

This lecture gives an introduction on RAM in digital design with a special focus on FPGAs. We will discuss how memory looks like in VHDL and how to interface with it.

HWMod
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

Hardware Modeling [VU] (191.011) – WS25 – RAM in VHDL (for FPGAs)

Florian Huemer & Sebastian Wiedemann & Dylan Baumann

WS 2025/26

RAM, or Random Access Memory, is an essential component in digital design, playing a key role in storing and accessing data efficiently. This lecture is about on-chip RAM, that is, memory integrated within the design itself – in the case of this course, this means inside the FPGA.

Introduction

- On-chip RAM is a fundamental building block in digital design

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

On-chip RAM provides fast and flexible data storage capabilities directly on silicon, which is crucial for applications requiring high-speed data processing and low-latency access, as it minimizes the delay associated with external memory communication.

Introduction

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

So far, in this course, we have only covered flip-flops and latches as storage elements. Compared to these components, RAM offers several advantages and trade-offs.

Introduction

HWMoD
WS25

RAM

Introduction

FPGA Memory

Simple DP RAM

True DP RAM

Dual-Clocked RAM

Implementation

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...

RAM uses an address-based access mechanism. This means that, in order to access specific data in the memory, an address has to be provided. The RAM decodes this address and selects the corresponding location in its memory-array for reading or writing data. This is in contrast to latches or flip-flops where data is accessed directly without address decoding, as each flip-flop is hardwired to specific parts of a circuit.

Introduction

HWMoD
WS25

RAM

Introduction

FPGA Memory

Simple DP RAM

True DP RAM

Dual-Clocked RAM

Implementation

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient

This approach allows RAM to be significantly more compact, leading to improved area efficiency and reduced power consumption.

Introduction

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower

Additionally, RAM can accommodate much larger memory capacities compared to flip-flops or latches, albeit with a trade-off in access speed, as RAM tends to be slightly slower.

Introduction

HWMoD
WS25

RAM

Introduction

FPGA Memory

Simple DP RAM

True DP RAM

Dual-Clocked RAM

Implementation

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?

On-chip RAM is indispensable in a wide range of design scenarios.

Introduction

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?
 - Buffers inside or in between modules (often in the form of FIFO buffers)

For instance, it is commonly employed as buffers within or between modules, frequently implemented as First In, First Out buffers to manage data flow.

Introduction

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?
 - Buffers inside or in between modules (often in the form of FIFO buffers)

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?
 - Buffers inside or in between modules (often in the form of FIFO buffers)
 - Caches for frequently accessed data in some external memory

RAM also serves as caches, which store frequently accessed data to reduce access times for external memory.

Introduction

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?
 - Buffers inside or in between modules (often in the form of FIFO buffers)
 - Caches for frequently accessed data in some external memory

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?
 - Buffers inside or in between modules (often in the form of FIFO buffers)
 - Caches for frequently accessed data in some external memory
 - Large look-up tables

Another critical application of RAM is in large look-up tables, enabling rapid access to precomputed values or data mappings.

Introduction

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?
 - Buffers inside or in between modules (often in the form of FIFO buffers)
 - Caches for frequently accessed data in some external memory
 - Large look-up tables

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?
 - Buffers inside or in between modules (often in the form of FIFO buffers)
 - Caches for frequently accessed data in some external memory
 - Large look-up tables
 - Program and data memory for processors to store instructions and data

Furthermore, processors may rely on on-chip RAM for program and data memory, storing instructions and data necessary for their operation.

Introduction

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- On-chip RAM is a fundamental building block in digital design
- Fast, low-latency, flexible data storage
- Compared to flip-flops/latches, RAM ...
 - uses an address-based access scheme
 - is more compact in terms of area and, hence, more power efficient
 - allows for higher-capacity memory
 - is slightly slower
- Where do we need RAM in a design?
 - Buffers inside or in between modules (often in the form of FIFO buffers)
 - Caches for frequently accessed data in some external memory
 - Large look-up tables
 - Program and data memory for processors to store instructions and data

A key characteristic of RAM is its capacity, which is determined by two factors: the data width and the depth.

Introduction (cont'd)

- RAM capacity (C)

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

The data width specifies the size of each data element stored in memory. This is the amount of data stored at each memory location that can be read or written in one memory access operation.

Introduction (cont'd)

- RAM capacity (C)
 - Data width (W): size of each data element stored per address

The depth of a memory refers to the number of memory locations available for storing data, effectively determining how many individual data elements the RAM can hold.

Introduction (cont'd)

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations

Usually, memory depth is defined by the width of its address bus, that is, its address width A . However, strictly speaking it is not required that a memory array really has locations for all 2^A possible address values.

Introduction (cont'd)

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $\Rightarrow D = 2^A$

Together, the product of the data width and depth define the total storage capacity of the RAM.

Introduction (cont'd)

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $\Rightarrow D = 2^A$
 - $C = W \times D$

In addition to capacity, RAM is also characterized by the types and number of independent memory access ports it supports. A single memory port supports one operation at a time, usually in a single clock cycle.

Introduction (cont'd)

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $\Rightarrow D = 2^A$
 - $C = W \times D$
- Memory access port types

These include read ports for retrieving stored data, write ports for inserting new data, and combined read/write ports, which allow both operations to be performed through a single port.

Introduction (cont'd)

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $\Rightarrow D = 2^A$
 - $C = W \times D$
- Memory access port types
 - read
 - write
 - read/write

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $2^A \leq D \leq 2^A$
 - $C = W \times D$
- Memory access port types
 - read
 - write
 - read/write
- Multiple independent ports are possible

Modern on-chip RAM architectures can support multiple independent ports that can be operated simultaneously.

Introduction (cont'd)

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $\Rightarrow D = 2^A$
 - $C = W \times D$
- Memory access port types
 - read
 - write
 - read/write
- Multiple independent ports are possible

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $2^A \leq D$
 - $C = W \times D$
- Memory access port types
 - read
 - write
 - read/write
- Multiple independent ports are possible
 - Simple dual-port RAM: one read and one write port

A common example is simple dual-port RAM, which features one read port and one write port, allowing simultaneous reading and writing operations. This type of RAM is very common in digital design, as it is necessary to implement FIFO buffers that can be read and written simultaneously.

Introduction (cont'd)

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $\Rightarrow D = 2^A$
 - $C = W \times D$
- Memory access port types
 - read
 - write
 - read/write
- Multiple independent ports are possible
 - Simple dual-port RAM: one read and one write port

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $2^A \leq D \leq 2^A$
 - $C = W \times D$
- Memory access port types
 - read
 - write
 - read/write
- Multiple independent ports are possible
 - Simple dual-port RAM: one read and one write port
 - True dual-port RAM: two read/write ports

A more advanced configuration, that is also very common, is true dual-port RAM, which provides two read/write ports. This design permits two fully independent read- or write-operations to occur simultaneously. In this lecture, we will look at examples for both of these types.

Introduction (cont'd)

- RAM capacity (C)
 - Data width (W): size of each data element stored per address
 - Depth (D): number of addressable memory locations
 - usually defined by the address width A
 - $\Rightarrow D = 2^A$
 - $C = W \times D$
- Memory access port types
 - read
 - write
 - read/write
- Multiple independent ports are possible
 - Simple dual-port RAM: one read and one write port
 - True dual-port RAM: two read/write ports

Dedicated on-chip memory blocks are a critical resource in FPGA design. They provide efficient storage and rapid access to data directly within the programmable fabric of the device.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource

HWMoD
WS25

RAM

- Introduction
- FPGA Memory**
- Simple DP RAM
- True DP RAM
- Dual-Clocked RAM
- Implementation

- └ RAM in VHDL (for FPGAs)
 - └ Memory Blocks in FPGAs
 - └ **Memory Blocks in FPGAs**

- Memory blocks are an important FPGA resource
- Logic-based memory (often) impractical
- Highest speed/lowest latency memory in a design

Attempting to implement large memories using the storage elements provided by general-purpose logic elements in an FPGA would require a significant number of resources, which is expensive and often impractical. Logic-based memory consumes excessive chip area and power, leaving fewer resources available for other tasks, which is why FPGAs are designed with dedicated memory blocks optimized for high-density storage and efficient access.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- └ RAM in VHDL (for FPGAs)
 - └ Memory Blocks in FPGAs
 - └ **Memory Blocks in FPGAs**

- Memory blocks are an important FPGA resource
- Logic-based memory (often) impractical
- Highest speed/lowest latency memory in a design

Implementing memory using general-purpose logic only makes sense for memories which are significantly smaller than the available minimum memory block sizes in a given FPGA, or if a certain critical performance goal has to be met. However, this is rarely a decision that you as the developer have to make, but is rather left to the synthesis tool.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design

HWMoD
WS25

RAM

- Introduction
- FPGA Memory**
- Simple DP RAM
- True DP RAM
- Dual-Clocked RAM
- Implementation

One of the key features of memory blocks in FPGAs is their high configurability and versatility.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of

One physical memory block on the chip can be used with a wide range of different data and address widths.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

Additionally, often various access port configurations are supported, allowing memory blocks to be tailored to meet specific application needs. Usually, FPGA block RAM supports at least simple, but commonly also true dual-port modes for simultaneous read and write operations. Often it is also possible to use different data and address widths on different ports of a block RAM.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width
 - number and type of access ports
 - control signals (e.g., read/write/byte enable)

RAM in VHDL (for FPGAs)

Memory Blocks in FPGAs

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often impractical)
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width
 - number and type of access ports
 - control signals (e.g., read/write/byte enable)
 - parity bits (error detection)

Some memory blocks even include built-in support for parity bits, which are essential for error detection and maintaining data integrity in critical systems.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width
 - number and type of access ports
 - control signals (e.g., read/write/byte enable)
 - parity bits (error detection)

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

Memory Blocks in FPGAs

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width
 - number and type of access ports
 - control signals (e.g., read/write/byte enable)
 - parity bits (error detection)
- Memory in modern FPGAs is (almost) always synchronous

Another defining characteristic of memory blocks in modern FPGAs is their synchronous operation.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width
 - number and type of access ports
 - control signals (e.g., read/write/byte enable)
 - parity bits (error detection)
- Memory in modern FPGAs is (almost) always synchronous

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

Memory Blocks in FPGAs

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width
 - number and type of access ports
 - control signals (e.g., read/write/byte enable)
 - parity bits (error detection)
- Memory in modern FPGAs is (almost) always synchronous
 - Read and write operations only happen at a clock edge

In synchronous memory, read and write actions occur only at the active clock edge, allowing for a seamless integration with other synchronous components and simplifying timing analysis.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width
 - number and type of access ports
 - control signals (e.g., read/write/byte enable)
 - parity bits (error detection)
- Memory in modern FPGAs is (almost) always synchronous
 - Read and write operations only happen at a clock edge

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

Memory Blocks in FPGAs

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width
 - number and type of access ports
 - control signals (e.g., read/write/byte enable)
 - parity bits (error detection)
- Memory in modern FPGAs is (almost) always synchronous
 - Read and write operations only happen at a clock edge
 - Some older FPGAs support asynchronous memory

In contrast, older FPGA architectures sometimes included asynchronous memory, which allowed operations to occur independently of the clock signal. While asynchronous memory provided more flexibility in certain contexts, it is less compatible with modern high-speed digital designs, where synchronization and timing consistency are paramount.

Memory Blocks in FPGAs

- Memory blocks are an important FPGA resource
 - Logic-based memory (often) impractical
 - Highest speed/lowest latency memory in a design
- Highly configurable in terms of
 - data width
 - address width
 - number and type of access ports
 - control signals (e.g., read/write/byte enable)
 - parity bits (error detection)
- Memory in modern FPGAs is (almost) always synchronous
 - Read and write operations only happen at a clock edge
 - Some older FPGAs support asynchronous memory

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- └ RAM in VHDL (for FPGAs)
- └ Memory Blocks in FPGAs
- └ Intel FPGAs

Table 3. Embedded Memory Blocks in Intel FPGA Devices

Device Family	MLAB (640 bits)	Memory Block Type				Logic Cell (LC)
		M9K (9 Kbits)	M144K (144 Kbits)	M10K (10 Kbits)	M20K (20 Kbits)	
Arria® II GX	Yes	Yes	-	-	-	Yes
Arria II GZ	Yes	Yes	Yes	-	-	Yes
Arria V	Yes	-	-	Yes	-	Yes
Intel Arria 10	Yes	-	-	-	Yes	Yes
Cyclone® IV	-	Yes	-	-	-	Yes
Cyclone V	Yes	-	-	Yes	-	Yes
Intel Cyclone 10 LP	-	Yes	-	-	-	Yes
Intel Cyclone 10 GX	Yes	-	-	-	Yes	Yes
MAX® II	-	-	-	-	-	Yes
Intel MAX 10	-	Yes	-	-	-	Yes
Stratix IV	Yes	Yes	Yes	-	-	Yes
Stratix V	Yes	-	-	-	Yes	Yes

Source: Embedded Memory User Guide

This slide shows an overview of the available memory resources in the various Intel FPGA device families. It outlines the types of memory blocks and their availability across device families.

Intel FPGAs

Table 3. Embedded Memory Blocks in Intel FPGA Devices

Device Family	Memory Block Type					
	MLAB (640 bits)	M9K (9 Kbits)	M144K (144 Kbits)	M10K (10 Kbits)	M20K (20 Kbits)	Logic Cell (LC)
Arria® II GX	Yes	Yes	-	-	-	Yes
Arria II GZ	Yes	Yes	Yes	-	-	Yes
Arria V	Yes	-	-	Yes	-	Yes
Intel Arria 10	Yes	-	-	-	Yes	Yes
Cyclone® IV	-	Yes	-	-	-	Yes
Cyclone V	Yes	-	-	Yes	-	Yes
Intel Cyclone 10 LP	-	Yes	-	-	-	Yes
Intel Cyclone 10 GX	Yes	-	-	-	Yes	Yes
MAX® II	-	-	-	-	-	Yes
Intel MAX 10	-	Yes	-	-	-	Yes
Stratix IV	Yes	Yes	Yes	-	-	Yes
Stratix V	Yes	-	-	-	Yes	Yes

Source: Embedded Memory User Guide

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- └ RAM in VHDL (for FPGAs)
- └ Memory Blocks in FPGAs
- └ Intel FPGAs

Table 3. Embedded Memory Blocks in Intel FPGA Devices

Device Family	MLAB (640 bits)	M9K (9 Kbits)	M144K (144 Kbits)	M10K (10 Kbits)	M20K (20 Kbits)	Logic Cell (LC)
Arria® II GX	Yes	Yes	-	-	-	Yes
Arria II GZ	Yes	Yes	Yes	-	-	Yes
Arria V	Yes	-	-	Yes	-	Yes
Intel Arria 10	Yes	-	-	-	Yes	Yes
Cyclone® IV	-	Yes	-	-	-	Yes
Cyclone V	Yes	-	-	Yes	-	Yes
Intel Cyclone 10 LP	-	Yes	-	-	-	Yes
Intel Cyclone 10 GX	Yes	-	-	-	Yes	Yes
MAX® II	-	-	-	-	-	Yes
Intel MAX 10	-	Yes	-	-	-	Yes
Stratix IV	Yes	Yes	Yes	-	-	Yes
Stratix V	Yes	-	-	-	Yes	Yes

Source: Embedded Memory User Guide

For instance, the Cyclone IV, which we use in the exercise part of this course, supports logic-based memory as well as dedicated M9K memory blocks.

Intel FPGAs

Table 3. Embedded Memory Blocks in Intel FPGA Devices

Device Family	Memory Block Type					
	MLAB (640 bits)	M9K (9 Kbits)	M144K (144 Kbits)	M10K (10 Kbits)	M20K (20 Kbits)	Logic Cell (LC)
Arria® II GX	Yes	Yes	-	-	-	Yes
Arria II GZ	Yes	Yes	Yes	-	-	Yes
Arria V	Yes	-	-	Yes	-	Yes
Intel Arria 10	Yes	-	-	-	Yes	Yes
Cyclone® IV	-	Yes	-	-	-	Yes
Cyclone V	Yes	-	-	Yes	-	Yes
Intel Cyclone 10 LP	-	Yes	-	-	-	Yes
Intel Cyclone 10 GX	Yes	-	-	-	Yes	Yes
MAX® II	-	-	-	-	-	Yes
Intel MAX 10	-	Yes	-	-	-	Yes
Stratix IV	Yes	Yes	Yes	-	-	Yes
Stratix V	Yes	-	-	-	Yes	Yes

Source: Embedded Memory User Guide

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- └ RAM in VHDL (for FPGAs)
- └ Memory Blocks in FPGAs
- └ Intel FPGAs

Table 3. Embedded Memory Blocks in Intel FPGA Devices

Device Family	MLAB (640 bits)	M9K (9 Kbits)	M144K (144 Kbits)	M10K (10 Kbits)	M20K (20 Kbits)	Logic Cell (LC)
Arria® II GX	Yes	Yes	-	-	-	Yes
Arria II GZ	Yes	Yes	Yes	-	-	Yes
Arria V	Yes	-	-	Yes	-	Yes
Intel Arria 10	Yes	-	-	-	Yes	Yes
Cyclone® IV	-	Yes	-	-	-	Yes
Cyclone V	Yes	-	-	Yes	-	Yes
Intel Cyclone 10 LP	-	Yes	-	-	-	Yes
Intel Cyclone 10 GX	Yes	-	-	-	Yes	Yes
MAX® II	-	-	-	-	-	Yes
Intel MAX 10	-	Yes	-	-	-	Yes
Stratix IV	Yes	Yes	Yes	-	-	Yes
Stratix V	Yes	-	-	-	Yes	Yes

Source: Embedded Memory User Guide

Other, more recent devices, such as the Arria 10 utilize larger M20K blocks for enhanced memory capacity. The table shows that memory resources in FPGA are quite diverse and vary greatly between device families.

Intel FPGAs

Table 3. Embedded Memory Blocks in Intel FPGA Devices

Device Family	Memory Block Type					
	MLAB (640 bits)	M9K (9 Kbits)	M144K (144 Kbits)	M10K (10 Kbits)	M20K (20 Kbits)	Logic Cell (LC)
Arria® II GX	Yes	Yes	-	-	-	Yes
Arria II GZ	Yes	Yes	Yes	-	-	Yes
Arria V	Yes	-	-	Yes	-	Yes
Intel Arria 10	Yes	-	-	-	Yes	Yes
Cyclone® IV	-	Yes	-	-	-	Yes
Cyclone V	Yes	-	-	Yes	-	Yes
Intel Cyclone 10 LP	-	Yes	-	-	-	Yes
Intel Cyclone 10 GX	Yes	-	-	-	Yes	Yes
MAX® II	-	-	-	-	-	Yes
Intel MAX 10	-	Yes	-	-	-	Yes
Stratix IV	Yes	Yes	Yes	-	-	Yes
Stratix V	Yes	-	-	-	Yes	Yes

Source: Embedded Memory User Guide

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

Simple Dual-Port RAM

Simple Dual-Port RAM

```
1 entity simple_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive;
5     WR_EN :
6       type t is std_ulogic;
7       type t is std_ulogic;
8     RD_EN : std_ulogic;
9     WR_ADDR : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
10    RD_ADDR : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
11    WR_DATA : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
12    RD_DATA : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
13  );
14  port (
15    clk : in std_ulogic;
16    wr_en : in std_ulogic;
17    rd_en : in std_ulogic;
18    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
19    rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
20    wr_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
21    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
22  );
23 end entity;
```

Let us now look at some VHDL code. This slide shows how an entity declaration for a typical simple dual-port memory in VHDL can look like. Of course the naming convention for the signal names is not standardized, and you will find many variations in practice. In this lecture we always use a common identifier to prefix the signals that belong to a specific memory port.

Simple Dual-Port RAM

```
1 entity simple_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive;
5   );
6   port (
7     clk : in std_ulogic;
8     -- read port
9     rd_en : in std_ulogic;
10    rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
11    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
12    -- write port
13    wr_en : in std_ulogic;
14    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
15    wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0)
16  );
17 end entity;
```

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
Read Access
Write Access
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

Simple Dual-Port RAM

Simple Dual-Port RAM

```
1 entity simple_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive;
5   );
6   port (
7     clk : in std_ulogic;
8     rd_en : in std_ulogic;
9     rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
10    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
11    wr_en : in std_ulogic;
12    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
13    wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0);
14  );
15 end entity;
```

The clock signal that controls all operations on both the read and the write port.

As already explained, memory blocks in modern FPGAs are always synchronous. Hence, we need a clock input to which all memory operations are synchronized to.

Simple Dual-Port RAM

```
1 entity simple_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive;
5   );
6   port (
7     clk : in std_ulogic;
8     -- read port
9     rd_en : in std_ulogic;
10    rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
11    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
12    -- write port
13    wr_en : in std_ulogic;
14    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
15    wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0);
16  );
17 end entity;
```

The clock signal that controls all operations on both the read and the write port.

HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation

RAM in VHDL (for FPGAs)

Simple Dual-Port RAM

Simple Dual-Port RAM

```
1 entity simple_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive;
5     )
6   port (
7     clk : in std_ulogic;
8     rd_en : in std_ulogic;
9     rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
10    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
11    wr_en : in std_ulogic;
12    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
13    wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0);
14    );
15 end entity;
```

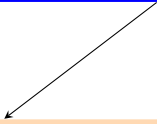
Memory read port

This group of signals represents the read-port of the RAM. It consists of an input for the address and an output for the read data. The read enable input (`rd_en`) is used to trigger the actual read operation. This is a signal that is not always present on RAM modules, since in some applications read should simply happen on every clock cycle.

Simple Dual-Port RAM

```
1 entity simple_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive;
5   );
6   port (
7     clk : in std_ulogic;
8     -- read port
9     rd_en : in std_ulogic;
10    rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
11    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
12    -- write port
13    wr_en : in std_ulogic;
14    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
15    wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0);
16  );
17 end entity;
```

Memory read port.



HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation

RAM in VHDL (for FPGAs)

Simple Dual-Port RAM

Simple Dual-Port RAM

```
1 entity simple_dp_ram is
2 generic (
3   ADDR_WIDTH : positive;
4   DATA_WIDTH : positive;
5 )
6 port (
7   clk : in std_ulogic;
8   rd_en : in std_ulogic;
9   rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
10  rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
11  wr_en : in std_ulogic;
12  wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
13  wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0);
14 );
15 end entity;
```

Memory write port

The last group of signals represents the write-port. Here, both the data and address signals are inputs. The write-enable (`wr_en`) signal controls when data is actually written to memory. Sometimes RAM also features byte-enable signals on the write-port to only selectively write certain parts of the input data.

Simple Dual-Port RAM

HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation

```
1 entity simple_dp_ram is
2 generic (
3   ADDR_WIDTH : positive;
4   DATA_WIDTH : positive;
5 );
6 port (
7   clk : in std_ulogic;
8   -- read port
9   rd_en : in std_ulogic;
10  rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
11  rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
12  -- write port
13  wr_en : in std_ulogic;
14  wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
15  wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0);
16 );
17 end entity;
```

Memory write port.

- RAM in VHDL (for FPGAs)
 - Simple Dual-Port RAM
 - Read Access**

```
clk  —
rd_en —
rd_addr [D/C]
rd_data [D/C]
```

Now that we know the interface signals, we can discuss how to interface with the shown RAM entity. For that purpose we assume that the RAM has a latency of one clock cycle. This means that the respective data is available in the next clock cycle from the point in time when the read was issued. Initially, both the input address as well as the output data are in a "don't care" state, which just means that their actual values are irrelevant for the purpose of this example. The read enable signal is zero.

Read Access

HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access**
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation

```
clk  —
rd_en —
rd_addr [D/C]
rd_data [D/C]
```

- RAM in VHDL (for FPGAs)
 - Simple Dual-Port RAM
 - Read Access**



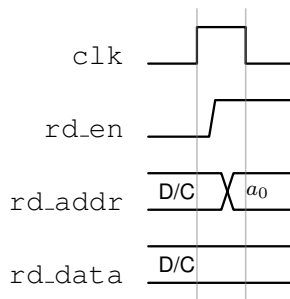
Now, let's say we want to read the data stored in the memory at address a_0 . Hence, we apply the respective address to the address input of the read port and set the read enable signal to high.

Read Access

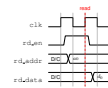
HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access**
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation



- └ RAM in VHDL (for FPGAs)
 - └ Simple Dual-Port RAM
 - └ **Read Access**



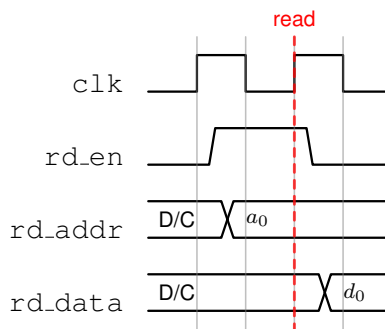
At the next rising clock edge – marked in red in the timing diagram – the RAM can now sample these inputs and output the value stored at address a_0 .

Read Access

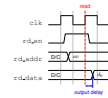
HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access**
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation



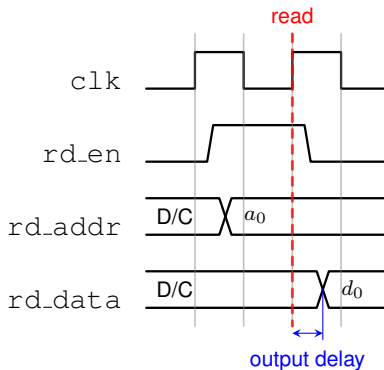
- └ RAM in VHDL (for FPGAs)
 - └ Simple Dual-Port RAM
 - └ **Read Access**



Note that, as also indicated in the timing diagram, this does not happen instantaneously but rather takes some time due to the intrinsic propagation delays and setup times associated with the memory access. The timing analysis has to take this access time value into account. However, we can safely assume that the value is available at the next rising clock edge. In this clock cycle no new read operation is performed on the memory, hence the read enable signal is reset to low.

Read Access

HWMoD
WS25



RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access**
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation

- └ RAM in VHDL (for FPGAs)
 - └ Simple Dual-Port RAM
 - └ **Read Access**



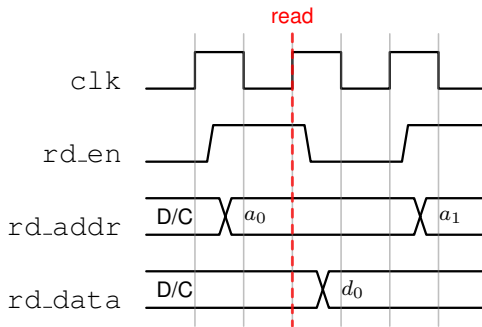
Now a second read is performed – this time on address a_1 . Note that since this read will only be initiated at the next rising clock edge the memory still outputs the value of the previous read operation.

Read Access

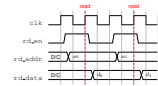
HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access**
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation



- └ RAM in VHDL (for FPGAs)
 - └ Simple Dual-Port RAM
 - └ **Read Access**



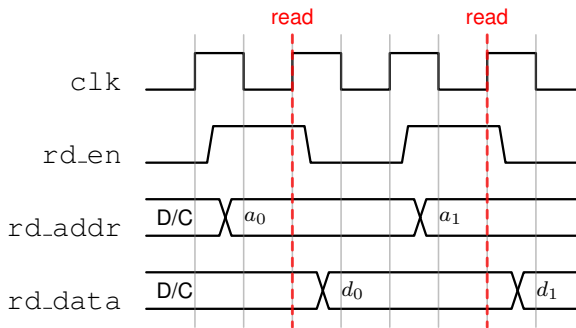
Again the read value can be retrieved after the next rising clock edge.

Read Access

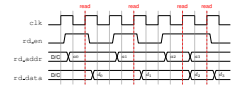
HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access**
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation

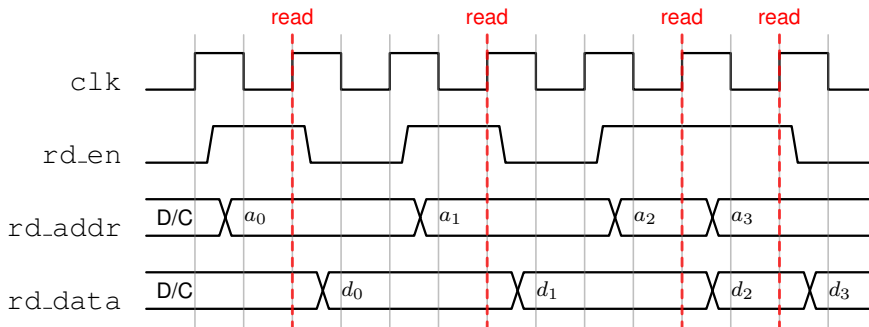


- └ RAM in VHDL (for FPGAs)
 - └ Simple Dual-Port RAM
 - └ **Read Access**



Reads can also be performed back-to-back, that is in consecutive clock cycles. This is shown for the read of addresses a_2 and a_3 in the timing diagram.

Read Access

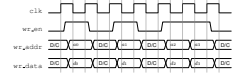


HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access**
- Write Access
- True DP RAM
- Dual-Clocked RAM
- Implementation

- RAM in VHDL (for FPGAs)
 - Simple Dual-Port RAM
 - Write Access**



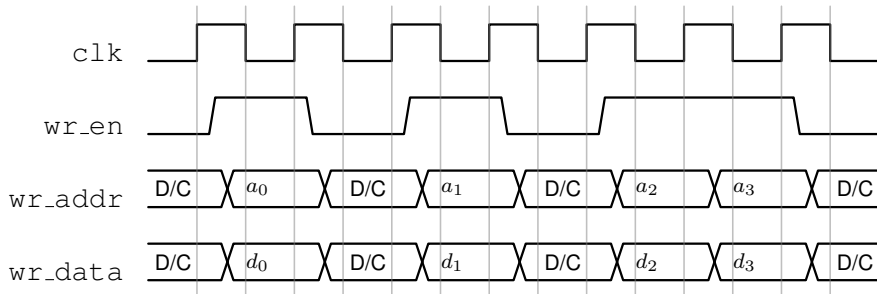
Now, let's turn to the write access, which is a little simpler than the read access.

Write Access

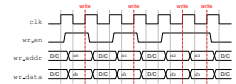
HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access
- Write Access**
- True DP RAM
- Dual-Clocked RAM
- Implementation



- RAM in VHDL (for FPGAs)
 - Simple Dual-Port RAM
 - Write Access**



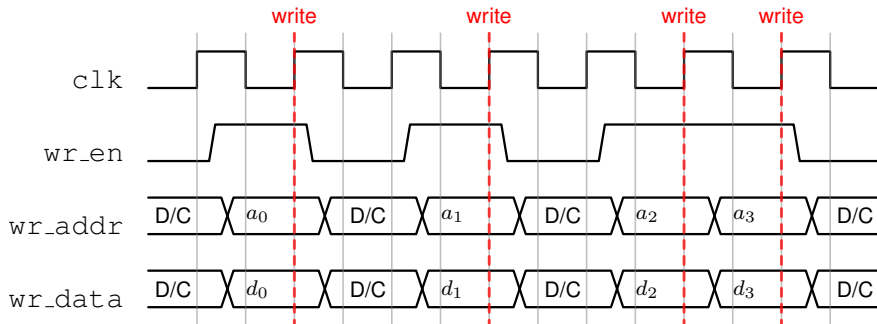
Whenever data should be written, the address and the associated data have to be applied at the respective inputs of the memory. If the write-enable signal (wr_en) is high at the rising edge of the clock signal, the data and address signals are sampled leading to the data being written to memory.

Write Access

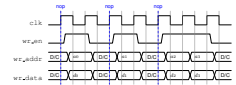
HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- Read Access
- Write Access**
- True DP RAM
- Dual-Clocked RAM
- Implementation



- RAM in VHDL (for FPGAs)
 - Simple Dual-Port RAM
 - Write Access**

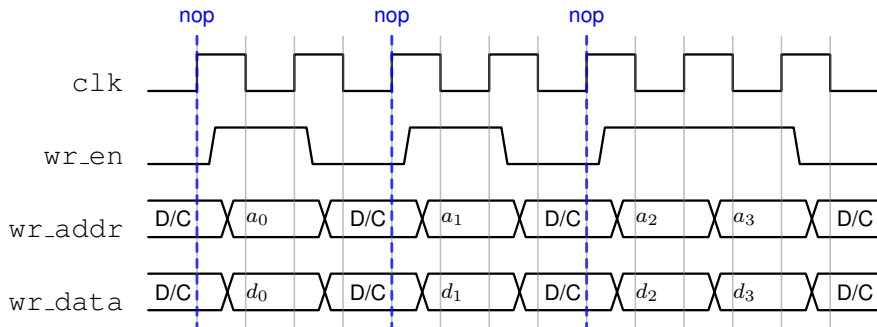


In all other cases the memory simply ignores the values of the data and address signals, which is indicated by the "don't care" values in the timing diagram. Please pause the video at this point and make sure that you understand the shown timing diagram.

Write Access

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
Read Access
Write Access
True DP RAM
Dual-Clocked RAM
Implementation



RAM in VHDL (for FPGAs)

True Dual-Port RAM

True Dual-Port RAM

```

1 entity true_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive;
5     M0 :
6     port (
7       clk : in std_ulogic;
8       rd0_en : in std_ulogic;
9       wr0_en : in std_ulogic;
10      rd0_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
11      wr0_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
12      rd0_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
13      rd1_en : in std_ulogic;
14      wr1_en : in std_ulogic;
15      rd1_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
16      wr1_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
17      rd1_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
18    );
19 end entity;

```

This slide shows an example for the interface of a true dual-port RAM.

True Dual-Port RAM

```

1 entity true_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive;
5   );
6   port (
7     clk : in std_ulogic;
8     -- read/write port 0
9     rw0_rd_en : in std_ulogic;
10    rw0_wr_en : in std_ulogic;
11    rw0_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
12    rw0_wr_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
13    rw0_rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
14    -- read/write port 1
15    rw1_rd_en : in std_ulogic;
16    rw1_wr_en : in std_ulogic;
17    rw1_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
18    rw1_wr_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
19    rw1_rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
20  );
21 end entity;

```

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

- True Dual-Port RAM
- True Dual-Port RAM

```
1 entity True_DP_RAM is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive
5   );
6   port (
7     clk : in std_ulogic;
8     -- read/write port 0
9     rw0_rd_en : in std_ulogic;
10    rw0_wr_en : in std_ulogic;
11    rw0_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
12    rw0_wr_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
13    rw0_rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
14    -- read/write port 1
15    rw1_rd_en : in std_ulogic;
16    rw1_wr_en : in std_ulogic;
17    rw1_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
18    rw1_wr_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
19    rw1_rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0)
20  );
21 end entity;
```

True read/write memory port

Notice that in our example both ports consist of the exact same signals. As already mentioned, this is not a strict requirement, as it is also possible and supported by many FPGAs to use different interface configurations on both ports.

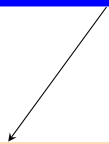
True Dual-Port RAM

HWMMod
WS25

- RAM
- Introduction
- FPGA Memory
- Simple DP RAM
- True DP RAM
- Dual-Clocked RAM
- Implementation

```
1 entity true_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive
5   );
6   port (
7     clk : in std_ulogic;
8     -- read/write port 0
9     rw0_rd_en : in std_ulogic;
10    rw0_wr_en : in std_ulogic;
11    rw0_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
12    rw0_wr_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
13    rw0_rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
14    -- read/write port 1
15    rw1_rd_en : in std_ulogic;
16    rw1_wr_en : in std_ulogic;
17    rw1_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
18    rw1_wr_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
19    rw1_rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0)
20  );
21 end entity;
```

First read/write memory port.



RAM in VHDL (for FPGAs)

- True Dual-Port RAM
- True Dual-Port RAM

```
1 entity True_DP_RAM is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive;
5     M0 :
6     M1 :
7     M2 :
8     M3 :
9     M4 :
10    M5 :
11    M6 :
12    M7 :
13    M8 :
14    M9 :
15    M10 :
16    M11 :
17    M12 :
18    M13 :
19    M14 :
20    M15 :
21    M16 :
22    M17 :
23    M18 :
24    M19 :
25    M20 :
26    M21 :
27    M22 :
28    M23 :
29    M24 :
30    M25 :
31    M26 :
32    M27 :
33    M28 :
34    M29 :
35    M30 :
36    M31 :
37    M32 :
38    M33 :
39    M34 :
40    M35 :
41    M36 :
42    M37 :
43    M38 :
44    M39 :
45    M40 :
46    M41 :
47    M42 :
48    M43 :
49    M44 :
50    M45 :
51    M46 :
52    M47 :
53    M48 :
54    M49 :
55    M50 :
56    M51 :
57    M52 :
58    M53 :
59    M54 :
60    M55 :
61    M56 :
62    M57 :
63    M58 :
64    M59 :
65    M60 :
66    M61 :
67    M62 :
68    M63 :
69    M64 :
70    M65 :
71    M66 :
72    M67 :
73    M68 :
74    M69 :
75    M70 :
76    M71 :
77    M72 :
78    M73 :
79    M74 :
80    M75 :
81    M76 :
82    M77 :
83    M78 :
84    M79 :
85    M80 :
86    M81 :
87    M82 :
88    M83 :
89    M84 :
90    M85 :
91    M86 :
92    M87 :
93    M88 :
94    M89 :
95    M90 :
96    M91 :
97    M92 :
98    M93 :
99    M94 :
100   M95 :
101   M96 :
102   M97 :
103   M98 :
104   M99 :
105   M100 :
106   M101 :
107   M102 :
108   M103 :
109   M104 :
110   M105 :
111   M106 :
112   M107 :
113   M108 :
114   M109 :
115   M110 :
116   M111 :
117   M112 :
118   M113 :
119   M114 :
120   M115 :
121   M116 :
122   M117 :
123   M118 :
124   M119 :
125   M120 :
126   M121 :
127   M122 :
128   M123 :
129   M124 :
130   M125 :
131   M126 :
132   M127 :
133   M128 :
134   M129 :
135   M130 :
136   M131 :
137   M132 :
138   M133 :
139   M134 :
140   M135 :
141   M136 :
142   M137 :
143   M138 :
144   M139 :
145   M140 :
146   M141 :
147   M142 :
148   M143 :
149   M144 :
150   M145 :
151   M146 :
152   M147 :
153   M148 :
154   M149 :
155   M150 :
156   M151 :
157   M152 :
158   M153 :
159   M154 :
160   M155 :
161   M156 :
162   M157 :
163   M158 :
164   M159 :
165   M160 :
166   M161 :
167   M162 :
168   M163 :
169   M164 :
170   M165 :
171   M166 :
172   M167 :
173   M168 :
174   M169 :
175   M170 :
176   M171 :
177   M172 :
178   M173 :
179   M174 :
180   M175 :
181   M176 :
182   M177 :
183   M178 :
184   M179 :
185   M180 :
186   M181 :
187   M182 :
188   M183 :
189   M184 :
190   M185 :
191   M186 :
192   M187 :
193   M188 :
194   M189 :
195   M190 :
196   M191 :
197   M192 :
198   M193 :
199   M194 :
200   M195 :
201   M196 :
202   M197 :
203   M198 :
204   M199 :
205   M200 :
206   M201 :
207   M202 :
208   M203 :
209   M204 :
210   M205 :
211   M206 :
212   M207 :
213   M208 :
214   M209 :
215   M210 :
216   M211 :
217   M212 :
218   M213 :
219   M214 :
220   M215 :
221   M216 :
222   M217 :
223   M218 :
224   M219 :
225   M220 :
226   M221 :
227   M222 :
228   M223 :
229   M224 :
230   M225 :
231   M226 :
232   M227 :
233   M228 :
234   M229 :
235   M230 :
236   M231 :
237   M232 :
238   M233 :
239   M234 :
240   M235 :
241   M236 :
242   M237 :
243   M238 :
244   M239 :
245   M240 :
246   M241 :
247   M242 :
248   M243 :
249   M244 :
250   M245 :
251   M246 :
252   M247 :
253   M248 :
254   M249 :
255   M250 :
256   M251 :
257   M252 :
258   M253 :
259   M254 :
260   M255 :
261   M256 :
262   M257 :
263   M258 :
264   M259 :
265   M260 :
266   M261 :
267   M262 :
268   M263 :
269   M264 :
270   M265 :
271   M266 :
272   M267 :
273   M268 :
274   M269 :
275   M270 :
276   M271 :
277   M272 :
278   M273 :
279   M274 :
280   M275 :
281   M276 :
282   M277 :
283   M278 :
284   M279 :
285   M280 :
286   M281 :
287   M282 :
288   M283 :
289   M284 :
290   M285 :
291   M286 :
292   M287 :
293   M288 :
294   M289 :
295   M290 :
296   M291 :
297   M292 :
298   M293 :
299   M294 :
300   M295 :
301   M296 :
302   M297 :
303   M298 :
304   M299 :
305   M300 :
306   M301 :
307   M302 :
308   M303 :
309   M304 :
310   M305 :
311   M306 :
312   M307 :
313   M308 :
314   M309 :
315   M310 :
316   M311 :
317   M312 :
318   M313 :
319   M314 :
320   M315 :
321   M316 :
322   M317 :
323   M318 :
324   M319 :
325   M320 :
326   M321 :
327   M322 :
328   M323 :
329   M324 :
330   M325 :
331   M326 :
332   M327 :
333   M328 :
334   M329 :
335   M330 :
336   M331 :
337   M332 :
338   M333 :
339   M334 :
340   M335 :
341   M336 :
342   M337 :
343   M338 :
344   M339 :
345   M340 :
346   M341 :
347   M342 :
348   M343 :
349   M344 :
350   M345 :
351   M346 :
352   M347 :
353   M348 :
354   M349 :
355   M350 :
356   M351 :
357   M352 :
358   M353 :
359   M354 :
360   M355 :
361   M356 :
362   M357 :
363   M358 :
364   M359 :
365   M360 :
366   M361 :
367   M362 :
368   M363 :
369   M364 :
370   M365 :
371   M366 :
372   M367 :
373   M368 :
374   M369 :
375   M370 :
376   M371 :
377   M372 :
378   M373 :
379   M374 :
380   M375 :
381   M376 :
382   M377 :
383   M378 :
384   M379 :
385   M380 :
386   M381 :
387   M382 :
388   M383 :
389   M384 :
390   M385 :
391   M386 :
392   M387 :
393   M388 :
394   M389 :
395   M390 :
396   M391 :
397   M392 :
398   M393 :
399   M394 :
400   M395 :
401   M396 :
402   M397 :
403   M398 :
404   M399 :
405   M400 :
406   M401 :
407   M402 :
408   M403 :
409   M404 :
410   M405 :
411   M406 :
412   M407 :
413   M408 :
414   M409 :
415   M410 :
416   M411 :
417   M412 :
418   M413 :
419   M414 :
420   M415 :
421   M416 :
422   M417 :
423   M418 :
424   M419 :
425   M420 :
426   M421 :
427   M422 :
428   M423 :
429   M424 :
430   M425 :
431   M426 :
432   M427 :
433   M428 :
434   M429 :
435   M430 :
436   M431 :
437   M432 :
438   M433 :
439   M434 :
440   M435 :
441   M436 :
442   M437 :
443   M438 :
444   M439 :
445   M440 :
446   M441 :
447   M442 :
448   M443 :
449   M444 :
450   M445 :
451   M446 :
452   M447 :
453   M448 :
454   M449 :
455   M450 :
456   M451 :
457   M452 :
458   M453 :
459   M454 :
460   M455 :
461   M456 :
462   M457 :
463   M458 :
464   M459 :
465   M460 :
466   M461 :
467   M462 :
468   M463 :
469   M464 :
470   M465 :
471   M466 :
472   M467 :
473   M468 :
474   M469 :
475   M470 :
476   M471 :
477   M472 :
478   M473 :
479   M474 :
480   M475 :
481   M476 :
482   M477 :
483   M478 :
484   M479 :
485   M480 :
486   M481 :
487   M482 :
488   M483 :
489   M484 :
490   M485 :
491   M486 :
492   M487 :
493   M488 :
494   M489 :
495   M490 :
496   M491 :
497   M492 :
498   M493 :
499   M494 :
500   M495 :
501   M496 :
502   M497 :
503   M498 :
504   M499 :
505   M500 :
506   M501 :
507   M502 :
508   M503 :
509   M504 :
510   M505 :
511   M506 :
512   M507 :
513   M508 :
514   M509 :
515   M510 :
516   M511 :
517   M512 :
518   M513 :
519   M514 :
520   M515 :
521   M516 :
522   M517 :
523   M518 :
524   M519 :
525   M520 :
526   M521 :
527   M522 :
528   M523 :
529   M524 :
530   M525 :
531   M526 :
532   M527 :
533   M528 :
534   M529 :
535   M530 :
536   M531 :
537   M532 :
538   M533 :
539   M534 :
540   M535 :
541   M536 :
542   M537 :
543   M538 :
544   M539 :
545   M540 :
546   M541 :
547   M542 :
548   M543 :
549   M544 :
550   M545 :
551   M546 :
552   M547 :
553   M548 :
554   M549 :
555   M550 :
556   M551 :
557   M552 :
558   M553 :
559   M554 :
560   M555 :
561   M556 :
562   M557 :
563   M558 :
564   M559 :
565   M560 :
566   M561 :
567   M562 :
568   M563 :
569   M564 :
570   M565 :
571   M566 :
572   M567 :
573   M568 :
574   M569 :
575   M570 :
576   M571 :
577   M572 :
578   M573 :
579   M574 :
580   M575 :
581   M576 :
582   M577 :
583   M578 :
584   M579 :
585   M580 :
586   M581 :
587   M582 :
588   M583 :
589   M584 :
590   M585 :
591   M586 :
592   M587 :
593   M588 :
594   M589 :
595   M590 :
596   M591 :
597   M592 :
598   M593 :
599   M594 :
600   M595 :
601   M596 :
602   M597 :
603   M598 :
604   M599 :
605   M600 :
606   M601 :
607   M602 :
608   M603 :
609   M604 :
610   M605 :
611   M606 :
612   M607 :
613   M608 :
614   M609 :
615   M610 :
616   M611 :
617   M612 :
618   M613 :
619   M614 :
620   M615 :
621   M616 :
622   M617 :
623   M618 :
624   M619 :
625   M620 :
626   M621 :
627   M622 :
628   M623 :
629   M624 :
630   M625 :
631   M626 :
632   M627 :
633   M628 :
634   M629 :
635   M630 :
636   M631 :
637   M632 :
638   M633 :
639   M634 :
640   M635 :
641   M636 :
642   M637 :
643   M638 :
644   M639 :
645   M640 :
646   M641 :
647   M642 :
648   M643 :
649   M644 :
650   M645 :
651   M646 :
652   M647 :
653   M648 :
654   M649 :
655   M650 :
656   M651 :
657   M652 :
658   M653 :
659   M654 :
660   M655 :
661   M656 :
662   M657 :
663   M658 :
664   M659 :
665   M660 :
666   M661 :
667   M662 :
668   M663 :
669   M664 :
670   M665 :
671   M666 :
672   M667 :
673   M668 :
674   M669 :
675   M670 :
676   M671 :
677   M672 :
678   M673 :
679   M674 :
680   M675 :
681   M676 :
682   M677 :
683   M678 :
684   M679 :
685   M680 :
686   M681 :
687   M682 :
688   M683 :
689   M684 :
690   M685 :
691   M686 :
692   M687 :
693   M688 :
694   M689 :
695   M690 :
696   M691 :
697   M692 :
698   M693 :
699   M694 :
700   M695 :
701   M696 :
702   M697 :
703   M698 :
704   M699 :
705   M700 :
706   M701 :
707   M702 :
708   M703 :
709   M704 :
710   M705 :
711   M706 :
712   M707 :
713   M708 :
714   M709 :
715   M710 :
716   M711 :
717   M712 :
718   M713 :
719   M714 :
720   M715 :
721   M716 :
722   M717 :
723   M718 :
724   M719 :
725   M720 :
726   M721 :
727   M722 :
728   M723 :
729   M724 :
730   M725 :
731   M726 :
732   M727 :
733   M728 :
734   M729 :
735   M730 :
736   M731 :
737   M732 :
738   M733 :
739   M734 :
740   M735 :
741   M736 :
742   M737 :
743   M738 :
744   M739 :
745   M740 :
746   M741 :
747   M742 :
748   M743 :
749   M744 :
750   M745 :
751   M746 :
752   M747 :
753   M748 :
754   M749 :
755   M750 :
756   M751 :
757   M752 :
758   M753 :
759   M754 :
760   M755 :
761   M756 :
762   M757 :
763   M758 :
764   M759 :
765   M760 :
766   M761 :
767   M762 :
768   M763 :
769   M764 :
770   M765 :
771   M766 :
772   M767 :
773   M768 :
774   M769 :
775   M770 :
776   M771 :
777   M772 :
778   M773 :
779   M774 :
780   M775 :
781   M776 :
782   M777 :
783   M778 :
784   M779 :
785   M780 :
786   M781 :
787   M782 :
788   M783 :
789   M784 :
790   M785 :
791   M786 :
792   M787 :
793   M788 :
794   M789 :
795   M790 :
796   M791 :
797   M792 :
798   M793 :
799   M794 :
800   M795 :
801   M796 :
802   M797 :
803   M798 :
804   M799 :
805   M800 :
806   M801 :
807   M802 :
808   M803 :
809   M804 :
810   M805 :
811   M806 :
812   M807 :
813   M808 :
814   M809 :
815   M810 :
816   M811 :
817   M812 :
818   M813 :
819   M814 :
820   M815 :
821   M816 :
822   M817 :
823   M818 :
824   M819 :
825   M820 :
826   M821 :
827   M822 :
828   M823 :
829   M824 :
830   M825 :
831   M826 :
832   M827 :
833   M828 :
834   M829 :
835   M830 :
836   M831 :
837   M832 :
838   M833 :
839   M834 :
840   M835 :
841   M836 :
842   M837 :
843   M838 :
844   M839 :
845   M840 :
846   M841 :
847   M842 :
848   M843 :
849   M844 :
850   M845 :
851   M846 :
852   M847 :
853   M848 :
854   M849 :
855   M850 :
856   M851 :
857   M852 :
858   M853 :
859   M854 :
860   M855 :
861   M856 :
862   M857 :
863   M858 :
864   M859 :
865   M860 :
866   M861 :
867   M862 :
868   M863 :
869   M864 :
870   M865 :
871   M866 :
872   M867 :
873   M868 :
874   M869 :
875   M870 :
876   M871 :
877   M872 :
878   M873 :
879   M874 :
880   M875 :
881   M876 :
882   M877 :
883   M878 :
884   M879 :
885   M880 :
886   M881 :
887   M882 :
888   M883 :
889   M884 :
890   M885 :
891   M886 :
892   M887 :
893   M888 :
894   M889 :
895   M890 :
896   M891 :
897   M892 :
898   M893 :
899   M894 :
900   M895 :
901   M896 :
902   M897 :
903   M898 :
904   M899 :
905   M900 :
906   M901 :
907   M902 :
908   M903 :
909   M904 :
910   M905 :
911   M906 :
912   M907 :
913   M908 :
914   M909 :
915   M910 :
916   M911 :
917   M912 :
918   M913 :
919   M914 :
920   M915 :
921   M916 :
922   M917 :
923   M918 :
924   M919 :
925   M920 :
926   M921 :
927   M922 :
928   M923 :
929   M924 :
930   M925 :
931   M926 :
932   M927 :
933   M928 :
934   M929 :
935   M930 :
936   M931 :
937   M932 :
938   M933 :
939   M934 :
940   M935 :
941   M936 :
942   M937 :
943   M938 :
944   M939 :
945   M940 :
946   M941 :
947   M942 :
948   M943 :
949   M944 :
950   M945 :
951   M946 :
952   M947 :
953   M948 :
954   M949 :
955   M950 :
956   M951 :
957   M952 :
958   M953 :
959   M954 :
960   M955 :
961   M956 :
962   M957 :
963   M958 :
964   M959 :
965   M960 :
966   M961 :
967   M962 :
968   M963 :
969   M964 :
970   M965 :
971   M966 :
972   M967 :
973   M968 :
974   M969 :
975   M970 :
976   M971 :
977   M972 :
978   M973 :
979   M974 :
980   M975 :
981   M976 :
982   M977 :
983   M978 :
984   M979 :
985   M980 :
986   M981 :
987   M982 :
988   M983 :
989   M984 :
990   M985 :
991   M986 :
992   M987 :
993   M988 :
994   M989 :
995   M990 :
996   M991 :
997   M992 :
998   M993 :
999   M994 :
1000  M995 :
1001  M996 :
1002  M997 :
1003  M998 :
1004  M999 :
1005  M1000 :
1006  M1001 :
1007  M1002 :
1008  M1003 :
1009  M1004 :
1010  M1005 :
1011  M1006 :
1012  M1007 :
1013  M1008 :
1014  M1009 :
1015  M1010 :
1016  M1011 :
1017  M1012 :
1018  M1013 :
1019  M1014 :
1020  M1015 :
1021  M1016 :
1022  M1017 :
1023  M1018 :
1024  M1019 :
1025  M1020 :
1026  M1021 :
1027  M1022 :
1028  M1023 :
1029  M1024 :
1030  M1025 :
1031  M1026 :
1032  M1027 :
1033  M1028 :
1034  M1029 :
1035  M1030 :
1036  M1031 :
1037  M1032 :
1038  M1033 :
1039  M1034 :
1040  M1035 :
1041  M1036 :
1042  M1037 :
1043  M1038 :
1044  M1039 :
1045  M1040 :
1046  M1041 :
1047  M1042 :
1048  M1043 :
1049  M1044 :
1050  M1045 :
1051  M1046 :
1052  M1047 :
1053  M1048 :
1054  M1049 :
1055  M1050 :
1056  M1051 :
1057  M1052 :
1058  M1053 :
1059  M1054 :
1060  M1055 :
1061  M1056 :
1062  M1057 :
1063  M1058 :
1064  M1059 :
1065  M1060 :
1066  M1061 :
1067  M1062 :
1068  M1063 :
1069  M1064 :
1070  M1065 :
1071  M1066 :
1072  M1067 :
1073  M1068 :
1074  M1069 :
1075  M1070 :
1076  M1071 :
1077  M1072 :
1078  M1073 :
1079  M1074 :
1080  M1075 :
1081  M1076 :
1082  M1077 :
1083  M1078 :
1084  M1079 :
1085  M1080 :
1086  M1081 :
1087  M1082 :
1088  M1083 :
1089  M1084 :
1090  M1085 :
1091  M1086 :
1092  M1087 :
1093  M1088 :
1094  M1089 :
1095  M1090 :
1096  M1091 :
1097  M1092 :
1098  M1093 :
1099  M1094 :
1100  M1095 :
1101  M1096 :
1102  M1097 :
1103  M1098 :
1104  M1099 :
1105  M1100 :
1106  M1101 :
1107  M1102 :
1108  M1103 :
1109  M1104 :
1110  M1105 :
1111  M1106 :
1112  M1107 :
1113  M1108 :
1114  M1109 :
1115  M1110 :
1116  M1111 :
1117  M1112 :
1118  M1113 :
1119  M1114 :
1120  M1115 :
1121  M1116 :
1122  M1117 :
1123  M1118 :
1124  M1119 :
1125  M1120 :
1126  M1121 :
1127  M1122 :
1128  M1123 :
1129  M1124 :
1130  M1125 :
1131  M1126 :
1132  M1127 :
1133  M1128 :
1134  M1129 :
1135  M1130 :
1136  M1131 :
1137  M1132 :
1138  M1133 :
1139  M1134 :
1140  M1135 :
1141  M1136 :
1142  M1137 :
1143  M1138 :
1144  M1139 :
1145  M1140 :
1146  M1141 :
1147  M1142 :
1148  M1143 :
1149  M1144 :
1150  M1145 :
1151  M1146 :
1152  M1147 :
1153  M1148 :
1154  M1149 :
1155  M1150 :
1156  M1151 :
1157  M1152 :
1158  M1153 :
1159  M1154 :
1160  M1155 :
1161  M1156 :
1162  M1157 :
1163  M1158 :
1164  M1159 :
1165  M1160 :
1166  M1161 :
1167  M1162 :
1168  M1163 :
1169  M1164 :
1170  M1165 :
1171  M1166 :
1172  M1167 :
1173  M1168 :
1174  M1169 :
1175  M1170 :
1176  M1171 :
1177  M1172 :
1178  M1173 :
1179  M1174 :
1180  M1175 :
1181  M1176 :
1182  M1177 :
1183  M1178 :
1184  M1179 :
1185  M1180 :
1186  M1181 :
1187  M1182 :
1188  M1183 :
1189  M1184 :
1190  M1185 :
1191  M1186 :
1192  M1187 :
1193  M1188 :
1194  M1189 :
1195  M1190 :
1196  M1191 :
1197  M1192 :
1198  M1193 :
1199  M1194 :
1200  M1195 :
1201  M1196 :
1202  M1197 :
1203  M1198 :
1204  M1199 :
1205  M1200 :
1206  M1201 :
1207  M1202 :
1208  M1203 :
1209  M1204 :
12
```

In practice, it is sometimes required to access one RAM block from two different clock domains of a design. Such scenarios often arise in clock-domain-crossing interfaces, which are crucial in complex systems where different subsystems operate on distinct clock frequencies. These situations require, what is referred to as dual-clock RAM.

Dual-Clock RAM

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- Separate/independent clock for each port of a dual-port memory ⇒ dual-clock memory
- Use case: clock-domain-crossing interfaces

Most modern FPGA block RAMs provide built-in support for such dual-clock operations, making them well-suited for such scenarios.

Dual-Clock RAM

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- Separate/independent clock for each port of a dual-port memory ⇒ dual-clock memory
- Use case: clock-domain-crossing interfaces
- Usually supported by FPGA block RAM

However, a key consideration when using dual-clock memory is the potential issue of simultaneous read and write operations targeting the same memory location. Such conflicts must be handled carefully to avoid undefined or erroneous behavior. Implementing robust synchronization and arbitration mechanisms is crucial to ensure reliable operation in these scenarios.

Dual-Clock RAM

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- Separate/independent clock for each port of a dual-port memory ⇒ dual-clock memory
- Use case: clock-domain-crossing interfaces
- Usually supported by FPGA block RAM
- Simultaneous read and write to the same location must be handled carefully

Hence, dual-clock memory is often used in the form of dual-clock or bisynchronous FIFOs. Bisynchronous FIFOs are an important tool in a designer's toolbox and heavily used for clock-domain-crossing interfaces.

Dual-Clock RAM

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

- Separate/independent clock for each port of a dual-port memory ⇒ dual-clock memory
- Use case: clock-domain-crossing interfaces
- Usually supported by FPGA block RAM
- Simultaneous read and write to the same location must be handled carefully
- ⇒ dual-clocked/bisynchronous FIFOs

RAM in VHDL (for FPGAs)

Dual-Clock RAM

Dual-Clocked RAM - Example

```
1 entity dualclock_dp_ram is
2
3   generic (
4     ADDR_WIDTH : positive;
5     DATA_WIDTH : positive;
6   );
7
8   port (
9     -- read port
10    rd_clk : in std_ulogic;
11    rd_en  : in std_ulogic;
12    rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
13    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
14
15    -- write port
16    wr_clk : in std_ulogic;
17    wr_en  : in std_ulogic;
18    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
19    wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0);
20  );
21 end entity;
```

This slide shows how a dual-clock version of the simple dual-port RAM from before can look like.

Dual-Clocked RAM - Example

```
1 entity dualclock_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive
5   );
6   port (
7     -- read port
8     rd_clk : in std_ulogic;
9     rd_en  : in std_ulogic;
10    rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
11    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
12    -- write port
13    wr_clk : in std_ulogic;
14    wr_en  : in std_ulogic;
15    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
16    wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0)
17  );
18 end entity;
```

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

Dual-Clock RAM

Dual-Clocked RAM - Example

```
1 entity dualclock_dp_ram is
2
3   generic
4     ADDR_WIDTH : positive;
5     DATA_WIDTH : positive;
6   );
7   port
8     rd_clk : in std_ulogic;
9     rd_en : in std_ulogic;
10    rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
11    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
12    -- write port
13    wr_clk : in std_ulogic;
14    wr_en : in std_ulogic;
15    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
16    wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0);
17  );
18 end entity;
```

Notice that instead of one clock input, now both the read and the write port have their own separate clock signals.

Dual-Clocked RAM - Example

```
1 entity dualclock_dp_ram is
2   generic (
3     ADDR_WIDTH : positive;
4     DATA_WIDTH : positive
5   );
6   port (
7     -- read port
8     rd_clk : in std_ulogic;
9     rd_en : in std_ulogic;
10    rd_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
11    rd_data : out std_ulogic_vector(DATA_WIDTH - 1 downto 0);
12    -- write port
13    wr_clk : in std_ulogic;
14    wr_en : in std_ulogic;
15    wr_addr : in std_ulogic_vector(ADDR_WIDTH - 1 downto 0);
16    wr_data : in std_ulogic_vector(DATA_WIDTH - 1 downto 0)
17  );
18 end entity;
```

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

Now that we know how RAM interfaces look like, and how to operate them, the question remains how they are actually implemented in VHDL. For RAM in FPGA design, there are two main approaches: instantiating RAM from a vendor library or letting the synthesis tool infer RAM based on the VHDL description. Each approach has its advantages and disadvantages.

RAM Implementation

HWMoD
WS25

RAM

- Introduction
- FPGA Memory
- Simple DP RAM
- True DP RAM
- Dual-Clocked RAM
- Implementation**

Vendor-Library Instantiation

Synthesis Inference

RAM in VHDL (for FPGAs)

Implementation

RAM Implementation

	Vendor-Library Instantiation	Synthesis Inference
Portability/Flexibility	Limited to vendor/device, requires vendor-specific knowledge	Generic and portable, simpler to work with and maintain

Designs which use vendor library components are inherently limited to specific devices or families of devices, which in turn makes these designs less portable. It also requires some level of familiarity with vendor-specific libraries and their configuration options. Furthermore, models for the library components are required for simulation, which can make things more complicated when multiple different simulation tools are used. Inferring memory, directly from VHDL code does not come with these issues and code relying on it is, thus, generally more portable and easier to maintain. It also allows to – for example – describe highly specialized memories that store values of some custom type.

RAM Implementation

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

	Vendor-Library Instantiation	Synthesis Inference
Portability/Flexibility	Limited to vendor/device, requires vendor-specific knowledge	Generic and portable, simpler to work with and maintain

RAM in VHDL (for FPGAs)

Implementation

RAM Implementation

	Vendor-Library Instantiation	Synthesis Inference
Portability/Flexibility	Limited to vendor/device, requires vendor-specific knowledge	Generic and portable, simpler to work with and maintain
Performance	Optimized for target hardware	May be suboptimal

On the other hand, instantiating library components can offer better performance in certain situations, as the designer has more direct control over the generated hardware. It also makes it possible to force the synthesis tool to use certain specific hardware resources in an FPGA. For the inference of memories during synthesis one has to be careful. When the description does not match the structure required for the available memory blocks, the performance can be severely impeded. However, there is no guarantee that this always leads to a performance hit or otherwise problematic hardware. Therefore, it is always a good idea to check if the hardware generated by the synthesis tool matches the memory you believe you described.

RAM Implementation

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

	Vendor-Library Instantiation	Synthesis Inference
Portability/Flexibility	Limited to vendor/device, requires vendor-specific knowledge	Generic and portable, simpler to work with and maintain
Performance	Optimized for target hardware	May be suboptimal

RAM in VHDL (for FPGAs)

Implementation

RAM Implementation

	Vendor-Library Instantiation	Synthesis Inference
Portability/Flexibility	Limited to vendor/device, requires vendor-specific knowledge	Generic and portable, simpler to work with and maintain
Performance	Optimized for target hardware	May be suboptimal
Special Features	Fully supported (e.g., ECC)	May be unavailable

Furthermore, in some cases there might be no other way than instantiating library components, as it might be the only way to access certain special hardware features, like error-detection.

RAM Implementation

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

	Vendor-Library Instantiation	Synthesis Inference
Portability/Flexibility	Limited to vendor/device, requires vendor-specific knowledge	Generic and portable, simpler to work with and maintain
Performance	Optimized for target hardware	May be suboptimal
Special Features	Fully supported (e.g., ECC)	May be unavailable

RAM in VHDL (for FPGAs)

Implementation

RAM Implementation

	Vendor-Library Instantiation	Synthesis Inference
Portability/Flexibility	Limited to vendor/device, requires vendor-specific knowledge	Generic and portable, simpler to work with and maintain
Performance	Optimized for target hardware	May be suboptimal
Special Features	Fully supported (e.g., ECC)	May be unavailable
Predictability	Deterministic	May cause mismatch in synthesis/simulation tool

Finally, predictability is a significant factor as well. Synthesis inference can sometimes lead to mismatches between synthesis and simulation tools, which may introduce challenges during verification. Specifically in edge cases, where the same memory location is read and written at the same time.

RAM Implementation

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

	Vendor-Library Instantiation	Synthesis Inference
Portability/Flexibility	Limited to vendor/device, requires vendor-specific knowledge	Generic and portable, simpler to work with and maintain
Performance	Optimized for target hardware	May be suboptimal
Special Features	Fully supported (e.g., ECC)	May be unavailable
Predictability	Deterministic	May cause mismatch in synthesis/simulation tool

RAM in VHDL (for FPGAs)

Implementation

Inferred RAM

```
1 architecture beh of simple_dp_ram is
2   subtype ram_entry_t is std_ulogic_vector(DATA_WIDTH - 1 downto 0);
3   type ram_t is array(0 to (2 ** ADDR_WIDTH) - 1) of ram_entry_t;
4   signal ram : ram_t := (others => (others => '0'));
5 begin
6   process (clk)
7   begin
8     if rising_edge(clk) then
9       if wr_en = '1' then
10        ram(to_integer(unsigned(wr_addr))) <= wr_data;
11      end if;
12      if rd_en = '1' then
13        rd_data <= ram(to_integer(unsigned(rd_addr)));
14      end if;
15    end if;
16  end process;
17 end architecture;
```

Finally, let's look at some VHDL code that can be used to infer RAM. Here, we see a possible architecture implementation for the simple dual-port RAM.

Inferred RAM

```
1 architecture beh of simple_dp_ram is
2   subtype ram_entry_t is std_ulogic_vector(DATA_WIDTH - 1 downto 0);
3   type ram_t is array(0 to (2 ** ADDR_WIDTH) - 1) of ram_entry_t;
4   signal ram : ram_t := (others => (others => '0'));
5 begin
6   process (clk)
7   begin
8     if rising_edge(clk) then
9       if wr_en = '1' then
10        ram(to_integer(unsigned(wr_addr))) <= wr_data;
11      end if;
12      if rd_en = '1' then
13        rd_data <= ram(to_integer(unsigned(rd_addr)));
14      end if;
15    end if;
16  end process;
17 end architecture;
```

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

Implementation

Inferred RAM

```
1 architecture beh of simple_dp_ram is
2   subtype ram_entry_t is std_ulogic_vector(DATA_WIDTH - 1 downto 0);
3   type ram_t is array(0 to (2 ** ADDR_WIDTH) - 1) of ram_entry_t;
4   signal ram : ram_t := (others => (others => '0'));
5 begin
6   if rd_en = '1' then
7     ram(to_integer(unsigned(rd_addr))) <= wr_data;
8   end if;
9   if rd_en = '1' then
10    rd_data <= ram(to_integer(unsigned(rd_addr)));
11  end if;
12 end process;
13 end architecture;
```

Use parentheses and signal reference for the signal that actually represents the memory array.

First some types and the signal that actually represents the memory array are declared.

Inferred RAM

```
1 architecture beh of simple_dp_ram is
2   subtype ram_entry_t is std_ulogic_vector(DATA_WIDTH - 1 downto 0);
3   type ram_t is array(0 to (2 ** ADDR_WIDTH) - 1) of ram_entry_t;
4   signal ram : ram_t := (others => (others => '0'));
5 begin
6   process (clk)
7   begin
8     if rising_edge(clk) then
9       if wr_en = '1' then
10        ram(to_integer(unsigned(wr_addr))) <= wr_data;
11      end if;
12      if rd_en = '1' then
13        rd_data <= ram(to_integer(unsigned(rd_addr)));
14      end if;
15    end if;
16  end process;
17 end architecture;
```

Type definitions and signal declaration for the signal that actually represents the memory array.

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

Implementation

Inferred RAM

```
1 architecture beh of simple_dp_ram is
2   subtype ram_entry_t is std_ulogic_vector(DATA_WIDTH - 1 downto 0);
3   type ram_t is array(0 to (2 ** ADDR_WIDTH) - 1) of ram_entry_t;
4   signal ram : ram_t := (others => (others => '0'));
5 begin
6   process (clk)
7   begin
8     if rising_edge(clk) then
9       if wr_en = '1' then
10        ram(to_integer(unsigned(wr_addr))) <= wr_data;
11      end if;
12      if rd_en = '1' then
13        rd_data <= ram(to_integer(unsigned(rd_addr)));
14      end if;
15    end if;
16  end process;
17 end architecture;
```

Synchronous process similar to what is used in the description of flip-flops.

Since we are describing synchronous RAM, we need a process whose body is only executed on rising clock edges.

Inferred RAM

```
1 architecture beh of simple_dp_ram is
2   subtype ram_entry_t is std_ulogic_vector(DATA_WIDTH - 1 downto 0);
3   type ram_t is array(0 to (2 ** ADDR_WIDTH) - 1) of ram_entry_t;
4   signal ram : ram_t := (others => (others => '0'));
5 begin
6   process (clk)
7   begin
8     if rising_edge(clk) then
9       if wr_en = '1' then
10        ram(to_integer(unsigned(wr_addr))) <= wr_data;
11      end if;
12      if rd_en = '1' then
13        rd_data <= ram(to_integer(unsigned(rd_addr)));
14      end if;
15    end if;
16  end process;
17 end architecture;
```

Synchronous process similar to what is used in the description of flip-flops.

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

RAM in VHDL (for FPGAs)

Implementation

Inferred RAM

```
1 architecture beh of simple_dp_ram is
2   subtype ram_entry_t is std_ulogic_vector(DATA_WIDTH - 1 downto 0);
3   type ram_t is array(0 to (2 ** ADDR_WIDTH) - 1) of ram_entry_t;
4   signal ram : ram_t := (others => (others => '0'));
5 begin
6   if rd_en = '1' then
7     ram(to_integer(unsigned(rd_addr))) <= wr_data;
8   end if;
9   if rd_en = '1' then
10    rd_data <= ram(to_integer(unsigned(rd_addr)));
11  end if;
12 end process;
13 end architecture;
```

In the process body the write-port is implemented by first checking the write-enable signal. If this signal is asserted, the address is used to index the memory array.

Inferred RAM

HWMoD
WS25

RAM
Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

```
1 architecture beh of simple_dp_ram is
2   subtype ram_entry_t is std_ulogic_vector(DATA_WIDTH - 1 downto 0);
3   type ram_t is array(0 to (2 ** ADDR_WIDTH) - 1) of ram_entry_t;
4   signal ram : ram_t := (others => (others => '0'));
5 begin
6   process (clk)
7   begin
8     if rising_edge(clk) then
9       if wr_en = '1' then
10        ram(to_integer(unsigned(wr_addr))) <= wr_data;
11      end if;
12      if rd_en = '1' then
13        rd_data <= ram(to_integer(unsigned(rd_addr)));
14      end if;
15    end if;
16  end process;
17 end architecture;
```

Implementation of the write port.

Thank you for listening! We recommend you to immediately take the self-check test in TUWEL, to see if you understood the material presented in this lecture.

HWMoD
WS25

RAM

Introduction
FPGA Memory
Simple DP RAM
True DP RAM
Dual-Clocked RAM
Implementation

Lecture Complete!