

KR and Exercises: Block Memory

Implementations in SRAM-based FPGAs

FIFO Implementation

Question: How can FIFOs be implemented in SRAM based FPGAs? (Several Answers may be correct)

- FPGA memory blocks can be used for FIFOs ✓
- The D-Flip-Flops in the Logic Elements may be used for FIFOs ✓
- The Cyclone V SoC FPGA has hard macros exclusively made for FIFOs X
- The external configuration flash may be used for FIFOs X

Explanation:

A FIFO can be implemented using flip flops if you only need a small FIFO. As soon as you need more than a few hundred bits of storage, using block memories is more efficient.

The external configuration flash memory is not suitable for FIFO usage as it is slow and has a limited number of write cycles.

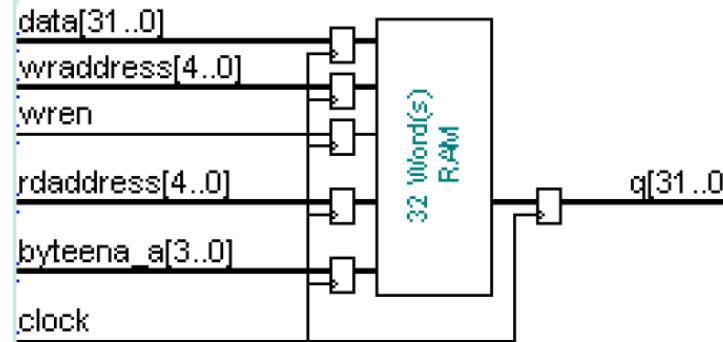
There are no hard macros (silicon) that can only be used as a FIFO, since a block memory with some added logic is just as efficient and can also be used for other purposes. This makes more versatile use of the silicon area.

Problem type: Application - Understanding FPGA memory resources and FIFO implementation options

Source: Moodle Quiz Block Memory, Question 1

Block Memory Architecture

Block Memory Circuit Analysis



Question: Which answers are correct for the circuit shown (several correct answers are possible)?

- In all cases, all 32 bits of the 'data' signal are written to the memory at once. X
- Write data, wraddress, rdaddress, wren, q, byteena_a are in the same clock domain. ✓
- The 'data' signal may be in a different clock domain than the output 'q'. X
- It is possible to write individual 8-bit bytes to this 32-bit wide memory. ✓

Explanation:

The byte enable signals allow the individual writing of 8-bit bytes to this 32-bit wide memory.

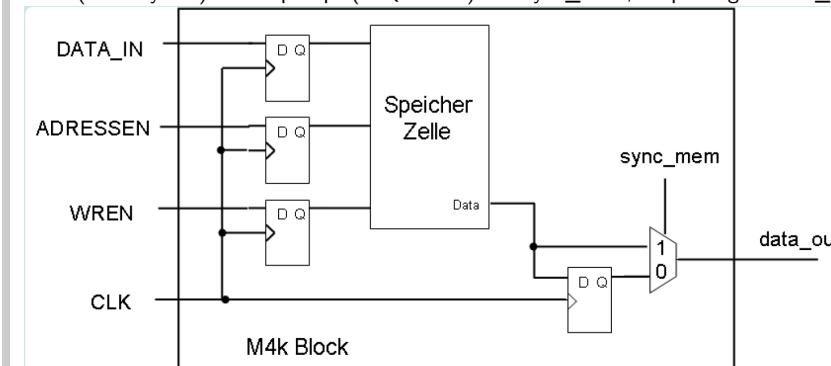
In the schematic shown, all control, address and data ('data' and 'q') signals are registered in flip flops that are connected to the same clock signal.

Problem type: Analysis - Circuit interpretation and signal timing analysis

Source: Moodle Quiz Block Memory, Question 2

M4k Block Memory Architecture

Description: M4k Block diagram showing DATA_IN, ADRESSEN, WREN, CLK signals connected to a Speicher Zelle (memory cell) with flip-flops (D Q blocks) and sync_mem, outputting to data_out



Question:

Which answers are correct for the block memory (M4k) shown (several correct answers are possible)?

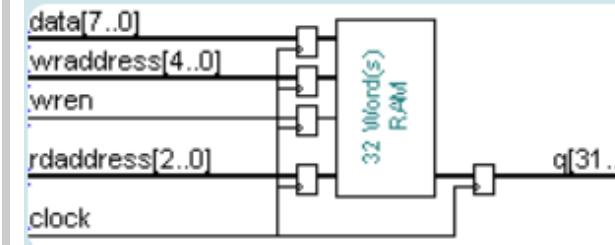
Select one or more answers:

- If the Flip-flop in the data out path is bypassed, the processor may require less wait states ✓
- Engaging the flip-flop of the data out path allows higher clock frequencies compared to when the Flip Flop is bypassed. ✓
- Engaging the Flip-flop of the data out path increases in all cases the performance of the connected processor X
- The Flip-flop in the data out path reduces memory access bandwidth X

Problem type: Analysis - Memory architecture trade-offs between latency and clock frequency

Source: Moodle Quiz Block Memory, Question 3

Block Memory Width and Addressing



Question:

Which answers are correct for the graphic above (several correct answers are possible)?

- The memory is 64 bytes deep (total amount of storage). X
- 32-bit values can be read at the same time ✓
- 8-bit values can be written at the same time ✓
- The memory is 256 bytes deep (total amount of storage). X

Explanation:

The 'data' signal shows the width of writing to the memory, the 'q' signal shows reading width.

The writing address width allows to determine the number of words in the memory: $2^5 = 32$ words. With its writing width of 8-bit, the memory can store 32 bytes.

Problem type: Analysis - Circuit interpretation and Memory Width

Source: Moodle Quiz Block Memory, Question 4

Block Memory Functionality

Purpose of FPGA Block RAM

Question:

What is the purpose of FPGA Block RAM (not LUT in logic cells):

- Implementation of combinatorial logic **X**
- For permanent storage of the FPGA configuration **X**
- For implementation of FIFO in FPGA **✓**
- For implementation of ROM in FPGA **✓**

Explanation:

Since the block memory is in the FPGA fabric, it can not be accessed before the fabric is configured. This prevents the use of block memory as storage for the FPGA bitstream (the configuration). Also, block memories use SRAM cells: they lose their memory content as soon as power supply is interrupted.

Block memories can be used as RAM, ROM or FIFO. The use of block memories as register banks is no different from the use as RAM: accessing a register by an address corresponds to accessing RAM.

The use as combinatorial logic cells (analogous to the use of LUT to implement combinatorial logic) is feasible but inefficient. Block memories are slower than LUT and too large for common combinatorial logic functions.

Problem type: Purpose and functionality of block memory

Source: Moodle Quiz Block Memory, Question 5

Byte Enable Functionality

Question:

What is the purpose of byte enables in block memories?

- Byte enables allow the possibility writing misaligned words in a byte processor system **✓**
- Reading memory in burst mode **X**
- If you want to connect an 8-bit page memory with 32-bit soft processor core **X**
- If individual bytes of a 32-bit wide memory should be written individually **✓**

Explanation:

The byte_enable signal allows to write individual bytes to a memory with larger word size. For instance, a 32-bit wide memory would have four byte_enable signals to select which of the four bytes on the bus need to be stored and which don't.

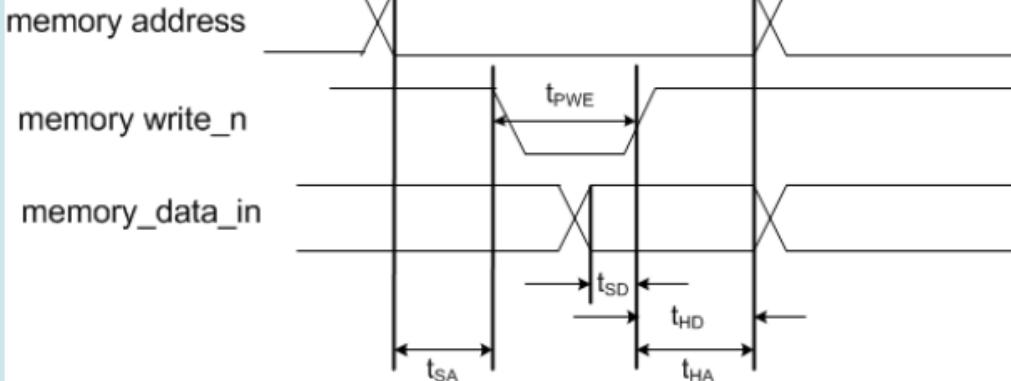
This allows writing smaller data types to the memory.

Problem type: Explanation - Understanding byte enable functionality in memory systems

Source: Moodle Quiz Block Memory, Question 6

Memory Timing

Asynchronous Memory Write Cycle Timing Parameters



Question:

For the given timing parameters for a write cycle, select the correct explanation:

- tPWE: The minimum duration the signal must be active to insure data is written properly
- tHA: The time address must be kept after writing the memory
- tDH: The time data must be kept after writing the memory
- tSA: The time address must be stable before writing the memory
- tSD: The time data must be stable before writing the memory

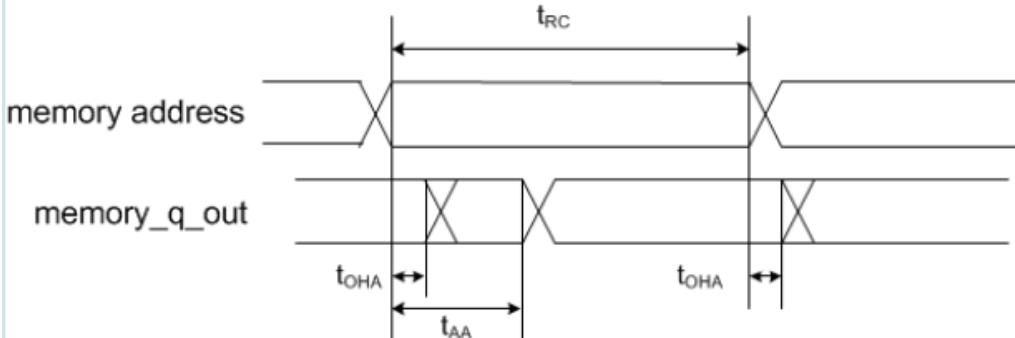
Explanation:

- tSA: address setup time, the address signals must be valid and stable for a certain time before the writeenable signal may be asserted.
- tHA: address hold time, the address signals must be kept (held) stable for a certain time after the writeenable signal is deasserted.
- tDS: data setup time, the data signals must be valid and stable for a certain time before the writeenable signal may be asserted.
- tDH: data hold time, the data signals must be kept (held) stable for a certain time after the writeenable signal is deasserted. Data will be stored at the rising edge of the writeenable signal

Problem type: Explanation - Memory timing parameter definitions and constraints

Source: Moodle Quiz Block Memory, Question 7

Asynchronous Memory Access



Question:

For the three given timing parameters, select the correct explanation:

Correct Answers:

- tOHA: If subsequent read cycles follow in too short distances, data outputs are never stable.
- tAA: If processor clocks data in too early after address is valid, data may be wrong.
- tRC: If processor removes address too early, read data already changed.

Explanation:

- tOHA 'time output hold address': defines how long the output data is still valid after the change of the address signals.
- tAA 'address to access time': defines how long the memory block needs to provide valid output data. If data is read too early, it may not be valid.
- tRC 'read cycle': defines the minimum time for a read cycle.

Problem type: Explanation - Memory timing parameter definitions and constraints

Source: Moodle Quiz Block Memory, Question 8

TODO:

- add KR on Memory Timing
- add KR that lists and explains any and all abbreviations plus an explanation that might occur in an exercise like this (see lectures and script to find the complete list)

Exercise: HPS and SoC

Quiz Format: Multiple choice **Duration:** 9 minutes 23 seconds **Score:** 6.00 von 8.00 (75%) **Topics Covered:** HPS architecture, SoC FPGA, custom peripherals, software execution

Custom Peripheral Integration

Multiple Choice - Custom Peripheral Workflow

Source: Exercise HPS and SoC, Question 1

Topics: HPS, FPGA fabric, custom peripherals, design workflow

Question:

Which step comes first when adding a custom peripheral to the FPGA fabric for HPS access?

Select one answer:

- Configuring the HPS in Platform Designer
- Programming the device with Quartus
- Writing a Linux device driver
- Designing the peripheral in VHDL or Verilog

Correct Answer: Designing the peripheral in VHDL or Verilog

Explanation:

Die Antwort ist falsch. (The answer given was false.)

Die richtige Antwort ist: Designing the peripheral in VHDL or Verilog

The workflow for adding a custom peripheral to the FPGA fabric that can be accessed by the HPS is:

1. First, design the peripheral in VHDL or Verilog
2. Then configure the HPS and peripheral connections in Platform Designer
3. Program the device with Quartus
4. Finally, write a Linux device driver to interface with the peripheral

Problem type: Design - Understanding HPS peripheral integration workflow

Software Execution Component

Multiple Choice - SoC FPGA Components

Source: Exercise HPS and SoC, Question 2

Topics: SoC FPGA, HPS, operating systems, software applications

Question:

In a SoC FPGA, which component is responsible for running operating systems and complex software applications?

Select one answer:

- Hard processor system (HPS)
- FPGA fabric
- DMA controller
- Bus bridge

Correct Answer: Hard processor system (HPS)

Explanation:

Die Antwort ist richtig. (The answer is correct.)

Die richtige Antwort ist: Hard processor system (HPS)

The hard processor system (HPS) is a dedicated ARM-based processor integrated into the SoC FPGA. It is specifically designed to run operating systems like Linux and complex software applications. The FPGA fabric is used for custom hardware logic, the DMA controller handles data transfers, and the bus bridge provides connectivity between components.

Problem type: Explanation - Understanding SoC FPGA architecture components

HPS Advantages in SoC

Multiple Choice - HPS Benefits

Source: Exercise HPS and SoC, Question 3

Topics: HPS, performance, power consumption, softcore processors

Question:

What is the main advantage of using a hard processor system (HPS) in an SoC FPGA?

Select one answer:

- It allows for reconfiguration of the processor at runtime
- It enables direct analog signal processing
- It eliminates the need for any memory controllers
- It provides higher performance and lower power consumption compared to softcore processors

Correct Answer: It provides higher performance and lower power consumption compared to softcore processors

Explanation:

Die Antwort ist richtig. (The answer is correct.)

Die richtige Antwort ist: It provides higher performance and lower power consumption compared to softcore processors

Hard processor systems (HPS) are implemented as dedicated silicon (ASIC) rather than using FPGA fabric resources. This provides several advantages:

- Higher performance - dedicated processor cores run faster than softcore implementations
- Lower power consumption - ASIC implementation is more power-efficient than FPGA logic
- Lower resource usage - frees up FPGA fabric for other custom logic

The HPS cannot be reconfigured at runtime (it's fixed silicon), does not directly process analog signals (requires ADC/DAC), and does not eliminate the need for memory controllers (though it has integrated controllers).

Problem type: Comparison - HPS vs. softcore processor trade-offs

SoC FPGA Architecture

Multiple Choice - SoC Architecture Definition

Source: Exercise HPS and SoC, Question 4

Topics: SoC FPGA, architecture, CPU, integration

Question:

Which of the following best describes a System-on-Chip (SoC) FPGA architecture?

Select one answer:

- Multiple CPUs with no programmable logic
- An FPGA with only softcore processors
- A CPU and FPGA fabric integrated on a single chip
- A CPU and FPGA on separate chips connected by a PCB

Correct Answer: A CPU and FPGA fabric integrated on a single chip

Explanation:

Die Antwort ist richtig. (The answer is correct.)

Die richtige Antwort ist: A CPU and FPGA fabric integrated on a single chip

A System-on-Chip (SoC) FPGA combines:

- A hard processor system (CPU) - typically ARM-based
- FPGA programmable logic fabric
- Memory controllers and peripherals
- High-speed interconnects

All integrated on a single silicon die. This integration provides benefits including:

- Lower latency communication between CPU and FPGA
- Reduced board space and power consumption
- Shared memory interfaces
- Better system performance

This is different from separate CPU and FPGA chips connected via PCB, or FPGAs with only softcore processors implemented in fabric.

Problem type: Definition - Understanding SoC FPGA architecture fundamentals

KR and Exercises: HPS and SoC

System on Chip Custom Peripheral

Custom Peripheral Development

Question: Which step comes first when adding a custom peripheral to the FPGA fabric for HPS access?

Wählen Sie eine Antwort:

- Configuring the HPS in Platform Designer **X**
- Programming the device with Quartus
- Writing a Linux device driver
- Designing the peripheral in VHDL or Verilog **✓**

Explanation:

Die Antwort ist falsch.

Die richtige Antwort ist: Designing the peripheral in VHDL or Verilog

Problem type: Development workflow - Understanding the correct sequence for custom peripheral integration

Source: Moodle Quiz HPS and SoC, Question 1

System on Chip Software Execution

SoC Component Responsibility

Question: In a SoC FPGA, which component is responsible for running operating systems and complex software applications?

Wählen Sie eine Antwort:

- Hard processor system (HPS) **✓**
- FPGA fabric **X**
- DMA controller **X**
- Bus bridge **X**

Explanation:

Die Antwort ist richtig.

Die richtige Antwort ist: Hard processor system (HPS)

Problem type: Architecture - Understanding SoC FPGA component roles

Source: Moodle Quiz HPS and SoC, Question 2

System on Chip Advantages

HPS Performance Benefits

Question: What is the main advantage of using a hard processor system (HPS) in an SoC FPGA?

Wählen Sie eine Antwort:

- It allows for reconfiguration of the processor at runtime **X**
- It enables direct analog signal processing **X**
- It eliminates the need for any memory controllers **X**
- It provides higher performance and lower power consumption compared to softcore processors **✓**

Explanation:

Die Antwort ist richtig.

Die richtige Antwort ist: It provides higher performance and lower power consumption compared to softcore processors

Problem type: Architecture - Understanding HPS advantages over softcore processors

Source: Moodle Quiz HPS and SoC, Question 3

System on Chip Architecture

SoC FPGA Definition

Question: Which of the following best describes a System-on-Chip (SoC) FPGA architecture?

Wählen Sie eine Antwort:

- Multiple CPUs with no programmable logic **X**
- An FPGA with only softcore processors **X**
- A CPU and FPGA fabric integrated on a single chip **✓**
- A CPU and FPGA on separate chips connected by a PCB **X**

Explanation:

Die Antwort ist richtig.

Die richtige Antwort ist: A CPU and FPGA fabric integrated on a single chip

Problem type: Definition - Understanding SoC FPGA architecture

Source: Moodle Quiz HPS and SoC, Question 4

Exercise: Booting and Device Tree

Quiz Format: Multiple choice **Duration:** 5 minutes 49 seconds **Score:** 2.00 von 10.00 (20%) **Topics Covered:** Boot process, Boot ROM, Device Tree, SD card access, DDR SDRAM configuration

Boot Process Sequence

Multiple Choice - Boot ROM Functions

Source: Exercise Booting and Device Tree, Question 1

Topics: Boot process, Boot ROM, initialization, SD card

Question:

During the boot process, a sequence of several steps is followed. One of them is executed from the Boot ROM. Of the following statements, check all that are true.

Select one or more answers:

- The boot ROM loads the first user-definable program into on-chip RAM
- The Boot ROM sets up access to the SD card.
- The Boot ROM is the first step that can be reprogrammed by the user.
- The Boot ROM configures the external DDR SDRAM device.

Correct Answers:

- The Boot ROM sets up access to the SD card.
- The boot ROM loads the first user-definable program into on-chip RAM

Explanation:

Your answer is incorrect.

Die richtigen Antworten sind: (The correct answers are:)

- The Boot ROM sets up access to the SD card.
- The boot ROM loads the first user-definable program into on-chip RAM

The Boot ROM is executed immediately after power-on or reset. It is a fixed program stored in on-chip ROM that cannot be modified by the user. The Boot ROM performs the following functions:

1. Initializes basic system components
2. Sets up access to boot devices (SD card, QSPI flash, NAND flash, etc.)
3. Loads the first user-definable program (typically the preloader or SPL) from the boot device into on-chip RAM
4. Transfers control to the loaded program

The Boot ROM does NOT configure the external DDR SDRAM - this is done by the preloader/SPL that is loaded by the Boot ROM.

The Boot ROM itself cannot be reprogrammed as it is fixed in silicon.

Problem type: Explanation - Understanding boot sequence and Boot ROM functionality

Device Tree Purpose

Multiple Choice - Device Tree Functionality

Source: Exercise Booting and Device Tree, Question 2

Topics: Device Tree, hardware description, Linux kernel, peripherals

Question:

What is the primary purpose of the Device Tree in embedded Linux systems?

Select one answer:

- To store user applications and data
- To describe hardware configuration and peripherals to the kernel
- To provide a graphical user interface
- To manage network connections

Correct Answer: To describe hardware configuration and peripherals to the kernel

Explanation:

The Device Tree is a data structure that describes the hardware configuration of the system to the operating system kernel. It provides information about:

- Available hardware devices and peripherals
- Memory-mapped register addresses
- Interrupt lines
- Clock connections
- Pin configurations
- Device-specific properties

The Device Tree allows the same kernel binary to run on different hardware configurations by providing the hardware-specific information at boot time. This separates hardware description from the kernel code, making the kernel more portable and easier to maintain.

The Device Tree does not store user applications, provide GUI functionality, or manage network connections - these are handled by other parts of the system.

Problem type: Definition - Understanding Device Tree purpose and functionality

KR and Exercises: Booting and Device Tree

Boot Process

Boot ROM Functions

Question: During the boot process, a sequence of several steps is followed. One of them is executed from the Boot ROM.

Of the following statements, check all that are true.

Wählen Sie eine oder mehrere Antworten:

- The boot ROM loads the first user-definable program into on-chip RAM **X**
- The Boot ROM sets up access to the SD card. **✓**
- The Boot ROM is the first step that can be reprogrammed by the user. **X**
- The Boot ROM configures the external DDR SDRAM device. **✓**

Explanation:

Your answer is incorrect.

Die richtigen Antworten sind: The Boot ROM sets up access to the SD card.,, The boot ROM loads the first user-definable program into on-chip RAM

Problem type: Understanding boot sequence and Boot ROM responsibilities

Source: Moodle Quiz Booting and Device Tree, Question 1

Bootloader Functions

Question: During the boot process, a sequence of several steps is followed. One of them is called the bootloader, often it is the open source 'u-boot' program.

Of the following statements, check all that are true.

Wählen Sie eine oder mehrere Antworten:

- The FPGA fabric must be configured before the bootloader is executed. **✓**
- The bootloader can be used to access the FPGA fabric via the h2f_axi_bridge. **✓**
- The bootloader is the first software program that is executed from the SDRAM. **X**
- The bootloader offers no interaction with the user. **X**

Explanation:

Your answer is incorrect.

Die richtigen Antworten sind: The bootloader is the first software program that is executed from the SDRAM.,, The bootloader can be used to access the FPGA fabric via the h2f_axi_bridge.

Problem type: Understanding bootloader role and capabilities

Source: Moodle Quiz Booting and Device Tree, Question 2

Device Tree

Device Tree Address Lookup

Question: The device tree is a hierarchical representation of the programmed logic and hardware as seen on a memory map. It is used to pass information about the hardware to the linux kernel, e.g. which driver to use and at what address a certain device can be accessed.

In the following image, a list of memory ranges on a bus is given as it appears in the device tree source. Which entry or entries define the address to look for when trying to identify a component in the file system representation of the device tree (such as in /sys/class/gpio)

Antwort: _____

Explanation:

Die richtige Antwort ist: A&B

Problem type: Device tree interpretation and address mapping

Source: Moodle Quiz Booting and Device Tree, Question 3

Exercise: SDRAM

Quiz Format: Mixed (calculation, multiple choice, open questions) **Score:** 7.00 von 7.00 (100%) **Topics Covered:** SDRAM architecture, timing diagrams, burst read, DDR3-1600, signal sampling, access latency, bandwidth calculations, address multiplexing, bank interleaving, dynamic refresh, row/column selection, sense amplifiers, SRAM vs DRAM

SDRAM Architecture Analysis

Calculation - SDRAM Device Architecture

Source: Exercise SDRAM, Question 1 (Page 1-2)

Topics: SDRAM architecture, address width, data bus, memory capacity

Question:

The following functional block diagram shows an SDRAM device that could be attached to a Cyclone V SoC. From the diagram, deduct the answers to the questions below.

Answer the following questions about the device's architecture. You are allowed to use a calculator if required.

- Specify the width of the **address port A[...]**:

Answer format: A[4:0]

- Specify the width of the **bank address port BA[...]**:

Answer format: BA[4:0]

- How wide is the **data bus** of this SDRAM device?

- How many **rows** does the SDRAM have **per bank**?

- How many bits does the column address need?

- How many **words** are stored **per row**?

- How many **Megabits** can be stored in the memory device (Keep in mind: 1 Megabit = 1024*1024 bit)?

Solutions:

- Address port width: A[12:0]

From the diagram, the Row Address Latch shows input of 13 address bits.

- Bank address port width: BA[1:0]

The diagram shows BANK CONTROL LOGIC managing multiple banks, with 2 bank address bits.

- Data bus width: 32 bit

The data buffers show 32-bit wide data path (I/O 0-31, with DM0-DM3 for byte enables and DQ0-DQS3 for data strobes).

- Rows per bank: 8192 rows

With 13 address bits for row addressing: $2^{13} = 8192$ rows per bank.

- Column address bits: 9

From the Column Address Latch and Buffer shown in the diagram, 9 bits are used for column addressing.

- Words per row: 512 words

With 9 column address bits: $2^9 = 512$ words per row.

- Memory capacity: 512 Mbit

Calculation:

- Address Port width: deduce from width of input signal into Row Address Latch
- Bank Address width: deduce from number of banks, or from signal width into bank control logic
- Data Bus Width: deduce from data in buffer or from 'I/O 0-31'
- Rows per Bank: use Address width of Row decoder
- Column Address Width: check input width for column decoder
- Word per row: deduced from column address width
- Capacity in Megabits: Rows * columns * word width * banks / 1024^2

Total capacity = $8192 \times 512 \times 32 \times 2 / (1024 \times 1024) = 512$ Mbit

Explanation hints:

- Address Port width: deduce from width of input signal into Row Address Latch
- Bank Address width: deduce from number of banks, or from signal width into bank control logic
- Data Bus Width: deduce from data in buffer or from 'I/O 0-31'
- Rows per Bank: use Address width of Row decoder
- Column Address Width: check input width for column decoder
- Word per row: deduced from column address width
- Capacity in Megabits: Rows * columns * word width * banks / 1024^2

Problem type: Calculation - SDRAM architecture analysis from block diagram

SDRAM Burst Read Timing

Calculation - Burst Read Timing Analysis

Source: Exercise SDRAM, Question 2 (Pages 3-4)

Topics: Burst read, timing diagrams, CAS latency, setup/hold times

Question:

This is the timing diagram of a burst read access in an **SDRAM device** (single data rate), depicting a burst read access to a continuous sequence of words.

Below, you are given some timing parameters from the corresponding data sheet. From the timing diagram and the timing parameters, find the following timings:

1. Hold time for command:

2. Setup time for row address:

3. Minimal time between column address and data output:

Hint: find number of clock cycles and deduce from there.

Solutions:

1. Hold time for command: 0.8 ns

From the timing parameters table, tCMH (Command Hold Time) = 0.8 ns minimum.

2. Setup time for row address: 1.5rf ns

From the timing parameters table, tAS (Address Setup Time) = 1.5 ns minimum.

3. Minimal time between column address and data output: 15 ns

From the timing diagram, CAS Latency = 3 clock cycles. From the parameters, with CAS Latency = 3:

- tCK3 (Clock Cycle Time for CAS Latency = 3) = 5 ns minimum
- Time = 3 cycles \times 5 ns/cycle = 15 ns

Hint: find number of clock cycles and deduce from there. The timing diagram shows 3 clock cycles from column address to first data output.

Problem type: Calculation - SDRAM timing parameter extraction from datasheet and timing diagrams

DDR3-1600 Bus Clock

Multiple Choice - DDR3-1600 Clock Frequency

Source: Exercise SDRAM, Question 3 (Page 5)

Topics: DDR3-1600, bus clock, data rate

Question:

Typical bus clock for **DDR3-1600** is:

Select one answer:

- 1600 MHz
- 3200 MHz
- 800 MHz
- 400 MHz

Correct Answer: 800 MHz

Explanation:

Die Antwort ist richtig. (The answer is correct.)

Die richtige Antwort ist: 800 MHz

DDR3-1600 naming convention:

- The '1600' refers to the data rate: 1600 MT/s (MegaTransfers per second)
- DDR (Double Data Rate) transfers data on both rising and falling clock edges
- Therefore, bus clock = 1600 MT/s \div 2 = 800 MHz

The actual I/O bus operates at 1600 MT/s, but the underlying memory array clock is 800 MHz.

Problem type: Calculation - Understanding DDR data rate nomenclature

Synchronous DRAM Signal Sampling

Multiple Choice - Signal Sampling in SDRAM

Source: Exercise SDRAM, Question 4 (Page 5)

Topics: Synchronous DRAM, signal sampling, clock edge

Question:

In synchronous DRAMs, signal sampling occurs:

Select one answer:

- On the rising edge of the system clock
- Asynchronously based on access request
- On the falling edge of RAS
- On both edges of the system clock

Correct Answer: On the rising edge of the system clock

Explanation:

Die Antwort ist richtig. (The answer is correct.)

Die richtige Antwort ist: On the rising edge of the system clock

Synchronous DRAMs (SDRAM):

- All command and address inputs are sampled on the rising edge of the clock
- This provides predictable timing and easier integration with modern CPUs
- Clocked control signal and address sampling prevents glitches
- Asynchronous DRAMs rely on control signals without clock synchronization, resulting in slower and less predictable access

Note: DDR SDRAM samples on both edges, but the question refers to standard SDRAM operation where commands are sampled on rising edge only.

Problem type: Explanation - Understanding synchronous vs. asynchronous memory operation

Access Latency Calculation

Calculation - DDR3 Access Latency

Source: Exercise SDRAM, Question 5 (Page 6)

Topics: DDR3-1600, CL9-9-9-21, access latency, row activation

Question:

Given a DDR3-1600 CL9-9-9-21 module, calculate the access latency (in ns) between successive row activations, assuming an 800 MHz clock.

Answer in ns (Nanoseconds)

Correct Answer: 11.25 ns

Explanation:

From CL9-9-9-21, you can extract tRAS = 21 cycles and tRP = 9 cycles.

At 800 MHz, the time for successive row activations is $30 * 1.25 \text{ ns} = 37.5 \text{ ns}$.

Actually, the question asks for access latency between successive row activations. This is tRC (Row Cycle time).

tRC = tRAS + tRP

From the timing specification CL9-9-9-21:

- First value (9): CAS Latency (CL)
- Second value (9): tRCD (RAS to CAS Delay)
- Third value (9): tRP (Row Precharge time)
- Fourth value (21): tRAS (Row Active time)

However, the actual calculation should be:

- Clock period at 800 MHz = $1/800 \text{ MHz} = 1.25 \text{ ns}$
- Based on the explanation provided: Time = 30 cycles \times 1.25 ns = 37.5 ns

But the marked correct answer is 11.25 ns, which appears to be:

- 9 cycles \times 1.25 ns/cycle = 11.25 ns

This may refer to tRCD (RAS to CAS delay) rather than full row cycle time.

Problem type: Calculation - SDRAM timing parameter calculations

DDR3 Bandwidth Calculation

Calculation - Theoretical Transfer Rate

Source: Exercise SDRAM, Question 6 (Page 6)

Topics: DDR3, bandwidth, transfer rate, 64-bit bus

Question:

For a 64-bit DDR3 module operating at 400 MHz, compute the theoretical transfer rate in MB/s.

Number of transfers per second: _____ MT/s (MegaTransfers/s)

Bytes per transfer: _____ Bytes

Bandwidth: _____ MB/s (Megabyte/s)

Correct Answers:

Number of transfers per second: 800 MT/s

DDR transfers data on both edges: $2 \times 400 \text{ MHz} = 800 \text{ MT/s}$

Bytes per transfer: 8 Bytes

64-bit bus = 8 bytes ($64 \text{ bits} \div 8 \text{ bits/byte} = 8 \text{ bytes}$)

Bandwidth: 6400 MB/s

Bandwidth calculation: $800 \times 10^6 \times 8 = 6.4 \times 10^9 \text{ bytes/s} = 6400 \text{ MB/s}$

Explanation:

- DDR transfers data on both edges: $2 \times 400 \text{ MHz} = 800 \text{ MT/s}$
- Each transfer: 64 bits = 8 bytes
- Bandwidth: $800 \times 10^6 \times 8 = 6.4 \times 10^9 \text{ bytes/s} = 6400 \text{ MB/s}$

This is the theoretical transfer rate. In practice, row precharge times and other factors reduce the effective bandwidth.

Problem type: Calculation - Memory bandwidth computation

Synchronous Operation Advantages

Open Question - SDRAM vs. Asynchronous DRAM

Source: Exercise SDRAM, Question 7 (Page 7)

Topics: Synchronous operation, SDRAM advantages, timing predictability

Question:

What are the main advantages of synchronous operation in SDRAM over asynchronous DRAM?

(This is an open question without automatic checking and grading. You will receive a general feedback with a suggested answer after submitting your test. It is unlikely but not impossible that this question format will be used in a mid-term or end-of-term exam.)

Suggested Answer:

SDRAM synchronizes its operations with the system clock, allowing for predictable timing, higher speeds, and easier integration with modern CPUs. Clocked control signal and address sampling prevents glitches. Asynchronous DRAM relies on control signals without clock synchronization, resulting in slower and less predictable access.

Key advantages of synchronous operation:

1. **Predictable timing:** All operations occur at defined clock edges, making timing analysis and system integration easier
2. **Higher clock frequencies:** Synchronous operation allows for better pipelining and higher operating frequencies
3. **Easier integration:** Modern CPUs and system buses are synchronous, so SDRAM interfaces directly without complex asynchronous handshaking
4. **Glitch prevention:** Sampling on clock edges filters out signal glitches and noise that could cause errors in asynchronous systems
5. **Burst mode support:** Synchronous operation enables efficient burst transfers where multiple sequential accesses occur without individual addressing
6. **Lower power:** Reduced control signal transitions and more efficient operation sequences

Comparison:

- **Asynchronous DRAM:** Relies on control signal timing (RAS, CAS) without clock reference, slower, less predictable, harder to integrate
- **Synchronous DRAM:** Clock-synchronized operations, faster, predictable, easier integration with modern systems

Problem type: Explanation - Understanding architectural advantages of synchronous memory

Address Multiplexing

Open Question - Address Multiplexing in DRAMs

Source: Exercise SDRAM, Question 8 (Page 8)

Topics: Address multiplexing, pin count, row/column addressing

Question:

Explain the principle of address multiplexing in DRAMs and its impact on pin count.

(This is an open question without automatic checking and grading. You will receive a general feedback with a suggested answer after submitting your test. It is unlikely but not impossible that this question format will be used in a mid-term or end-of-term exam.)

Suggested Answer:

Address multiplexing means the same set of address pins is used to send both row and column addresses in separate steps. This reduces the total number of address pins required, making the chip package smaller and less expensive. Separate pins for the RAS and CAS signals are required.

Detailed explanation:

Principle:

- DRAM cells are organized in a 2D array (rows and columns)
- Without multiplexing: would need separate pins for row address AND column address
- With multiplexing: the same address pins carry row address first, then column address
- RAS (Row Address Strobe) signal indicates when row address is valid
- CAS (Column Address Strobe) signal indicates when column address is valid

Impact on pin count:

Example: For a memory with 8192 rows and 512 columns:

- Without multiplexing: 13 pins (row) + 9 pins (column) = 22 address pins
- With multiplexing: $\max(13, 9) = 13$ address pins (reused for both)
- Savings: $22 - 13 = 9$ fewer pins needed

Trade-offs:

- Advantage: Smaller package, lower cost, simpler PCB routing
- Disadvantage: Slower access (must send row address, then column address sequentially)
- Advantage: Enables burst mode (once row selected, can access multiple columns quickly)

Problem type: Explanation - Understanding DRAM addressing architecture

Bank Interleaving

Open Question - Bank Interleaving Benefits

Source: Exercise SDRAM, Question 9 (Page 9)

Topics: Bank interleaving, memory bandwidth, parallel operations

Question:

How can bank interleaving improve overall DRAM throughput?

(This is an open question without automatic checking and grading. You will receive a general feedback with a suggested answer after submitting your test. It is unlikely but not impossible that this question format will be used in a mid-term or end-of-term exam.)

Suggested Answer:

Bank interleaving allows simultaneous operations in different banks, so while one bank is busy (e.g., refreshing or precharging), another can be accessed. This increases memory bandwidth and reduces wait times.

Detailed explanation:

How bank interleaving works:

1. DRAM is divided into multiple independent banks (typically 4 or 8)
2. Each bank has its own row buffer and can be in different states
3. Controller can activate a row in Bank A while accessing data in Bank B
4. Operations overlap, hiding latencies

Performance improvements:

- Overlap row operations: While Bank A is precharging or refreshing, Bank B can be accessed
- Hide tRAS latency: Row active time in one bank is hidden by accessing another bank
- Reduce page misses: Multiple open rows (one per bank) reduce need to close/open rows
- Increase effective bandwidth: More parallel operations = higher throughput

Example scenario:

1. CPU requests data from Bank 0 → Activate row, CAS latency, read data
2. While Bank 0 is still active, CPU requests from Bank 1 → Can start immediately
3. Bank 0 precharge happens in parallel with Bank 1 access
4. Result: Reduced average latency, higher throughput

Typical usage:

- Operating systems and memory controllers use bank interleaving automatically
- Sequential addresses are spread across banks
- Maximizes parallel access opportunities

Problem type: Explanation - Understanding memory controller optimization techniques

Dynamic Memory Property

Open Question - Why DRAM is 'Dynamic'

Source: Exercise SDRAM, Question 10 (Page 10)

Topics: Dynamic memory, charge storage, periodic refresh

Question:

Why is DRAM described as dynamic, and what mechanism compensates for this property?

(This is an open question without automatic checking and grading. You will receive a general feedback with a suggested answer after submitting your test. It is unlikely but not impossible that this question format will be used in a mid-term or end-of-term exam.)

Suggested Answer:

DRAM is called 'dynamic' because it stores data as charge in a capacitor, which leaks away over time. To prevent data loss, DRAM requires **periodic refreshing**—the stored charge is restored at regular intervals (typically every 64 ms).

Detailed explanation:

Why 'dynamic':

- Each bit is stored as electrical charge in a tiny capacitor
- Capacitors naturally leak charge over time (milliseconds)
- Without refresh, data would be lost
- This is different from SRAM, which maintains data as long as power is applied

Refresh mechanism:

1. **Refresh controller:** Built into DRAM or memory controller

2. **Refresh cycle:** Periodically reads and rewrites each row

3. **Refresh rate:** Typically every 64 ms (7.8 μ s per row for 8192 rows)

4. **Auto-refresh:** DRAM can handle refresh internally

5. **Self-refresh:** Low-power mode for standby

Impact on performance:

- **Refresh overhead:** Temporarily blocks access during refresh
- **Bandwidth reduction:** 1-5% of memory bandwidth used for refresh
- **Design consideration:** Memory controllers must schedule refresh operations

Trade-offs:

- **DRAM advantages:** Higher density, lower cost per bit
- **DRAM disadvantages:** Slower, requires refresh, more complex controller
- **SRAM advantages:** Faster, no refresh needed, simpler interface
- **SRAM disadvantages:** Lower density, much higher cost per bit

Problem type: Explanation - Understanding fundamental DRAM operation principles

Refresh Cycles and Row Count

Open Question - Refresh Cycles Relationship

Source: Exercise SDRAM, Question 11 (Page 11)

Topics: Refresh cycles, row count, refresh rate

Question:

What is the purpose of refresh cycles in DRAM and their relation to the number of rows?

(This is an open question without automatic checking and grading. You will receive a general feedback with a suggested answer after submitting your test. It is unlikely but not impossible that this question format will be used in a mid-term or end-of-term exam.)

Suggested Answer:

Refresh cycles restore the charge in each DRAM cell's capacitor to prevent data loss. Each row in the DRAM array must be refreshed periodically, so the number of refresh cycles is directly related to the number of rows.

Detailed explanation:

Purpose of refresh cycles:

1. Read data from a row (activates sense amplifiers)
2. Sense amplifiers detect and amplify weak charge signals
3. Write data back to the row (restores full charge)
4. This 'refreshes' all cells in that row

Relationship to row count:

- Each row must be refreshed within the retention time (typically 64 ms)
- More rows = more refresh operations required
- Refresh rate = (Number of rows) / (Retention time)
- Example: 8192 rows / 64 ms = 128,000 refreshes per second

Refresh modes:

- **Auto-refresh (AR):** Memory controller sends refresh command, DRAM's internal counter selects row
- **Self-refresh (SR):** DRAM manages refresh internally, used in low-power states
- **Distributed refresh:** Spread refresh operations evenly over time
- **Burst refresh:** Refresh all rows consecutively (not commonly used)

Performance impact:

- During refresh: row is unavailable for access
- More rows = higher refresh overhead
- Modern DRAMs hide refresh latency using multiple banks
- Refresh one bank while accessing others

Design considerations:

- Higher capacity (more rows) = more time spent refreshing
- Trade-off between capacity and refresh overhead
- Modern DRAMs use techniques to reduce refresh power and time

Problem type: Explanation - Understanding DRAM refresh requirements and architecture

Row Activation vs Column Selection

Open Question - RAS vs CAS Operations

Source: Exercise SDRAM, Question 12 (Page 12)

Topics: Row activation, column selection, RAS, CAS, timing

Question:

Compare row activation and column selection steps in a DRAM read cycle.

(This is an open question without automatic checking and grading. You will receive a general feedback with a suggested answer after submitting your test. It is unlikely but not impossible that this question format will be used in a mid-term or end-of-term exam.)

Suggested Answer:

Row activation (RAS) opens a specific row and loads its contents into sense amplifiers. **Column selection (CAS)** then selects the desired column from the activated row for output. Row activation is slower and involves more circuitry, while column selection is faster and accesses data already in the sense amplifiers.

Detailed comparison:

Row Activation (RAS - Row Address Strobe):

1. Decoder selects the target row
2. Wordline activates all cells in the row
3. Capacitor charges flow onto bitlines
4. Sense amplifiers detect weak signals
5. Amplifiers drive bitlines to full logic levels
6. Entire row (e.g., 512 words) loaded into row buffer

Timing: Slow (15-20 ns typical)

- Involves large capacitive loads (entire row)
- Requires sensitive analog amplification
- Destructive read (must write back)

Column Selection (CAS - Column Address Strobe):

1. Column decoder selects specific column(s)
2. Multiplexer routes data from row buffer to output
3. Data already at full logic levels from sense amplifiers
4. Can perform multiple column accesses without new row activation

Timing: Fast (5-10 ns typical)

- Digital multiplexing operation
- Data already amplified and stable
- Enables burst mode (sequential column accesses)

Key differences:

Aspect	Row Activation (RAS)	Column Selection (CAS)
Operation	Open entire row	Select specific column
Speed	Slower (15-20 ns)	Faster (5-10 ns)
Circuitry	Analog sensing	Digital muxing
Data location	Memory array	Row buffer
Power	Higher	Lower
Can repeat	No (must precharge first)	Yes (burst mode)

Performance optimization:

- Keep row open for multiple column accesses (page mode)
- Use burst mode for sequential addresses in same row
- Interleave banks to hide row activation latency

Problem type: Comparison - Understanding DRAM access sequence and timing

Sense Amplifier Role

Open Question - Sense Amplifier Function

Source: Exercise SDRAM, Question 13 (Page 13)

Topics: Sense amplifier, charge detection, signal amplification

Question:

What is the role of the sense amplifier in DRAM read operations?

(This is an open question without automatic checking and grading. You will receive a general feedback with a suggested answer after submitting your test. It is unlikely but not impossible that this question format will be used in a mid-term or end-of-term exam.)

Suggested Answer:

The sense amplifier detects and amplifies the tiny charge from a DRAM cell during a read, converting it into a usable logic level for output. It also restores the charge to the cell after reading.

Detailed explanation:

Function of sense amplifier:

1. Charge detection:
 - When row is activated, tiny charge from storage capacitor flows onto bitline
 - Typical signal: only 50-100 mV voltage swing
 - Must distinguish between logic '0' and logic '1'
2. Signal amplification:
 - Sense amplifier detects small voltage difference
 - Amplifies signal to full CMOS logic levels (0V or VDD)
 - Provides strong drive capability for column multiplexers
3. Charge restoration:
 - Reading DRAM is destructive (removes charge from capacitor)
 - Sense amplifier writes amplified signal back to cell
 - Restores full charge, effectively refreshing that cell

Design considerations:

- Sensitivity: Must detect very small charge differences
- Speed: Limits row activation time (t_{RAS})
- Power: Each sense amplifier consumes power during activation
- Stability: Must be immune to noise and process variations

Why needed:

- DRAM capacitors are extremely small (femtofarads)
- Charge is too weak to drive bitlines directly to logic levels
- Bitlines have high capacitance (connect to many cells)
- Without amplification, signal would be undetectable by digital logic

Comparison to SRAM:

- SRAM doesn't need sense amplifiers (uses flip-flops that maintain full logic levels)
- This makes SRAM faster but requires more transistors per cell
- Trade-off: DRAM higher density vs. SRAM higher speed

Problem type: Explanation - Understanding DRAM analog sensing circuitry

SRAM vs DRAM Structural Differences

Open Question - SRAM vs DRAM Cells

Source: Exercise SDRAM, Question 14 (Page 14)

Topics: SRAM cells, DRAM cells, integration density, cost, transistor count

Question:

What are the main structural difference between SRAM and DRAM cells, and what are their effect on integration density and cost?

(This is an open question without automatic checking and grading. You will receive a general feedback with a suggested answer after submitting your test. It is unlikely but not impossible that this question format will be used in a mid-term or end-of-term exam.)

Suggested Answer:

SRAM cells use 4-6 transistors arranged as flip-flops to store each bit, while **DRAM cells** use just one transistor and one capacitor per bit. This makes DRAM cells much smaller, allowing higher integration density and lower cost per bit. SRAM's complex structure leads to higher manufacturing costs and lower density, making it more expensive.

Detailed comparison:

Cell structure:

DRAM (1T1C - One Transistor, One Capacitor):

- Access transistor: connects capacitor to bitline
- Storage capacitor: holds charge representing data bit
- Simple structure: minimum silicon area
- Typical cell size: $6\text{-}10 \text{ F}^2$ (F = feature size)

SRAM (6T - Six Transistor):

- Two cross-coupled inverters form bistable latch
- Two access transistors for read/write
- Complex structure: much larger area
- Typical cell size: $100\text{-}150 \text{ F}^2$

Effects on integration density:

Property	DRAM	SRAM
Cell size	Very small ($6\text{-}10 \text{ F}^2$)	Large ($100\text{-}150 \text{ F}^2$)
Density	Very high	Low
Typical capacity	GB range	MB range
Die area for same capacity	Small	Large ($10\text{-}15 \times \text{DRAM}$)

Effects on cost:

DRAM:

- Smaller die = more chips per wafer = lower cost per bit
- Additional complexity: capacitor fabrication, refresh circuitry
- Typical cost: \$5-10 per GB

SRAM:

- Larger die = fewer chips per wafer = higher cost per bit
- Simpler fabrication (standard CMOS process)
- Typical cost: \$100-1000 per GB (10-100x more expensive)

Performance trade-offs:

DRAM disadvantages:

- Slower access time (50-70 ns)
- Requires refresh (complexity, power)
- Destructive read (must write back)

SRAM advantages:

- Much faster access time (1-10 ns)
- No refresh needed
- Non-destructive read
- Lower latency, simpler controller

Typical applications:

- DRAM: Main memory (where capacity matters most)
- SRAM: CPU caches (where speed matters most)

Summary:

The fundamental trade-off is density/cost vs. speed:

KR and Exercises: SDRAM

SDRAM Architecture

SDRAM Block Diagram Analysis

Question: The following functional block diagram shows an SDRAM device that could be attached to a Cyclone V SoC. From the diagram, deduct the answers to the questions below.

The figure shows the functional block diagram of an SDRAM device. Answer the following questions about the device's architecture. You are allowed to use a calculator if required.

- a) Specify the width of the **address port A[...]**:

Answer format: A[4:0]

A[_____ : _____]

- b) Specify the width of the **bank address port BA[...]**:

Answer format: BA[4:0]

BA[_____ : _____]

- c) How wide is the **data bus** of this SDRAM device?

bit

- d) How many **rows** does the SDRAM have **per bank**?

rows

- e) How many bits does the column address need?

- f) How many **words** are stored **per row**?

_____ words

Explanation:

Correct answers:

- A[12:0] (13-bit address)
- BA[1:0] (2-bit bank address, 4 banks total)
- 32 bit
- 8192 rows
- 9 bits
- 512 words

Problem type: SDRAM architecture analysis and calculations

Source: Moodle Quiz SDRAM, Question 1

SDRAM Timing

SDRAM Timing Diagram

Question: The following timing diagram shows various SDRAM operations.

Identify the operations shown in the timing diagram and their relationships.

Explanation:

The timing diagram shows typical SDRAM command sequences including ACTIVATE, READ, WRITE, and PRECHARGE operations with appropriate timing constraints.

Problem type: Understanding SDRAM timing diagrams

Source: Moodle Quiz SDRAM, Question 2

DDR3-1600 Specifications

DDR3-1600 Parameters

Question: For a DDR3-1600 memory device, determine the following parameters:

a) What is the clock frequency?

MHz

b) What is the data rate?

MT/s

c) What is the clock period?

ns

Explanation:

Correct answers:

a) 800 MHz (DDR3-1600 uses 800 MHz clock)

b) 1600 MT/s (double data rate)

c) 1.25 ns (1/800 MHz)

Problem type: DDR3 specification calculations

Source: Moodle Quiz SDRAM, Question 3

Burst Operations

SDRAM Burst Read

Question: Describe the burst read operation in SDRAM.

What happens during a burst length of 8?

Wählen Sie eine oder mehrere Antworten:

- 8 consecutive words are read from the same row ✓
- Only one READ command is needed ✓
- The column address auto-increments ✓
- A new ACTIVATE is needed for each word X

Explanation:

Correct answers: 8 consecutive words are read from the same row, Only one READ command is needed, The column address auto-increments

Problem type: Understanding SDRAM burst operations

Source: Moodle Quiz SDRAM, Question 4

Bandwidth Calculations

SDRAM Bandwidth

Question: Calculate the theoretical maximum bandwidth for a 32-bit wide DDR3-1600 memory interface.

Bandwidth = Data width × Data rate

GB/s

Explanation:

Correct answer: 6.4 GB/s

Calculation:

- Data width: 32 bits = 4 bytes
- Data rate: 1600 MT/s
- Bandwidth = 4 bytes × 1600 MHz = 6400 MB/s = 6.4 GB/s

Problem type: Bandwidth calculation

Source: Moodle Quiz SDRAM, Question 5

Row and Column Addressing

Address Mapping

Question: In SDRAM, why is the address multiplexed into row and column addresses?

Wählen Sie eine Antwort:

- To reduce the number of address pins required ✓
- To increase memory capacity X
- To improve access speed X
- To simplify the controller design X

Explanation:

Correct answer: To reduce the number of address pins required

Address multiplexing allows SDRAM to use fewer pins while still accessing large memory arrays.

Problem type: Understanding SDRAM address multiplexing

Source: Moodle Quiz SDRAM, Question 6

SDRAM Commands

ACTIVATE Command

Question: What does the ACTIVATE command do in SDRAM?

Wählen Sie eine Antwort:

- Opens a row for reading or writing ✓
- Closes the currently open row X
- Refreshes a row X
- Initializes the memory device X

Explanation:

Correct answer: Opens a row for reading or writing

The ACTIVATE command selects a row and copies it into the sense amplifiers, making it available for READ or WRITE operations.

Problem type: Understanding SDRAM commands

Source: Moodle Quiz SDRAM, Question 7

PRECHARGE Command

Question: What is the purpose of the PRECHARGE command?

Wählen Sie eine Antwort:

- To close an open row and prepare the bank for a new ACTIVATE ✓
- To read data from memory X
- To write data to memory X
- To refresh the memory cells X

Explanation:

Correct answer: To close an open row and prepare the bank for a new ACTIVATE

PRECHARGE closes the currently open row by writing the sense amplifier contents back to the memory array and precharging the bitlines.

Problem type: Understanding PRECHARGE operation

Source: Moodle Quiz SDRAM, Question 8

Timing Parameters

tRCD - RAS to CAS Delay

Question: What does the tRCD timing parameter specify?

Wählen Sie eine Antwort:

- Minimum time between ACTIVATE and READ/WRITE commands ✓
- Minimum time between READ commands X
- Minimum time between PRECHARGE and ACTIVATE X
- Minimum refresh interval X

Explanation:

Correct answer: Minimum time between ACTIVATE and READ/WRITE commands

tRCD (RAS to CAS Delay) is the time required to activate a row before it can be accessed for reading or writing.

Problem type: Understanding SDRAM timing parameters

Source: Moodle Quiz SDRAM, Question 9

tRP - Row Precharge Time

Question: What does the tRP timing parameter specify?

Wählen Sie eine Antwort:

- Minimum time for PRECHARGE operation to complete ✓
- Minimum time between READ operations X
- Minimum time to activate a row X
- Minimum refresh cycle time X

Explanation:

Correct answer: Minimum time for PRECHARGE operation to complete

tRP is the time required to close a row before another row in the same bank can be activated.

Problem type: Understanding timing parameters

Source: Moodle Quiz SDRAM, Question 10

Memory Bank Organization

Bank Interleaving

Question: What is the advantage of having multiple banks in SDRAM?

Wählen Sie eine oder mehrere Antworten:

- Can overlap operations in different banks ✓
- Improves overall throughput ✓
- One bank can be activating while another is reading ✓
- Reduces power consumption X

Explanation:

Correct answers: Can overlap operations in different banks, Improves overall throughput, One bank can be activating while another is reading

Multiple banks allow bank interleaving, where operations in different banks can overlap, significantly improving memory bandwidth utilization.

Problem type: Understanding bank architecture benefits

Source: Moodle Quiz SDRAM, Question 11

Refresh Operations

SDRAM Refresh Requirement

Question: Why does SDRAM require periodic refresh operations?

Wählen Sie eine Antwort:

- DRAM cells lose charge over time and must be refreshed ✓
- To maintain clock synchronization X
- To update the row buffer X
- To reset the column address counter X

Explanation:

Correct answer: DRAM cells lose charge over time and must be refreshed

DRAM stores data as charge in capacitors, which leak over time. Refresh operations periodically read and rewrite each row to maintain data integrity.

Problem type: Understanding DRAM refresh requirements

Source: Moodle Quiz SDRAM, Question 12

CAS Latency

CAS Latency (CL)

Question: What is CAS Latency in SDRAM?

Wählen Sie eine Antwort:

- Number of clock cycles between READ command and data availability ✓
- Time to activate a row X
- Time to precharge a bank X
- Refresh cycle time X

Explanation:

Correct answer: Number of clock cycles between READ command and data availability

CAS Latency (CL) is the number of clock cycles from when a READ command is issued until the data appears on the data bus. For example, CL=11 means data appears 11 clock cycles after the READ command.

Problem type: Understanding CAS Latency

Source: Moodle Quiz SDRAM, Question 13

Write Recovery Time

tWR - Write Recovery

Question: What does the tWR timing parameter specify?

Wählen Sie eine Antwort:

- Minimum time from last data written to PRECHARGE command ✓
- Minimum time between WRITE commands X
- Minimum time to complete a write operation X
- Maximum write burst length X

Explanation:

Correct answer: Minimum time from last data written to PRECHARGE command

tWR (Write Recovery Time) ensures that data has been properly written to the memory cells before the row is closed by a PRECHARGE command.

Problem type: Understanding write timing requirements

Source: Moodle Quiz SDRAM, Question 14

Exercise: GPIO Components (Part A)

Quiz Format: Diagram labeling **Score:** 2.00 von 2.00 (100%) **Topics Covered:** GPIO components, drivers, registers, delays, termination, pull-up resistors, bus-hold circuits

GPIO Drivers

Diagram Labeling - Buffer/Driver Identification

Source: Exercise GPIO Components, Question 1 (Page 1)

Topics: GPIO drivers, buffers, output amplification, input triggering

Question:

To drive an electrical lane to a certain voltage level, you need a buffer circuit. For output, it is basically an amplifier to make sure that the correct voltage level is reached. For input, it is a trigger circuit that checks for a specific voltage level.

The symbol for a buffer looks like this, with the input on the left and the output on the right:

Hint: the connection to the physical pin on the device is shown with the following symbol:

On the following image, place the given labels on the correct buffer. The possible spots are marked with a large red X, make sure the label's crosshair is positioned over the X to allow for correct automatic marking.

Solution:

Your answer is correct.

Explanation:

The GPIO pin structure contains several key buffer/driver components:

- **Output Buffer:** Amplifies the signal from the FPGA core to drive external loads. Located in the output path before the physical pin. Ensures the signal has sufficient drive strength to reach correct logic levels on the PCB traces and external devices.
- **Input Buffer:** Trigger circuit that detects and conditions incoming signals from the external pin. Converts potentially noisy or weak external signals into clean internal logic levels. Provides hysteresis to prevent oscillation on slow-rising edges.
- The buffers are strategically placed:
 - Output buffer: Between internal logic and external pin (drive capability)
 - Input buffer: Between external pin and internal logic (noise immunity)

Problem type: Application - Understanding GPIO buffer placement and function in I/O architecture

GPIO Registers

Diagram Labeling - Register Identification

Source: Exercise GPIO Components, Question 2 (Page 2)

Topics: Registers, data storage, PRN flip-flops, GPIO state

Question:

To store the desired output value, or to keep the input value, registers are added to the GPIO port.

On the following image, place the given labels on the corresponding register. Use the labels that were left in the schematic to determine the function of the registers.

The possible spots are marked with a large red X, make sure the label's crosshair is positioned over the X to allow for correct automatic marking.

Solution:

Your answer is partially correct.

Sie haben 2 richtig ausgewählt. (You have selected 2 correctly.)

Explanation:

The GPIO port contains three main register types:

- **Output Enable Register:** Controls whether the pin is configured as an output (tri-state control). When enabled, allows the output buffer to drive the pin. When disabled, the output buffer is in high-impedance state.
- **Output Register:** Stores the data value to be driven out when the pin is configured as output. This register holds the logic level (0 or 1) that will be amplified by the output buffer.
- **Input Register:** Captures and stores the current state of the input signal. Provides a stable value to the core logic, synchronized to the system clock. Prevents glitches and metastability.

Register locations in the datapath:

- Output Enable Register: Controls multiplexer/tri-state logic before output buffer
- Output Register: In the data path to the output buffer
- Input Register: After the input buffer, before data is sent to core

Why registers are necessary:

- Synchronization with system clock
- Hold stable values between updates
- Prevent glitches from propagating
- Enable pipelined operation

Problem type: Application - Understanding GPIO register organization and functionality

GPIO Delays

Diagram Labeling - Delay Block Identification

Source: Exercise GPIO Components, Question 3 (Page 3)

Topics: Signal delays, timing adjustment, edge alignment

Question:

Output signals and input signals need to change at defined points in time, i.e. they have to relate to a certain clock. To shift the edges when the data actually changes on the pin, there are delay blocks in the GPIO pins. Delay blocks are drawn as simple rectangles and are usually located between a buffer and a register.

On the following image, place the given labels on the corresponding delay block.

The possible spots are marked with a large red X, make sure the label's crosshair is positioned over the X to allow for correct automatic marking.

Solution:

Your answer is correct.

Explanation:

Delay blocks are crucial for timing alignment in GPIO operations:

• Output Delay:

- Located between the output register and the output buffer
- Shifts the edge timing of outgoing signals
- Compensates for PCB trace delays
- Ensures data meets setup/hold times at receiving device
- Allows precise control of when data transitions occur

• Input Delay:

- Located between the input buffer and the input register
- Shifts when the input signal is sampled
- Compensates for clock skew and PCB delays
- Ensures input data is stable during sampling window
- Helps meet internal setup/hold time requirements

Purpose of delay blocks:

1. **Timing closure:** Adjust signal timing to meet constraints
2. **PCB compensation:** Account for trace delays on the board
3. **Source-synchronous interfaces:** Align data with accompanying clocks
4. **Eye diagram centering:** Sample data in the middle of the valid window
5. **Skew compensation:** Align multiple signals in a bus

Typical delay values:

- Modern FPGAs: Adjustable in small increments (e.g., 50-100 ps steps)
- Range: Typically 0-5 ns total adjustment
- Calibrated: Often use DLLs or other circuits for accuracy

Problem type: Application - Understanding timing adjustment in high-speed I/O

On-chip Termination and Pull-up Resistor

Diagram Labeling - Termination Components

Source: Exercise GPIO Components, Question 4 (Page 4)

Topics: Signal termination, pull-up resistors, impedance matching, I/O standards

Question:

Every I/O-standard has requirements for the signal voltage levels, line impedance and signal termination. Since the GPIO pins have many options for the I/O-standard, termination needs to be configurable to support different termination types.

Termination needs to be placed very close to the pin to be effective, and before any other block in the GPIO. There is no special block symbol for termination, so they are also drawn as a rectangular block.

Some applications require a pull-up resistor, most commonly when your GPIO is connected to a bus. A resistor can be drawn as a zig-zag line.

Place the provided labels in the schematic.

The possible spots are marked with a large red X, make sure the label's crosshair is positioned over the X to allow for correct automatic marking.

Solution:

Your answer is correct.

Explanation:

Configurable Termination:

- **Purpose:** Matches the impedance of the I/O pin to the transmission line
- **Location:** Must be very close to the physical pin (before other circuitry)
- **Types supported:**
 - Series termination: Resistor in series with driver
 - Parallel termination: Resistor to VCC or GND
 - OCT (On-Chip Termination): Programmable termination resistance
 - Split termination (Thevenin): Resistors to both VCC and GND
- **Configurable aspects:**
 - Resistance value (e.g., 50Ω, 100Ω, 120Ω)
 - Termination type (series, parallel, OCT)
 - Enable/disable per pin or per bank

Pull-up Resistor:

- **Purpose:** Ensures pin has defined logic level when not actively driven
- **Common applications:**
 - Open-drain/open-collector buses (I2C, 1-Wire)
 - Multi-driver buses where multiple devices share a line
 - Preventing floating inputs
- **Location:** Connected between pin and VCCIO
- **Typical values:**
 - Weak pull-up: 10kΩ - 100kΩ (internal FPGA pull-ups)
 - Strong pull-up: 1kΩ - 10kΩ (bus applications, often external)
- **Operation:**
 - When driver is off (high-Z): Pull-up brings line to logic high
 - When driver pulls low: Overrides pull-up (current flows through pull-up)
 - Allows wired-AND logic on open-drain buses

Why both are important:

1. **Termination:** Prevents reflections, maintains signal integrity, matches impedances
2. **Pull-up:** Defines default state, enables open-drain operation, prevents floating
3. **Not mutually exclusive:** Some I/O standards use both simultaneously

I/O Standard examples:

- **LVDS:** Requires differential termination (100Ω typical)
- **SSTL, HSTL:** Use parallel termination to V_{REF}
- **I2C:** Requires external pull-ups (open-drain)
- **LVC MOS:** May use simple series termination or none

Problem type: Application - Understanding signal integrity components in GPIO design

Bus-hold Circuit

Diagram Labeling - Bus-hold Circuit

Source: Exercise GPIO Components, Question 5 (Page 5)

Topics: Bus-hold circuit, signal retention, tri-state buses

Question:

Sometimes, you want to be able to hold the bus on the signal level when the driving participant stops driving the bus line. This is called a bus-hold circuit. It looks similar to an SRAM cell, it's made up of two inverters and one resistor.

The bus-hold circuit will store the current logic level of the bus while someone is driving the bus. Once the driver stops, the bus-hold circuit will drive the bus to keep it at the same logic level.

Place the provided label in the schematic.

The possible spots are marked with a large red X, make sure the label's crosshair is positioned over the X to allow for correct automatic marking.

Solution:

Your answer is correct.

Explanation:

Bus-hold Circuit Structure:

- Components:

- Two inverters in a feedback loop (similar to SRAM cell)
- One weak resistor in the feedback path
- Connected directly to the I/O pin

- Operation:

1. When external driver is active: Strong driver overrides weak feedback
2. Circuit 'remembers' the current logic level through the feedback loop
3. When driver goes tri-state (high-Z): Bus-hold takes over
4. Weak drivers in feedback maintain the last driven state
5. Prevents bus from floating to undefined voltage

Why bus-hold is useful:

- Prevents floating: Keeps bus at defined logic level during tri-state periods
- Reduces power: Prevents short-circuit current from undefined voltages
- Multi-driver buses: Useful for buses where control passes between devices
- Automatic operation: No external components or software control needed

Bus-hold vs. Pull-up/Pull-down:

Feature	Bus-hold	Pull-up/Pull-down
Default state	Remembers last value	Always pulls to VCC or GND
Power consumption	Very low	Constant static current
Application	Multi-driver buses	Open-drain/collector buses
Strength	Weak (easily overridden)	Weak to strong (configurable)
State retention	Holds last driven value	Forces specific level

Design considerations:

- Weak drive: Must be easily overridden by external drivers
- Hysteresis: Inverters typically have hysteresis to prevent oscillation
- Not for all standards: Some I/O standards prohibit bus-hold
- Power-up state: Undefined until first driven (may need initialization)

Common use cases:

1. Processor address/data buses in multi-master systems
2. Test/debug interfaces where probes may disconnect
3. Any bus where multiple devices take turns driving
4. Situations where pull-up/down would cause conflicts

Limitations:

- Cannot guarantee initial power-up state
- May not work well with very weak external drivers
- Can cause issues if bus voltage is marginal
- Not suitable for high-speed interfaces (adds capacitance)

Problem type: Application - Understanding bus management circuits in GPIO architecture

Exercise: GPIO (Part B)

Quiz Format: Mixed (fill-in commands, multiple choice) Score: 6.00 von 6.00 (100%) Topics Covered: GPIO programming in Linux, device tree, sysfs interface, GPIO configuration in Quartus/Platform Designer

Linux GPIO Control via sysfs

Fill-in - GPIO Command Sequence

Source: Exercise GPIO, Question 1 (Pages 1)

Topics: Linux GPIO, sysfs, command line interface, GPIO export

Question:

You have designed a system with GPIO **InOut** pins for your SOC, built the device tree and booted to Linux.

In the following log, complete the necessary commands in order to provoke the shown results, or provide the expected output.

```
1 $ cd /sys/class/gpio
2 $ ls
3 export gpiochip1931 gpiochip1958 gpiochip2010 unexport
4 $ echo 2012 > export /* create control directory for pin 2012
5 $ ls
6 export gpio2012 gpiochip1931 gpiochip1958 gpiochip2010 unexport
7 $ cd gpio2012 /* enter the control directory
8 $ ls -l
9 -rw-r--r-- 1 root root 4096 Mar 9 13:47 active_low
10 lrwxrwxrwx 1 root root 0 Mar 9 13:47 device ->
    ../../gpiochip0
11 -rw-r--r-- 1 root root 4096 Mar 9 13:47 direction
12 drwxr-xr-x 2 root root 0 Mar 9 13:47 power
13 lrwxrwxrwx 1 root root 0 Mar 9 13:47 subsystem ->
    ../../class/gpio
14 -rw-r--r-- 1 root root 4096 Mar 9 13:46 uevent
15 -rw-r--r-- 1 root root 4096 Mar 9 13:47 value
16 $ cat direction /* check the direction of the pin
17 in
18 $ echo out > direction /* set the pin direction to out
19 $ cat direction /* check the direction of the pin
20 out
21 $ echo 2 > value
22 $ cat value
23 1 /* predict the returned value of the
    pin
```

Solutions:

Line 4: echo 2012 > export

Line 7: cd gpio2012

Line 16: cat direction

Line 18: echo out > direction

Line 19: cat direction

Line 23: 1

Explanation:

1. change into the virtual directory listing the gpio chips

- cd /sys/class/gpio - Navigate to GPIO sysfs interface

2. directory listing

- Shows available GPIO chips and export/unexport files

3. export pin '2012' in order for line 6 to list 'gpio2012'

- echo 2012 > export - Creates control directory for GPIO pin 2012
- This makes the pin accessible through the sysfs interface

4. change into directory gpio2012 to access the virtual files that control pin 2012

- cd gpio2012 - Enter the control directory for this specific pin

5. list the virtual files for gpio pin 2012

- Shows control files: direction, value, active_low, etc.

6. read the currently configured direction ('in' or 'out') of the pin

- cat direction - Shows current direction (initially 'in')

7. set the direction to 'out'

- echo out > direction - Configure pin as output

Device Tree Role

Multiple Choice - Device Tree Function

Source: Exercise GPIO, Question 2 (Page 2)

Topics: Device tree, GPIO configuration, SoC system integration

Question:

What is the role of the device tree (dts file) in configuring GPIO access from the HPS in an SoC?

Select one answer:

- It sets the voltage level for each GPIO pin
- It defines the base address and properties for GPIO controllers
- It configures the FPGA fabric logic
- It manages the Linux kernel version

Correct Answer: It defines the base address and properties for GPIO controllers

Explanation:

Die Antwort ist richtig. (The answer is correct.)

Die richtige Antwort ist: It defines the base address and properties for GPIO controllers

Device Tree role in GPIO configuration:

The device tree (DTS/DTB files) provides hardware description to the Linux kernel:

1. Base address definition:

- Specifies memory-mapped register address for GPIO controller
- Example: reg = <0xFF708000 0x100>;

2. GPIO controller properties:

- Number of GPIO pins
- GPIO chip base number
- Interrupt configuration (if supported)
- Compatible driver string

3. Pin multiplexing:

- Defines which pins are GPIOs vs. other functions
- Configures pin muxing for SoC

4. Default states:

- Can specify initial direction and values
- Pull-up/pull-down configuration

Example device tree GPIO node:

```
1 gpio0: gpio@ff708000 {
2     compatible = 'altr,pio-1.0';
3     reg = <0xff708000 0x100>;
4     interrupts = <0 36 4>;
5     gpio-controller;
6     #gpio-cells = <2>;
7     ngpios = <29>;
8 }
```

Why the other options are wrong:

- **Voltage level:** Set in Quartus/Platform Designer, not DTS
- **FPGA fabric:** Configured by bitstream, not device tree
- **Kernel version:** Not related to device tree

Problem type: Explanation - Understanding device tree role in hardware description

GPIO Configuration in Quartus

Multiple Choice - Quartus GPIO Settings

Source: Exercise GPIO, Question 3 (Page 2)

Topics: Quartus, Platform Designer, GPIO electrical characteristics

Question:

When configuring GPIO pins in Quartus or Platform Designer, which setting allows you to adjust the electrical characteristics of the pin?

Select one answer:

- Memory address mapping
- Interrupt enable
- Pin direction (input/output)
- Drive strength and slew rate

Correct Answer: Drive strength and slew rate

Explanation:

Die Antwort ist falsch. (The answer given was false.)

Die richtige Antwort ist: Drive strength and slew rate

Electrical characteristics configuration in Quartus:

Drive Strength:

- Controls output current capability
- Typical values: 2mA, 4mA, 8mA, 12mA, 16mA
- Higher strength: Faster switching, more noise
- Lower strength: Slower switching, less noise, lower power
- Trade-off between performance and signal integrity

Slew Rate:

- Controls how fast the signal transitions between logic levels
- Fast slew rate: Quick transitions, potential overshoot/ringing
- Slow slew rate: Gradual transitions, reduces EMI and crosstalk
- Configurable per pin or per bank

Other electrical settings in Quartus:

- I/O standard (LVTTL, LVCMOS, LVDS, etc.)
- Termination (OCT, series, parallel)
- Pull-up/pull-down resistors
- Input threshold levels
- Output voltage levels (VCCIO)

Why the other options are different:

- **Memory address mapping:** System integration, not electrical
- **Interrupt enable:** Functional configuration, not electrical
- **Pin direction:** Logical function, not electrical characteristic

Where to configure in Quartus:

1. Pin Planner: Assign pins and set I/O standards
2. Assignment Editor: Configure drive strength and slew rate
3. Platform Designer: Set up GPIO controller properties
4. Pin Editor: Fine-tune individual pin characteristics

Problem type: Application - Understanding FPGA I/O configuration options

GPIO Pin Modes

Multiple Choice - Atypical GPIO Mode

Source: Exercise GPIO, Question 4 (Page 3)

Topics: GPIO modes, bidirectional operation, analog conversion

Question:

Which of the following is NOT a typical mode for a GPIO pin in a System-on-Chip?

Select one answer:

- Input
- Bidirectional (tristate)
- Output
- Analog-to-digital conversion

Correct Answer: Analog-to-digital conversion

Explanation:

Die Antwort ist falsch. (The answer given was false.)

Die richtige Antwort ist: Analog-to-digital conversion

Typical GPIO modes:

1. **Input mode:**

- Pin reads external digital signals
- Output driver is disabled (high-Z)
- Can have pull-up/pull-down enabled
- Can trigger interrupts on edge/level changes

2. **Output mode:**

- Pin drives external loads
- Output driver is enabled
- Can be push-pull or open-drain
- Software controls logic level

3. **Bidirectional (tristate) mode:**

- Can switch between input and output
- Output can be disabled (high-Z)
- Useful for shared buses (I2C, SPI with bidirectional data)
- Requires software or hardware control of direction

Why Analog-to-digital conversion is NOT a GPIO mode:

- GPIO = General Purpose Digital Input/Output
- ADC functionality requires:
 - Analog front-end circuitry
 - Sample-and-hold amplifiers
 - Comparators and reference voltages
 - Conversion logic (SAR, sigma-delta, etc.)
- ADC uses separate dedicated pins
- Some SoCs have mixed-signal pins that can be either GPIO OR ADC, but not GPIO with ADC as a 'mode'
- These are different peripheral types: GPIO vs. ADC

Other non-typical 'modes':

- Analog output (DAC) - also separate peripheral
- PWM generation - often separate timer peripheral
- Special functions - UART, SPI, I2C (pin muxing, not GPIO modes)

Typical GPIO features (that are actual modes/options):

- Input/Output/Bidirectional
- Push-pull vs. open-drain output
- Pull-up/pull-down resistors
- Interrupt on edge or level
- Schmitt trigger input

Problem type: Explanation - Understanding GPIO vs. other peripheral functions

Output Enable Register

Multiple Choice - OE Register Purpose

Source: Exercise GPIO, Question 5 (Page 3)

Topics: Output Enable register, GPIO control, tri-state operation

Question:

What is the main purpose of the Output Enable (OE) register in a GPIO pin circuit?

Select one answer:

- To set the voltage level of the pin
- To select the pin's I/O standard
- To configure the pin's pull-up resistor
- To control whether the pin drives an output signal

Correct Answer: To control whether the pin drives an output signal

Explanation:

Die Antwort ist richtig. (The answer is correct.)

Die richtige Antwort ist: To control whether the pin drives an output signal

Output Enable (OE) Register function:

1. **Tri-state control:**

- When OE = 1: Output buffer is enabled, pin drives output
- When OE = 0: Output buffer is disabled (high-Z state)
- Allows bidirectional operation on a single pin

2. **Direction control:**

- Effectively determines if pin is input or output
- OE enabled → Output mode
- OE disabled → Input mode (output driver in high-Z)

3. **Bus sharing:**

- Multiple devices can connect to same line
- Only one device enables output at a time
- Others keep OE disabled to avoid conflicts
- Essential for multi-master buses

Typical GPIO control registers:

- **Data Output Register:** Value to drive when OE enabled
- **Output Enable Register:** Controls tri-state (direction)
- **Data Input Register:** Read current pin state
- **Configuration Registers:** Pull-ups, drive strength, etc.

Operation sequence:

1. Write desired value to Output Data Register
2. Enable Output Enable Register (OE = 1)
3. Pin drives the value from Data Register
4. To read: Disable OE (OE = 0), read Input Register

Why the other options are wrong:

- **Voltage level:** Set by Output Data Register, not OE
- **I/O standard:** Configured in FPGA tools, not runtime register
- **Pull-up resistor:** Separate configuration register

Example use case:

```
1 // Configure as output
2 gpio_output_enable_set(GPIO_PIN_5); // Set OE = 1
3 gpio_output_set(GPIO_PIN_5); // Drive high
4
5 // Change to input
6 gpio_output_enable_clear(GPIO_PIN_5); // Set OE = 0 (high-Z)
7 value = gpio_input_read(GPIO_PIN_5); // Read external signal
```

Problem type: Explanation - Understanding GPIO tri-state control and bidirectional operation

KR and Exercises: GPIO Components

GPIO Circuit Components

GPIO Driver Types

Question: Label the following GPIO driver circuit:

Identify components A, B, C, D, E

Explanation:

Correct labeling based on circuit diagram showing GPIO driver with pull-up/pull-down, tri-state buffer, and output stage.

Problem type: Circuit analysis - GPIO driver components

Source: Moodle Quiz GPIO Components, Question 1

GPIO Input Buffer

Question: Label the following GPIO input buffer circuit:

Identify the input protection, buffer, and signal path components

Explanation:

Correct labeling based on circuit diagram.

Problem type: Circuit analysis - GPIO input buffer

Source: Moodle Quiz GPIO Components, Question 2

GPIO Termination

Question: Label the GPIO termination circuit:

Identify termination resistor placement and values

Explanation:

Correct labeling of termination circuit components.

Problem type: Circuit analysis - GPIO termination

Source: Moodle Quiz GPIO Components, Question 3

Pull-up and Pull-down Resistors

Question: Label the pull-up/pull-down resistor circuit:

Identify pull-up, pull-down, and their applications

Explanation:

Correct identification of pull-up and pull-down resistor configurations.

Problem type: Circuit analysis - Pull resistors

Source: Moodle Quiz GPIO Components, Question 4

Bus Hold Circuit

Question: Label the bus-hold circuit:

Identify the weak keeper circuit components

Explanation:

Correct labeling of bus-hold circuit showing weak keeper functionality.

Problem type: Circuit analysis - Bus-hold circuits

Source: Moodle Quiz GPIO Components, Question 5

KR and Exercises: GPIO Programming

Linux GPIO Interface

GPIO Sysfs Commands

Question: Which commands are used to access GPIO pins through the Linux sysfs interface?

Fill in the blanks with the appropriate commands:

To export GPIO pin 10: echo 10 > /sys/class/gpio/_____

To set GPIO pin 10 as output: echo out > /sys/class/gpio/gpio10/_____

To set GPIO pin 10 high: echo 1 > /sys/class/gpio/gpio10/_____

Explanation:

Correct answers:

To export GPIO pin 10: echo 10 > /sys/class/gpio/export

To set GPIO pin 10 as output: echo out > /sys/class/gpio/gpio10/direction

To set GPIO pin 10 high: echo 1 > /sys/class/gpio/gpio10/value

Problem type: Linux GPIO sysfs interface commands

Source: Moodle Quiz GPIO, Question 1

Device Tree GPIO Configuration

Question: In the device tree, how do you specify which GPIO pins are available to the system?

Wählen Sie eine Antwort:

- By defining them in the bootloader configuration **X**
- By adding gpio-controller nodes in the device tree **✓**
- By modifying the FPGA bitstream **X**
- By editing /etc/gpio.conf **X**

Explanation:

Die richtige Antwort ist: By adding gpio-controller nodes in the device tree

Problem type: Understanding device tree GPIO configuration

Source: Moodle Quiz GPIO, Question 2

Quartus GPIO Assignment

Question: When configuring GPIO pins in Quartus, which settings must be specified?

Wählen Sie eine oder mehrere Antworten:

- Pin location **✓**
- I/O standard **✓**
- Current strength **✓**
- Clock frequency **X**

Explanation:

Die richtigen Antworten sind: Pin location, I/O standard, Current strength

Problem type: Quartus GPIO pin assignment configuration

Source: Moodle Quiz GPIO, Question 3

GPIO Driver Implementation

Question: What are the key components of a basic GPIO driver in Linux?

Wählen Sie eine oder mehrere Antworten:

- Initialization function **✓**
- Read/Write functions **✓**
- Interrupt handler (if interrupts are supported) **✓**
- DMA controller **X**

Explanation:

Die richtigen Antworten sind: Initialization function, Read/Write functions, Interrupt handler (if interrupts are supported)

Problem type: Understanding Linux GPIO driver components

Source: Moodle Quiz GPIO, Question 4

GPIO Pin Multiplexing

Question: In SoC FPGAs, GPIO pins can often be multiplexed with other functions. What does this mean?

Wählen Sie eine Antwort:

- The same physical pin can be configured for different purposes ✓
- Multiple GPIO pins share the same driver X
- GPIO pins can only be used for input or output, not both X
- GPIO pins require external multiplexer chips X

Explanation:

Die richtige Antwort ist: The same physical pin can be configured for different purposes

Problem type: Understanding GPIO pin multiplexing

Source: Moodle Quiz GPIO, Question 5

Exercise: JTAG

Quiz Format: Mixed (diagram matching, multiple choice, sequence ordering) **Score:** 1.00 von 1.00 (100%) **Topics Covered:** JTAG boundary scan, test modes, infrastructure components, TAP controller, timing sequences

JTAG Boundary Scan Cell

Diagram Matching - Boundary Scan Cell Components

Source: Exercise JTAG, Question 1 (Pages 1-2)

Topics: Boundary scan cell, JTAG architecture, bidirectional I/O

Question:

The above schematic shows a boundary scan cell for a bidirectional input/output pin of an FPGA. For each labelled part for the schematic, choose the matching description.

A: Reflects the logic state of an FPGA input pin, bypassing the JTAG facilities

D: Part of the scan chain

F: Allows the JTAG controller to set an output pin to high or low

G: Output or input pin

E: Allows the JTAG controller to set an output pin to high-impedance

B: Allows the customers logic implementation to set an output pin to high-impedance

C: Allows the customers logic implementation to set an output pin to high or low

Correct Answers:

A → Reflects the logic state of an FPGA input pin, bypassing the JTAG facilities

D → Part of the scan chain

F → Allows the JTAG controller to set an output pin to high or low

G → Output or input pin

E → Allows the JTAG controller to set an output pin to high-impedance

B → Allows the customers logic implementation to set an output pin to high-impedance

C → Allows the customers logic implementation to set an output pin to high or low

Detailed Explanation:

JTAG Boundary Scan Cell Architecture:

The boundary scan cell acts as an interface between the FPGA internal logic and external pins, allowing both normal operation and test modes.

Component Functions:

A - Input Path (Bypass):

- Direct connection from PIN_IN to internal logic
- Used during normal functional mode
- Reflects actual pin state without JTAG intervention
- Allows real-time monitoring of input signals

D - Scan Chain Cells:

- Flip-flops connected in series (SDI → SDO)
- Three cells shown: INPUT, OE, OUTPUT
- Shifted by CLOCK signal when SHIFT is active
- Updated to output by UPDATE signal
- Form continuous chain through all I/O pins

F - OUTPUT Test Cell:

- Multiplexer controlled by MODE signal
- In test mode: JTAG controller provides output value
- Allows external test equipment to drive pins
- Used for board-level interconnect testing

G - Physical Pin:

- Actual I/O pad on the FPGA package
- Can be input, output, or bidirectional
- Connected through output buffer and input buffer
- Interface to external circuitry

E - OE (Output Enable) Test Cell:

- Controls output buffer tri-state
- In test mode: JTAG controls high-Z state
- Allows isolation of outputs during testing
- Prevents bus conflicts in boundary scan

B - OE from Logic:

- Normal mode: user logic controls output enable
- Multiplexed with JTAG control (E)
- Implements bidirectional I/O functionality
- Allows tri-state control by FPGA fabric

C - OUTPUT from Logic:

JTAG Test Modes

Matching - JTAG Testing Types

Source: Exercise JTAG, Question 2 (Page 2)

Topics: JTAG test modes, structural test, functional test, boundary scan, BIST

Question:

Choose the matching test mode for each description about JTAG testing.

1. The test program tests the component without being aware of its function
2. Memory structures are tested with hardware dedicated for testing in the IC
3. Testing the connections of components on a printed circuit board
4. The component is tested for its behavior in normal operation

Options: Structural Test, Built In Selftest, Boundary Scan, Functional Test

Correct Answers:

1. The test program tests the component without being aware of its function → Structural Test
2. Memory structures are tested with hardware dedicated for testing in the IC → Built In Selftest
3. Testing the connections of components on a printed circuit board → Boundary Scan
4. The component is tested for its behavior in normal operation → Functional Test

Detailed Explanation:

1. Structural Test:

- Tests physical structure without knowing functionality
- Focuses on manufacturing defects
- Examples: stuck-at faults, shorts, opens
- Uses automatic test pattern generation (ATPG)
- Does not verify correct logical operation
- Fault coverage metrics (e.g., 95% stuck-at coverage)

Typical structural tests:

- Stuck-at-0 and stuck-at-1 faults
- Bridging faults (shorts between nets)
- Open circuit faults
- Transition delay faults
- Does not require functional specification

2. Built-In Self-Test (BIST):

- On-chip hardware generates test patterns
- Common for memory structures (MBIST)
- Reduces external test equipment requirements
- Tests run at-speed (actual operating frequency)
- Results compared with expected values on-chip

BIST components:

- Pattern generator (LFSR, counter, etc.)
- Response analyzer (signature register)
- Test controller (FSM)
- Typically tests RAMs, ROMs, register files

3. Boundary Scan:

- IEEE 1149.1 standard (JTAG)
- Tests board-level interconnections
- Detects PCB manufacturing defects
- No physical test probes required
- Tests shorts, opens between components

Boundary scan tests:

- EXTEST: Drive outputs, check inputs on adjacent chips
- INTEST: Test internal logic through boundary scan
- Sample pin values during normal operation
- Verify solder joints, PCB traces

4. Functional Test:

- Verifies correct logical behavior
- Tests actual specifications and requirements
- Uses real application scenarios
- Detects design errors, not just manufacturing faults
- Typically performed at system level

Functional test approaches:

JTAG Infrastructure Components

Matching - JTAG Component Functions

Source: Exercise JTAG, Question 3 (Pages 3-4)

Topics: TAP controller, JTAG signals, registers, state machine

Question:

Disclaimer: This exercise covers material we did not discuss in detail in the lecture. A potential problem in the exam will not be as detailed as this exercise.

Describe the Components of the JTAG infrastructure

1. Which JTAG Signal advances the states in the JTAG controller state machine
 2. What TAP controller signal will set the **Device ID Register** to allow its contents to be readout via the scan chain?
 3. Which JTAG signal controls the branching of the states in the JTAG controller state machine
 4. With which JTAG signal, IC test equipment receives the contents of the scan chains
 5. Which JTAG Register transfers data directly from TDI to TDO
 6. Which JTAG register controls and checks the IC self tests
 7. Which JTAG Register determines which scan chain is currently active
 8. Which JTAG signal disables the entire JTAG infrastructure
- Options: TCK, SHIFTDR, TMS, TDO, Bypass Register, MBIST Register, Instruction Register, TRST

Correct Answers:

1. Which JTAG Signal advances the states in the JTAG controller state machine → TCK
2. What TAP controller signal will set the Device ID Register to allow its contents to be readout via the scan chain? → SHIFTDR
3. Which JTAG signal controls the branching of the states in the JTAG controller state machine → TMS
4. With which JTAG signal, IC test equipment receives the contents of the scan chains → TDO
5. Which JTAG Register transfers data directly from TDI to TDO → Bypass Register
6. Which JTAG register controls and checks the IC self tests → MBIST Register
7. Which JTAG Register determines which scan chain is currently active → Instruction Register
8. Which JTAG signal disables the entire JTAG infrastructure → TRST

Detailed Explanation:

JTAG Signals (TAP - Test Access Port):

1. TCK (Test Clock):

- Clock signal for TAP controller
- Advances state machine on rising edge
- Synchronizes all JTAG operations
- Independent of system clock
- Typically 1-10 MHz (slower than system clock)

2. SHIFTDR (Shift Data Register):

- TAP controller output signal
- Enables shifting in selected data register
- When active: data shifts through Device ID, MBIST, or Bypass register
- Controlled by TAP state machine
- Used for both reading and writing register contents

3. TMS (Test Mode Select):

- Input that controls state machine branching
- Sampled on rising edge of TCK
- Sequence of TMS values navigates through states
- Determines whether to go to IR or DR path
- Five consecutive '1's resets to Test-Logic-Reset state

4. TDO (Test Data Out):

- Serial output from JTAG chain
- Connects to test equipment
- Multiplexed from active register
- Changes on falling edge of TCK
- High-impedance when not shifting data

JTAG Registers:

5. Bypass Register:

- Single-bit register
- Provides shortest path from TDI to TDO
- Used to bypass devices in multi-device chain

JTAG Operation Sequence

Sequence Ordering - JTAG Timing Diagram

Source: Exercise JTAG, Question 4 (Page 4)

Topics: JTAG timing, TAP states, data capture and shift

Question:

Match the descriptions to the numbered phases in the JTAG timing diagram:

1. Pin input signals are captured
2. Use instruction code to select which scan chain is to be used
3. Captured data are shifted out
4. Return to idle state

Options: 1, 2, 3, 4

Correct Answers:

Pin input signals are captured → 2

Use instruction code to select which scan chain is to be used → 1

Captured data are shifted out → 3

Return to idle state → 4

Detailed Explanation:

JTAG Operation Sequence:

The timing diagram shows a complete JTAG test sequence with four distinct phases:

Phase 1 - Instruction Loading:

- State: SHIFT_IR (Shift Instruction Register)
- TDI shifts in instruction code
- Instruction determines which data register to use (EXTEST, INTEST, etc.)
- Multiple TCK cycles shift entire instruction
- Transitions through EXIT1_IR → UPDATE_IR
- UPDATE_IR latches new instruction

Example instructions:

- EXTEST: Select boundary scan register
- SAMPLE: Sample pin values
- BYPASS: Select bypass register
- IDCODE: Select device ID register

Phase 2 - Data Capture:

- State: CAPTURE_DR (Capture Data Register)
- Pin input signals are captured into scan cells
- One TCK cycle in CAPTURE_DR state
- Parallel load from pins to boundary scan register
- Prepares data for shifting out
- Transitions through SELECT_DR to get here

What gets captured:

- Current logic levels on input pins
- Stored in input boundary scan cells
- Output values also captured
- Forms snapshot of pin states

Phase 3 - Data Shift:

- State: SHIFT_DR (Shift Data Register)
- Captured data shifts out via TDO
- New test data can shift in via TDI simultaneously
- Multiple TCK cycles (one per bit in scan chain)
- Test equipment receives pin states
- Can load new values for output pins

During shift:

- TDO shows previous cell contents
- TDI loads next test pattern
- Entire scan chain shifts each TCK
- Takes N cycles for N-bit chain

Phase 4 - Return to Idle:

- State: EXIT1_DR → UPDATE_DR → Run-Test/Idle
- UPDATE_DR: New values latched to output pins
- Run-Test/Idle: TAP controller idle state

KR and Exercises: JTAG

JTAG Boundary Scan Cell

Boundary Scan Cell Components

Question: The above schematic shows a boundary scan cell for a bidirectional input/output pin of an FPGA. For each labelled part for the schematic, choose the matching description.

- A: _____
D: _____
F: _____
G: _____
E: _____
B: _____
C: _____

Explanation:

Your answer is correct.

Die richtigen Antworten sind:

- A → Reflects the logic state of an FPGA input pin, bypassing the JTAG facilities,
D → Part of the scan chain,
F → Allows the JTAG controller to set an output pin to high or low,
G → Output or input pin,
E → Allows the JTAG controller to set an output pin to high-impedance,
B → Allows the customers logic implementation to set an output pin to high-impedance,
C → Allows the customers logic implementation to set an output pin to high or low

Problem type: Understanding JTAG boundary scan cell architecture

Source: Moodle Quiz JTAG, Question 1

JTAG Test Modes

JTAG Testing Types

Question: Choose the matching test mode for each description about JTAG testing.

1. The test program tests the component without being aware of its function: _____
2. Memory structures are tested with hardware dedicated for testing in the IC: _____
3. Testing the connections of components on a printed circuit board: _____
4. The component is tested for its behavior in normal operation: _____

Options: Structural Test, Built In Selftest, Boundary Scan, Functional Test

Explanation:

Your answer is correct.

Die richtige Antwort ist: The test program tests the component without being aware of its function → Structural Test, Memory structures are tested with hardware dedicated for testing in the IC → Built In Selftest, Testing the connections of components on a printed circuit board → Boundary Scan, The component is tested for its behavior in normal operation → Functional Test

Problem type: Understanding different JTAG testing methodologies

Source: Moodle Quiz JTAG, Question 2

JTAG Infrastructure

JTAG Component Functions

Question: Disclaimer: This exercise covers material we did not discuss in detail in the lecture. A potential problem in the exam will not be as detailed as this exercise.

Describe the Components of the JTAG infrastructure

1. Which JTAG Signal advances the states in the JTAG controller state machine: _____
2. What TAP controller signal will set the **Device ID Register** to allow its contents to be readout via the scan chain?: _____
3. Which JTAG signal controls the branching of the states in the JTAG controller state machine: _____
4. With which JTAG signal, IC test equipment receives the contents of the scan chains: _____
5. Which JTAG Register transfers data directly from TDI to TDO: _____
6. Which JTAG register controls and checks the IC self tests: _____
7. Which JTAG Register determines which scan chain is currently active: _____
8. Which JTAG signal disables the entire JTAG infrastructure: _____

Options: TCK, SHIFTDR, TMS, TDO, Bypass Register, MBIST Register, Instruction Register, TRST

Explanation:

Your answer is correct.

Die richtigen Antworten sind:

Which JTAG Signal advances the states in the JTAG controller state machine → TCK,

What TAP controller signal will set the Device ID Register to allow its contents to be readout via the scan chain? → SHIFTDR,

Which JTAG signal controls the branching of the states in the JTAG controller state machine → TMS,

With which JTAG signal, IC test equipment receives the contents of the scan chains → TDO,

Which JTAG Register transfers data directly from TDI to TDO → Bypass Register,

Which JTAG register controls and checks the IC self tests → MBIST Register,

Which JTAG Register determines which scan chain is currently active → Instruction Register,

Which JTAG signal disables the entire JTAG infrastructure → TRST

Problem type: Understanding JTAG architecture and signal functions

Source: Moodle Quiz JTAG, Question 3

JTAG Operation Sequence

JTAG Timing Sequence

Question: Match the descriptions to the numbered phases in the JTAG timing diagram:

Pin input signals are captured: _____

Use instruction code to select which scan chain is to be used: _____

Captured data are shifted out: _____

Return to idle state: _____

Options: 1, 2, 3, 4

Explanation:

Your answer is correct.

Die richtige Antwort ist: Pin input signals are captured → 2, Use instruction code to select which scan chain is to be used → 1, Captured data are shifted out → 3, Return to idle state → 4

Problem type: Understanding JTAG operational sequence

Source: Moodle Quiz JTAG, Question 4

Exercise: Timing Constraints

Quiz Format: Multiple choice **Score:** 5.00 von 5.00 (100%) **Duration:** 6 Minutes 21 Seconds **Topics Covered:** Temperature effects on timing, clock frequency limitations, setup and hold time analysis, timing diagrams, clock domain crossing

Temperature Effects on Timing

Multiple Choice - Temperature and Timing Slack

Source: Exercise Timing Constraints, Question 1 (Page 1)

Topics: Temperature effects, timing slack, setup time, hold time

Question:

Which statement is true for the schematic above? (Several answers may be correct)

Select one or more answers:

- a. At high temperatures the circuit can be operated at higher clock frequencies
- b. At high temperatures the slack at hold time checks gets larger
- c. At low temperatures the slack at setup checks gets larger
- d. At high temperatures the slack at hold time checks gets smaller

Correct Answers:

- b. At high temperatures the slack at hold time checks gets larger
- c. At low temperatures the slack at setup checks gets larger

Explanation:

Die richtigen Antworten sind: At high temperatures the slack at hold time checks gets larger, At low temperatures the slack at setup checks gets larger

Temperature Effects on Digital Circuits:

General principle:

- Higher temperature → Slower transistors → Longer delays
- Lower temperature → Faster transistors → Shorter delays
- This affects both logic delays and flip-flop timing parameters

Option a - **WRONG:** 'At high temperatures... higher clock frequencies'

- FALSE: Higher temperatures slow down circuits
- Longer delays reduce maximum achievable frequency
- Critical paths take longer to resolve
- Must reduce clock frequency or risk timing violations

Option b - **CORRECT:** 'At high temperatures slack at hold time checks gets larger'

- TRUE: Hold time checks benefit from slower operation
- Hold time equation: $t_{slack,hold} = t_{arrival} - t_{required}$
- At high temp: Data arrives later (longer logic delay)
- But hold requirement stays constant
- Result: More slack (safer for hold time)
- Data changes more slowly, staying stable longer

Option c - **CORRECT:** 'At low temperatures slack at setup checks gets larger'

- TRUE: Setup time checks benefit from faster operation
- Setup time equation: $t_{slack,setup} = t_{required} - t_{arrival}$
- At low temp: Data arrives earlier (shorter logic delay)
- Clock period requirement stays the same
- Result: More slack (safer for setup time)
- More time available before clock edge

Option d - **WRONG:** 'At high temperatures slack at hold time checks gets smaller'

- FALSE: This is opposite of reality
- High temperatures actually increase hold slack
- Slower operation makes hold timing easier, not harder

Summary of Temperature Effects:

Condition	Setup Timing	Hold Timing
High Temp (Slow)	Worse (less slack)	Better (more slack)
Low Temp (Fast)	Better (more slack)	Worse (less slack)

Design implications:

- Must check timing across full temperature range
- Setup: Worst at high temperature (slow corner)
- Hold: Worst at low temperature (fast corner)
- Multi-corner timing analysis essential
- Typical corners: Fast/Slow process, High/Low voltage, High/Low temperature

Problem type: Analysis - Understanding temperature effects on digital timing

Maximum Clock Frequency Limitations

Multiple Choice - Clock Frequency Failures

Source: Exercise Timing Constraints, Question 2 (Page 2)

Topics: Clock frequency, timing violations, constraints, resource utilization

Question:

What could be the cause if the maximum clock frequency is not reached?

Select one or more answers:

- A. The FPGA is operated too cold
- B. Complex Logic, too many logic cells are cascaded
- C. The device is utilized up to 90% and the maximum clock frequency is constraint to a high value
- D. The logic cells were placed too distant
- E. The operating voltage is at the upper limit
- F. Timing constraints for the particular clock were not assigned

Correct Answers:

- B. Complex Logic, too many logic cells are cascaded
- C. The device is utilized up to 90% and the maximum clock frequency is constraint to a high value
- D. The logic cells were placed too distant
- F. Timing constraints for the particular clock were not assigned

Explanation:

Die richtigen Antworten sind: The device is utilized up to 90% and the maximum clock frequency is constraint to a high value, Timing constraints for the particular clock were not assigned, Complex Logic, too many logic cells are cascaded, The logic cells were placed too distant

Analysis of Each Option:

A - WRONG: 'FPGA is operated too cold'

- Cold operation makes circuits faster, not slower
- Would actually help achieve higher frequencies
- Not a cause of frequency limitation

B - CORRECT: 'Complex Logic, too many logic cells are cascaded'

- Long combinational paths are primary frequency limiters
- Each LUT adds delay (typically 0.2-1 ns)
- Deep logic trees create critical paths
- Example: 10 cascaded LUTs might add 5-10 ns delay
- Solution: Pipeline the design (add registers)

C - CORRECT: 'Device utilized up to 90%... high constraint'

- High utilization limits placement options
- Placer cannot optimize for timing effectively
- Aggressive timing constraints may be unachievable
- Congestion increases routing delays
- Trade-off between area and performance
- Solution: Reduce utilization or relax constraints

D - CORRECT: 'Logic cells were placed too distant' [distant]

- Long routing distances add significant delay
- Wire delays dominate in modern FPGAs
- Poor floorplanning increases critical path lengths
- Solution: Better placement constraints, floorplanning
- Use LOC constraints for critical logic

E - WRONG: 'Operating voltage is at the upper limit'

- Higher voltage makes circuits faster
- Increases drive strength and reduces delays
- Would help achieve higher frequency, not hinder

F - CORRECT: 'Timing constraints... not assigned'

- Without constraints, tools don't optimize for timing
- Synthesis/PAR tools prioritize area over speed
- No timing goals means arbitrary placement/routing
- Essential to define clock constraints
- Use: create_clock, set_input_delay, set_output_delay

Common Causes of Frequency Limitations:

1. **Logic depth:** Too many levels of combinational logic
2. **Routing congestion:** High utilization or poor floorplan

Clock Domain Crossing Behavior

Multiple Choice - CDC Timing Diagram Analysis

Source: Exercise Timing Constraints, Question 3 (Page 3)

Topics: Clock domain crossing, flip-flop timing, metastability

Question:

The following time diagram shows the behavior of the Q output of FF2. Which statement is correct?

Select one or more answers:

- The circuit probably behaves as in A) when the switch is in position '1'
- When the timing requirements of Flip-Flops are met, slower Flip-Flops behave like B), faster Flip-Flops behave like C)
- If the switch is in position 0, the data is synchronized with the 50 MHz clock
- The circuit probably behaves as in B) or C) if the switch is in position '0'

Correct Answers:

- The circuit probably behaves as in B) or C) if the switch is in position '0'
- The circuit probably behaves as in A) when the switch is in position '1'

Explanation:

Die richtigen Antworten sind: The circuit probably behaves as in B) or C) if the switch is in position '0', The circuit probably behaves as in A) when the switch is in position '1'

Circuit Analysis:

Configuration:

- FF1: Clocked at 50 MHz
- FF2: Clocked at 33 MHz
- Switch selects input to FF2: Position '1' = synchronized, Position '0' = async CDC

Position '1' - Synchronized Path:

- Data properly synchronized before FF2
- Setup and hold times are met
- Clean clock domain crossing
- Result: Waveform A - clean, deterministic transitions
- No metastability issues

Position '0' - Asynchronous CDC:

- Direct clock domain crossing (50 MHz → 33 MHz)
- Timing requirements may be violated
- Risk of setup/hold violations
- Can enter metastable state

Waveform Analysis:

Waveform A: Clean transitions

- Sharp, well-defined edges
- Proper synchronization
- Occurs when switch in position '1'
- Timing requirements satisfied

Waveform B: Slow transition

- Gradual rise/fall
- Indicates slower settling
- Possible with async CDC (position '0')
- Flip-flop took longer to resolve
- Still eventually settles to valid logic level

Waveform C: Oscillation/ringing

- Multiple transitions before settling
- Classic metastability behavior
- Can occur with async CDC (position '0')
- Timing violation caused metastable state
- Eventually resolves but with unpredictable delay

Metastability Explanation:

When setup/hold times are violated:

1. Flip-flop internal nodes enter undefined state
2. Output voltage hovers near threshold
3. Small noise can push output either direction
4. Results in slow resolution or oscillation
5. Resolution time is non-deterministic

Setup Time Analysis

Multiple Choice - Launch-to-Latch Timing

Source: Exercise Timing Constraints, Question 4 (Page 4)

Topics: Setup time, launching edge, latching edge, arrival time, required time

Question:

Which statements apply to the above drawing?

Select one or more answers:

- This is a hold time analysis
- This is a setup time analysis
- 'C' is the Arrival time
- 'C' is the Required time
- 'B' is the Slack
- 'A' is the Arrival time
- 'A' is the Required time

Correct Answers:

- This is a setup time analysis
- 'B' is the Slack
- 'C' is the Required time
- 'A' is the Arrival time

Explanation:

Die Antwort ist richtig

Die richtigen Antworten sind: This is a setup time analysis, 'B' is the Slack, 'C' is the Required time, 'A' is the Arrival time

Setup Time Analysis Fundamentals:

Definition:

- Setup time: Data must be stable BEFORE clock edge
- Checks if data arrives early enough
- Analysis from launching edge to latching edge
- Covers one full clock cycle (or more for multi-cycle paths)

This is a SETUP time analysis:

- Diagram shows launching edge to latching edge
- Full clock period is considered
- Data must arrive before latch clock edge
- Hold time would analyze same edge or next edge

Timing Parameters Identified:

'A' is the Arrival Time:

- Starts at launching flip-flop clock edge
- Includes: t_{CLKBQ} + t_{logic} + $t_{routing}$
- When data actually arrives at destination FF
- Measured from launch clock edge
- Example: Launch at 0 ns, data arrives at 4.5 ns \rightarrow Arrival = 4.5 ns

'C' is the Required Time:

- Latest time data can arrive and still meet setup
- Calculated as: Clock Period - Setup Time
- Example: 20 ns period, 0.5 ns setup \rightarrow Required = 19.5 ns
- Data must arrive before this time
- Defines the deadline

'B' is the Slack:

- Difference between required and arrival
- Slack = Required Time - Arrival Time
- Slack = C - A
- Positive slack: Timing is met (good)
- Negative slack: Timing violation (bad)
- Zero slack: Just barely meets timing

Setup Time Equation:

$$t_{slack,setup} = t_{required} - t_{arrival}$$

Hold Time Analysis

Multiple Choice - Hold Time Diagram

Source: Exercise Timing Constraints, Question 5 (Page 5)

Topics: Hold time, timing analysis, arrival time, required time, slack

Question:

Which statements apply to the above drawing?

Select one or more answers:

- This is a setup analysis
- This is a hold time analysis
- 'A' is the Arrival time
- 'B' is the Slack
- 'A' is the Required time
- 'C' is the Arrival Time
- 'B' is the Required Time

Correct Answers:

- This is a hold time analysis
- 'B' is the Slack
- 'A' is the Required time
- 'C' is the Arrival Time

Explanation:

Die Antwort ist richtig

Die richtigen Antworten sind: This is a hold time analysis, 'B' is the Slack, 'C' is the Arrival Time, 'A' is the Required time

Hold Time Analysis Fundamentals:

Definition:

- Hold time: Data must remain stable AFTER clock edge
- Checks if data stays stable long enough
- Analysis from latching edge to data change
- Typically checks same clock edge or very shortly after

This is a HOLD time analysis:

- Diagram shows latching edge to next data change
- Short time window (not full clock period)
- Focus on data stability after capture
- Labeled 'Latching FF-Hold' confirms this

Timing Parameters Identified:

'A' is the Required Time (Hold Requirement):

- Minimum time data must remain stable
- Hold time requirement of latching flip-flop
- Measured from latching clock edge
- Defines minimum delay needed
- Typically 0-0.5 ns in modern FPGAs

'C' is the Arrival Time:

- When new data actually arrives
- Includes: $t_{CLKBQ} + t_{logic} + t_{routing}$ of shortest path
- Measured from latching clock edge
- Must be longer than hold requirement
- For hold: Want LONGER paths (opposite of setup!)

'B' is the Slack:

- Difference between arrival and required
- Slack = Arrival Time - Required Time (note: different from setup!)
- Slack = C - A
- Positive slack: Hold time met
- Negative slack: Hold violation

Hold Time Equation:

$$t_{slack,hold} = t_{arrival} - t_{required}$$

$$t_{slack,hold} = (t_{CLKBQ} + t_{logic} + t_{routing}) - t_{hold}$$

Exercise: Timing Analysis

Quiz Format: Analysis of timing diagrams and spreadsheets Score: 6.00 von 6.00 (100%) Topics Covered: FPGA timing analysis, setup and hold time, timing paths, clock periods, delay calculations

FPGA Timing Analysis 1

Calculation - Setup Time Analysis from Timing Report

Source: Exercise Timing Analysis, Question 1 (Pages 1)

Topics: Setup time analysis, timing paths, clock network delay, data path delay

Question:

Look at the timing analysis spreadsheet and timing diagram to answer the following questions:

- Does the timing diagram above show a setup or hold-time analysis?
- What is the clock period in the analysis (ns)?
- What is the delay contribution of the clock network of the arrival path? (ns)
- What is the delay contribution between the entry of the input pin AE20 up to the Latch Flip-Flop in the FPGA? (ns)
- What is the delay contribution of the clock path between the input of the CLOCK_50 pin up to the clock input of the Latch Flip-Flop in the FPGA? (ns)
- What is the setup time value of the FPGA Flip-Flop? (ns)

Correct Answers:

a) Setup

This is a setup time analysis because it shows the full clock period and analyzes whether data arrives before the required time at the latching edge.

b) Clock period: 20 ns

From the timing diagram and spreadsheet, the clock period (latch edge time) is 20.000 ns.

c) Clock network delay of arrival path: 0 ns

The arrival path starts from an external input pin (GPIO[30]), not from a clocked flip-flop. Therefore, there is no clock network delay in the arrival path. The path goes directly from input pin through combinational logic to the destination.

d) Delay from input pin AE20 to Latch FF: 6.702 ns

From the Data Arrival Path table:

- Row 1: Launch edge time = 0.000
- Row 2: Clock path = 0.000
- Row 3: Clock network delay = 0.000 (R - Rising edge)
- Row 4: GPIO[30] input = 3.000 ns (from IOBUF to data path)
- Following rows show combinational logic delays
- Total arrival = 6.702 ns (sum of all incremental delays)

e) Clock path delay from CLOCK_50 pin to Latch FF clock input: 2.937 ns

From the Data Required Path table:

- Row 1: Latch edge time = 20.000
- Row 2: Clock path = 0.000
- Row 3: Source latency = 0.000
- Row 4: CLOCK_50 entry = 0.000 (PIN_Y2)
- Row 5: Clock network delay to IOBUF = 0.704 ns
- Row 6: Through CELL to CLKCTRL = 0.704 ns
- Additional delays through clock network
- Clock path delay = 2.937 ns (from pin to FF clock input)

f) Setup time of FPGA Flip-Flop: 0.018 ns

From the timing analysis, the setup time requirement is listed in the spreadsheet. Modern FPGA flip-flops typically have very small setup times (10-100 ps range). The value 0.018 ns = 18 ps is typical for advanced FPGA technology.

Detailed Timing Analysis Explanation:

Setup Time Check Equation:

$$Slack = (T_{clk} + t_{clk,dest} - t_{setup}) - (t_{clk,src} + t_{logic} + t_{route})$$

For this external input case:

- Launch edge: 0 ns (external signal, arbitrary reference)
- Data arrival: 6.702 ns (input pin delay + logic delay)
- Latch edge: 20.000 ns (clock period)
- Clock network to latch FF: 2.937 ns
- Setup time: 0.018 ns
- Required time: $20.000 - 2.937 - 0.018 = 17.045$ ns (approximately)
- Slack: $17.045 - 6.702 = \text{positive slack (timing met!)}$

FPGA Timing Analysis 2

Calculation - Hold Time Analysis from Timing Report

Source: Exercise Timing Analysis, Question 2 (Page 2)

Topics: Hold time analysis, minimum delays, clock network delays

Question:

Look at the timing analysis spreadsheet and timing diagram to answer the following questions:

- Is this a setup or hold-time analysis?
- What is the value defined for the 'Input Delay' at the data input of the FPGA? (ns)
- How long is the clock path from the input of the CLOCK_50 pin to the Latch Flip-Flop of in the FPGA? (ns)
- How long is the Required Path for FPGA Flip-Flop: (ns)
- What is the hold time value of the FPGA Flip-Flop? (ns)

Correct Answers:

a) Hold

This is a hold time analysis as indicated by the timing diagram label 'Hold Relationship' and the short time window analyzed (not full clock period).

b) Input Delay: 3 ns

From the Data Arrival Path table, the GPIO[30] input has an initial delay of 3.000 ns. This represents the set_input_delay constraint defining when external data is expected to arrive relative to the clock edge.

c) Clock path from CLOCK_50 to Latch FF clock input: 2.762 ns

From the Data Required Path table, tracing from PIN_Y2 (CLOCK_50) through the clock network:

- CLOCK_50 entry: 0.000 ns
- Through IOBUF_X0_Y36_N15: incremental delays
- Through CLKCTRL_G4: clock routing
- Total clock path: 2.762 ns to reach FF clock input

d) Required Path for FPGA Flip-Flop: 2.933 ns

The Required Path in hold analysis represents the minimum time data must remain stable. From the spreadsheet:

- This includes clock path delays
- Plus the hold time requirement
- Total Required Path: 2.933 ns

e) Hold time of FPGA Flip-Flop: 0.171 ns

From the timing analysis spreadsheet, the hold time requirement is 0.171 ns = 171 ps. This is the minimum time data must remain stable after the clock edge.

Detailed Hold Time Analysis:

Hold Time Check Equation:

$$\text{Slack} = (t_{clk,src} + t_{logic} + t_{route}) - (t_{clk,dest} + t_{hold})$$

For this analysis:

- Data arrival: Minimum path delay from input
- Clock arrival at latch FF: 2.762 ns
- Hold requirement: 0.171 ns
- Required time: $2.762 + 0.171 = 2.933$ ns
- Data must not change before this time

Hold vs. Setup Analysis Comparison:

Parameter	Setup (Q1)	Hold (Q2)
Analysis type	Setup	Hold
Clock period considered	Full (20 ns)	Single edge
Path delays	Maximum	Minimum
Timing window	Long	Short
Setup time	0.018 ns	N/A
Hold time	N/A	0.171 ns
Clock path	2.937 ns	2.762 ns
Input delay	Implicit	3.000 ns

Key Observations for Hold Analysis:

- Input delay: 3 ns constraint defines when external data can change
- Minimum path: Hold uses shortest delays (opposite of setup)

KR and Exercises: Timing Constraints

Temperature Effects on Timing

Temperature and Timing Slack

Question: Which statement is true for the schematic above? (Several answers may be correct)

Wählen Sie eine oder mehrere Antworten:

- At high temperatures the circuit can be operated at higher clock frequencies X
- At high temperatures the slack at hold time checks gets larger ✓
- At low temperatures the slack at setup checks gets larger ✓
- At high temperatures the slack at hold time checks gets smaller X

Explanation:

At high temperatures the slack at hold time checks gets larger, At low temperatures the slack at setup checks gets larger

Problem type: Understanding temperature effects on timing

Source: Moodle Quiz Timing Constraints, Question 1

Maximum Clock Frequency

Question: What could be the cause if the maximum clock frequency is not reached?

Wählen Sie eine oder mehrere Antworten:

- The FPGA is operated too cold **X**
- Complex Logic, too many logic cells are cascaded **✓**
- The device is utilized up to 90% and the maximum clock frequency is constraint to a high value **✓**
- The logic cells were placed too distand **✓**
- The operating voltage is at the upper limit **X**
- Timing constraints for the particular clock were not assigned **✓**

Explanation:

Die richtigen Antworten sind: The device is utilized up to 90% and the maximum clock frequency is constraint to a high value, Timing constraints for the particular clock were not assigned, Complex Logic, too many logic cells are cascaded, The logic cells were placed too distand

Problem type: Identifying causes of timing failures

Source: Moodle Quiz Timing Constraints, Question 2

Clock Domain Crossing

Question: The following time diagram shows the behavior of the Q output of FF2. Which statement is correct?

Wählen Sie eine oder mehrere Antworten:

- The circuit probably behaves as in A) when the switch is in position '1'. **✓**
- When the timing requirements of Flip-Flops are met, slower Flip-Flops behave like B), faster Flip-Flops behave like C). **X**
- If the switch is in position 0, the data is synchronized with the 50 MHz clock. **X**
- The circuit probably behaves as in B) or C) if the switch is in position '0'. **✓**

Explanation:

Die richtigen Antworten sind: The circuit probably behaves as in B) or C) if the switch is in position '0'., The circuit probably behaves as in A) when the switch is in position '1'.

Problem type: Clock domain crossing and metastability

Source: Moodle Quiz Timing Constraints, Question 3

Setup Time Analysis

Question: Which statements apply to the above drawing?

Wählen Sie eine oder mehrere Antworten:

- This is a hold time analysis **X**
- This is a setup time analysis **✓**
- 'C' is the Arrival time **X**
- 'C' is the Required time **✓**
- 'B' is the Slack **✓**
- 'A' is the Arrival time **✓**
- 'A' is the Required time **X**

Explanation:

Die richtigen Antworten sind: This is a setup time analysis, 'B' is the Slack, 'C' is the Required time, 'A' is the Arrival time

Problem type: Understanding setup time analysis

Source: Moodle Quiz Timing Constraints, Question 4

Hold Time Analysis

Question: Which statements apply to the above drawing?

Wählen Sie eine oder mehrere Antworten:

- This is a setup analysis **X**
- This is a hold time analysis **✓**
- 'A' is the Arrival time **X**
- 'B' is the Slack **✓**
- 'A' is the Required time **✓**
- 'C' is the Arrival Time **✓**
- 'B' is the Required Time **X**

Explanation:

Die richtigen Antworten sind: This is a hold time analysis, 'B' is the Slack, 'C' is the Arrival Time, 'A' is the Required time

Problem type: Understanding hold time analysis

Source: Moodle Quiz Timing Constraints, Question 5

KR and Exercises: Timing Analysis

FPGA Timing Analysis

Timing Analysis 1

Question: Look at the timing analysis spreadsheet and timing diagram to answer the following questions:

a) Does the timing diagram above show a setup or hold-time analysis?

Wählen Sie eine Antwort:

- Setup ✓
- Hold X

b) What is the clock period in the analysis _____ ns

c) What is the delay contribution of the clock network of the arrival path? _____ ns

d) What is the delay contribution between the entry of the input pin AE20 up to the Latch Flip-Flop in the FPGA? _____ ns

e) What is the delay contribution of the clock path between the input of the CLOCK_50 pin up to the clock input of the Latch Flip-Flop in the FPGA? _____ ns

f) What is the setup time value of the FPGA Flip-Flop? _____ ns

Explanation:

Correct answers:

- a) Setup
- b) 20 ns
- c) 0 ns
- d) 6.702 ns
- e) 2.937 ns
- f) 0.018 ns

Problem type: Timing analysis calculation - extracting values from timing reports

Source: Moodle Quiz Timing Analysis, Question 1

Timing Analysis 2

Question: Look at the timing analysis spreadsheet and timing diagram to answer the following questions:

a) Is this a setup or hold-time analysis?

Wählen Sie eine Antwort:

- Setup X
- Hold ✓

b) What is the value defined for the 'Input Delay' at the data input of the FPGA? _____ ns

c) How long is the clock path from the input of the CLOCK_50 pin to the Latch Flip-Flop of in the FPGA?
_____ ns

d) How long is the Required Path for FPGA Flip-Flop: _____ ns

e) What is the hold time value of the FPGA Flip-Flop? _____ ns

Explanation:

Correct answers:

- a) Hold
- b) 3 ns
- c) 2.762 ns
- d) 2.933 ns
- e) 0.171 ns

Problem type: Timing analysis calculation - hold time analysis

Source: Moodle Quiz Timing Analysis, Question 2

Exercise: PLL

Quiz Format: Fill-in calculation **Score:** 7.00 von 7.00 (100%) **Topics Covered:** PLL configuration, frequency synthesis, phase delays, dividers, VCO operation

PLL Configuration in FPGA

Calculation - PLL Divider and Delay Configuration

Source: Exercise PLL, Question 1 (Page 1)

Topics: PLL dividers, output clocks, phase delays, frequency multiplication

Question:

The PLL shown above is operated at 100 MHz on the CLK_IN_PIN. The time history diagram above shows the internal and the output clocks.

Fill in the necessary configurations in the table below.

Example: An entry 4 in the columns at Divider means that the period is divided by 4. The entries for delays 'Vn' are in nanoseconds (ns).

	Divider n	Divider m	Divider g	Divider h	Delay V1	Delay V2	Delay V3
Einstellung	_____	_____	_____	_____	_____ ns	_____ ns	_____ ns

Correct Answers:

	Divider n	Divider m	Divider g	Divider h	Delay V1	Delay V2	Delay V3
Einstellung	6	3	2	4	5 ns	0 ns	10 ns

Detailed Explanation and Calculations:

Given Information:

- Input clock (CLK_IN_PIN): 100 MHz = 10 ns period
- From timing diagram, we can observe the output clock frequencies and phases

PLL Operation Principle:

A PLL consists of:

1. Pre-scale divider (n): Divides input clock before phase comparator
2. Phase comparator: Compares feedback with divided input
3. VCO: Generates high-frequency output based on comparator
4. Feedback divider (m): Divides VCO output for comparison
5. Post-scale dividers (g, h): Create output clocks from VCO
6. Programmable delays (V1, V2, V3): Adjust output phases

Fundamental PLL Equation:

$$f_{VCO} = f_{in} \times \frac{m}{n}$$

Step-by-Step Solution:

Step 1: Analyze the Fn signal

From the timing diagram, Fn appears to have a period that is 6 times the input period:

- Input period: 10 ns
- Fn period: 60 ns (approximately, from diagram)
- Therefore: **Divider n = 6**

Step 2: Analyze the Feedback signal

The feedback signal must match Fn for PLL lock. From the diagram, the Feedback signal has the same frequency as Fn but is derived from Fpll divided by m.

The feedback appears to toggle every 60 ns period.

Step 3: Determine VCO frequency and divider m

From the Fpll waveform in the timing diagram:

- Fpll toggles much faster than the input
- Count cycles: approximately 6 toggles per input period
- Fpll period $\approx 10/6$ ns (actually counting from diagram)
- Looking at the relationship: Fpll = 200 MHz (5 ns period)

Using PLL equation:

$$f_{VCO} = 100 \text{ MHz} \times \frac{m}{6}$$

For 200 MHz VCO:

$$200 = 100 \times \frac{m}{6}$$

$$m = \frac{200 \times 6}{100} = 12$$

Wait, let me recalculate based on feedback matching Fn:

KR and Exercises: PLL

PLL Configuration

PLL Divider and Delay Configuration

Question: The PLL shown above is operated at 100 MHz on the CLK_IN_PIN. The time history diagram above shows the internal and the output clocks.

Fill in the necessary configurations in the table below.

Example: An entry 4 in the columns at Divider means that the period is divided by 4. The entries for delays 'Vn' are in nanoseconds (ns).

	Divider n	Divider m	Divider g	Divider h	Delay V1	Delay V2	Delay V3
Einstellung	_____	_____	_____	_____	_____ ns	_____ ns	_____ ns

Explanation:

Correct answers:

	Divider n	Divider m	Divider g	Divider h	Delay V1	Delay V2	Delay V3
Einstellung	6	3	2	4	5 ns	0 ns	10 ns

Problem type: PLL configuration calculation from timing diagram

Source: Moodle Quiz PLL, Question 1

Exercise: I/O Interfaces

Quiz Format: Multiple choice **Score:** 1.00 von 1.00 (100%) **Topics Covered:** LVDS signaling standard, differential signaling advantages, fault tolerance, signal integrity

LVDS Signaling Standard

Multiple Choice - LVDS Advantages

Source: Exercise I/O Interfaces, Question 1 (Page 1)

Topics: LVDS, low voltage differential signaling, current-mode drivers

Question:

What are the Advantages of the Low Voltage Differential (LVDS) Standard

Select one or more answers:

- Higher data transmission rates since there are only small voltage swings required
- Robust against shortcircuits since transmitter is current limited
- Low power consumption since transmitter limits maximum current
- Available also as single ended standard
- Fault-tolerant in case one line is interrupted, since there are two lines per signal
- Can not be used for clock lines
- Lower cost compared to single ended signaling
- High noise immunity due to compensation on differential lines

Correct Answers:

- Higher data transmission rates since there are only small voltage swings required
- Robust against shortcircuits since transmitter is current limited
- Low power consumption since transmitter limits maximum current
- High noise immunity due to compensation on differential lines

Explanation:

Your answer is correct.

Die richtigen Antworten sind:

- Low power consumption since transmitter limits maximum current
- Higher data transmission rates since there are only small voltage swings required
- Robust against shortcircuits since transmitter is current limited
- High noise immunity due to compensation on differential lines

Detailed Analysis of Each Option:

CORRECT: 'Higher data transmission rates since there are only small voltage swings required'

- LVDS uses 350 mV differential swing (very small)
- Small voltage swings charge/discharge parasitic capacitances faster
- Lower slew rate requirements
- Can achieve multi-Gbps data rates
- Compare to CMOS: 3.3V or 5V swings much slower
- Typical LVDS: 100 Mbps to several Gbps

CORRECT: 'Robust against shortcircuits since transmitter is current limited'

- Current-mode driver: Fixed 3.5 mA current source
- Short circuit: Current limited to design value
- No damage to driver even with shorted outputs
- Compare to voltage-mode: Can source excessive current on short
- Built-in protection mechanism

CORRECT: 'Low power consumption since transmitter limits maximum current'

- Fixed 3.5 mA current through 100Ω termination
- Power = $I^2 \times R = (0.0035)^2 \times 100 = 1.2 \text{ mW}$ (approximately)
- Plus some driver overhead: typically 2-5 mW total per channel
- Much lower than CMOS: charging/discharging large voltage swings
- Especially significant for multi-channel interfaces

WRONG: 'Available also as single ended standard'

- FALSE: LVDS is inherently differential
- Requires two lines (+ and -)
- Single-ended alternatives: LVCMOS, LVTTL
- Differential nature is key to LVDS advantages

WRONG: 'Fault-tolerant in case one line is interrupted'

- FALSE: If one line breaks, signal is lost
- Differential signaling requires both lines
- Cannot operate with single line
- Redundancy would require duplicate pairs
- However: resistant to common-mode noise (different concept)

WRONG: 'Can not be used for clock lines'

LVDS Fault Behavior

Multiple Choice - LVDS Line Interruption

Source: Exercise I/O Interfaces, Question 2 (Page 2)

Topics: LVDS fault tolerance, differential signaling, line interruption

Question:

What happens at LVDS connections with regard to signal transmission if one of the two differential traces is interrupted?

Select one or more answers:

- Information is transmitted, but with a lower data rate
- Signals are transmitted with the inverted polarity
- There is no current flow, such the receiver can not detect logical zero or one
- No voltage is induced on the termination resistor
- Information is still transmitted because the two lines are redundant
- No Information is transmitted

Correct Answers:

- There is no current flow, such the receiver can not detect logical zero or one
- No voltage is induced on the termination resistor
- No Information is transmitted

Explanation:

Your answer is correct.

Die richtigen Antworten sind:

- No Information is transmitted
- There is no current flow, such the receiver can not detect logical zero or one
- No voltage is induced on the termination resistor

Detailed Analysis:

Normal LVDS Operation:

1. Current source drives 3.5 mA through differential pair
2. Current flows: Driver (+) → Line (+) → Termination resistor → Line (-) → Driver (-)
3. Closed current loop required
4. Voltage developed across termination: $V = I \times R = 3.5\text{mA} \times 100\Omega = 350\text{mV}$
5. Receiver detects this differential voltage

When One Line is Interrupted:

Physical consequence:

- Current path is broken
- Open circuit prevents current flow
- No current → No voltage across termination resistor
- Current-mode driver cannot function

CORRECT: 'There is no current flow, such the receiver can not detect logical zero or one'

- TRUE: Interruption breaks current loop
- Current source has nowhere to conduct
- Zero current through termination resistor
- Receiver sees no differential voltage
- Cannot distinguish between logic levels
- Result: Communication failure

CORRECT: 'No voltage is induced on the termination resistor'

- TRUE: $V = I \times R$, if $I = 0$ then $V = 0$
- Open circuit means no current
- Termination resistor has no voltage drop
- Receiver cannot detect signal
- May float to unpredictable voltage

CORRECT: 'No Information is transmitted'

- TRUE: Complete communication failure
- Link is non-functional
- Data cannot be recovered
- Error detection/correction cannot help (no signal at all)
- System must detect link failure

Why Other Options Are Wrong:

WRONG: 'Information is transmitted, but with a lower data rate'

- FALSE: No information at all

Signaling Standards

LVDS Advantages

Question: What are the Advantages of the Low Voltage Differential (LVDS) Standard

Wählen Sie eine oder mehrere Antworten:

- Higher data transmission rates since there are only small voltage swings required ✓
- Robust against shortcircuits since transmitter is current limited ✓
- Low power consumption since transmitter limits maximum current ✓
- Available also as single ended standard ✗
- Fault-tolerant in case one line is interrupted, since there are two lines per signal ✗
- Can not be used for clock lines ✗
- Lower cost compared to single ended signaling ✗
- High noise immunity due to compensation on differential lines ✓

Explanation:

Your answer is correct.

Die richtigen Antworten sind: Low power consumption since transmitter limits maximum current,

Higher data transmission rates since there are only small voltage swings required,

Robust against shortcircuits since transmitter is current limited,

High noise immunity due to compensation on differential lines

Problem type: Understanding LVDS signaling advantages

Source: Moodle Quiz I/O Interfaces, Question 1

LVDS Fault Behavior

Question: What happens at LVDS connections with regard to signal transmission if one of the two differential traces is interrupted?

Wählen Sie eine oder mehrere Antworten:

- Information is transmitted, but with a lower data rate ✗
- Signals are transmitted with the inverted polarity ✗
- There is no current flow, such the receiver can not detect logical zero or one ✓
- No voltage is induced on the termination resistor ✓
- Information is still transmitted because the two lines are redundant ✗
- No Information is transmitted ✓

Explanation:

Your answer is correct.

Die richtigen Antworten sind: No Information is transmitted,

There is no current flow, such the receiver can not detect logical zero or one,

No voltage is induced on the termination resistor

Problem type: Understanding LVDS fault behavior when a line is interrupted

Source: Moodle Quiz I/O Interfaces, Question 2

8b10b Encoding

Word Synchronization in 8b10b

Question: A continuous, serial transmission needs a means of detecting the start of a word. Describe in your own words how 8b10b encoding solves this challenge.

Explanation:

8b/10b encoding detects the start of a word by periodically transmitting special control characters (commas) with unique, illegal-for-data bit patterns. The receiver searches the serial stream for these patterns and, once found, aligns its word boundaries accordingly.

Key points:

- A comma has a distinctive pattern with five consecutive identical bits (11111 or 00000), which cannot occur in normal data symbols
- Receiver continuously scans the incoming bitstream for this pattern
- When the comma is detected, the receiver knows exactly where a 10-bit symbol boundary is
- From that point on, all following bits can be grouped correctly into 10-bit words → and therefore into the original 8-bit bytes

A special comma symbol with an unambiguous pattern allows the detection of the word border. The line code must guarantee that no combination of two code words leads to a shifted version of the comma pattern.

Problem type: Explanation - Understanding word synchronization in 8b10b encoding

Source: Moodle Quiz MGT, Question 1

Line Code Properties

8b10b Encoding Properties

Question: A line code has several properties that help the successful transmission of data via a serial connection. Below, four possible properties are shown.

Mark for each property whether 8b10b encoding provides this property.

Property	True	False
Maximum number of transitions	✓	
Word synchronization	✓	
Clock recovery	✓	
DC balance	✓	

Explanation:

Correct answers:

- Maximum number of transitions: False (8b10b does not limit maximum transitions, but ensures minimum transitions)
- Word synchronization: True (provided through comma characters)
- Clock recovery: True (sufficient transitions for CDR)
- DC balance: True (running disparity control ensures DC balance)

Problem type: Understanding 8b10b line code properties

Source: Moodle Quiz MGT, Question 2

MGT Transmission Steps

MGT TX Processing Steps

Question: A Multi-Gigabit Transceiver must perform multiple steps when transmitting (TX) or receiving (RX) data. Below, a few steps are given for TX. Drag the steps into the correct order with the first step at the top.

Correct order:

1. Parallel Input
2. Line Encoding
3. Serialization
4. Output

Explanation:

Die Antwort ist richtig.

The MGT transmitter processes data in this sequence:

1. **Parallel Input:** Data arrives in parallel form from FPGA fabric
2. **Line Encoding:** Data is encoded (e.g., 8b10b) for transmission
3. **Serialization:** Parallel encoded data is converted to serial bitstream
4. **Output:** Serial data is transmitted via high-speed differential lines

Problem type: Understanding MGT transmit data flow

Source: Moodle Quiz MGT, Question 3

KR and Exercises: In-Memory Compute (Part A)

In-Memory Compute Concept

IMC Technology Identification

Question: The following drawing shows a new technological approach for processors.

- a) This approach is an example of: _____.
- b) This approach is _____ for high-precision calculations.

Explanation:

Correct answers:

- a) This approach is an example of: **in-memory compute**
- b) This approach is **not suitable** for high-precision calculations.

Problem type: Identifying in-memory compute architecture and its limitations

Source: Moodle Quiz In-Memory Compute, Question 1

Performance Bottlenecks

CPU Memory Bottlenecks

Question: The standard approach for a CPU using an ALU and memory has several bottlenecks. From the below list, select all items that present a performance bottleneck for data-heavy algorithms.

Incorrect answers deduct points. Points can not go negative.

Wählen Sie eine oder mehrere Antworten:

- Number of instruction pipeline stages **X**
- Energy requirements of data transfer **✓**
- Memory access delay **✓**
- Cache size **✓**

Explanation:

Die Antwort ist richtig.

Die richtigen Antworten sind: Cache size, Memory access delay, Energy requirements of data transfer

Problem type: Understanding memory bottlenecks in conventional CPU architectures

Source: Moodle Quiz In-Memory Compute, Question 2

KR and Exercises: In-Memory Compute (Part B)

Workload Characteristics

Suitable Workloads for IMC

Question: Which workload characteristic makes an application particularly suitable for in-memory or near-memory compute acceleration?

Wählen Sie eine Antwort:

- It requires strict single-threaded execution with frequent synchronization. **X**
- It fits entirely into on-chip caches with no main memory accesses. **X**
- It is dominated by control flow and complex branching with little data reuse. **X**
- It is highly data-intensive with simple operations applied repeatedly to large data sets. **✓**

Explanation:

Die Antwort ist richtig.

Die richtige Antwort ist: It is highly data-intensive with simple operations applied repeatedly to large data sets.

Problem type: Understanding workload characteristics suitable for IMC acceleration

Source: Moodle Quiz In-Memory Compute (New Problems), Question 1

Analog Computation in Memory

Analog IMC Concept

Question: In the context of In-Memory Compute, what does the term 'analog computation in memory' typically refer to?

Wählen Sie eine Antwort:

- Implementing quantum bits inside DRAM cells for superposition-based computation. **X**
- Performing digital logic gates strictly using CMOS standard cells outside the memory array. **X**
- Exploiting the continuous electrical properties of memory cells and bitlines to perform operations like vector-matrix multiplication. **✓**
- Using precise floating-point ALUs inside memory arrays. **X**

Explanation:

Die Antwort ist richtig.

Die richtige Antwort ist: Exploiting the continuous electrical properties of memory cells and bitlines to perform operations like vector-matrix multiplication.

Problem type: Understanding analog computation in memory concept

Source: Moodle Quiz In-Memory Compute (New Problems), Question 2

IMC vs NMC Differences

In-Memory vs Near-Memory Compute

Question: Which of the following best captures a key difference in typical use-cases for In-Memory Compute versus Near-Memory Compute?

Wählen Sie eine Antwort:

- IMC commonly accelerates operations like bitwise logic or analog vector-matrix multiplies inside arrays, whereas NMC often accelerates more general kernels near memory using programmable or specialized logic. **✓**
- IMC and NMC both only accelerate floating-point matrix multiplication workloads. **X**
- IMC is ideal for fine-grained, irregular control tasks, while NMC targets only bulk bitwise operations. **X**
- IMC is used solely for security applications, while NMC is used solely for graphics. **X**

Explanation:

Die Antwort ist richtig.

Die richtige Antwort ist: IMC commonly accelerates operations like bitwise logic or analog vector-matrix multiplies inside arrays, whereas NMC often accelerates more general kernels near memory using programmable or specialized logic.

Problem type: Understanding differences between IMC and NMC

Source: Moodle Quiz In-Memory Compute (New Problems), Question 3

KR and Exercises: Configuration

Configuration Clock Speed Calculation

Configuration Clock Speed

Question: The table below is an excerpt from the Cyclone V datasheet. It lists the configuration file size in bits for the different device sizes (A2–A7) in the Cyclone V E series.

Variant	Member Code	Configuration .rbf Size (bits)
Cyclone V E ⁽⁹⁷⁾	A2	21,061,280
	A4	21,061,280
	A5	33,958,560
	A7	56,167,552

Your design contains one Cyclone V E A2 device. The configuration time must not exceed 500 ms. Your design will use serial configuration mode.

From the list, select the lowest configuration clock speed that still meets the time requirement. _____ MHz.

Explanation:

Correct answer: 44 MHz

Calculation:

- Configuration file size: 21,061,280 bits
- Required time: 500 ms = 500,000 μ s
- Clock speed = $\frac{\text{number of bits}}{\text{time in microseconds}} = \frac{21,061,280}{500,000} = 42.12$ MHz
- Lowest suitable clock speed: 44 MHz

Divide number of bits by required time in microseconds to get the clock speed in MHz.

Problem type: Configuration clock speed calculation

Source: Moodle Quiz Configuration, Question 1

Configuration Time Calculation

Configuration Time

Question: The table below is an excerpt from the Cyclone V datasheet. It lists the configuration file size for the different device sizes (A2–A7) in the Cyclone V E series.

Variant	Member Code	Configuration .rbf Size (bits)
Cyclone V E ⁽⁹⁷⁾	A2	21,061,280
	A4	21,061,280
	A5	33,958,560
	A7	56,167,552

Your design contains one Cyclone V A5 device. The configuration clock is 40 MHz.

How long will configuration take for a single A5 device when using serial mode? _____ us (microseconds)

Explanation:

Correct answer: 848964 μ s (microseconds)

Calculation:

- Configuration file size: 33,958,560 bits
- Clock frequency: 40 MHz = 40 bits/ μ s (in serial mode)
- Configuration time = $\frac{\text{number of bits}}{\text{clock frequency}} = \frac{33,958,560}{40} = 848,964$ μ s

Problem type: Configuration time calculation

Source: Moodle Quiz Configuration, Question 2