NTNU
Norwegian University of
Science and Technology

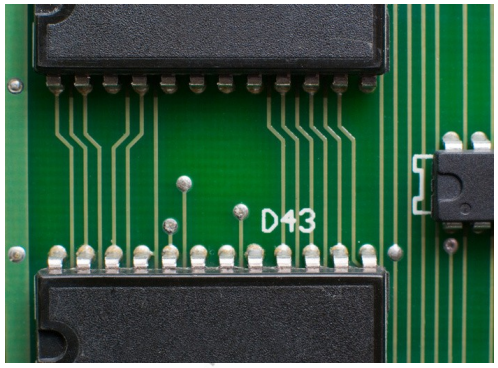
Lecture 5: Computing Platforms

Asbjørn Djupdal
ARM Norway, IDI NTNU
2013

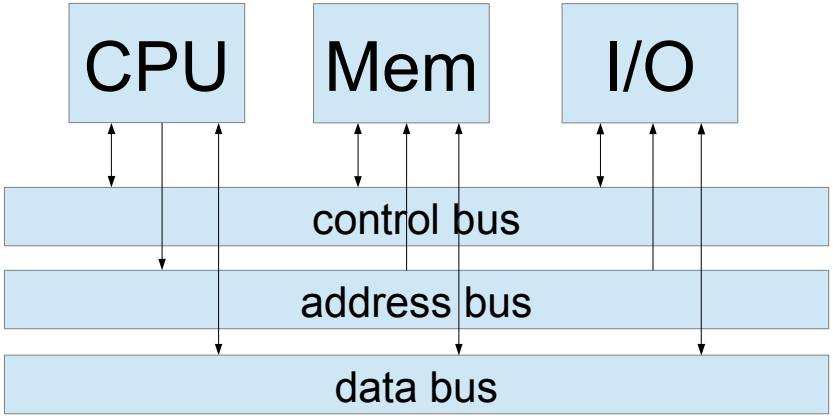
Lecture overview

- Bus based systems
  - Timing diagrams
  - Bus protocols
  - Various busses
- Basic I/O devices
- RAM
- Custom logic
  - FPGA
- Debug systems
  - JTAG
  - Logic analyzers

The bus

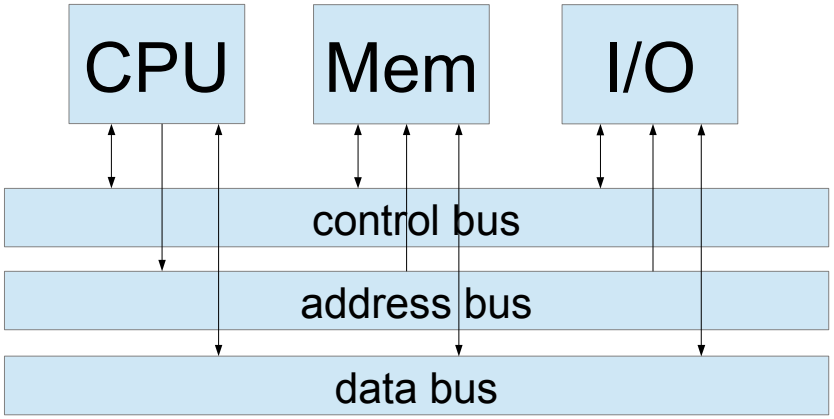


- A bus is a collection of wires
  - Provide mechanisms for data transfer, control and signaling
  - Multiplexing is possible
- The rules that governs the communication between two devices is called a protocol
- The master initiates transfers, slaves respond



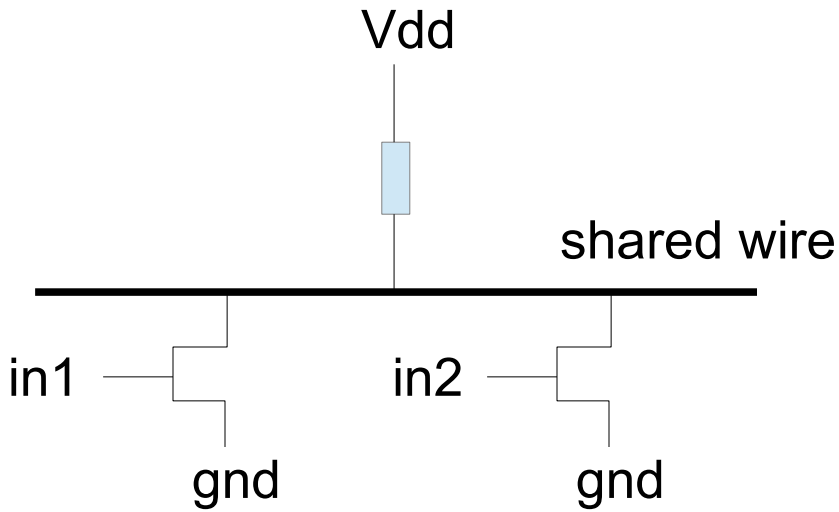
Typical bus signals

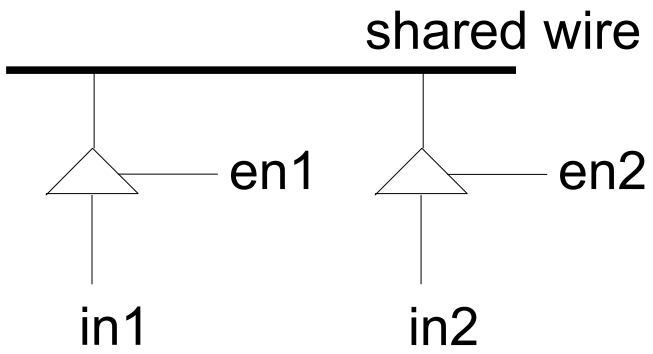
- *Address bus: 2^n wires specifying an address*
- *Data bus: 2^m wires specifying a data word, often bidirectional*
- *Control bus: Various wires for managing the bus protocol*
  - *enq, ack, rw, irq, ...*



Bidirectional busses

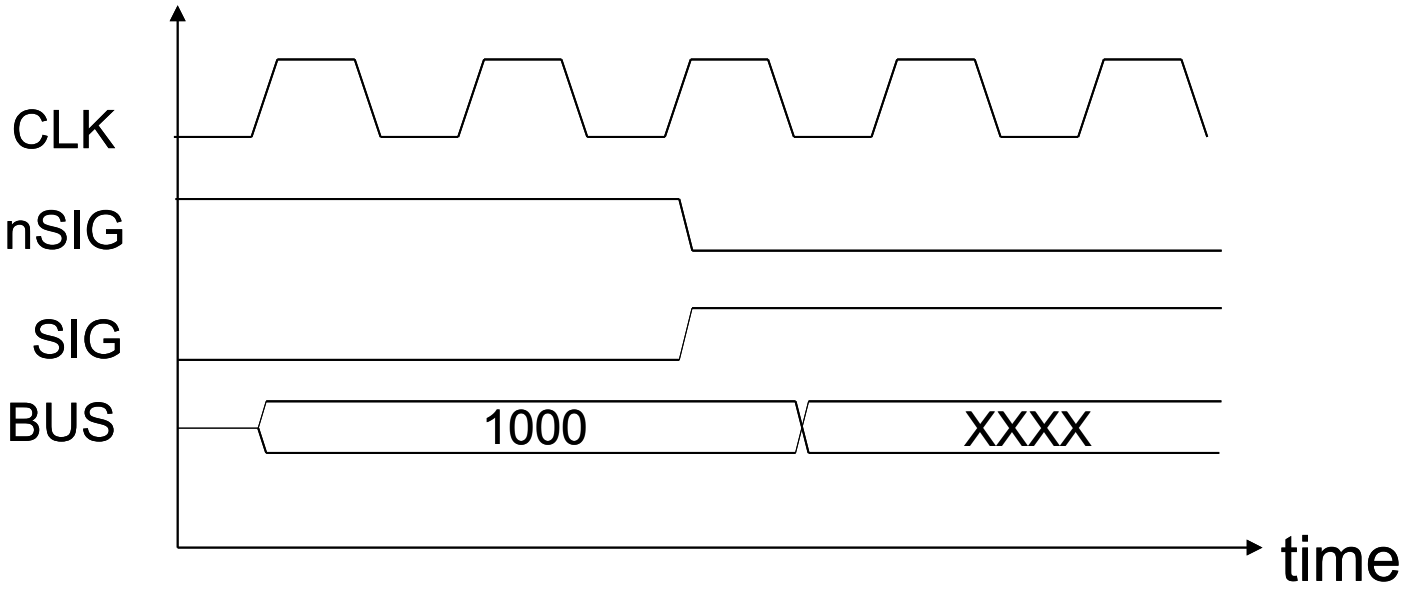
- How to design a bus that several devices can drive?
- Must avoid driving the same wire to both 0 and 1 (short circuit) at the same time
- Solutions:
  - Tristate buffers
  - Pull-up resistors and open collector (drain)





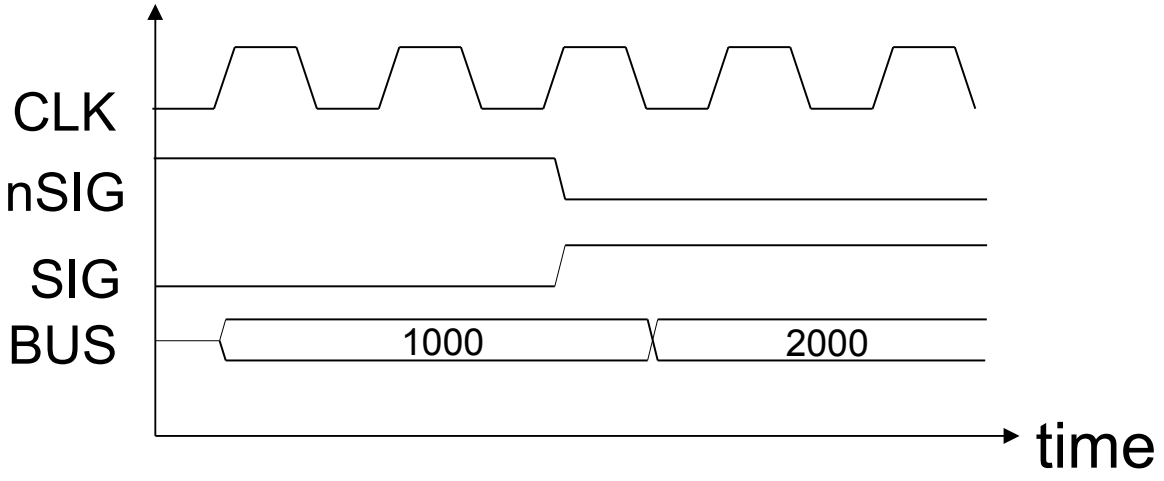
Timing diagrams

- Timing diagrams can be used to show how the bus protocol works

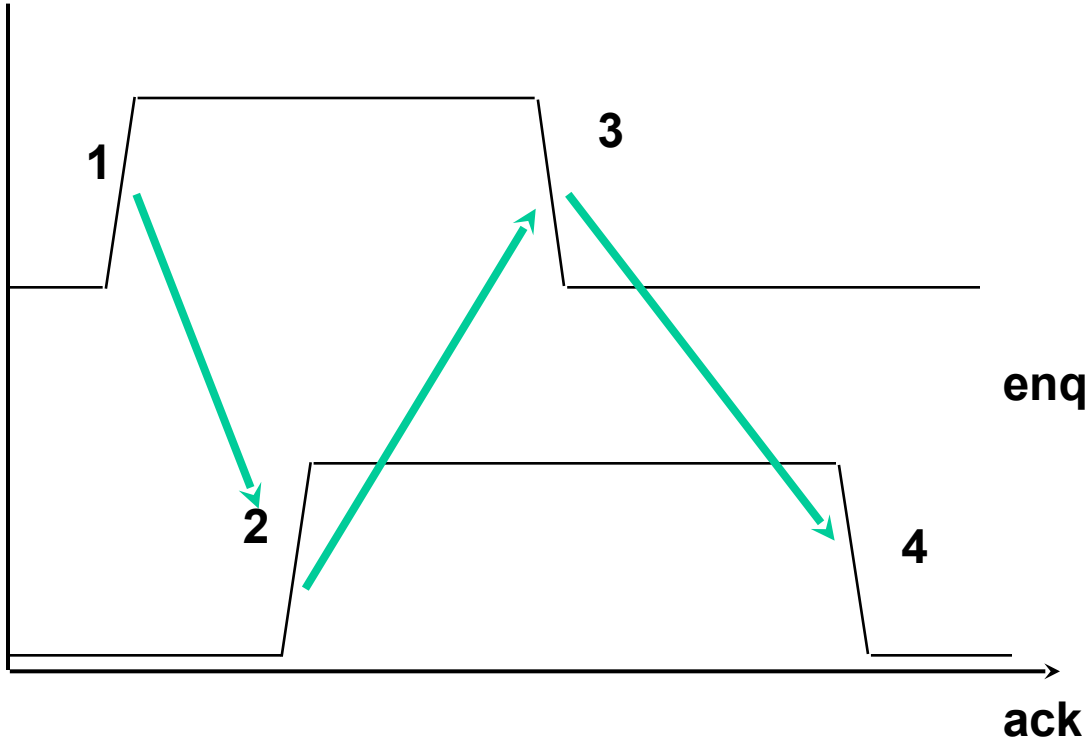


Terminology

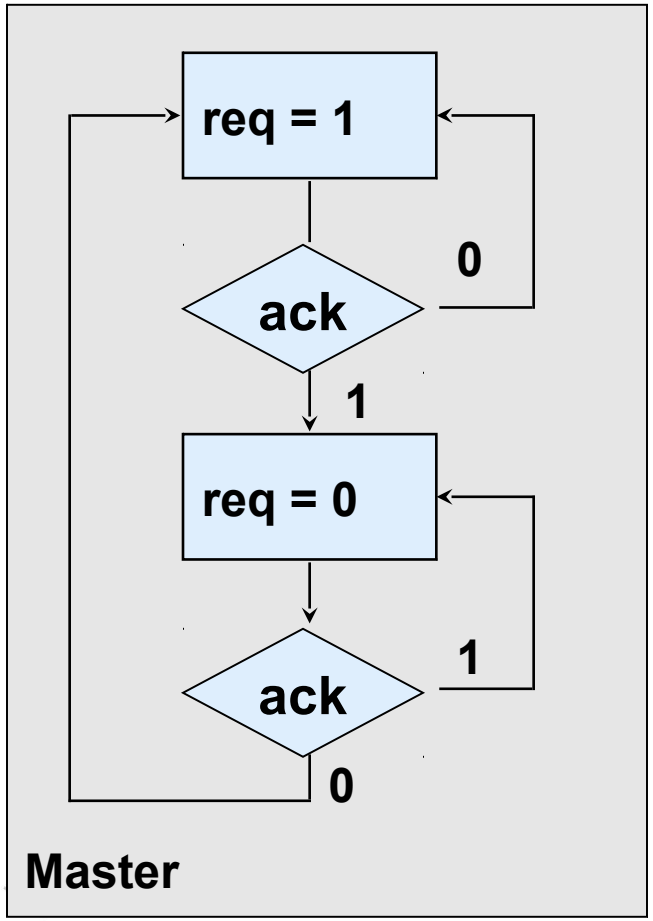
- Active high
  - System takes action on logical 1
- Active low
  - System takes action on a logical 0
  - Offers increased immunity to electrical noise
  - Typical notation: nSIGNAL, SIGNAL' (e.g nIRQ, IRQ')
- Rising edge
- Falling edge
- Not driven

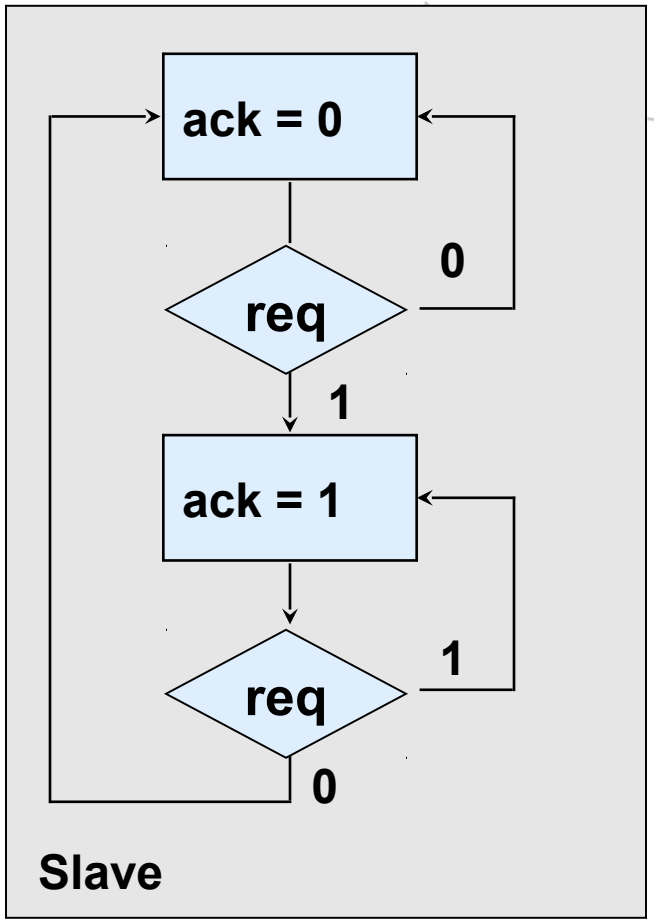


Bus protocol Four-cycle handshake: Read

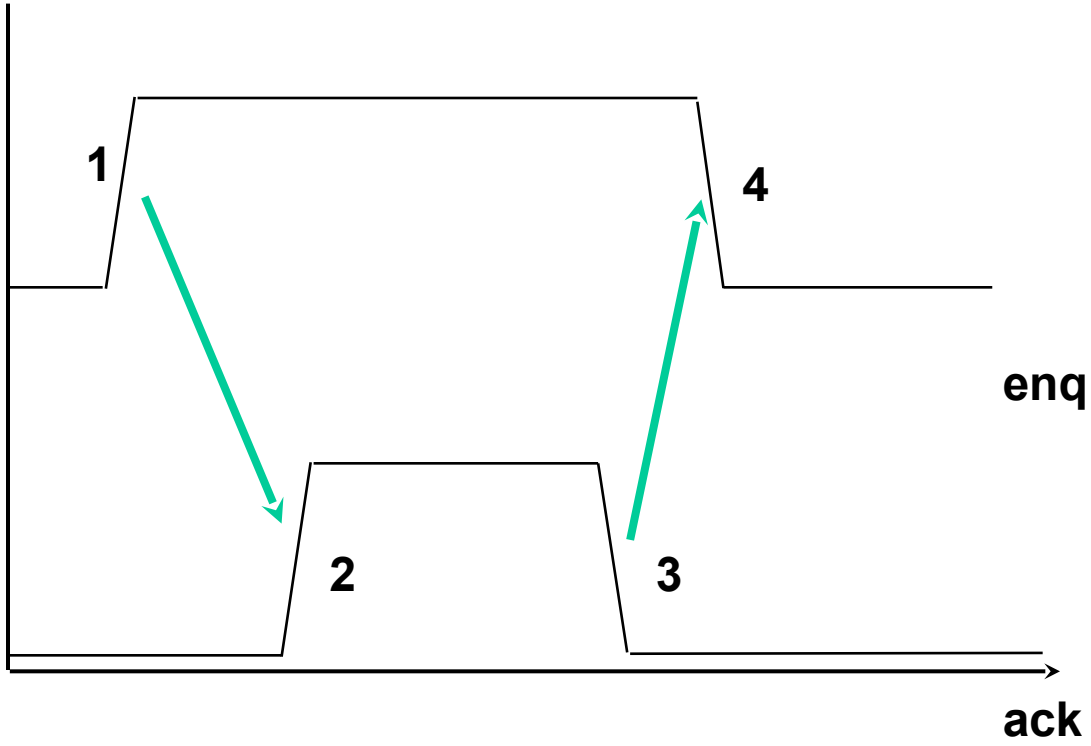


Protocol state machine

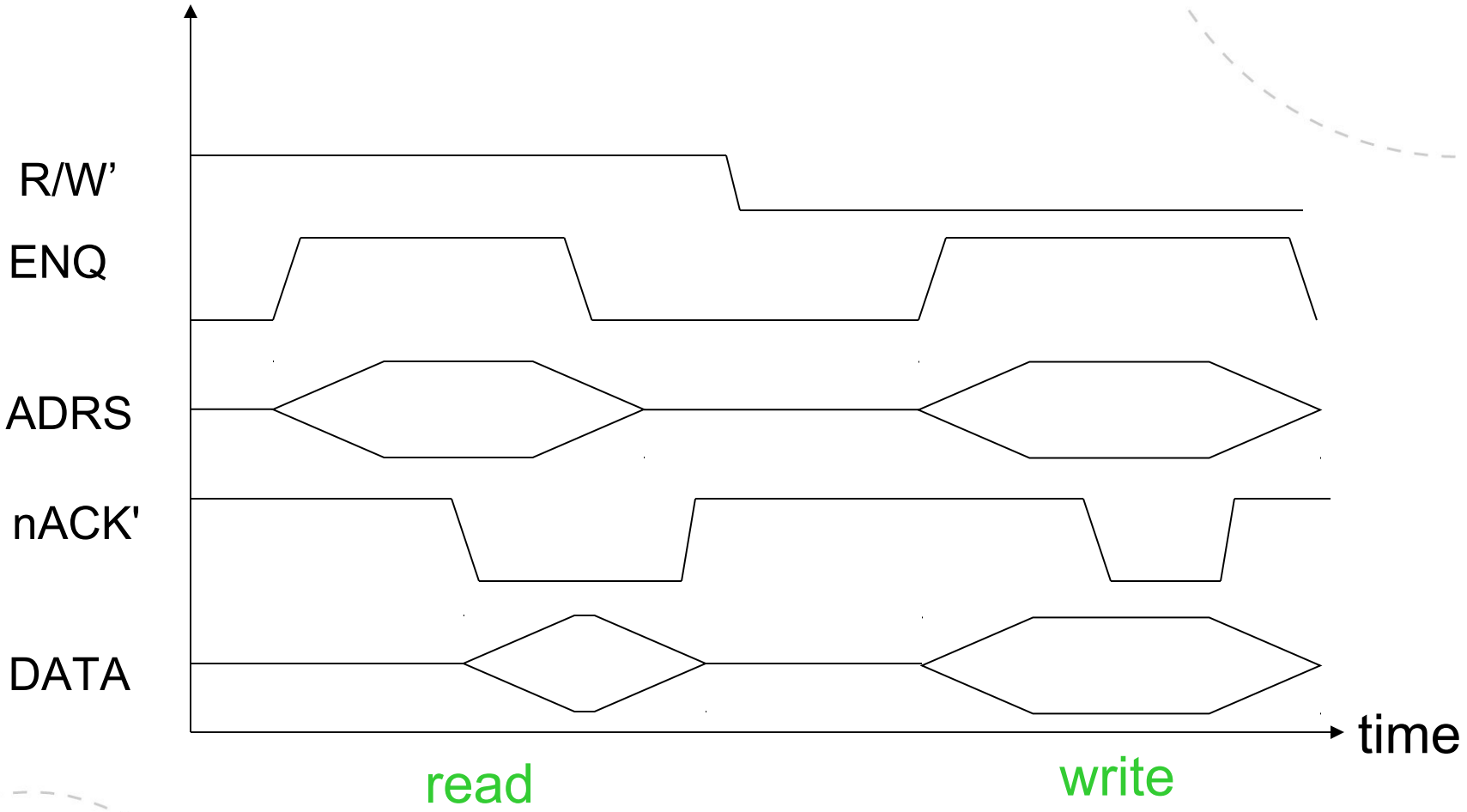




Bus protocol Four-cycle handshake: Write

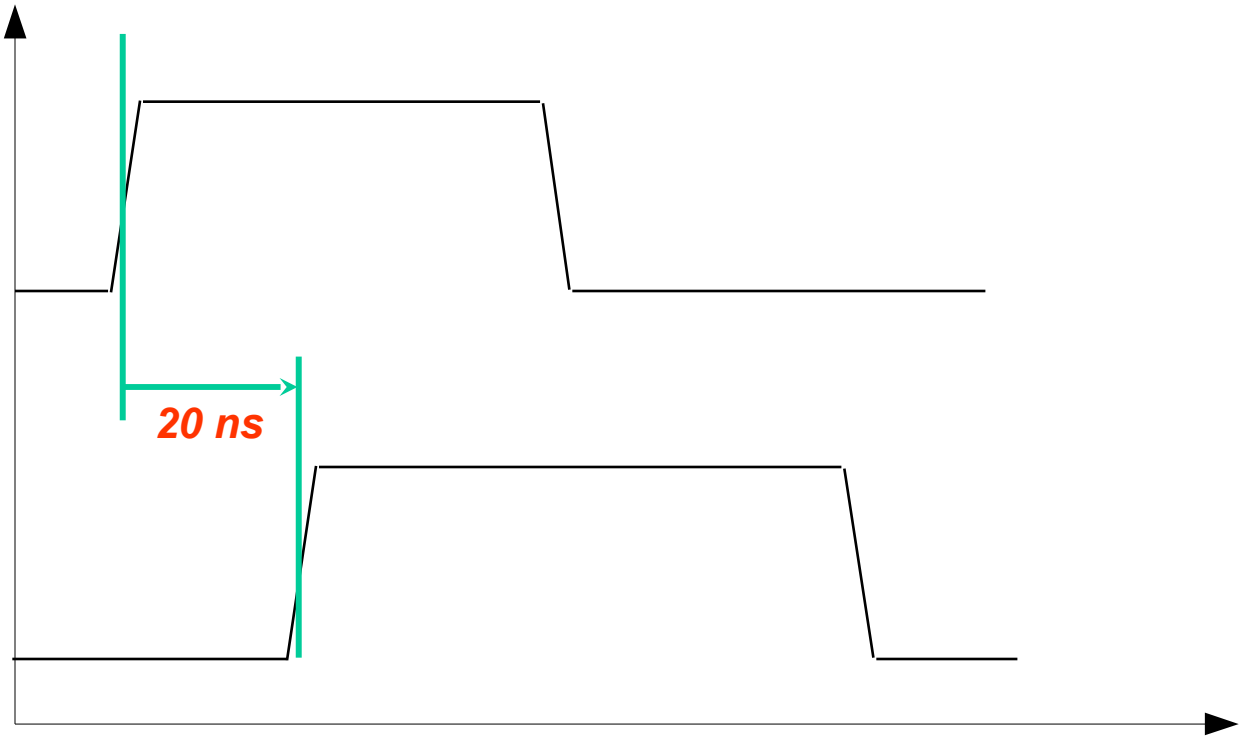


Typical asynchronous bus access



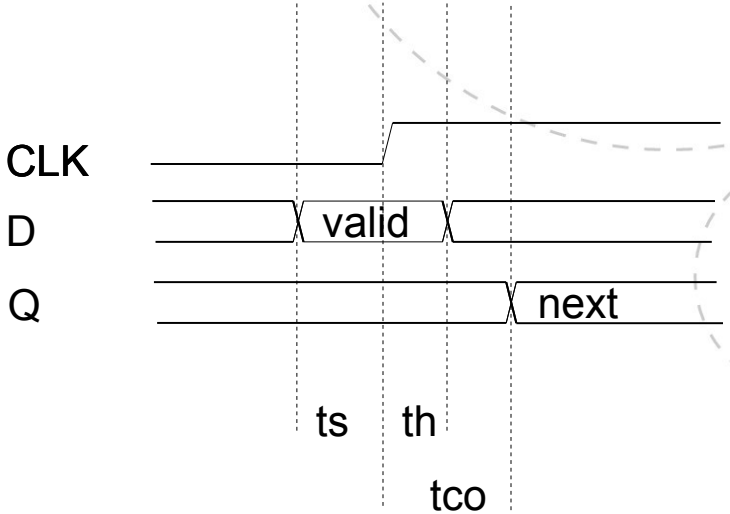
Timing constraints

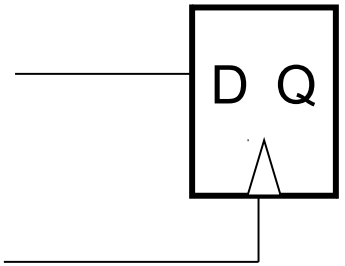
- Minimum time between two events



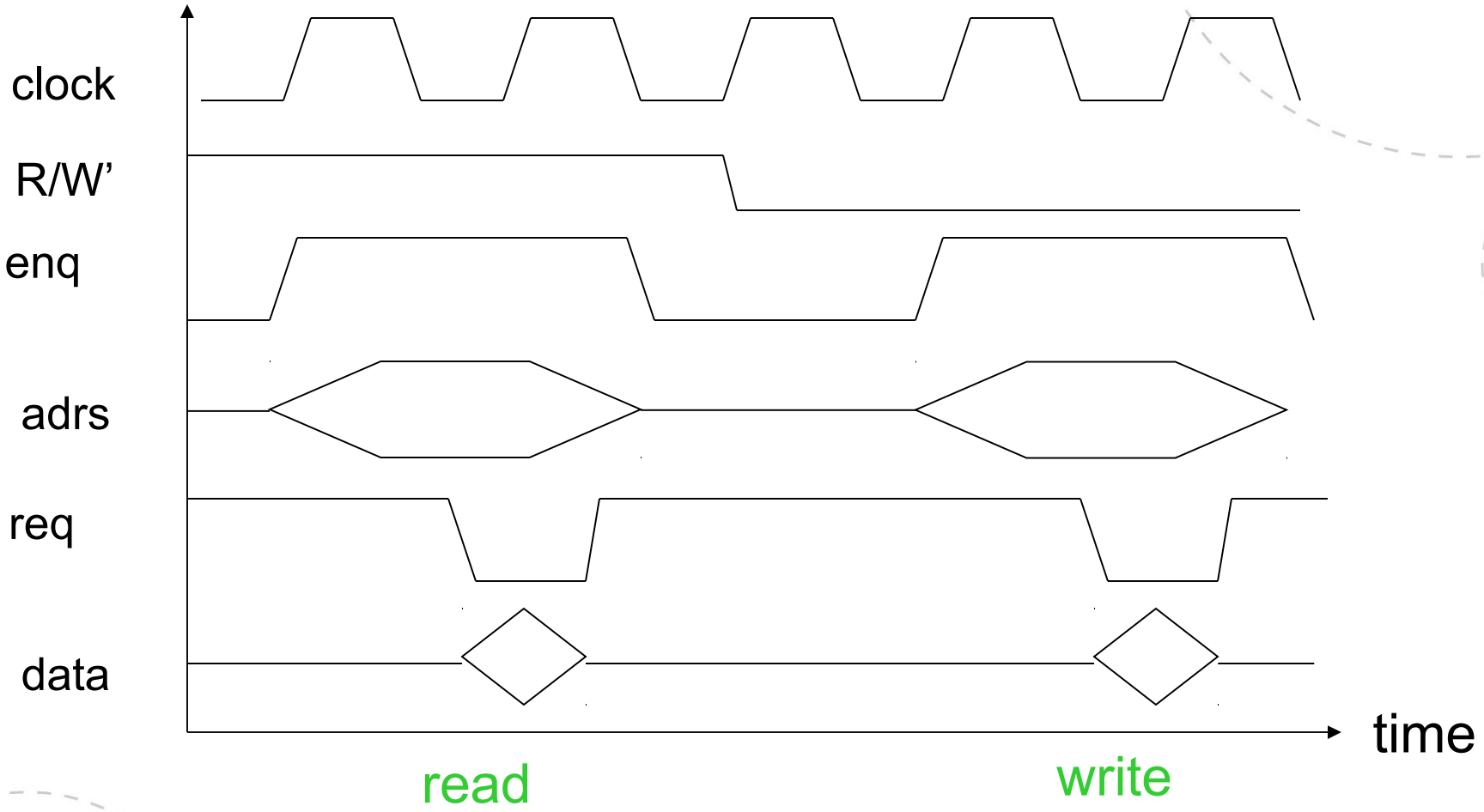
Origin of timing constraints

- Control signals needs to be read by the receiver
- D-latches often used on inputs to avoid synchronization problems
- Clocked busses have D-flipflops
- Latches and flipflops get undefined results if inputs change at the wrong time
  - ts: setup time
  - th: hold time
  - tco: clock to output time





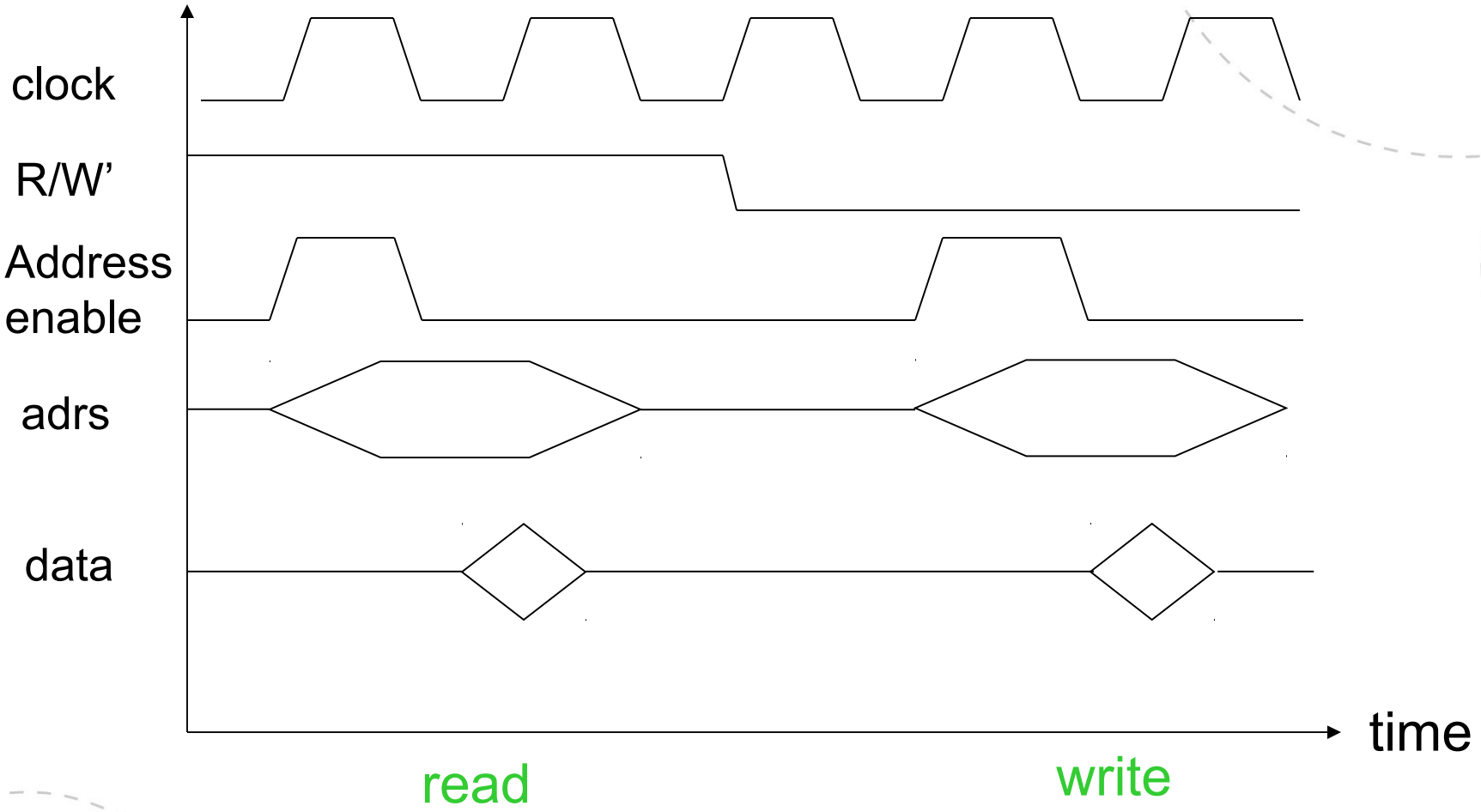
Typical synchronous bus access



Are handshakes always necessary?

- Only when response time cannot be guaranteed in advance
  - Data dependent delay
  - Component variations
- If you know the delay, a handshake is not needed
- Synchronous bus

Bus access, no handshake

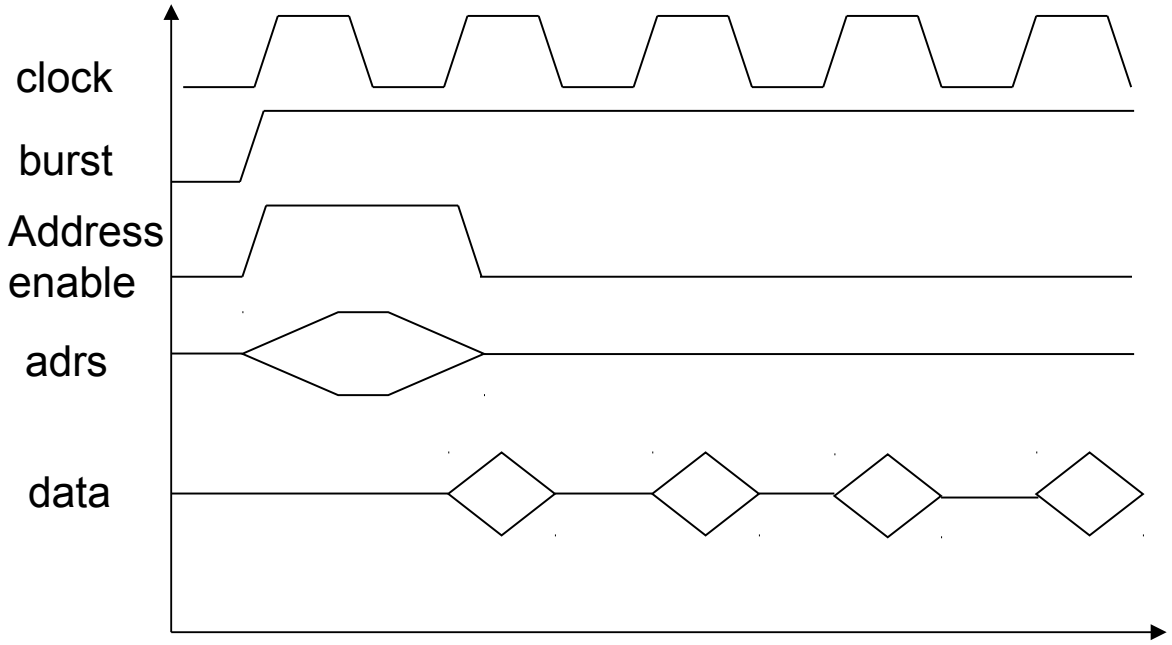


Wait states

- Only for synchronous busses
- A device may not be able to deliver data on the next clock cycle
- CPU must wait for data to be ready
- Bus cycles without activity is called *wait states*

Burst transfer

- Some busses support transferring multiple words in one request
- Saves lots of time wasted on initiating requests multiple times

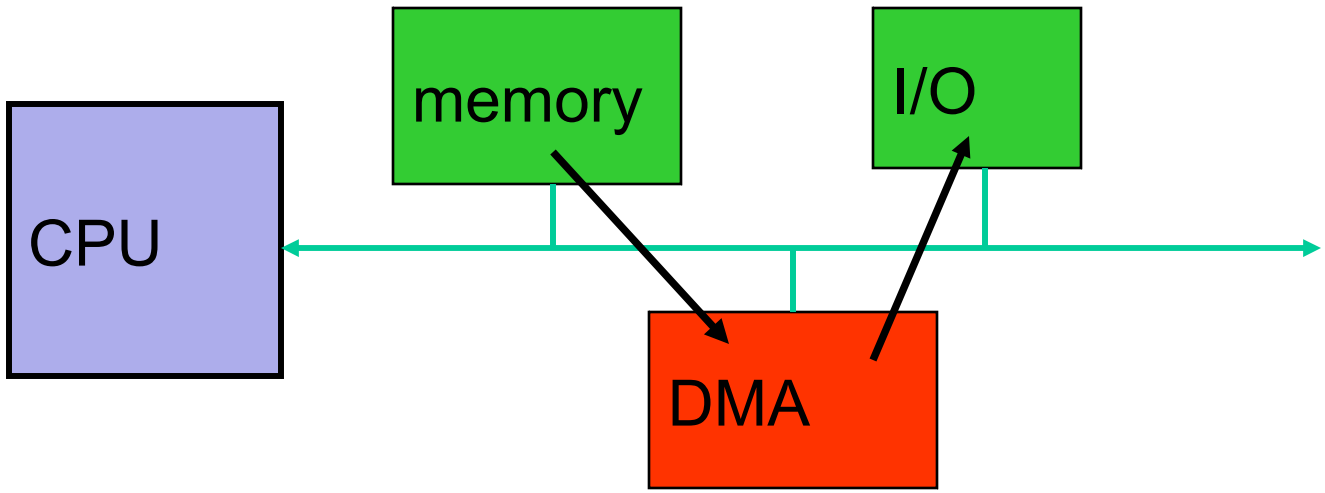


Bus masters

- The bus master can initiate traffic on the bus
- CPU is always a bus master
- Can have more than one:
  - Multiple CPUs
  - DMA units
- An arbitration mechanism must be part of the bus spec to handle multiple master

Direct Memory Access (DMA)

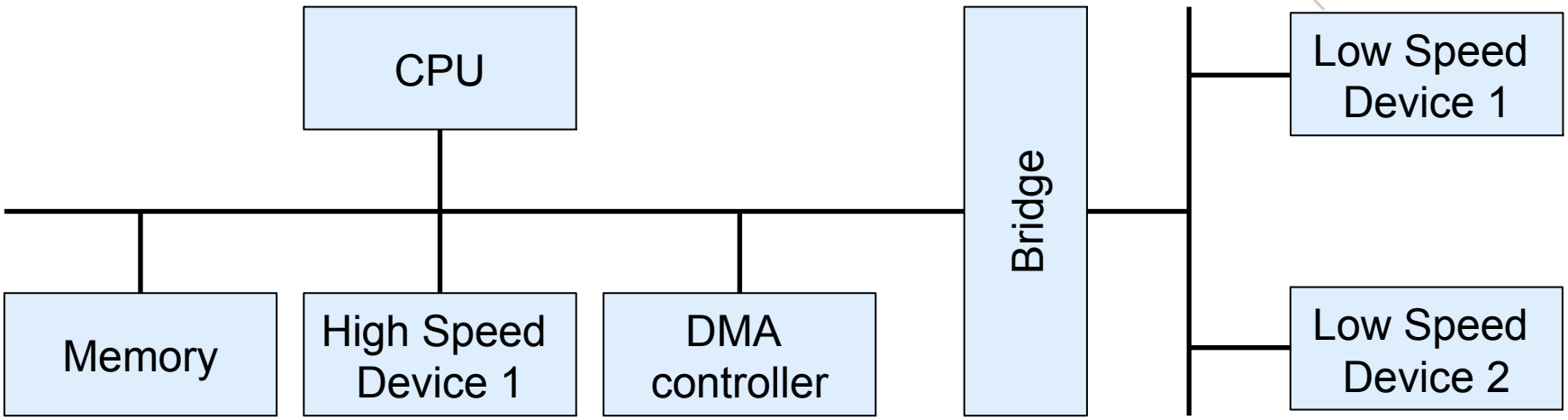
- DMA provides parallelism on bus by controlling transfers without using the CPU



DMA operation

- CPU sets up DMA transfer
  - Start address
  - Length
  - Transfer block length
  - Style of transfer
- DMA controller performs transfer, interrupt when done
- CPU and DMA can not use the bus at the same time
  - CPU have caches and might not need much access to the bus
  - Arbitration mechanism might be “cycle stealing”: CPU stalls the DMA unit while using the bus

Bus hierarchy

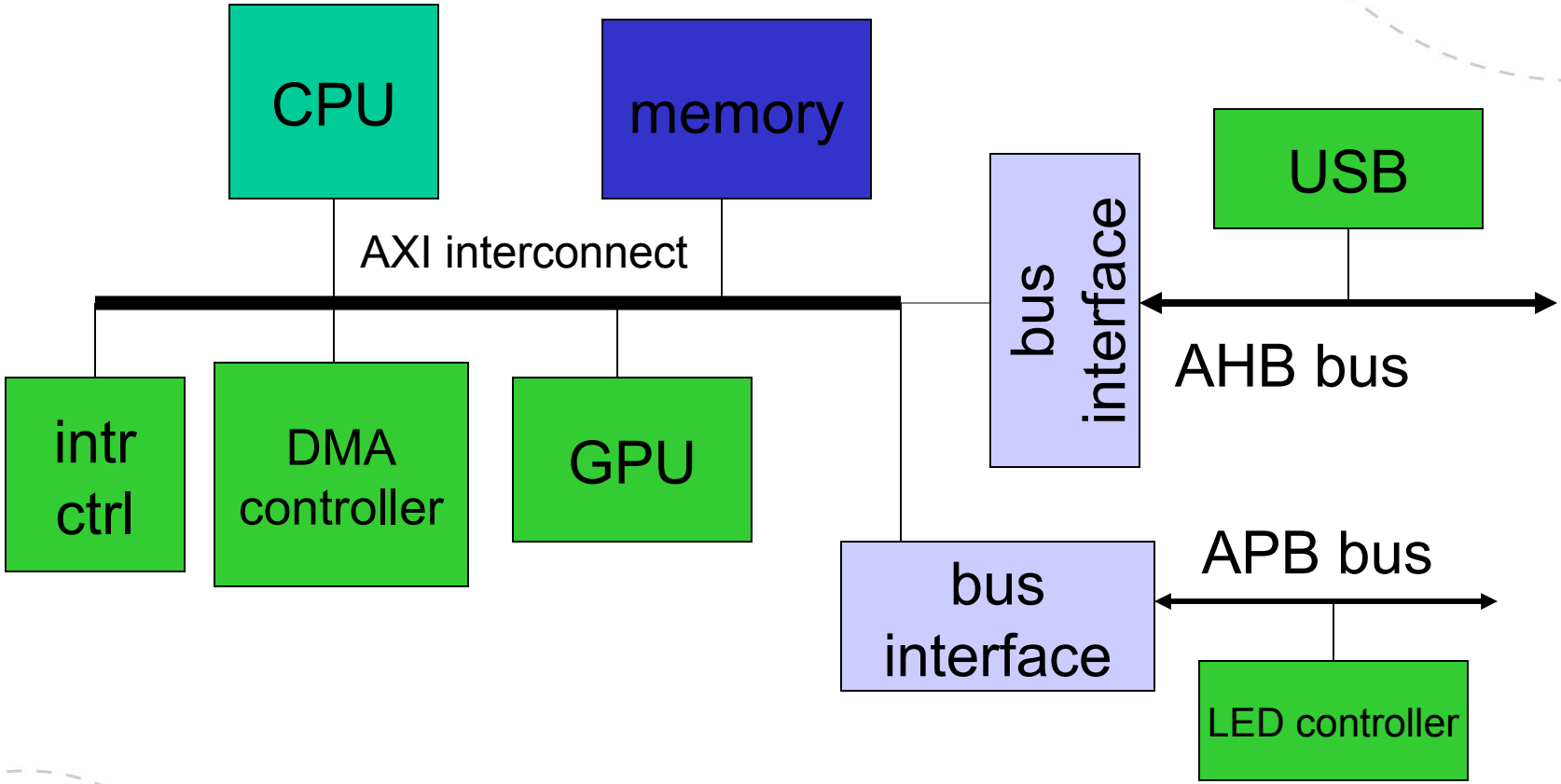


- Bus performance limited by the slowest device
- Solution: Hierarchy of busses

ARM busses

- Typically used on-chip
- ARM cores connect to the various ARM busses
- AMBA (Advanced Microcontroller Bus Architecture)
  - Open standard
  - Many compatible devices (all ARM devices)
  - Several variants
    - Advanced Peripherals Bus (APB)
    - Advanced High-performance Bus (AHB)
    - Advanced eXtensible Interface (AXI)

ARM bus hierarchy



APB

- Simple
- Synchronous
- Allows wait states

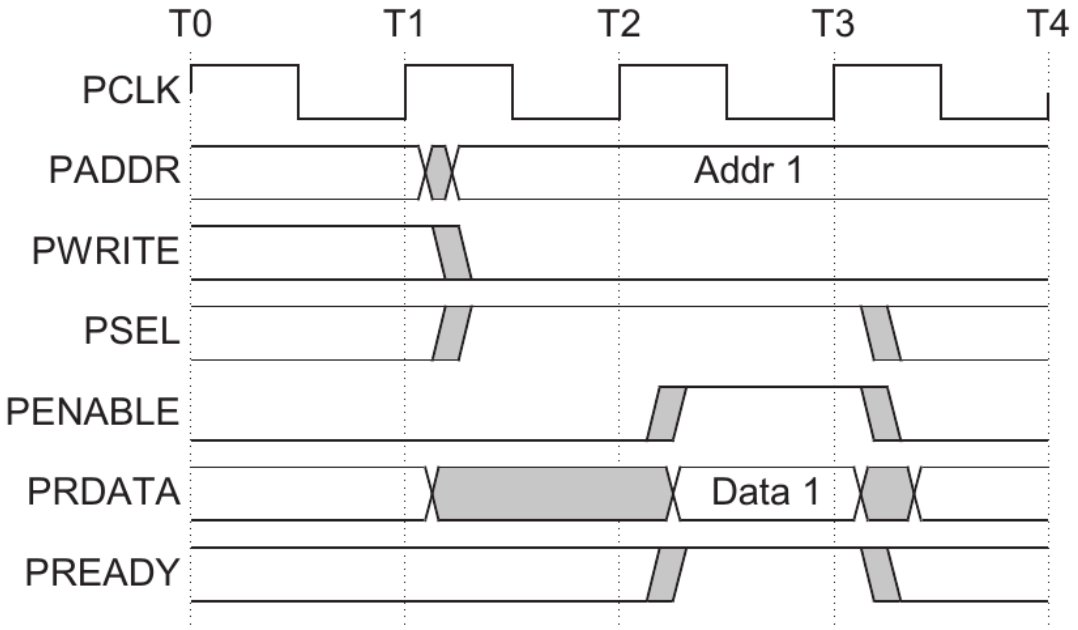


Figure 3-4 Read transfer with no wait states

AHB

- Medium complexity
- Allows burst transfer

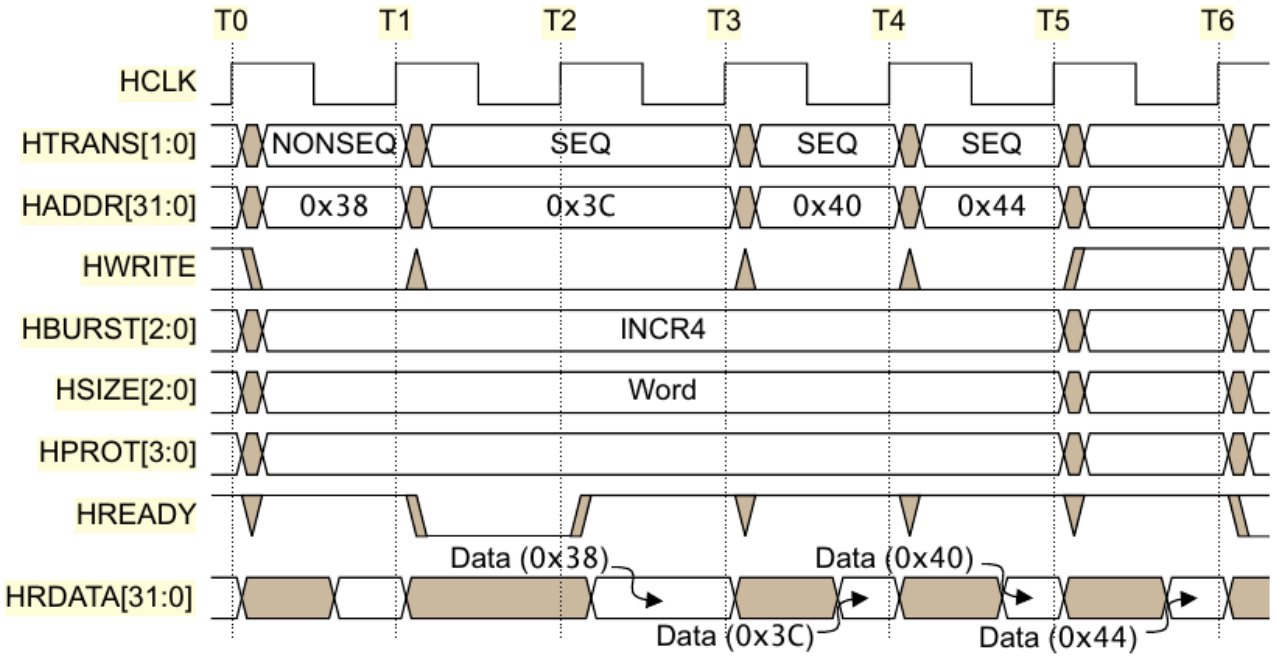


Figure 3-9 Four-beat incrementing burst

AXI

- High performance
- High complexity
- Independent channels
  - Read address
  - Read data
  - Write address
  - Write data
  - Write response
- Multiple outstanding requests
- Out of order transactions

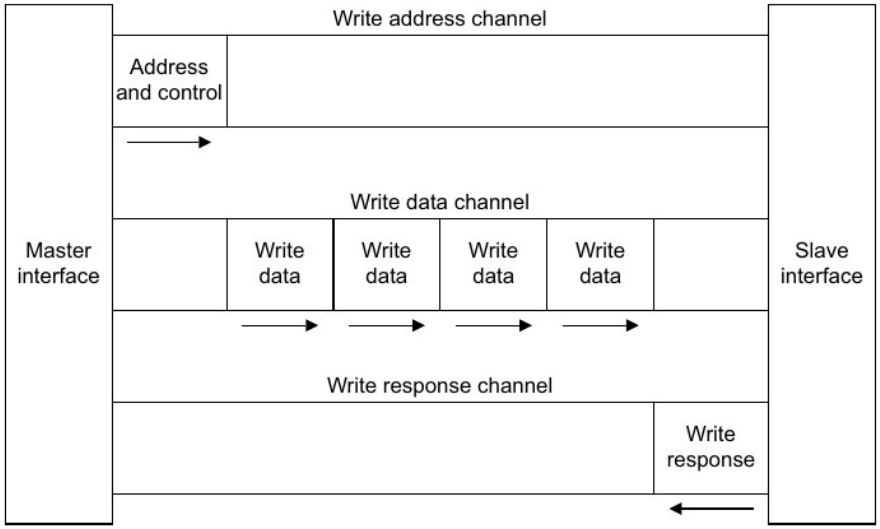


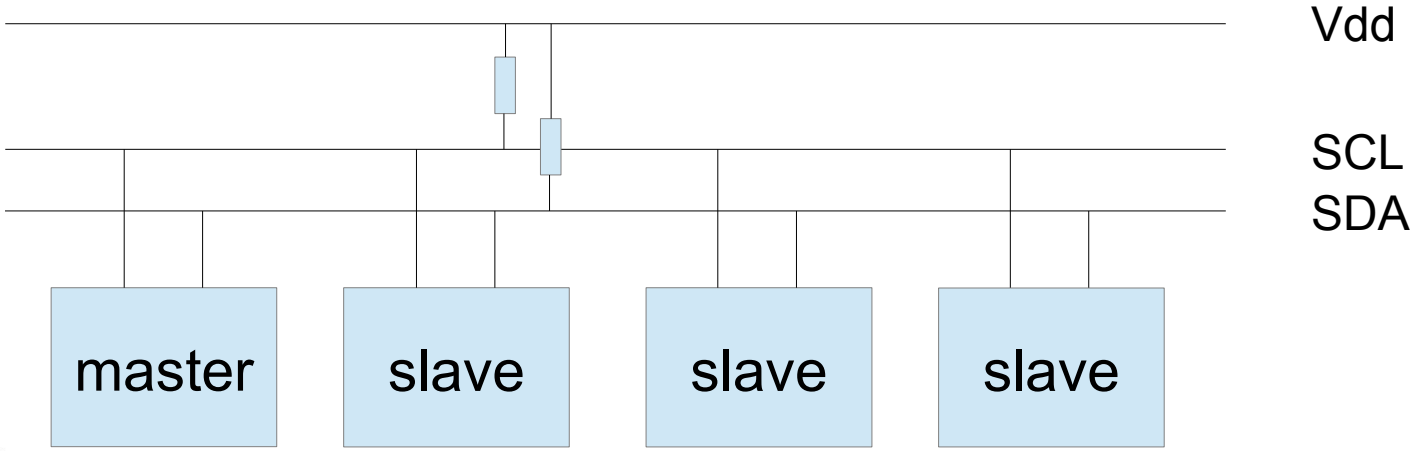
Figure A1-2 Channel architecture of writes

Device interfacing

- Often need to use components from various sources
- Glueless interface
  - Device is designed to work with the bus used by your project
- Glue logic
  - Needed when device interface is not compatible with your bus
  - Either custom made (FPGA) or standard components
- Many bus standards for on-board peripherals exists
  - I2C
  - SPI

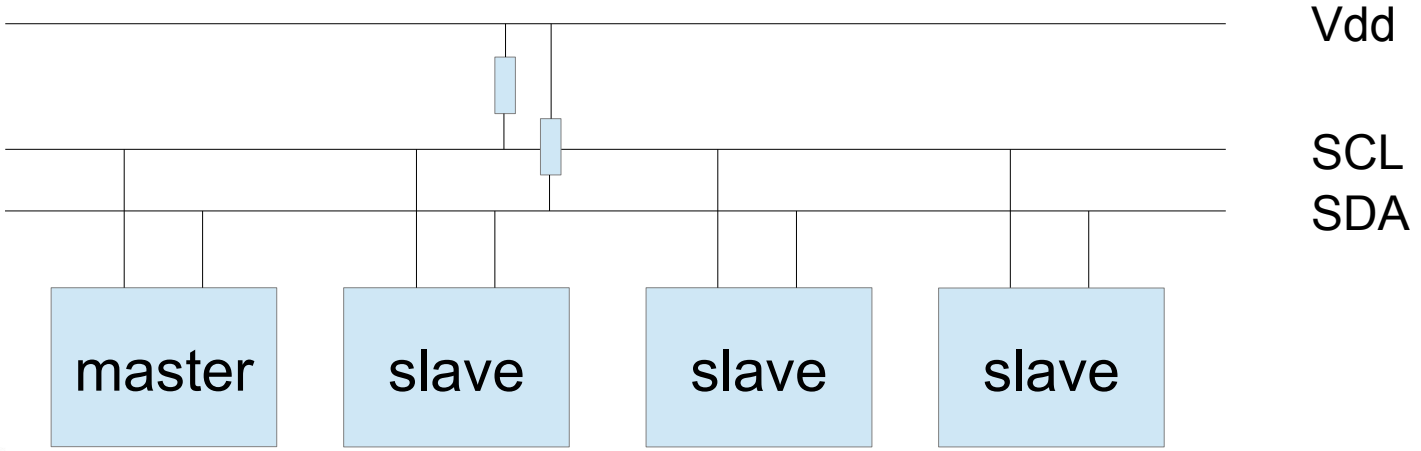
I2C

- Inter-Integrated Circuit bus
- Simple, both HW and SW
  - Two wires (clk and data) with pull-up
  - Low speed
- Typical applications:
  - RTC
  - Low speed DACs and ADCs
  - Reading HW monitors and sensors



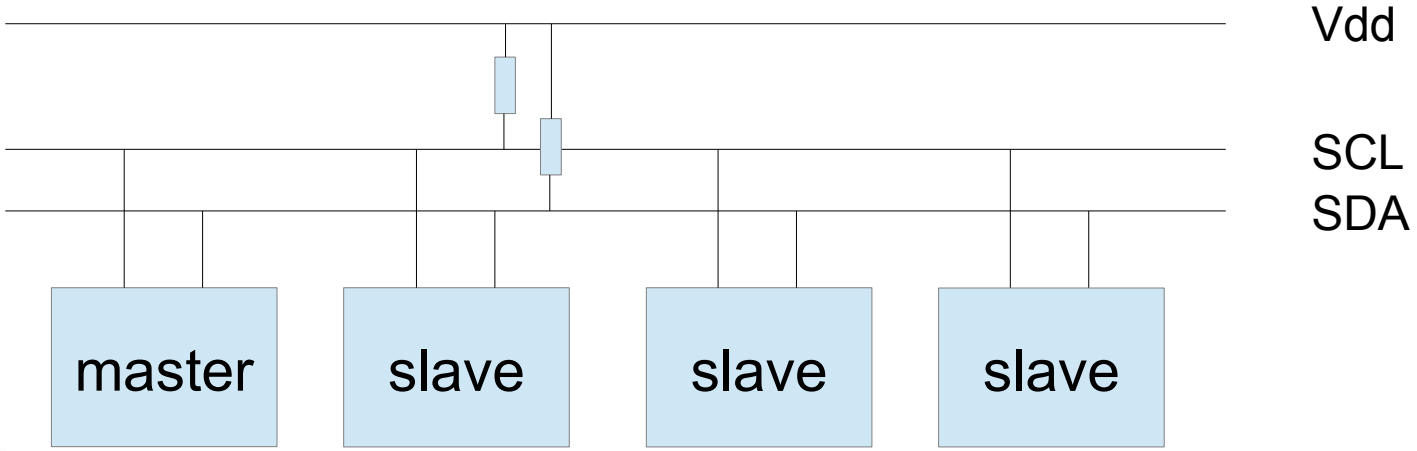
I2C

- Each slave has a 7 bit address
- Master
  - starts clock
  - pulls SDA low (start bit)
  - sends 7 bit slave address on SDA
  - sends R/W bit
- Slave
  - responds by pulling SDA low
- Data transfer starts



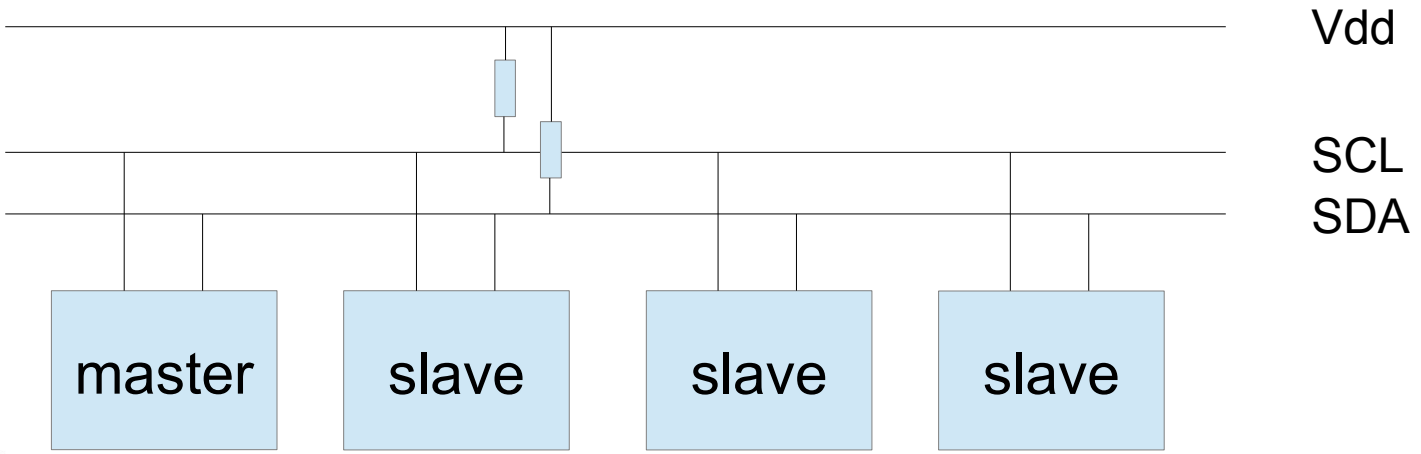
I2C

- Slaves may introduce wait states
  - Pulls SCL low while busy
- Master must stall transfer while SCL is low



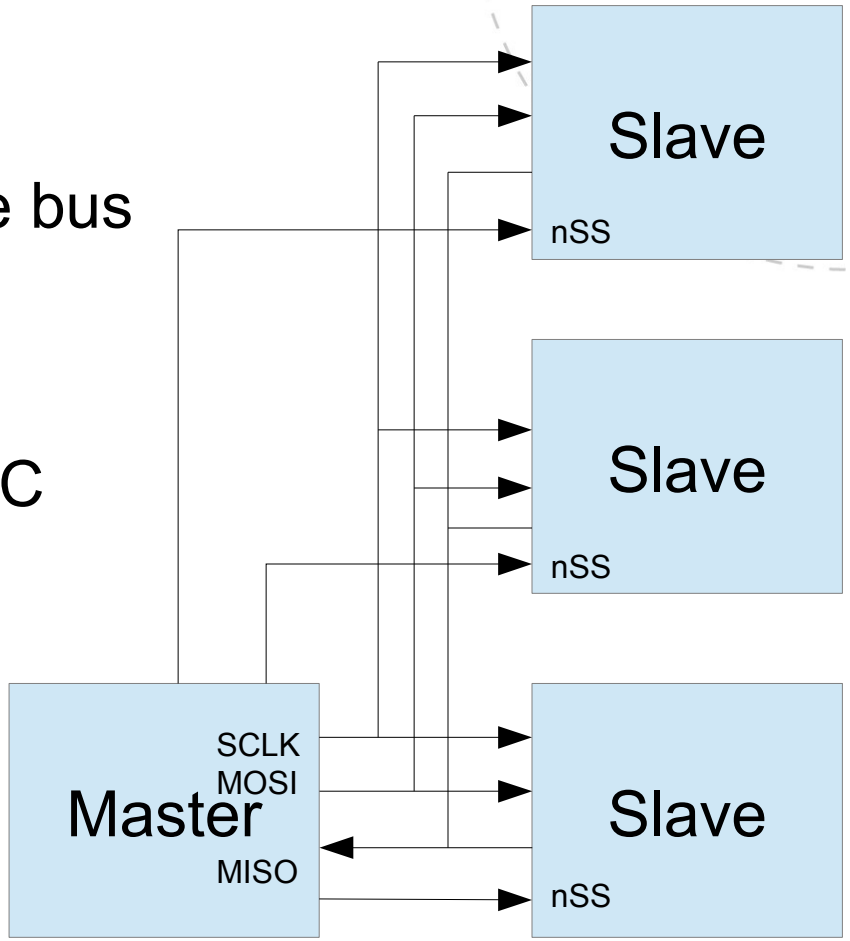
I2C

- Supports multiple masters
- Deterministic arbitration mechanism
  - Two masters starts transmitting at the same time
  - The first that sees SDA have a different value than expected gives up and retries later



SPI

- Serial Peripheral Interface bus
- Four wires
- Simple HW and SW
- Higher throughput than I2C
- Applications:
  - EEPROMs
  - Flash
  - LCD
  - SD cards



Secure Digital

- Peripheral bus for Flash memory cards (SD) and various IO devices (SDIO)
- Well defined protocol, both on physical and packet level
- Can be operated in SPI mode
  - All SPI controllers can be used to communicate with SD(IO) cards
  - Higher speed is possible in SD mode

Lecture overview

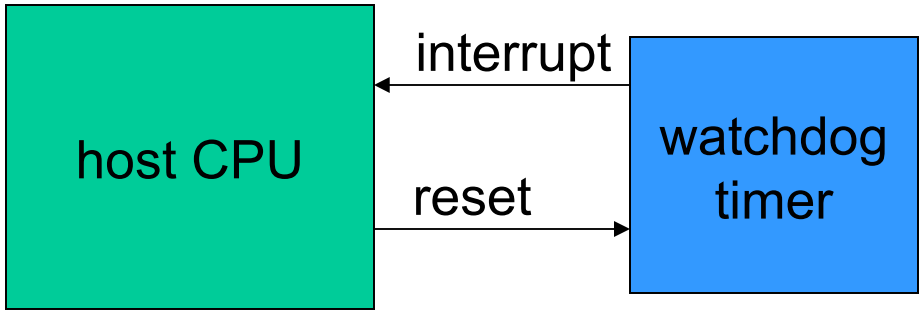
- Bus based systems
  - Timing diagrams
  - Bus protocols
  - Various busses
- Basic I/O devices
- RAM
- Custom logic
  - FPGA
- Debug systems
  - JTAG
  - Logic analyzers

Timers and counters

- Very similar
  - A Timer is incremented by a clock signal
  - A counter is incremented by an occasional signal
- Many different behaviours
  - May have interrupt on rollover
  - May have interrupt when reaching a set value
  - Count up or down

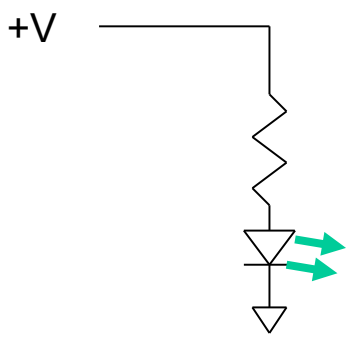
Watchdog timer

- Timer which is periodicalily reset by system software
- If not reset within a set time, it generates an interrupt or resets the host
- Purpose: Reset a system automatically if it hangs



Interfacing the digital world

- General Purpose IO (GPIO)
  - Most microcontrollers have GPIO pins that can be configured as inputs or outputs
  - Input pins can be read by SW
  - Output pins can be set to 0 or 1 by SW
  - Often possible to get interrupts when input pins change state
  - Very useful
    - LED control
    - Switch input
    - 7 segment display
    - Bit bang (low speed) busses



GPIO output pin

Bit banging

- Many slow busses can be driven by GPIO pins, with protocol entirely controlled by SW
- Useful in many embedded contexts where the appropriate bus controller is not available in the chosen microcontroller
- Examples:
  - I2C
  - SPI
  - SD
  - ATA
- Bit banging is useful, but inefficient
  - Typically needs the CPUs full attention while communicating

Interfacing the analog world

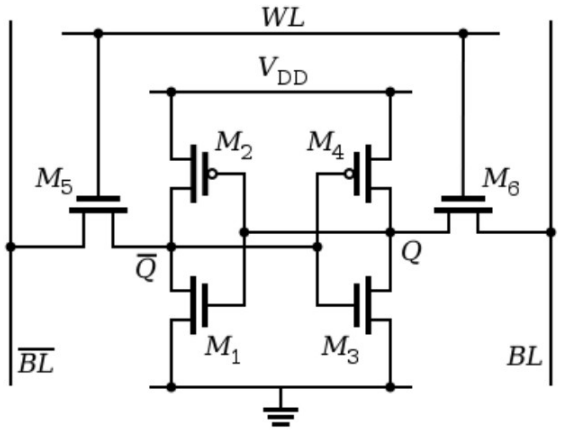
- Analog to digital conversion (ADC)
  - Samples an analog signal and converts it to a digital value
  - I.e voltage on input pin is converted to a digital number, to be read by the CPU
  - Used e.g to digitize sound
- Digital to analog conversion (DAC)
  - Digital number is converted to a voltage on the output pin
  - Used e.g. to play sounds

Lecture overview

- Bus based systems
  - Timing diagrams
  - Bus protocols
  - Various busses
- Basic I/O devices
- RAM
- Custom logic
  - FPGA
- Debug systems
  - JTAG
  - Logic analyzers

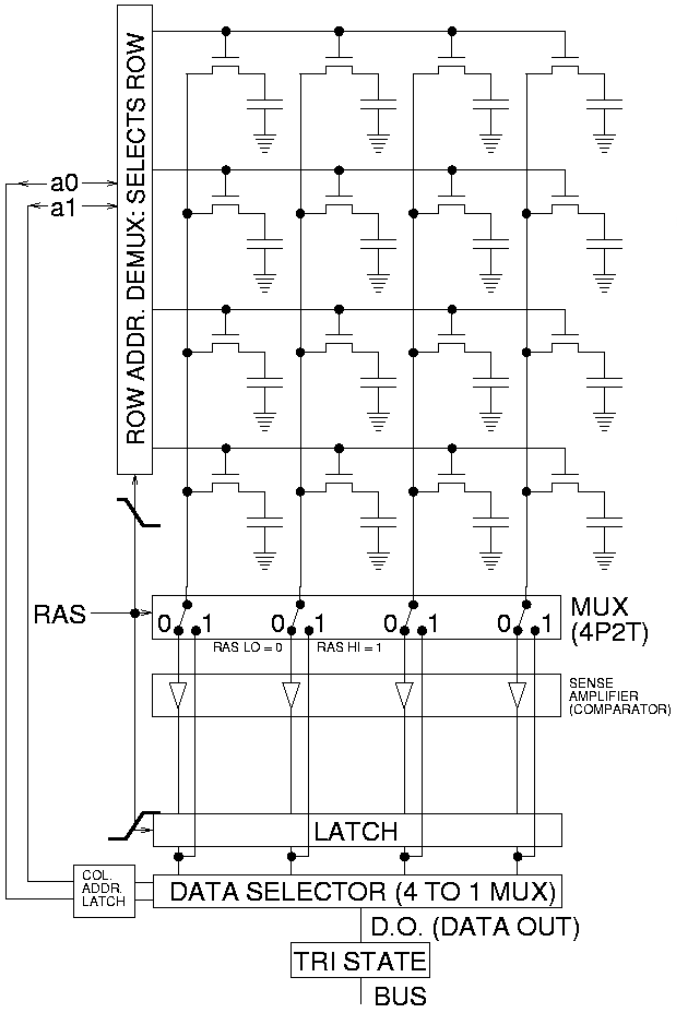
RAM

- Random access memory: Can retrieve from a random location at any time (as opposed to sequential access)
- Types:
  - Static RAM:
    - 6 transistors (two inverters and two access transistors)
    - Stores value until powerdown
    - Fast, big, used in cache and small embedded systems
  - Dynamic RAM:
    - Value stored in a capacitor
    - Must be refreshed periodically
    - Used in large off-chip memory



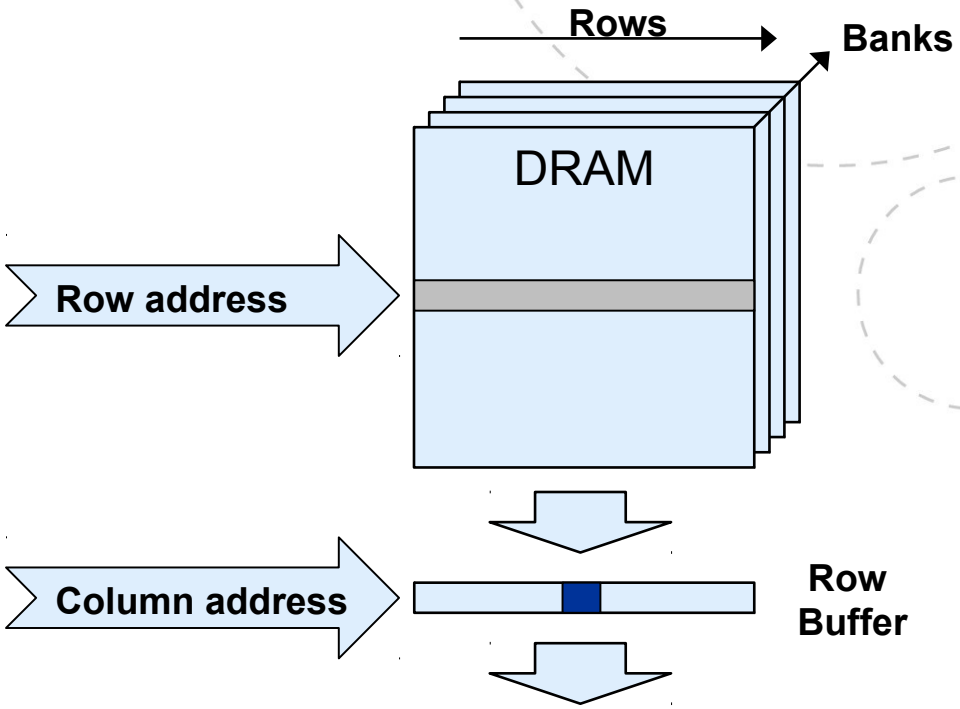
DRAM array

1. Precharge bitlines
2. Open row
   - Sense amps helps drive bitlines high or low
3. Latch in row
4. Mux out wanted column

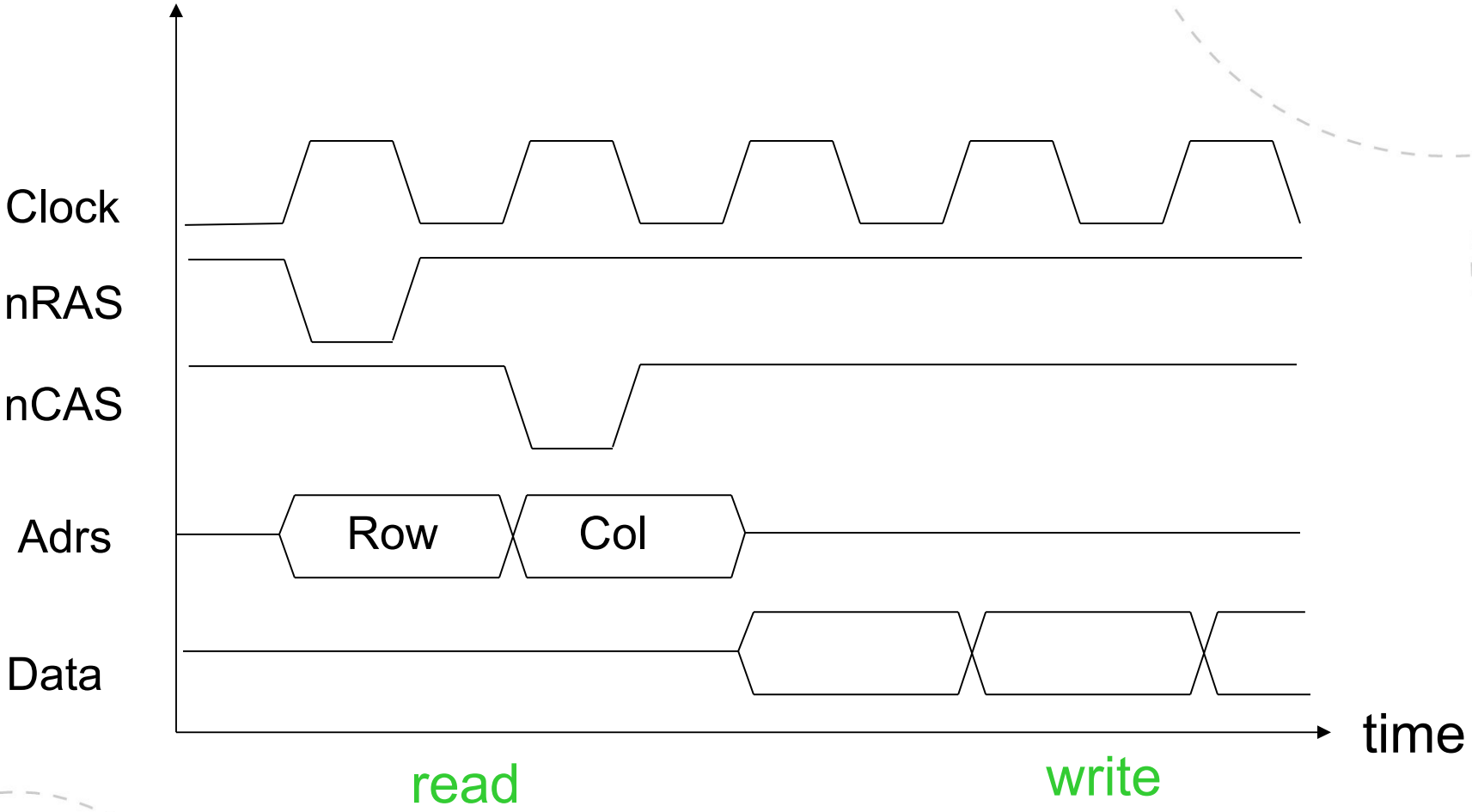


DRAM

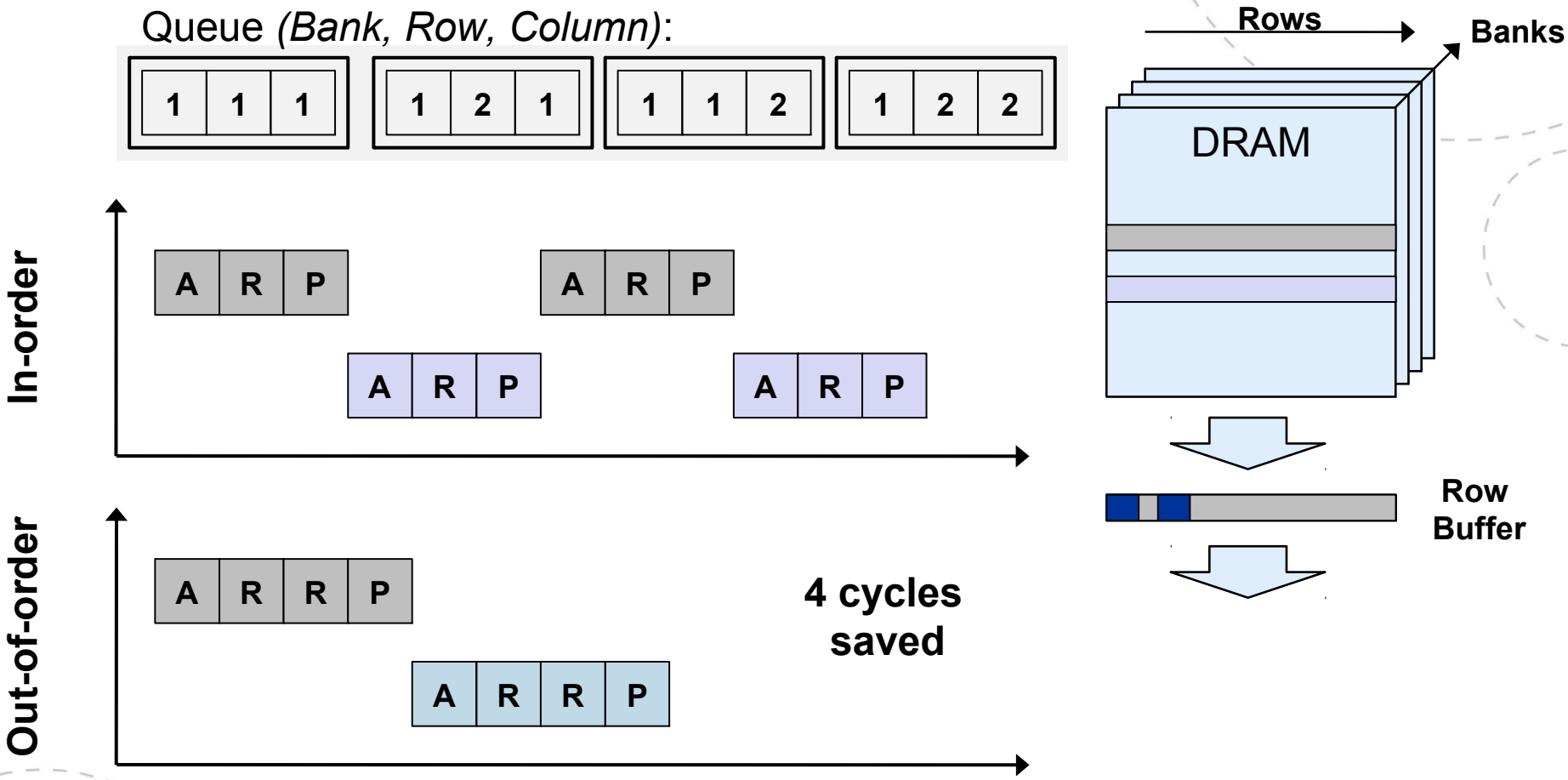
- Interfaces:
  - DRAM: Asynchronous bus interface
  - SDRAM: Synchronous bus interface
  - DDR SDRAM: Syncrhonous bus interface with data transfer on both rising and falling edge
- Need DRAM controller
- Performance
  - Multiple banks
  - Multiple channels



SDRAM transfer



Memory request scheduling



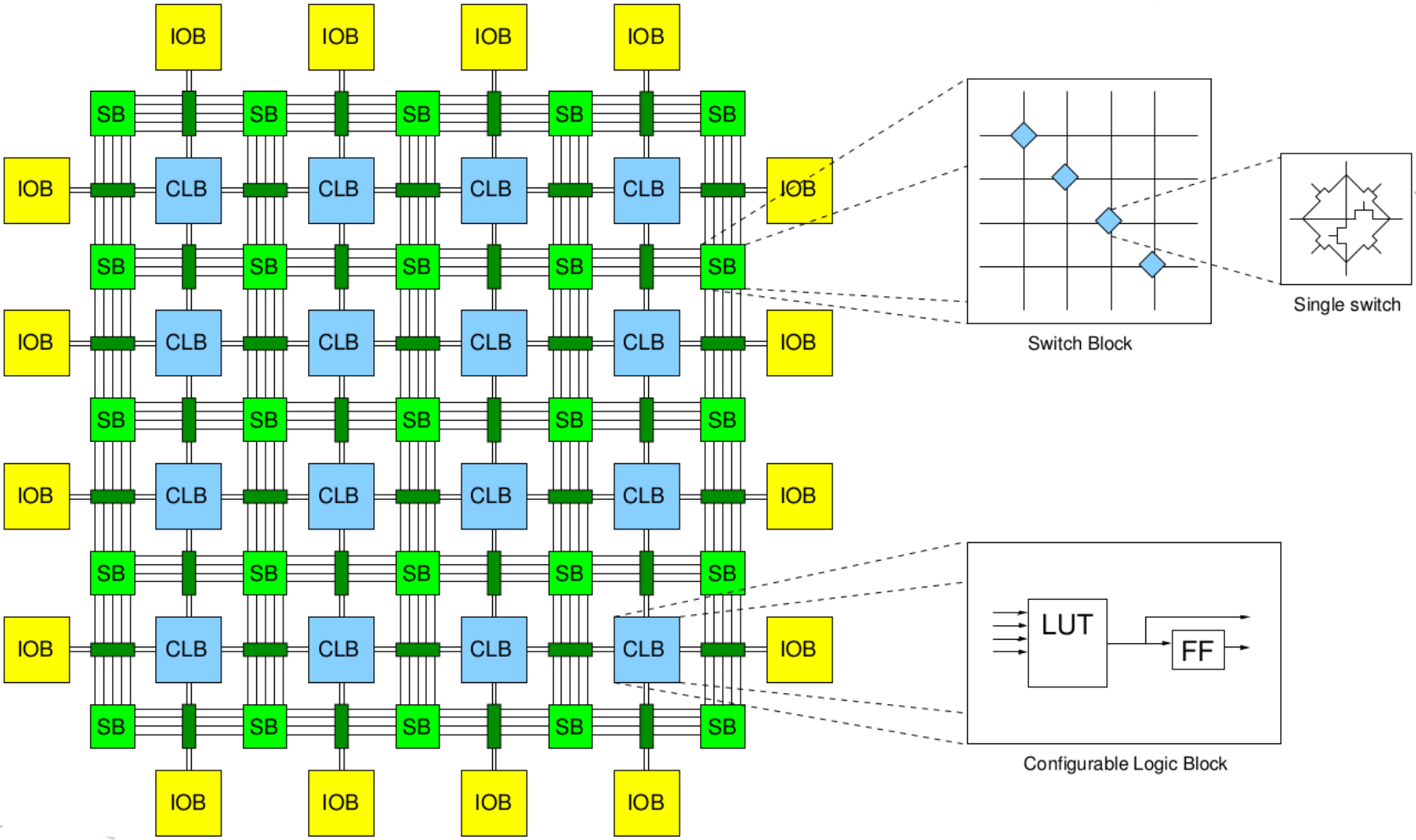
Lecture overview

- Bus based systems
  - Timing diagrams
  - Bus protocols
  - Various busses
- Basic I/O devices
- RAM
- Custom logic
  - FPGA
- Debug systems
  - JTAG
  - Logic analyzers

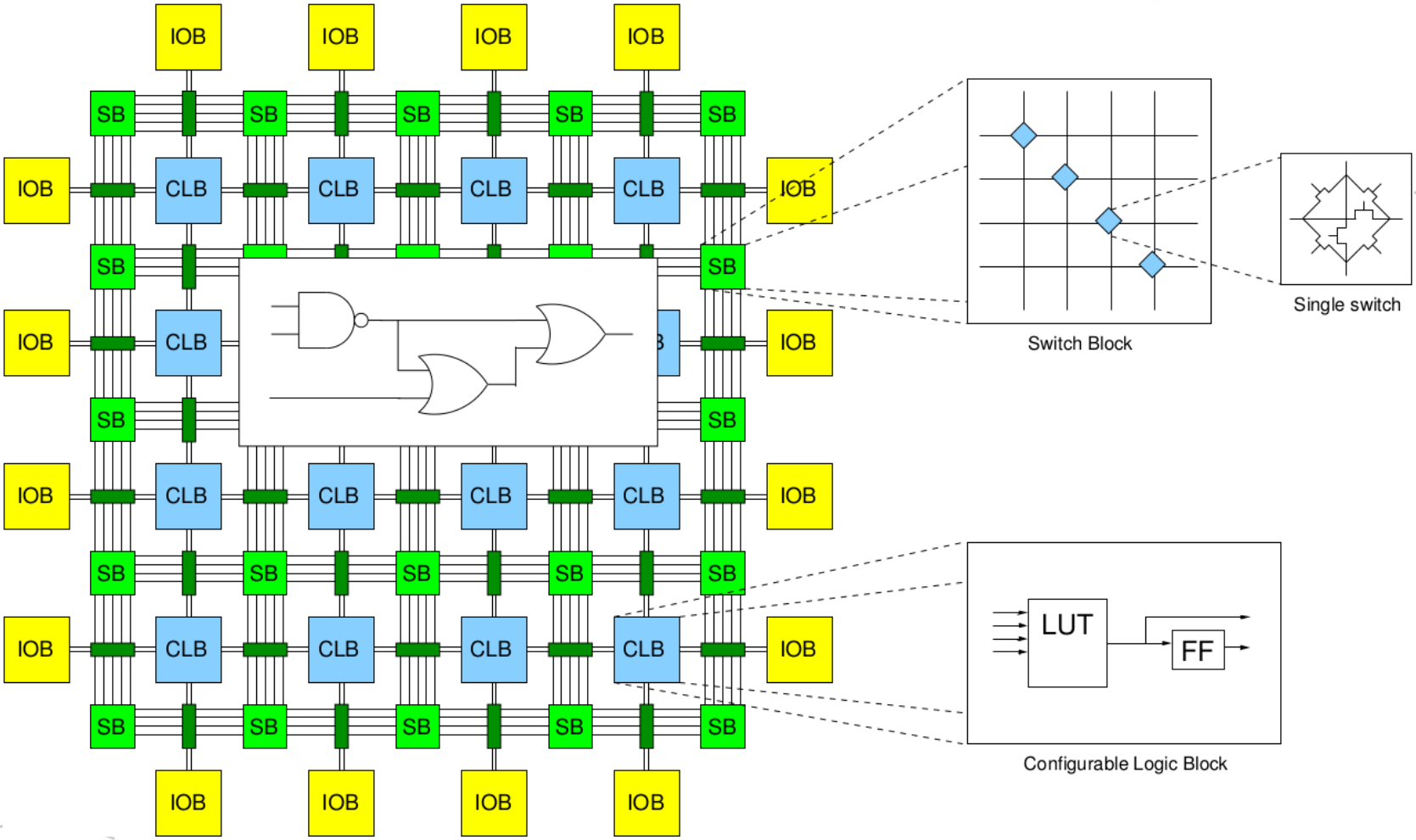
Adding custom logic to a board

- Application Specific Integrated Circuits (ASICs)
  - Design your own chip
  - Expensive
  - Takes time and money
- Field Programmable Gate Arrays (FPGAs)
  - General purpose, configurable logic chip
  - Can implement any kind of digital logic
  - Exists in many different sizes
  - Much cheaper than ASIC (for low volumes)
  - Faster and simpler to design for
  - Lower performance than ASIC using equivalent process technology

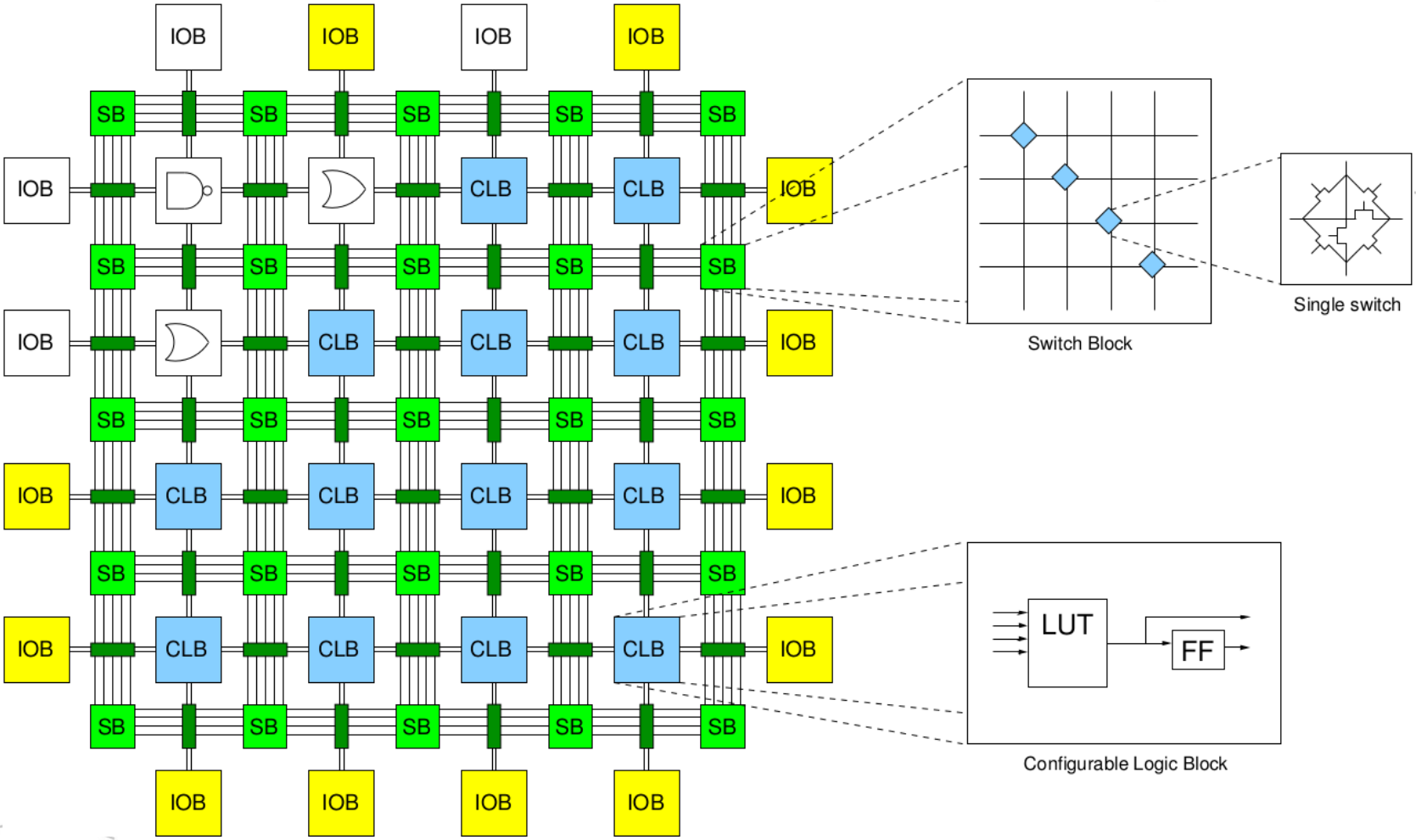
Field Programmable Gate Arrays



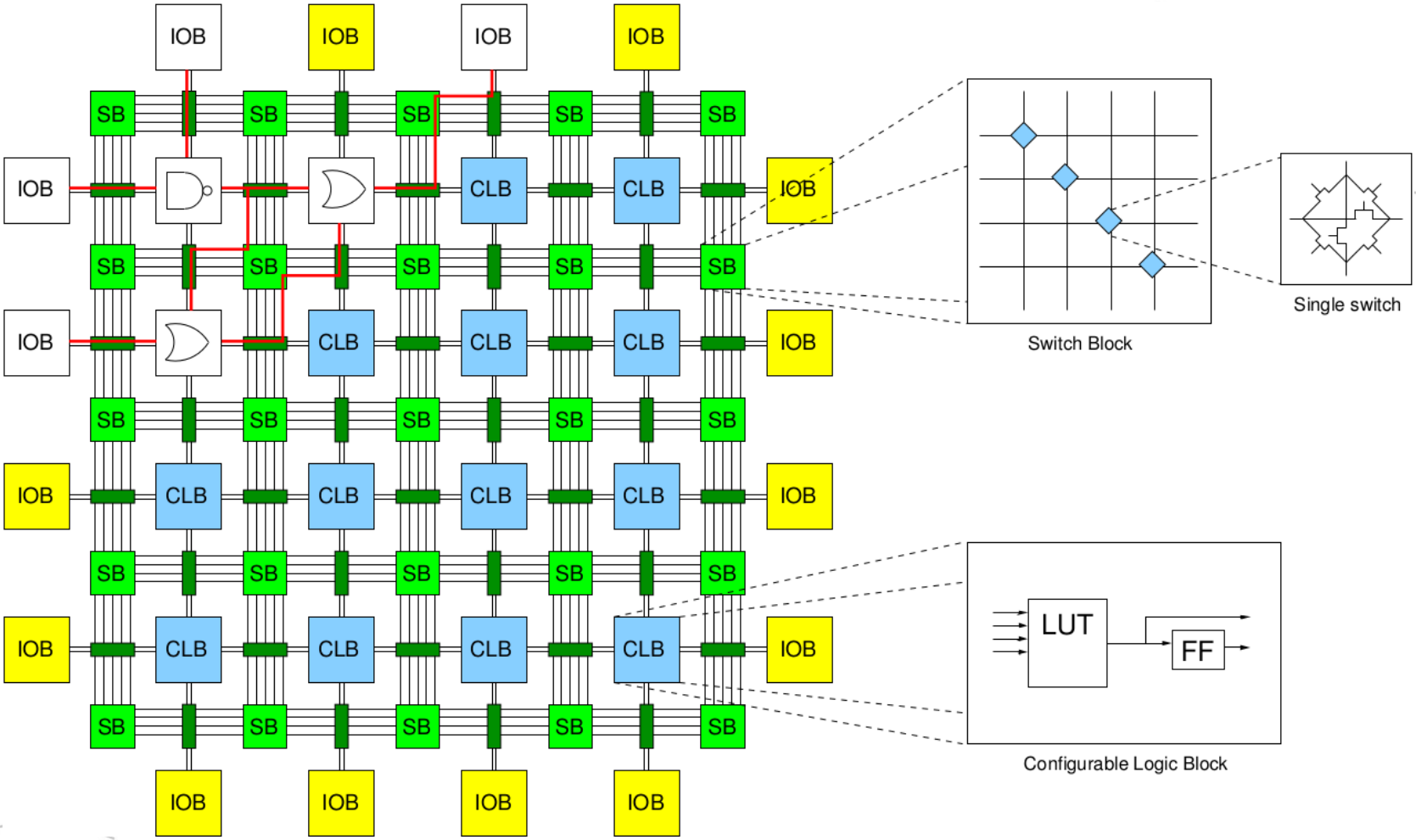
Field Programmable Gate Arrays



Field Programmable Gate Arrays



Field Programmable Gate Arrays



Designing for FPGAs

- Hardware Description Language (HDL)
- Textual specification of which flipflops and gates are needed, and how to connect them
- The FPGA tool flow can transform the HDL file and automatically synthesize, place and route and produce the final FPGA configuration file.
- Digital design with HDL:
  - Looks like programming, but is not
  - Just as for programming: Takes years to master

Lecture overview

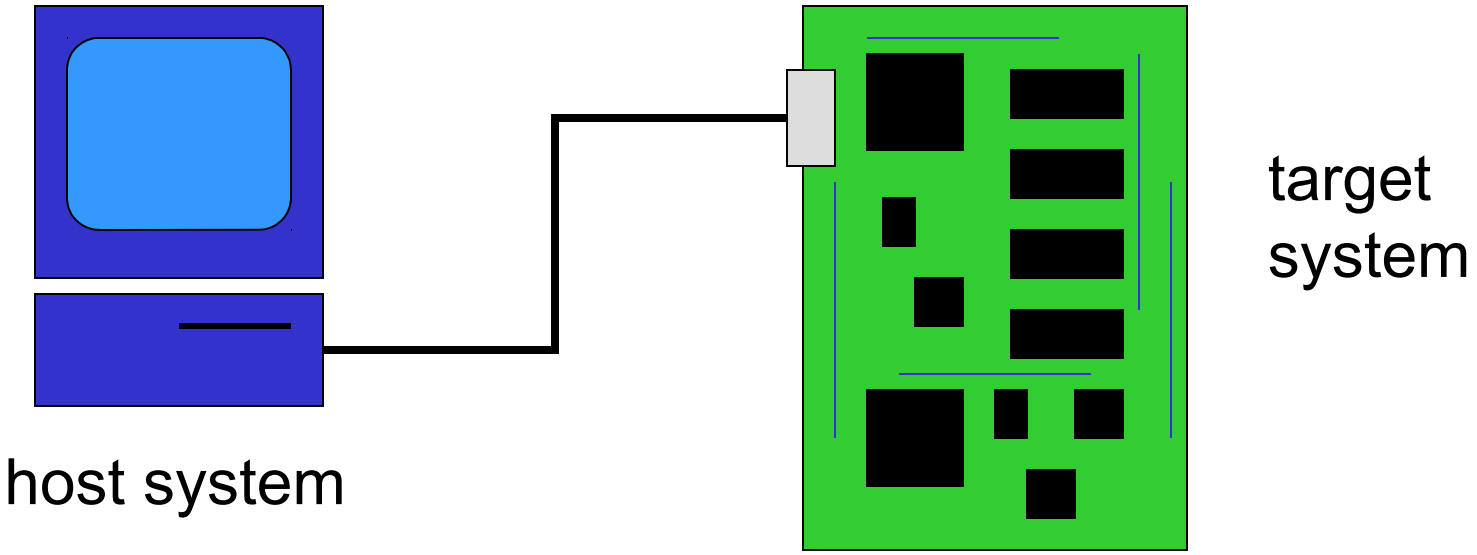
- Bus based systems
  - Timing diagrams
  - Bus protocols
  - Various busses
- Basic I/O devices
- RAM
- Custom logic
  - FPGA
- Debug systems
  - JTAG
  - Logic analyzers

HW prototype

- Evaluation boards are very useful
- Includes CPU, memory, some I/O devices
- Manufacturer often provides the design files which can be helpful when designing a custom board
- Many evaluation boards provide prototyping areas with access to various signals
  - Easy to add new HW modules

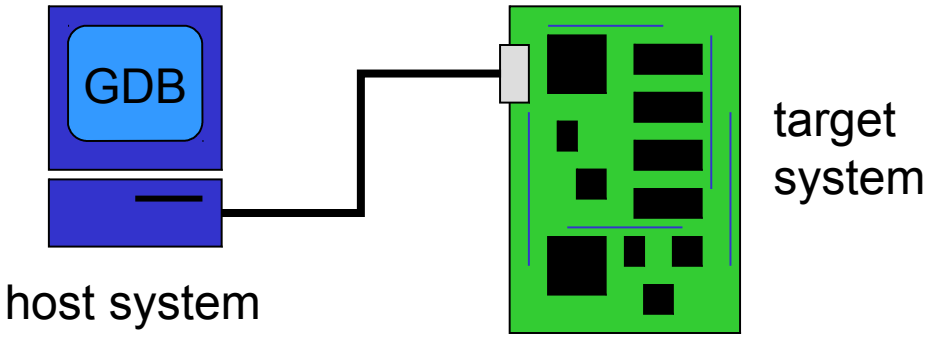
Development and debugging

- Use a host system to prepare software for target system



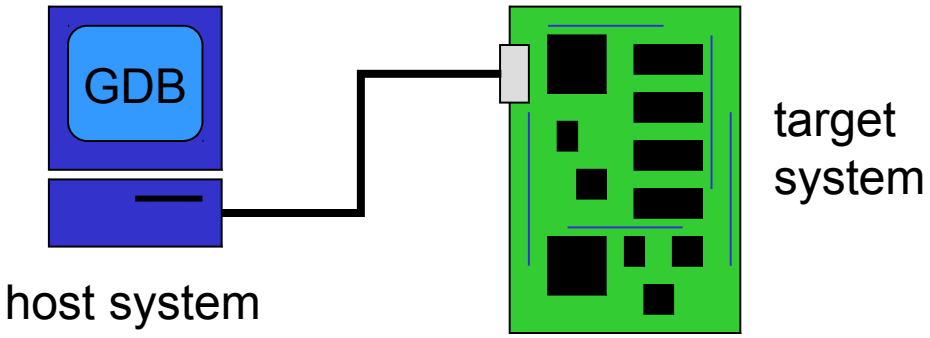
Host based tools

- Cross compilers
  - Compiles code on host for target system
- Cross debugger
  - Displays target state, allows target system to be controlled (single stepping, etc)



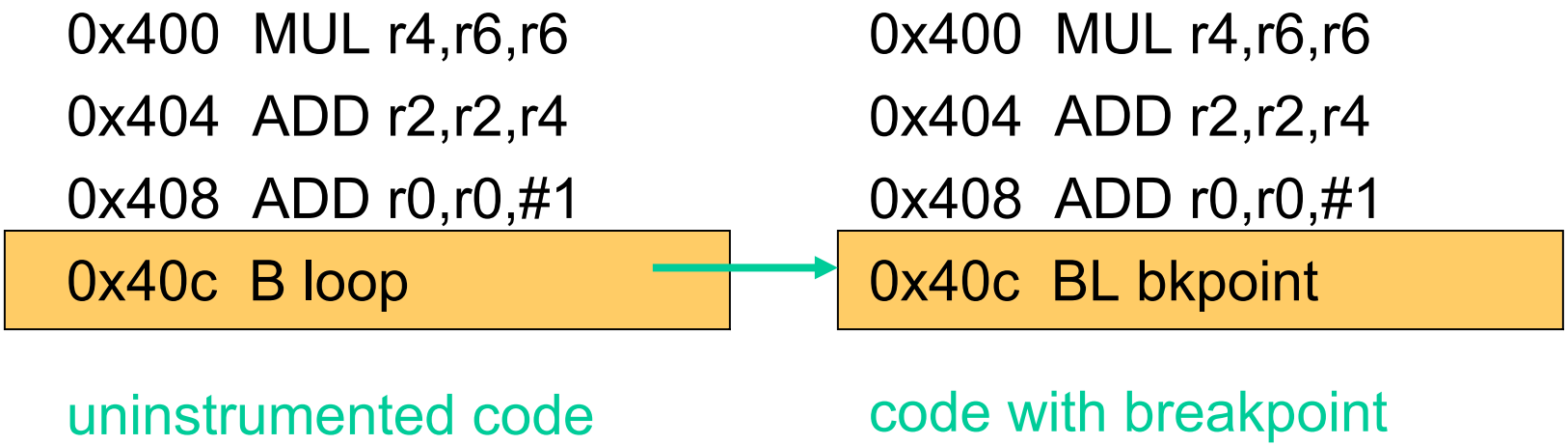
Software debuggers

- A monitor program residing on the target provides basic debugger functions
- Debugger should have a minimal footprint in memory
- User program must be careful not to destroy the debugger SW itself



Breakpoints

- A breakpoint allows the user to stop execution, examine system state, and change state
- Replace the breakpointed instruction with a subroutine call to the monitor program
  - Must preserve state, just like an interrupt handler

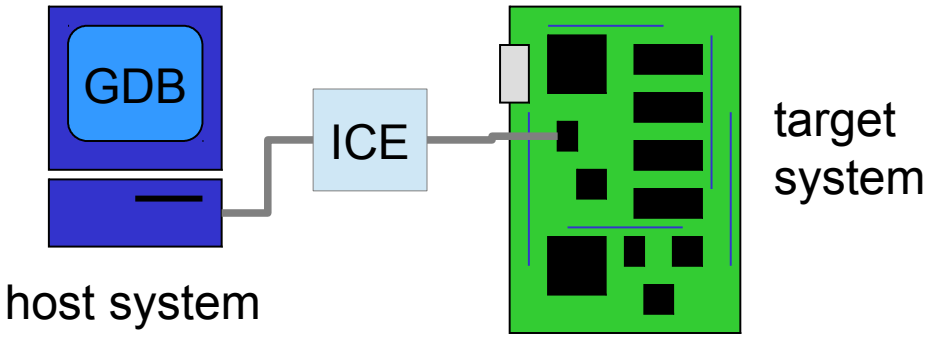


Using HW for debugging

- Necessary if things goes wrong even before the SW debugger loads
- LEDs
  - Very useful for simple debugging
  - Most boards provide one or more LEDs connected to some GPIO pins
    - Your custom HW prototype should also include LEDs
  - “The printf() of embedded systems”
    - Starting point when nothing else works
    - Fallback when all else fails

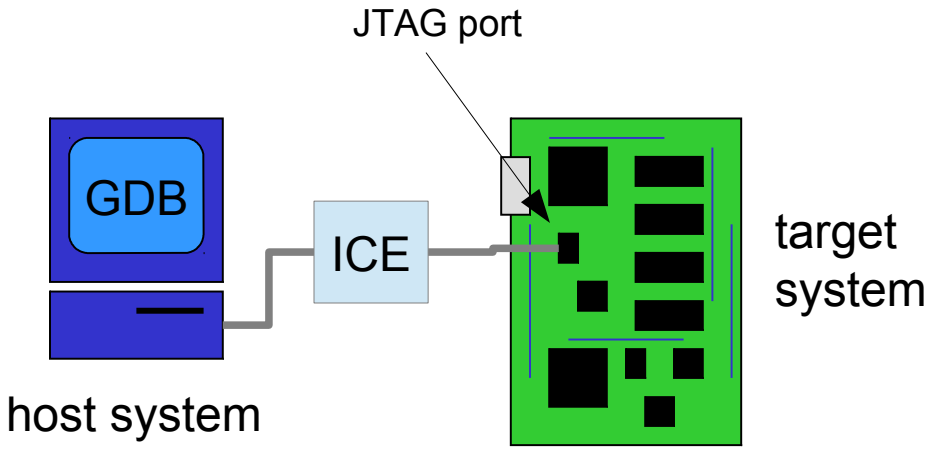
In Circuit Emulators (ICE)

- Many microcontrollers have hardware support for debugging
- Can connect a special HW device through a debug port which can
  - Read state
  - Change state
  - Single step
- Requires no SW support on the target
- Can often interface with your typical debugger (gdb)



JTAG (Joint Test Action Group)

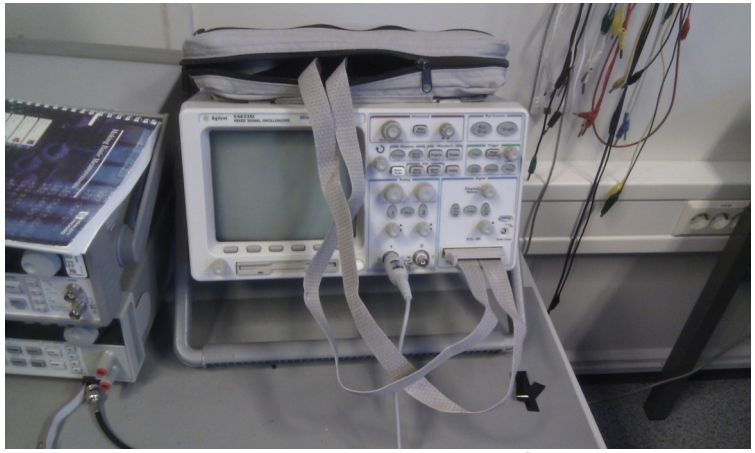
- IEEE standard for debug access to various HW
  - CPUs
  - FPGAs
  - ...
- Useful for
  - Boundary scan (manufacture testing)
  - Read out state
  - Program flash
  - Provide HW debugging access to CPUs
  - Many non-standard extensions
    - Often requires vendors own JTAG HW adapter and SW for full functionality



Logic analyzers

- Not everything can be debugged properly with debuggers (SW or HW)
  - External bus behaviour
  - How various components on the board talks together
- Solution: Logic analyzer or Oscilloscope
  - Oscilloscope provides full analog waveform of a signal on the board
  - If you suspect a logical fault, not a signal quality fault, a logic analyzer is a better choice

Logic analyzers



- An array of low-grade oscilloscopes
  - Displays 0 or 1 instead of the actual voltage
- Samples all inputs at regular intervals
  - Based on internal clock or external clock
- Presents the values in a nice timing diagram on-screen
- Logic analyzers often understands many bus protocols and displays data properly according to protocol

The hardware lab

Next lecture

- Next week
- Guest lecturer Snorre Aunet (IET)
  - Low power design on the physical level