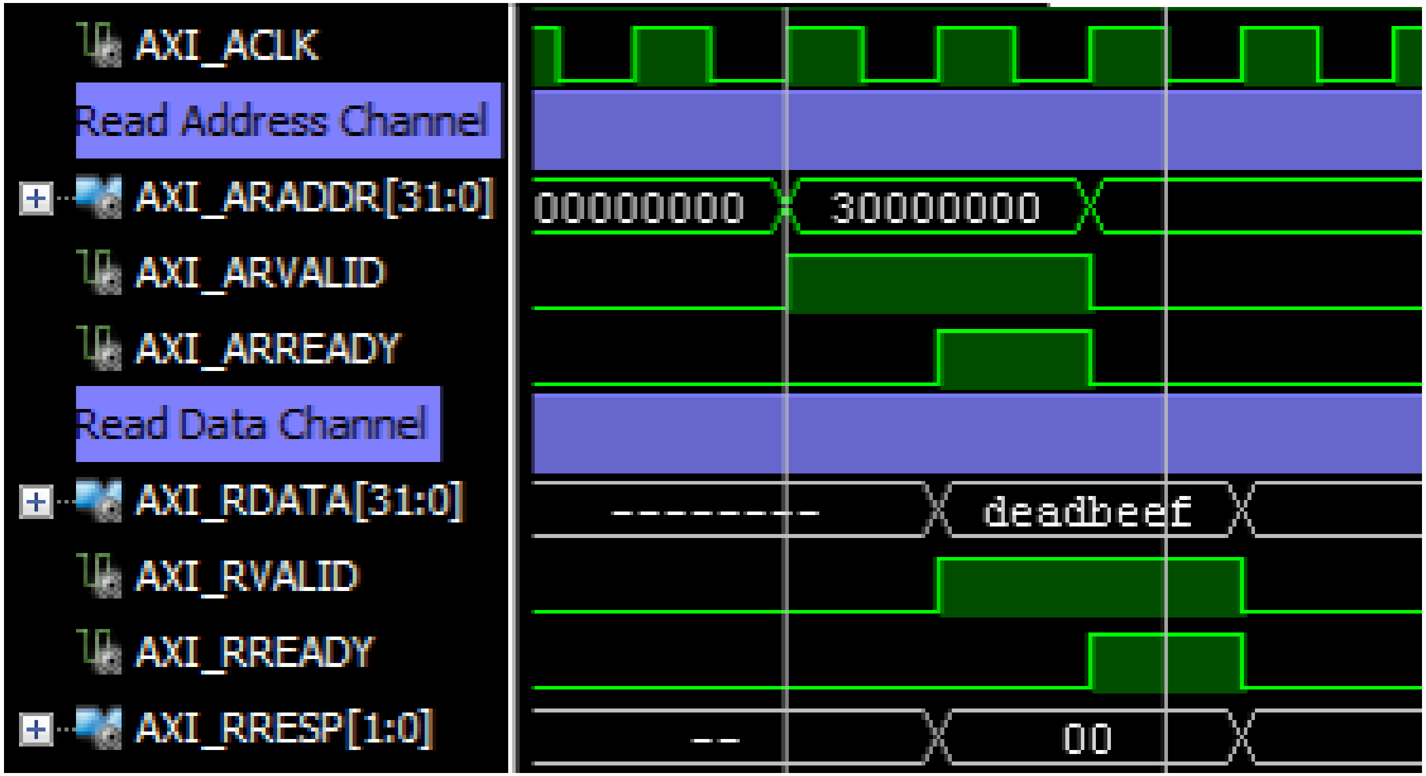


AXI4-lite Read Data Channel			
Signal Name	**Size**	**Driven by**	**Description**
S_AXI_RDATA	32 bits	Slave	Data bus from the slave peripheral to the AXI interconnect.
S_AXI_RVALID	1 bit	Slave	Valid signal, asserting that the S_AXI_RDATA can be sampled by the Master.
S_AXI_RREADY	1 bit	Master	Ready signal, indicating that the Master is ready to accept the value on the other signals.
S_AXI_RRESP	2 bits	Slave	A "Response" status signal showing whether the transaction completed successfully or whether there was an error.

AXI4-lite Response Signalling		
RRESP State [1:0]	**Condition**	**Description**
00	OKAY	**"OKAY"** The data was received successfully, and there were no errors.
01	EXOKAY	**"Exclusive Access OK"** This state is only used in the full implementation of AXI4, and therefore cannot occur when using AXI4-Lite.
10	SLVERR	**"Slave Error"** The slave has received the address phase of the transaction correctly, but needs to signal an error condition to the master. This often results in a retry condition occurring.
11	DECERR	**"Decode Error"** This condition is not normally asserted by a peripheral, but can be asserted by the AXI interconnect logic which sits between the slave and the master. This condition is usually used to indicate that the address provided doesn't exist in the address space of the AXI interconnect.

AXI4-lite Read Example

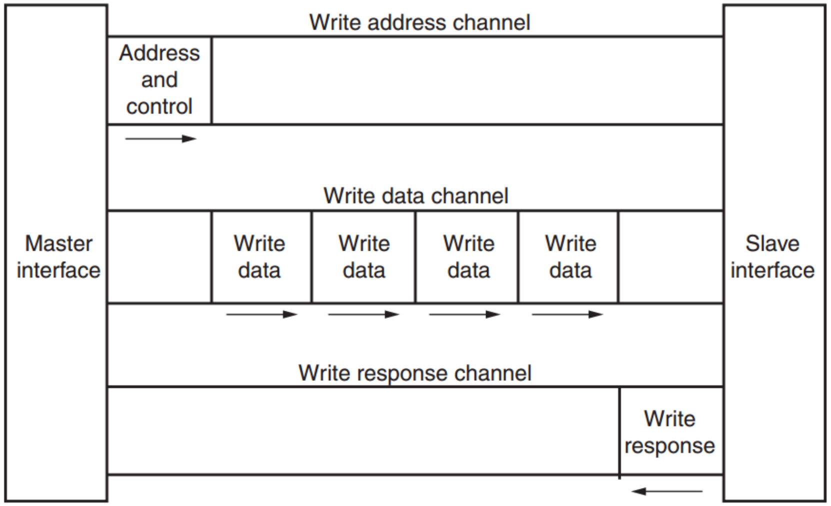


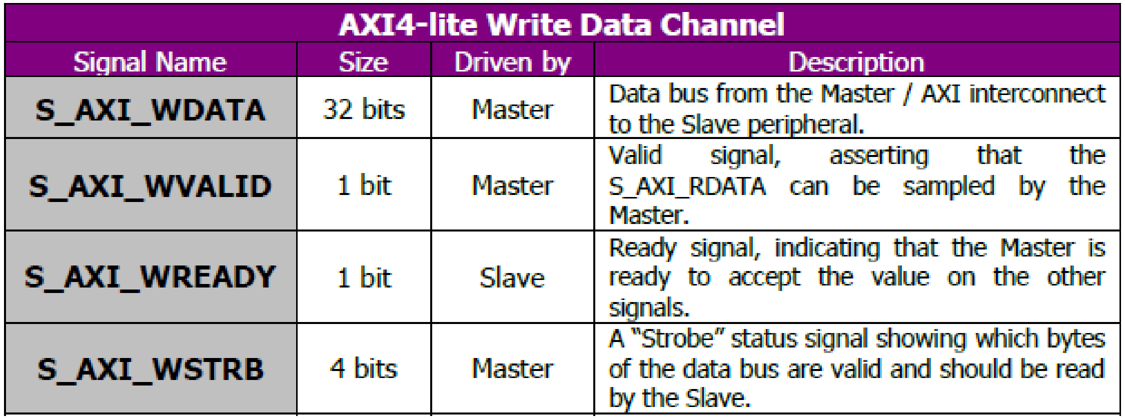


[redacted]

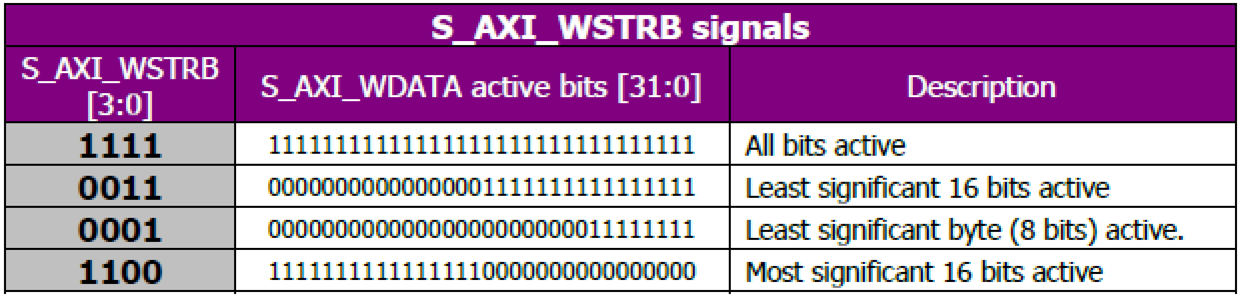
AXI4-lite Write Data Channel

(Address channel similar to Read; see Write example)



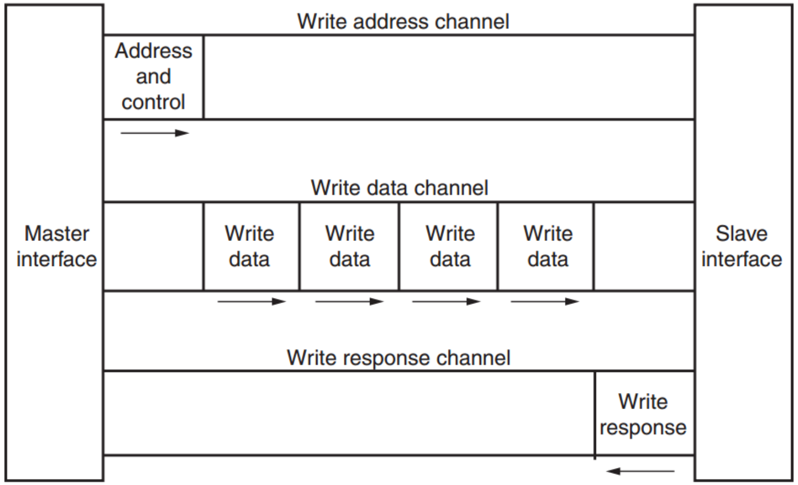


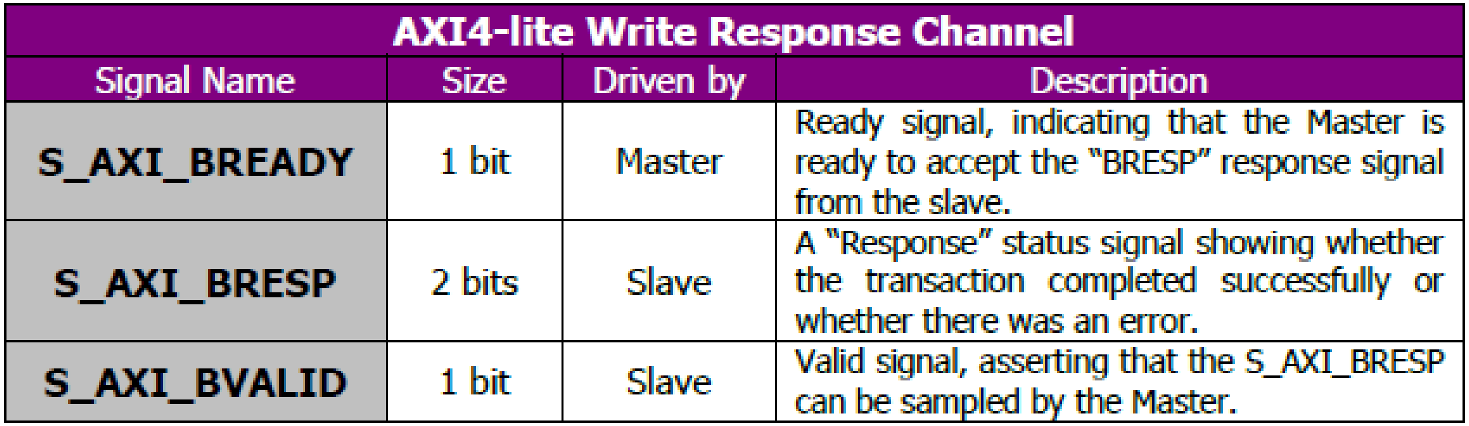
AXI4-lite Write Data Channel			
Signal Name	**Size**	**Driven by**	**Description**
S_AXI_WDATA	32 bits	Master	Data bus from the Master / AXI interconnect to the Slave peripheral.
S_AXI_WVALID	1 bit	Master	Valid signal, asserting that the S_AXI_RDATA can be sampled by the Master.
S_AXI_WREADY	1 bit	Slave	Ready signal, indicating that the Master is ready to accept the value on the other signals.
S_AXI_WSTRB	4 bits	Master	A "Strobe" status signal showing which bytes of the data bus are valid and should be read by the Slave.



S_AXI_WSTRB signals		
S_AXI_WSTRB [3:0]	**S_AXI_WDATA active bits [31:0]**	**Description**
1111	11111111111111111111111111111111	All bits active
0011	00000000000000001111111111111111	Least significant 16 bits active
0001	00000000000000000000000011111111	Least significant byte (8 bits) active.
1100	11111111111111110000000000000000	Most significant 16 bits active

AXI4-lite Write Response Channel

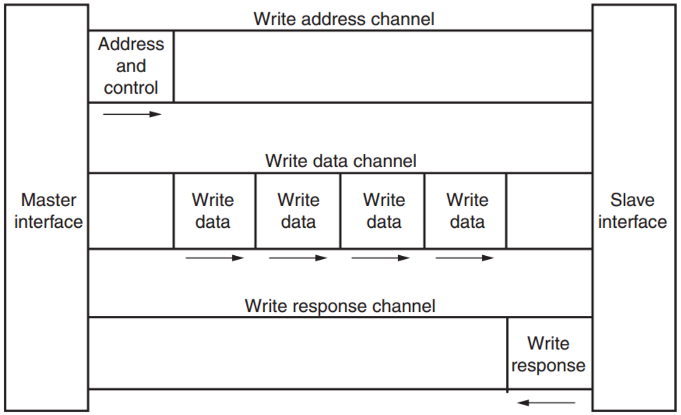


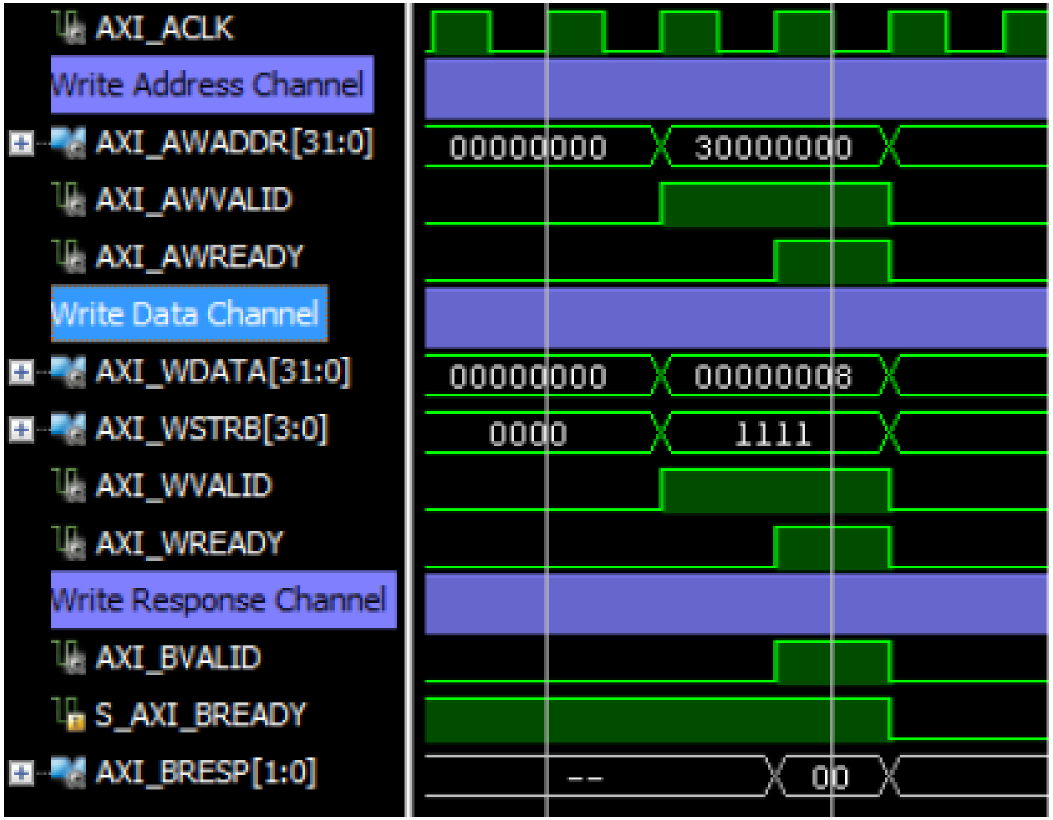


AXI4-lite Write Response Channel			
Signal Name	Size	Driven by	Description
S_AXI_BREADY	1 bit	Master	Ready signal, indicating that the Master is ready to accept the "BRESP" response signal from the slave.
S_AXI_BRESP	2 bits	Slave	A "Response" status signal showing whether the transaction completed successfully or whether there was an error.
S_AXI_BVALID	1 bit	Slave	Valid signal, asserting that the S_AXI_BRESP can be sampled by the Master.

[redacted]

AXI4-lite Write Example





[redacted]

AXI4-lite use of Valid and Ready signals

A frequently misunderstood use of the Valid and Ready signals, and one which often results in incorrect and illegal implementations of the AXI4-lite protocol, is the assumption that the sender can/must wait for "Ready" to be asserted by the receiver before it asserts its "Valid" signal. This is an illegal use of the handshaking signals and can result in a deadlock situation arising. Ready can be asserted before Valid, but the sender must never wait for Ready as a pre-condition to commencing the transaction.

This important aspect of the AXI4-lite protocol can be easily remembered by applying the "Assert and wait" rule. Never use the "Wait before Assert" approach, because this is illegal.

"Assert Ready and wait for Valid" ✓

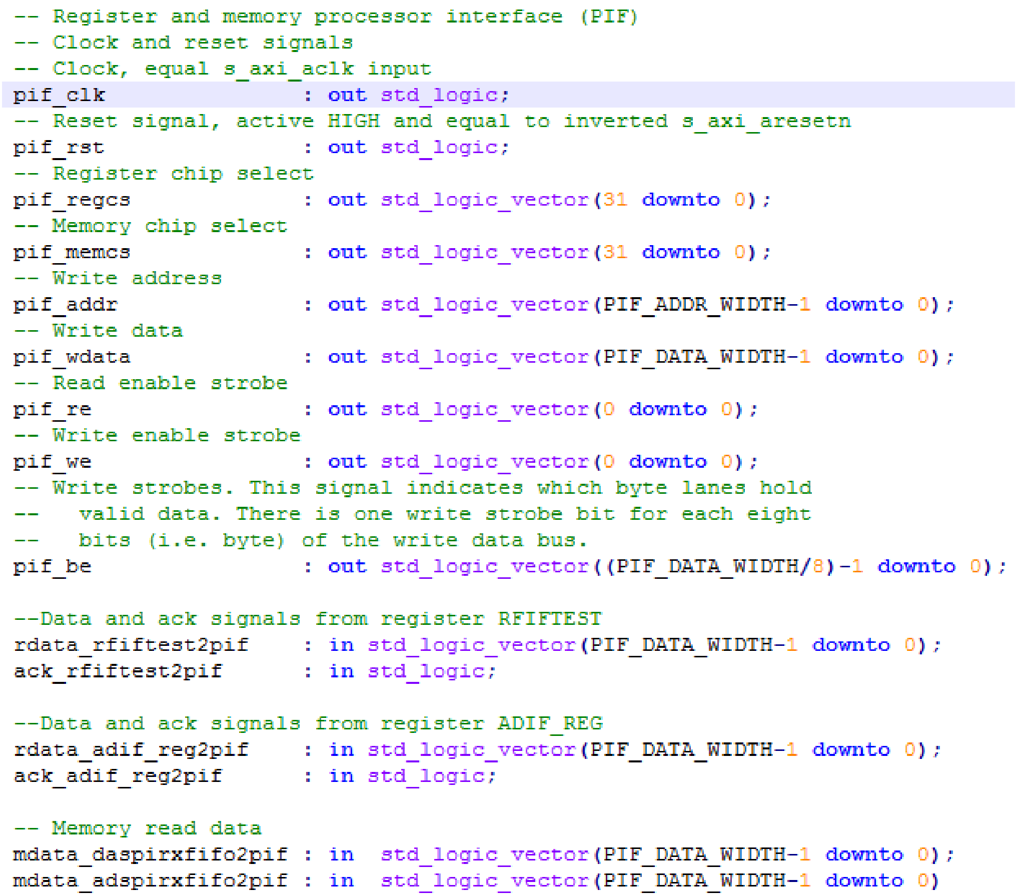
"Assert Valid and wait for Ready" ✓

"Wait for Ready before asserting Valid" ✗

AXI4

- Like AXI4-lite, but with additional features
  - Bursts of up to 256 beats
  - Exclusive access (https://blogs.synopsys.com/vip-central/2016/08/24/amba-axi-exclusive-access-de-mystified)
  - Memory management / coherency
  - Quality of service (https://community.arm.com/soc/b/blog/posts/quality-of-service-in-arm-systems-an-overview)
- Can be translated into AXI4-lite by AXI interconnects modules

AXI4Lite to internal shared bus bridge;
Processor InterFace (PIF) shared bus signals

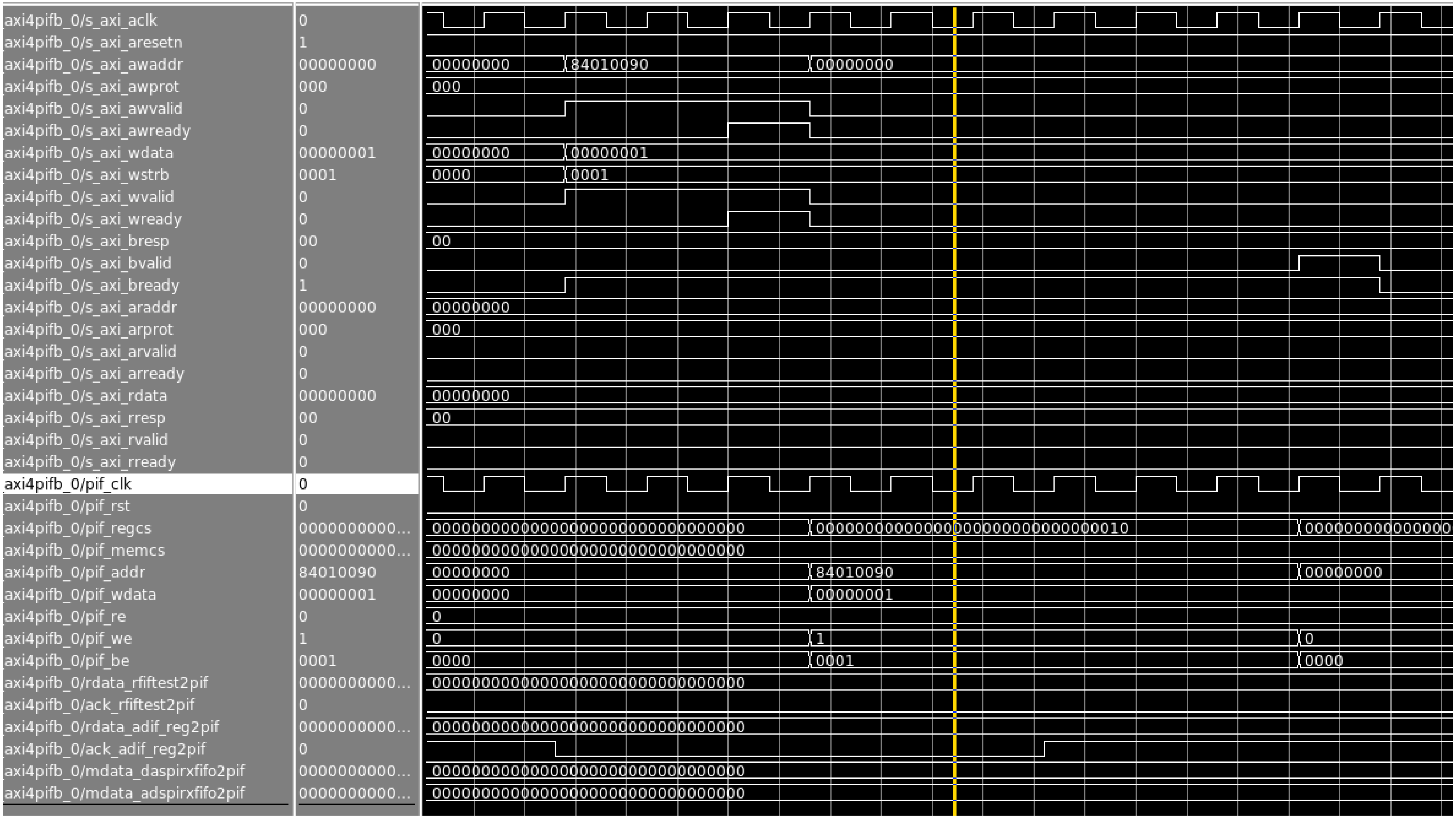


```
-- Register and memory processor interface (PIF)
-- Clock and reset signals
-- Clock, equal s_axi_aclk input
pif_clk               : out std_logic;
-- Reset signal, active HIGH and equal to inverted s_axi_aresetn
pif_rst               : out std_logic;
-- Register chip select
pif_regcs             : out std_logic_vector(31 downto 0);
-- Memory chip select
pif_memcs             : out std_logic_vector(31 downto 0);
-- Write address
pif_addr              : out std_logic_vector(PIF_ADDR_WIDTH-1 downto 0);
-- Write data
pif_wdata             : out std_logic_vector(PIF_DATA_WIDTH-1 downto 0);
-- Read enable strobe
pif_re                : out std_logic_vector(0 downto 0);
-- Write enable strobe
pif_we                : out std_logic_vector(0 downto 0);
-- Write strobes. This signal indicates which byte lanes hold
--   valid data. There is one write strobe bit for each eight
--   bits (i.e. byte) of the write data bus.
pif_be                : out std_logic_vector((PIF_DATA_WIDTH/8)-1 downto 0);

--Data and ack signals from register RFIFTEST
rdata_rfiftest2pif    : in std_logic_vector(PIF_DATA_WIDTH-1 downto 0);
ack_rfiftest2pif      : in std_logic;

--Data and ack signals from register ADIF_REG
rdata_adif_reg2pif    : in std_logic_vector(PIF_DATA_WIDTH-1 downto 0);
ack_adif_reg2pif      : in std_logic;

-- Memory read data
mdata_daspirxfifo2pif : in  std_logic_vector(PIF_DATA_WIDTH-1 downto 0);
mdata_adspirxfifo2pif : in  std_logic_vector(PIF_DATA_WIDTH-1 downto 0)
```

- PIF is an in-house developed low performance shared bus.
- Register access with acknowlede signal (ack_*) to support different clock domains.
- Separate register select signals and memory select signals.
- Memory (RAM and FIFO) access without acknowledge signal due to known access time with same clock domain (i.e. processor clock) to 2-port BRAM/FIFO.

[redacted]

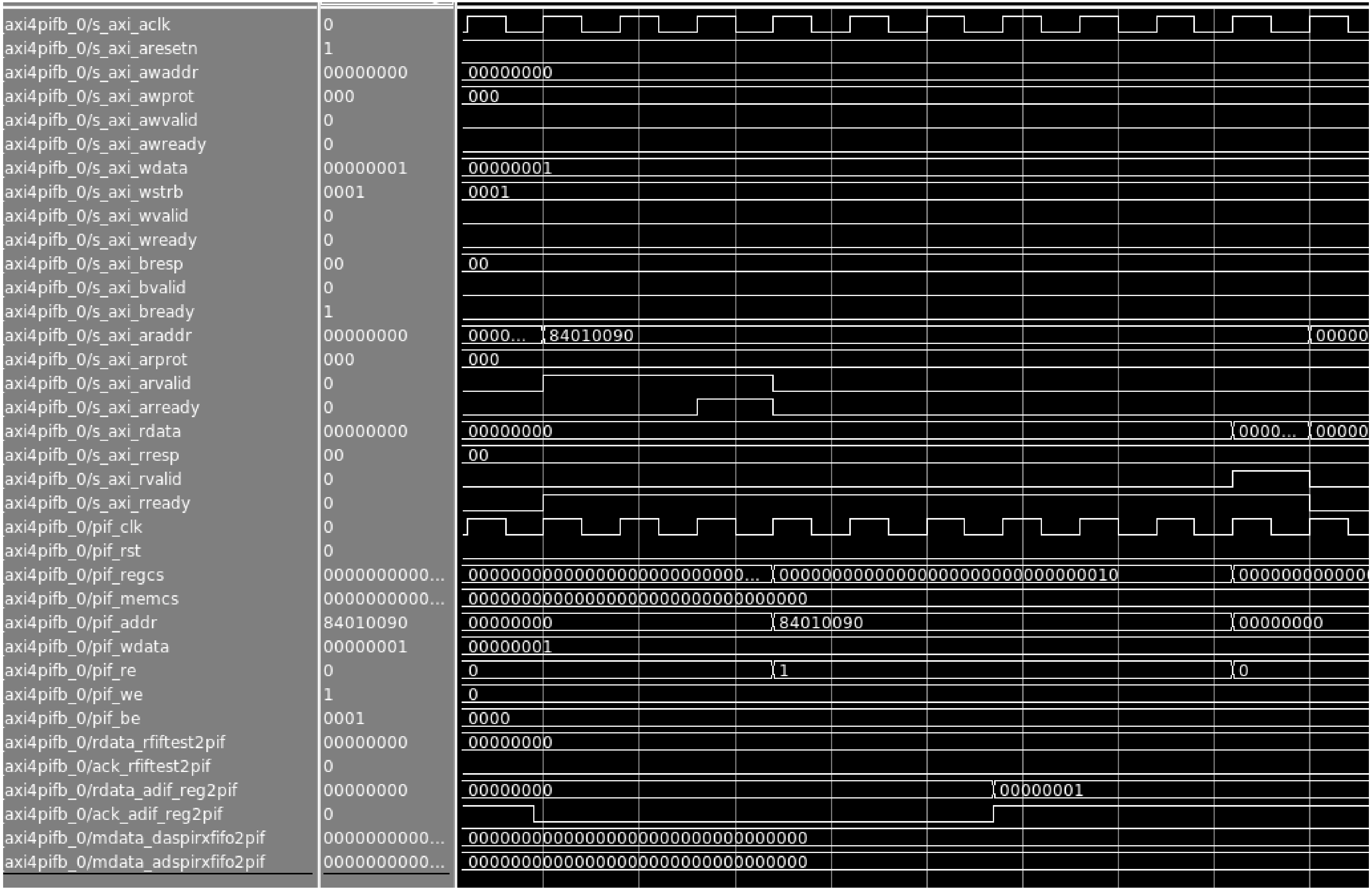
AXI4Lite to PIF bus bridge; Register write access example



axi4pifb_0/s_axi_aclk	0
axi4pifb_0/s_axi_aresetn	1
axi4pifb_0/s_axi_awaddr	00000000
axi4pifb_0/s_axi_awprot	000
axi4pifb_0/s_axi_awvalid	0
axi4pifb_0/s_axi_awready	0
axi4pifb_0/s_axi_wdata	00000001
axi4pifb_0/s_axi_wstrb	0001
axi4pifb_0/s_axi_wvalid	0
axi4pifb_0/s_axi_wready	0
axi4pifb_0/s_axi_bresp	00
axi4pifb_0/s_axi_bvalid	0
axi4pifb_0/s_axi_bready	1
axi4pifb_0/s_axi_araddr	00000000
axi4pifb_0/s_axi_arprot	000
axi4pifb_0/s_axi_arvalid	0
axi4pifb_0/s_axi_arready	0
axi4pifb_0/s_axi_rdata	00000000
axi4pifb_0/s_axi_rresp	00
axi4pifb_0/s_axi_rvalid	0
axi4pifb_0/s_axi_rready	0
axi4pifb_0/pif_clk	0
axi4pifb_0/pif_rst	0
axi4pifb_0/pif_regcs	0000000000...
axi4pifb_0/pif_memcs	0000000000...
axi4pifb_0/pif_addr	84010090
axi4pifb_0/pif_wdata	00000001
axi4pifb_0/pif_re	0
axi4pifb_0/pif_we	1
axi4pifb_0/pif_be	0001
axi4pifb_0/rdata_rfiftest2pif	0000000000...
axi4pifb_0/ack_rfiftest2pif	0
axi4pifb_0/rdata_adif_reg2pif	0000000000...
axi4pifb_0/ack_adif_reg2pif	0
axi4pifb_0/mdata_daspirxfifo2pif	0000000000...
axi4pifb_0/mdata_adspirxfifo2pif	0000000000...

[redacted]

AXI4Lite to PIF bus bridge; Register read access example



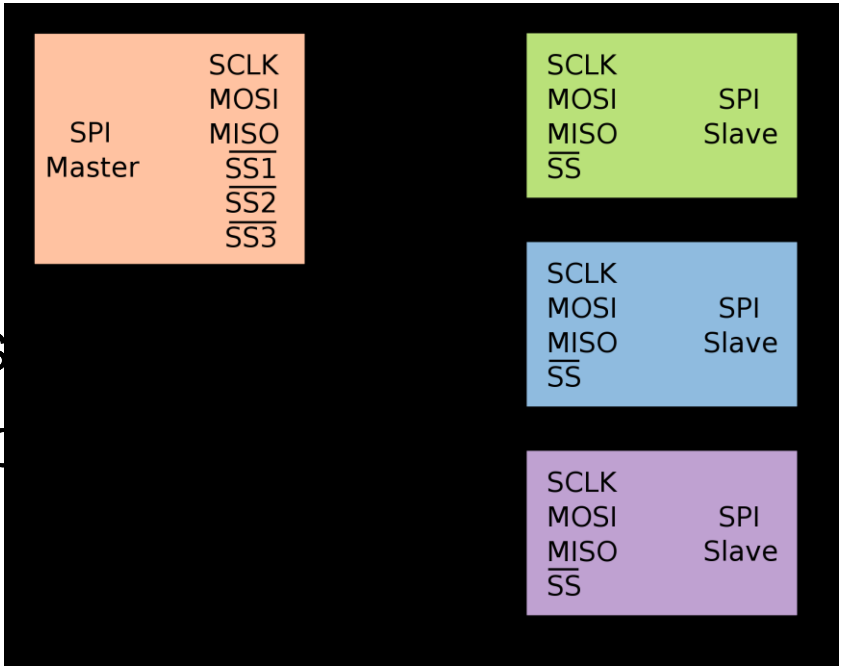
axi4pifb_0/s_axi_aclk	0
axi4pifb_0/s_axi_aresetn	1
axi4pifb_0/s_axi_awaddr	00000000
axi4pifb_0/s_axi_awprot	000
axi4pifb_0/s_axi_awvalid	0
axi4pifb_0/s_axi_awready	0
axi4pifb_0/s_axi_wdata	00000001
axi4pifb_0/s_axi_wstrb	0001
axi4pifb_0/s_axi_wvalid	0
axi4pifb_0/s_axi_wready	0
axi4pifb_0/s_axi_bresp	00
axi4pifb_0/s_axi_bvalid	0
axi4pifb_0/s_axi_bready	1
axi4pifb_0/s_axi_araddr	00000000
axi4pifb_0/s_axi_arprot	000
axi4pifb_0/s_axi_arvalid	0
axi4pifb_0/s_axi_arready	0
axi4pifb_0/s_axi_rdata	00000000
axi4pifb_0/s_axi_rresp	00
axi4pifb_0/s_axi_rvalid	0
axi4pifb_0/s_axi_rready	0
axi4pifb_0/pif_clk	0
axi4pifb_0/pif_rst	0
axi4pifb_0/pif_regcs	0000000000...
axi4pifb_0/pif_memcs	0000000000...
axi4pifb_0/pif_addr	84010090
axi4pifb_0/pif_wdata	00000001
axi4pifb_0/pif_re	0
axi4pifb_0/pif_we	1
axi4pifb_0/pif_be	0001
axi4pifb_0/rdata_rfiftest2pif	00000000
axi4pifb_0/ack_rfiftest2pif	0
axi4pifb_0/rdata_adif_reg2pif	00000000
axi4pifb_0/ack_adif_reg2pif	0
axi4pifb_0/mdata_daspirxfifo2pif	0000000000...
axi4pifb_0/mdata_adspirxfifo2pif	0000000000...

Serial Peripheral Interconnect (SPI)

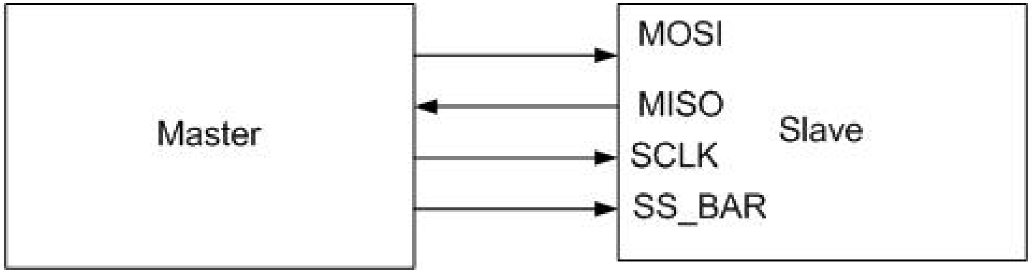
- A serial protocol in embedded systems (proposed by Motorola)
- Four-wire communication protocol
  - SCLK — Serial Clock
  - MOSI/SIMO — Master Output, Slave Input
  - MISO/SOMI — Master Input, Slave Output
  - SS/CS — Slave Select/Chip Select
  - May also come with bidirectional data in a 3 wire bus
- Single master device and with one or more slave devices
- Higher throughput than I2C and can do “stream transfers”
- No arbitration required
- But; has no slave acknowledgment (master could be talking to thin air and not even know it)
- Used to communicate across small distances

SPI Protocol

- Wires:
  - Master Out Slave In (MOS
  - Master In Slave Out (MISO
  - System Clock (SCLK)
  - Slave Select 1…N
- Master Set Slave Select low
- Master Generates Clock
- Shift registers shift in and out data



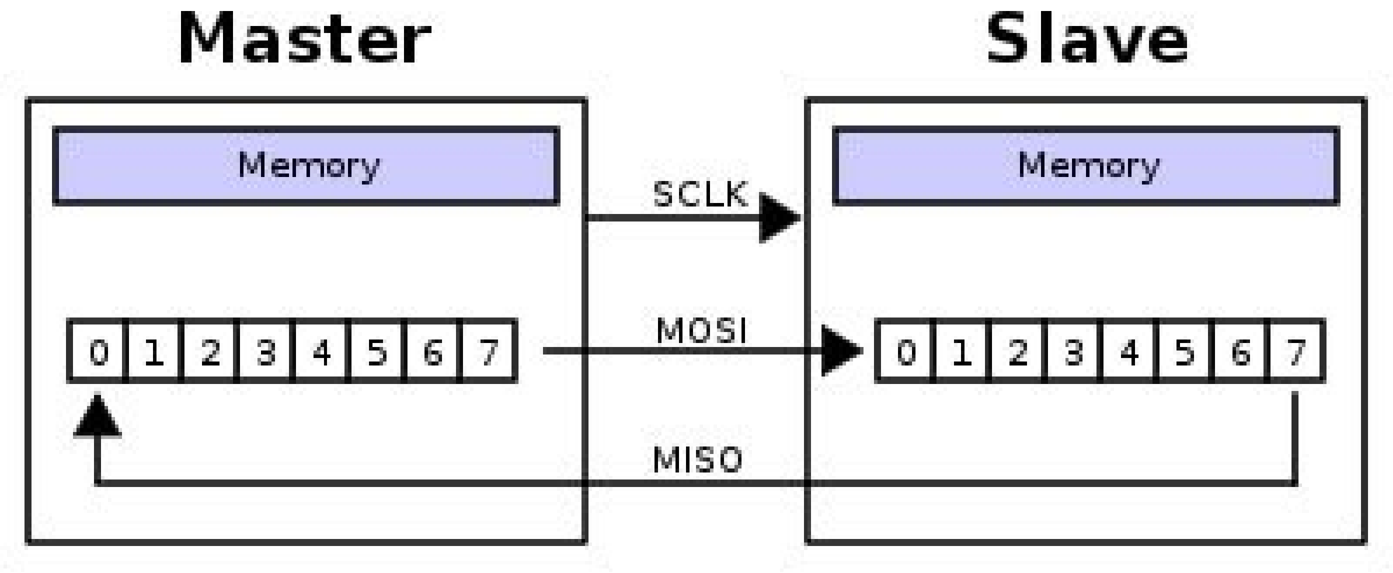
SPI Wires in Detail



- MOSI – Carries data out of Master to Slave
- MISO – Carries data from Slave to Master
  - Both signals happen for every transmission
- SS_BAR – Unique line to select a slave
- SCLK – Master produced clock to synchronize data transfer

[redacted]

SPI uses a “shift register” model of communications



Master shifts out data to Slave, and shifts in data from Slave

SPI communication; 3-wire Analog Devices AD9683 ADC

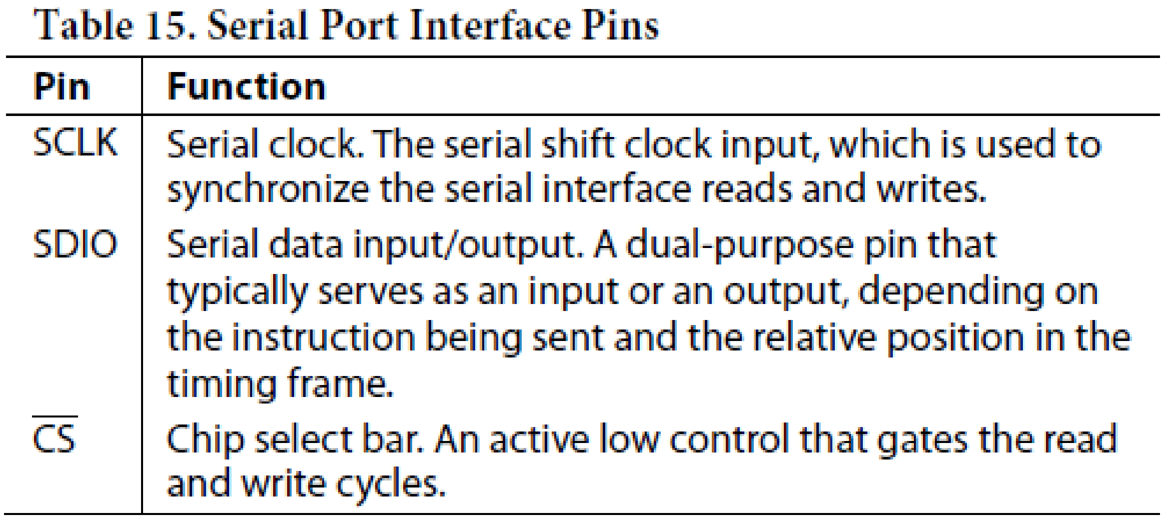
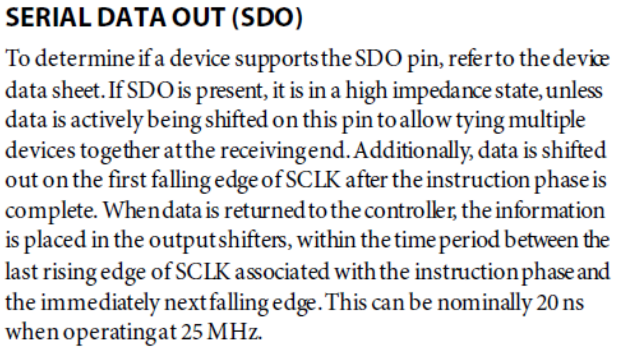


Table 15. Serial Port Interface Pins

Pin	Function
SCLK	Serial clock. The serial shift clock input, which is used to synchronize the serial interface reads and writes.
SDIO	Serial data input/output. A dual-purpose pin that typically serves as an input or an output, depending on the instruction being sent and the relative position in the timing frame.
$\overline{CS}$	Chip select bar. An active low control that gates the read and write cycles.

Example Analog Devices SPI protocol:

AN-877 Application Note; Interfacing to High Speed ADCs via SPI



SERIAL DATA OUT (SDO)

To determine if a device supports the SDO pin, refer to the device data sheet. If SDO is present, it is in a high impedance state, unless data is actively being shifted on this pin to allow tying multiple devices together at the receiving end. Additionally, data is shifted out on the first falling edge of SCLK after the instruction phase is complete. When data is returned to the controller, the information is placed in the output shifters, within the time period between the last rising edge of SCLK associated with the instruction phase and the immediately next falling edge. This can be nominally 20 ns when operating at 25 MHz.

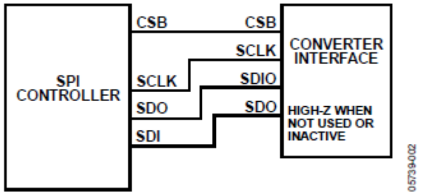
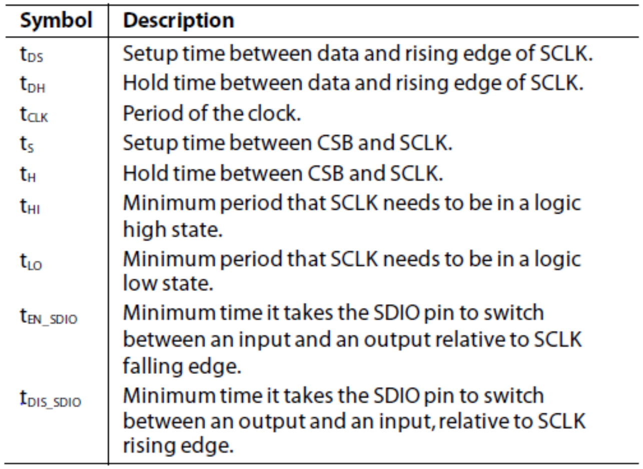


Figure 3. 3-Wire Control

Table 1. Serial Timing Specifications[1]



Symbol	Description
t_{DS}	Setup time between data and rising edge of SCLK.
t_{DH}	Hold time between data and rising edge of SCLK.
t_{CLK}	Period of the clock.
t_S	Setup time between CSB and SCLK.
t_H	Hold time between CSB and SCLK.
t_{HI}	Minimum period that SCLK needs to be in a logic high state.
t_{LO}	Minimum period that SCLK needs to be in a logic low state.
t_{EN_SDIO}	Minimum time it takes the SDIO pin to switch between an input and an output relative to SCLK falling edge.
t_{DIS_SDIO}	Minimum time it takes the SDIO pin to switch between an output and an input, relative to SCLK rising edge.

[1] See device data sheet for minimum and maximum ratings.

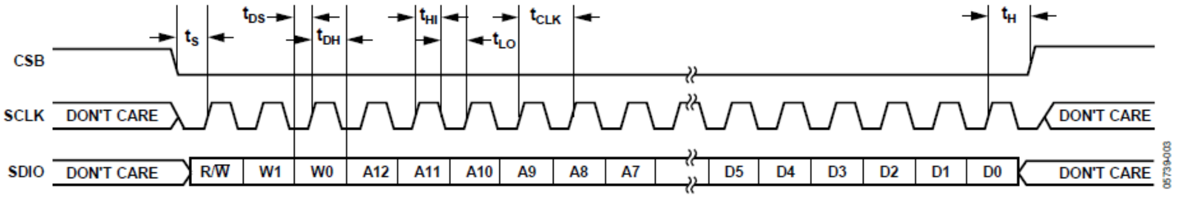
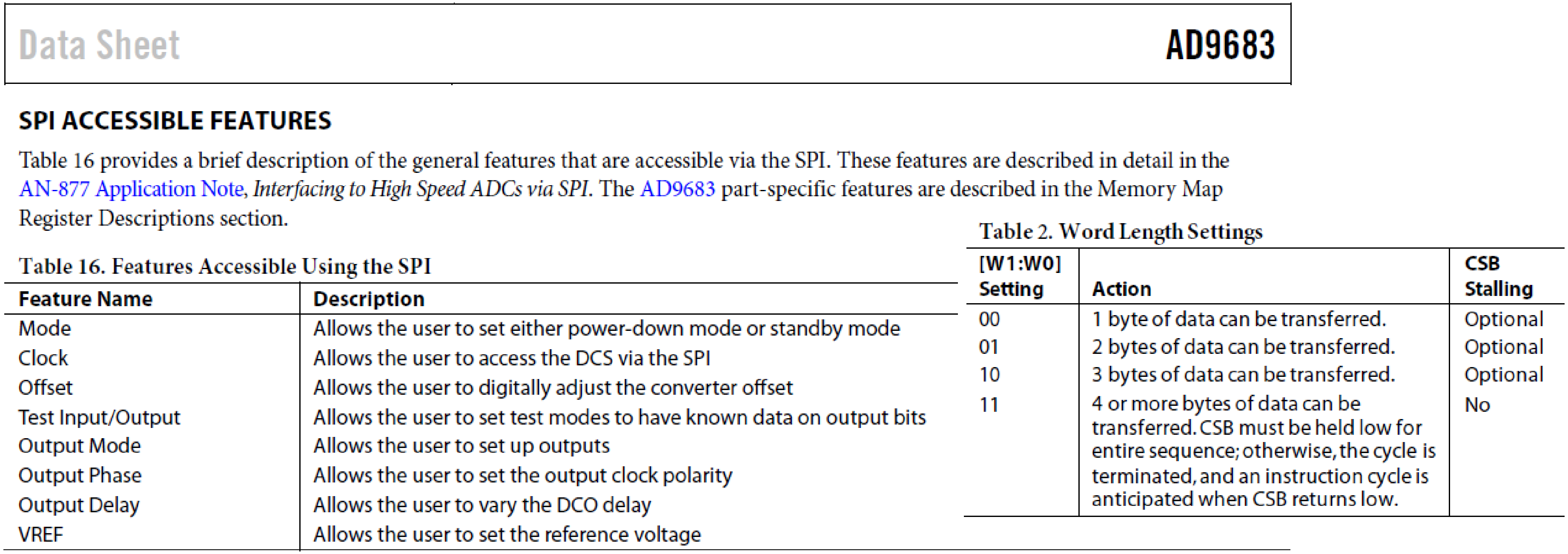


Figure 4. Setup and Hold Timing Measurements

[redacted]



Data Sheet AD9683

SPI ACCESSIBLE FEATURES

Table 16 provides a brief description of the general features that are accessible via the SPI. These features are described in detail in the AN-877 Application Note, *Interfacing to High Speed ADCs via SPI*. The AD9683 part-specific features are described in the Memory Map Register Descriptions section.

Table 16. Features Accessible Using the SPI

Feature Name	Description
Mode	Allows the user to set either power-down mode or standby mode
Clock	Allows the user to access the DCS via the SPI
Offset	Allows the user to digitally adjust the converter offset
Test Input/Output	Allows the user to set test modes to have known data on output bits
Output Mode	Allows the user to set up outputs
Output Phase	Allows the user to set the output clock polarity
Output Delay	Allows the user to vary the DCO delay
VREF	Allows the user to set the reference voltage

Table 2. Word Length Settings

[W1:W0] Setting	Action	CSB Stalling
00	1 byte of data can be transferred.	Optional
01	2 bytes of data can be transferred.	Optional
10	3 bytes of data can be transferred.	Optional
11	4 or more bytes of data can be transferred. CSB must be held low for entire sequence; otherwise, the cycle is terminated, and an instruction cycle is anticipated when CSB returns low.	No

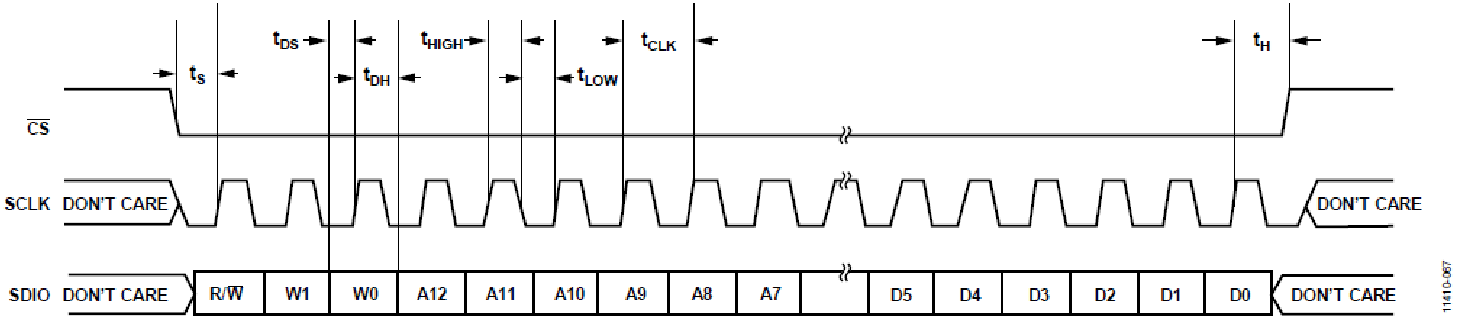


Figure 67. Serial Port Interface Timing Diagram