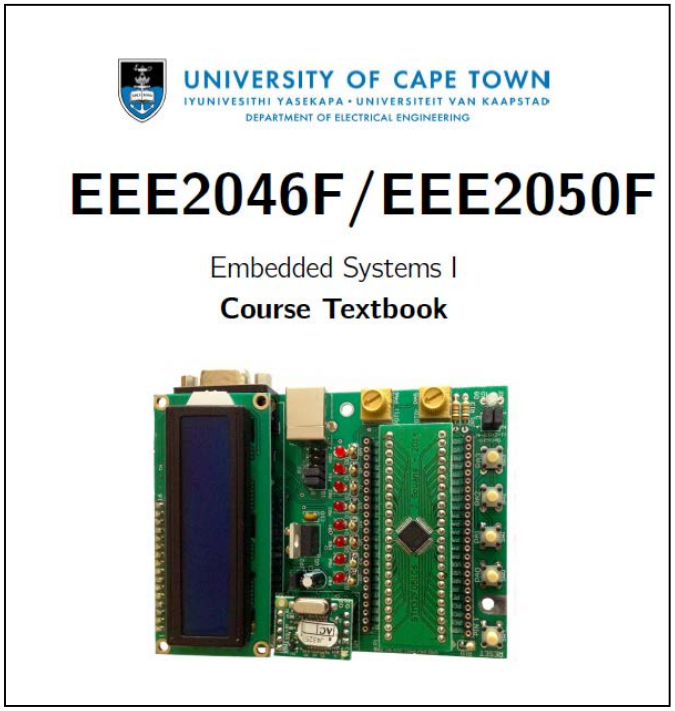
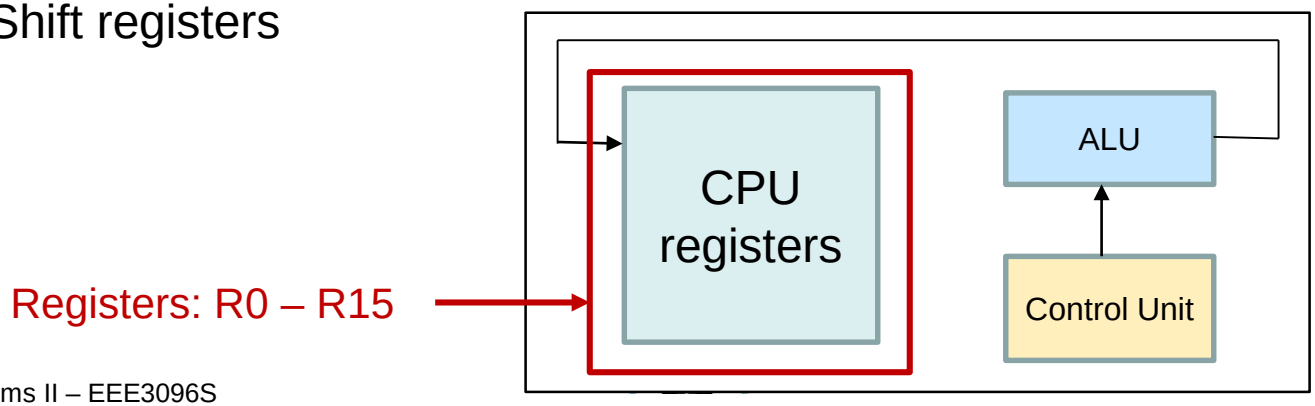
Digital Building Blocks
Registers



Chapter 4.3
"Digital Building Blocks" by R. Verrinder
EEE2046F/EEE2050F Embedded Systems I notes

Registers
Overview

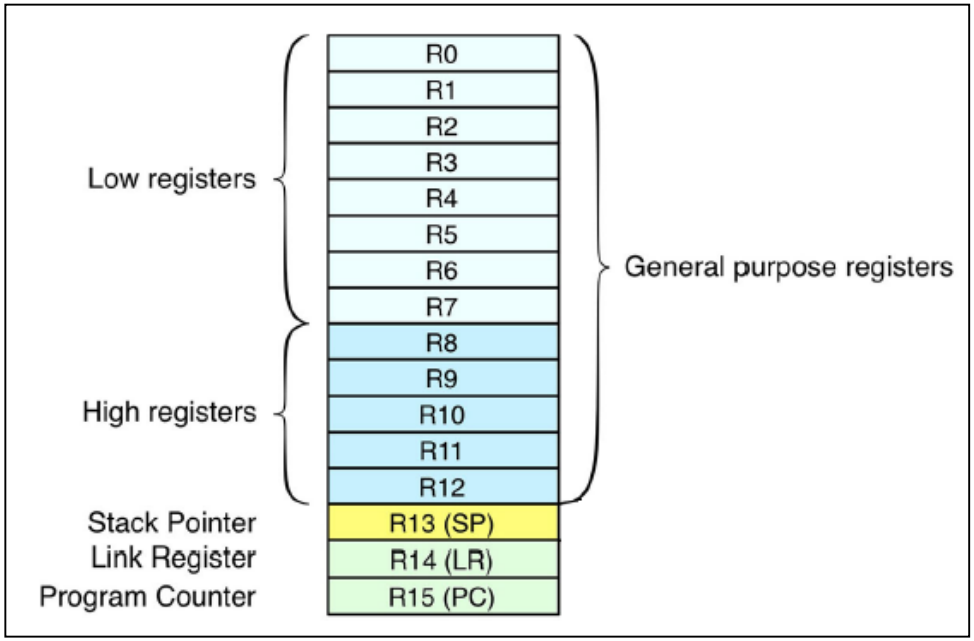
- What is a register?
  - A register is a memory type used in a processor
- How are registers used?
  - Registers are used to store data in binary format. Thus, a n-bit register can store a n-bit binary number
- Types of registers covered
  - Reset and enable registers
  - Register files
  - Shift registers



Simplified block diagram of generic CPU

Registers
A n-bit register: functionality

- Functionality of a n-bit register
  - Store n-bits of data
  - Reset n-bits of data
  - Write new n-bits of data into the register

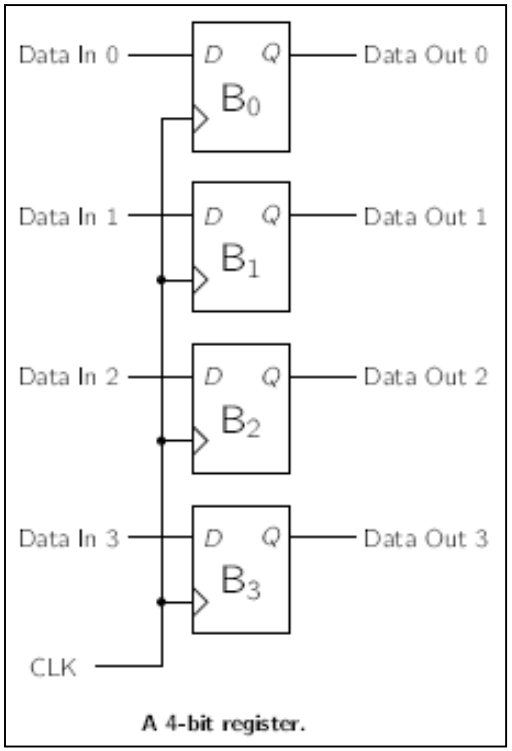


CPU registers of the Cortex M0

Registers
A practical n-bit register

- A n-bit register can be constructed using n D flip flops

- Example: a 4 bit register
  - CLK: connects to all flip flops
  - Data input: connects to the D inputs
  - On ↑ edge of the clock: Q = D
  - Data is read from the Q outputs



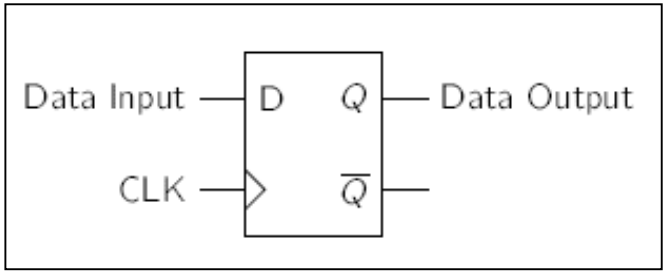
A 4-bit register.

Registers
Building blocks of a register: D flip flops

- Functionality of a D flip-flop for register usage
  - Storage: Q output stores the current output

Operation of a basic D flip flop

D	CLK	Next State of Q
X	0	$Q = Q_{prev}$
X	1	$Q = Q_{prev}$



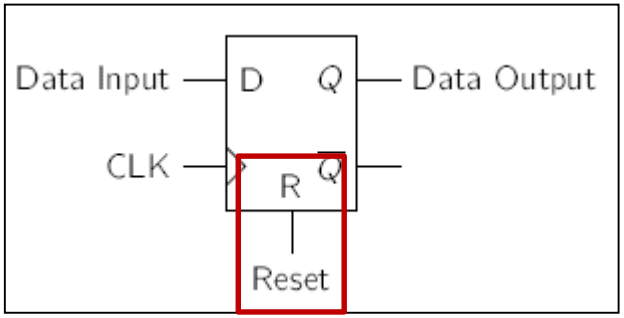
D flip-flop symbol

Registers
Building blocks of a register: D flip flops

- Functionality of a D flip-flop for register usage
  - Storage: Q output stores the current output
  - Reset: R input used to reset Q = 0

Operation of a D flip flop with reset

R	D	CLK	Next State of Q
0	X	↑	Q = D
1	X	↑	Q = 0



D flip-flop with reset symbol

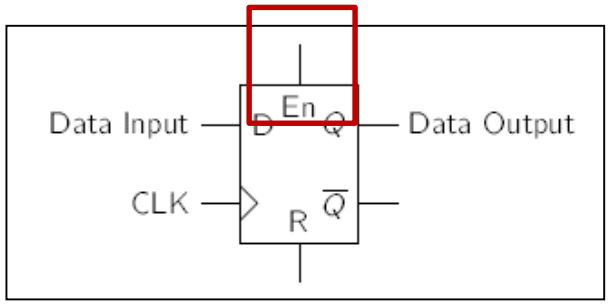
Registers
Building blocks of a register: D flip flops

- Functionality of a D flip-flop for register usage
  - Storage: Q output stores the current output
  - Reset: R input used to reset Q = 0
  - Write: Enable (En) used to load input data to the output
    - En = 0, $Q = Q_{prev}$ on ↑ CLK
    - En = 1, Q = D on ↑ CLK

Operation of a D flip flop with enable

En	R	D	CLK	Next State of Q
0	0	X	↑	$Q = Q_{prev}$
1	0	X	↑	Q = D

↑ represents the rising edge of the clock



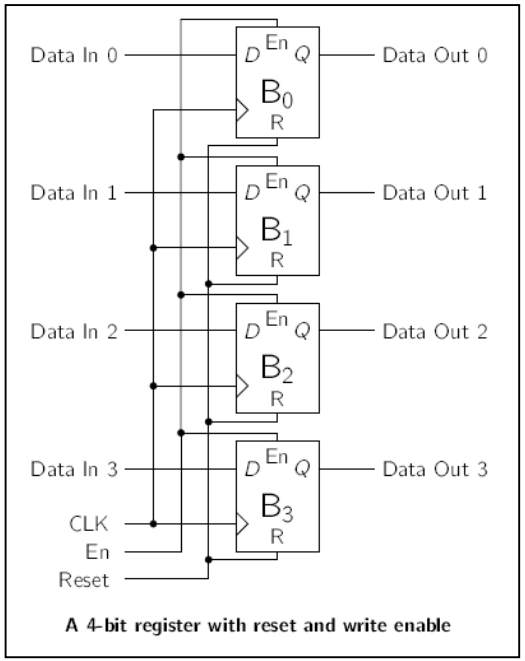
D flip-flop with reset and enable

Registers
A practical n-bit register with reset and enable

- A n-bit register with reset and enable functionality can be constructed using n D flip-flops with R and En pins
- Example: a 4-bit register with reset and write enable
  - Storage: current outputs are stored in Q
  - Reset: R = 1 sets Q = 0
  - Write: En = 1 activates D flip flop.

    On ↑ CLK, Q = D_{in}
  - Disable write: En = 0 deactivates D flip flops.

    On ↑ CLK, Q = Q_{prev}



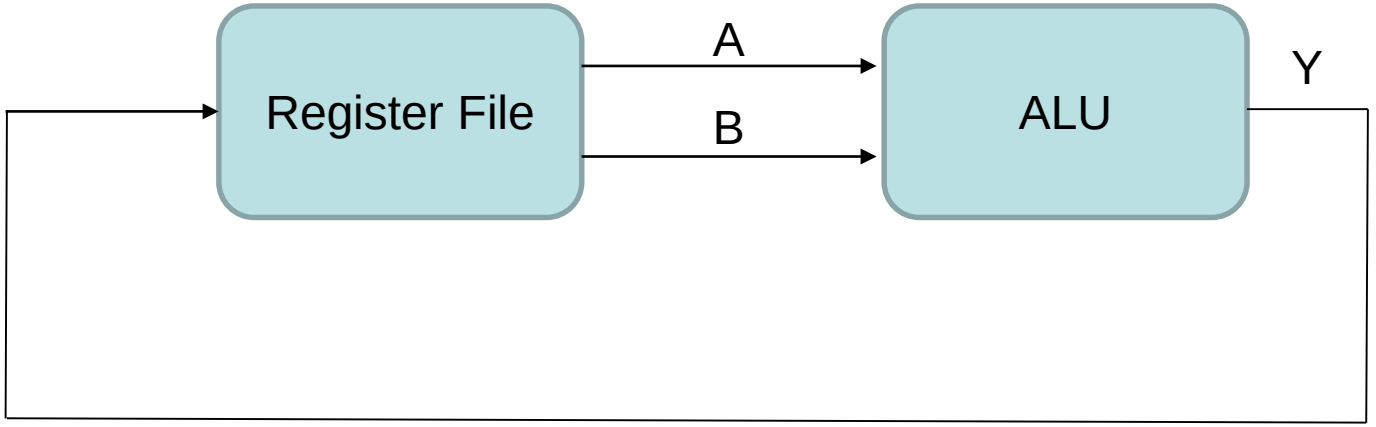
A 4-bit register with reset and write enable

Register Files

Register Files
Introduction

- A register file is a basic memory structure that is used in a CPU to perform the following functionality:
  - Output the values A and B that are inputs to the ALU
  - Store the output Y from the ALU

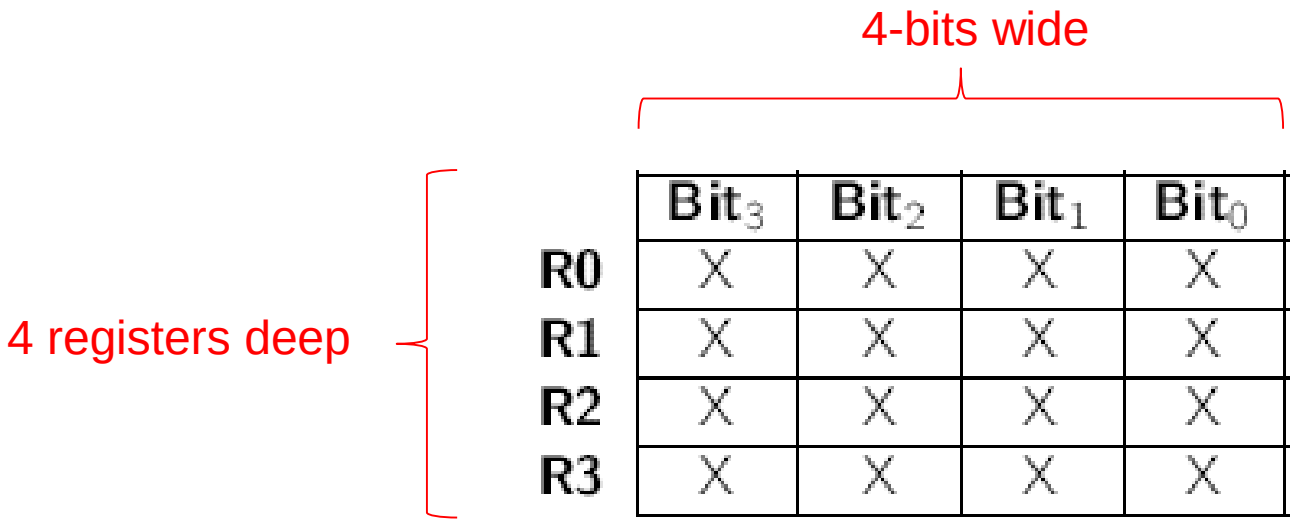
- The registers are typically as wide as the word length of the CPU and used as a temporary storage structure



Register Files
A 4-bit register file

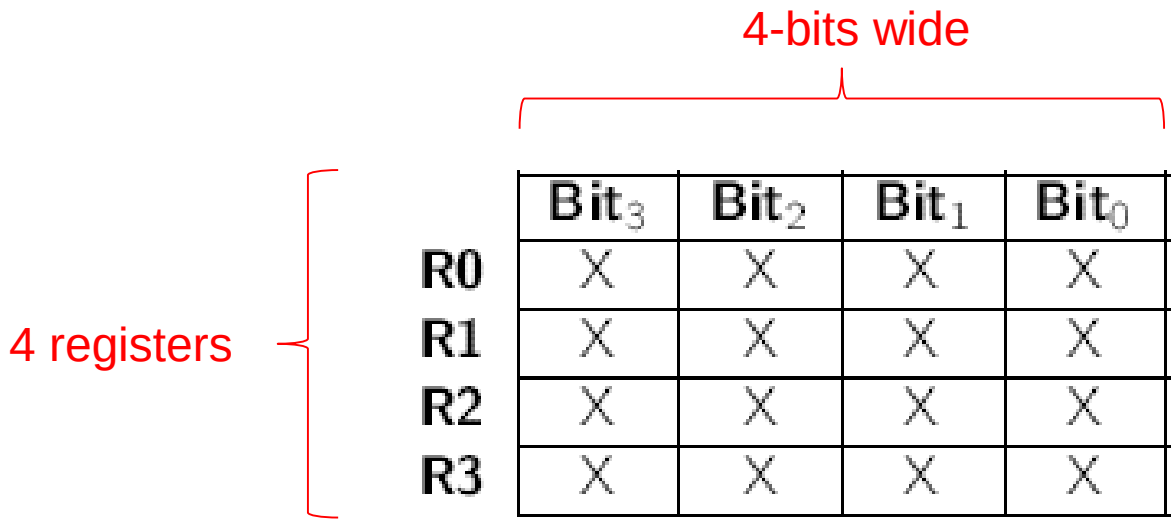
- Let's look at an example of a 4-bit register file to understand its functionality

- Properties of a 4-bit register file
  - 4 registers deep,
  - Each register is 4 bit wide



Register Files
A 4-bit register file: functionality

- A register file has the following functionality:
  - Read
  - Reset
  - Write
  - Disable write

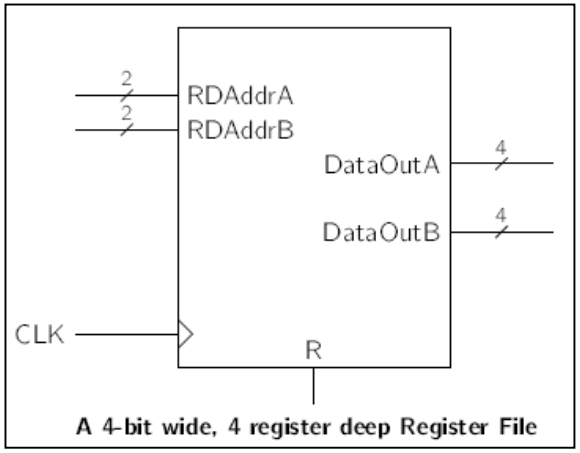


Register Files

A 4-bit register file: read functionality

- Two outputs, A and B, can be read out of the register file using the following steps
  1. Set the 2-bit read address for A on RDAddrA
  2. Set the 2-bit read address for B on RDAddrB
  3. Ensure that Reset is cleared: R = 0
  4. On the next ↑ of the CLK, DataOutA and DataOutB will have the contents of the registers at addresses RDAddrA & RDAddrB respectively

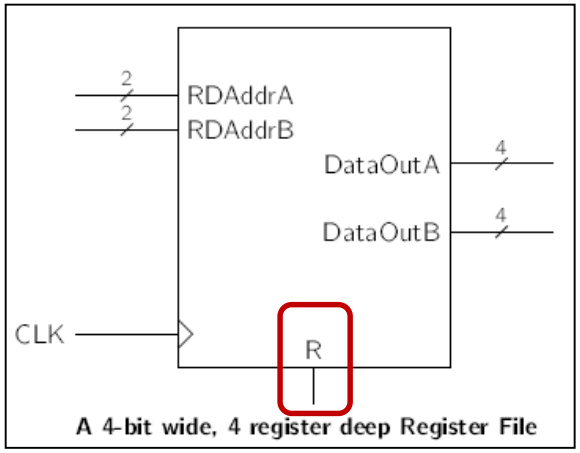
Address		Register
0	0	R0
0	1	R1
1	0	R2
1	1	R3



A 4-bit wide, 4 register deep Register File

Register Files
A 4-bit register file: reset functionality

- All four registers can be reset using the following steps:
  1. Ensure that Rest is set: R = 1
  2. On the next ↑ of the CLK, all 4 registers will have the binary value ‘0000’



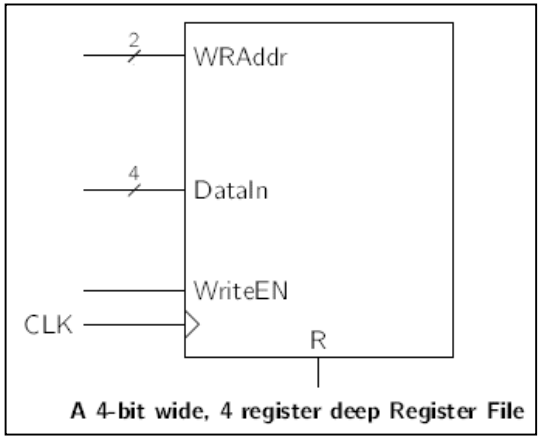
A 4-bit wide, 4 register deep Register File

Register Files

A 4-bit register file: write functionality

- A four-bit binary value can be written to one of the four registers in the following way:
  1. Set the 2-bit write address on WRAddr
  2. Let DataIn = 4 bit binary value to be written
  3. Set WriteEn = 1, and ensure R = 0
  4. On the next ↑ of the CLK, the register contents specified by WRAddr will contain the 4-bit binary value

Address		Register
0	0	R0
0	1	R1
1	0	R2
1	1	R3



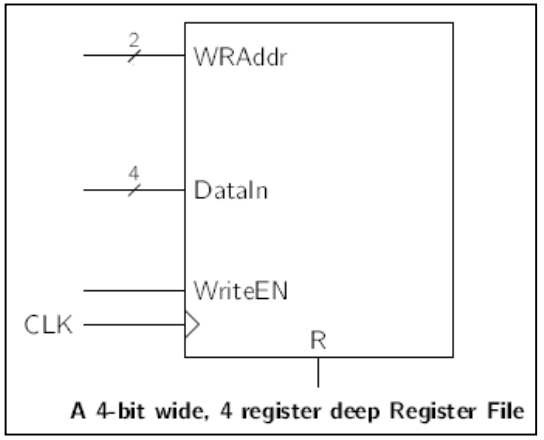
A 4-bit wide, 4 register deep Register File

Register Files

A 4-bit register file: disable write functionality

- The write functionality is disabled in the following way:
  1. Set WriteEn = 0, and ensure R = 0
  2. On the next ↑ of the CLK, the register contents of the register file, R0 – R4, will remain the same

Address		Register
0	0	R0
0	1	R1
1	0	R2
1	1	R3

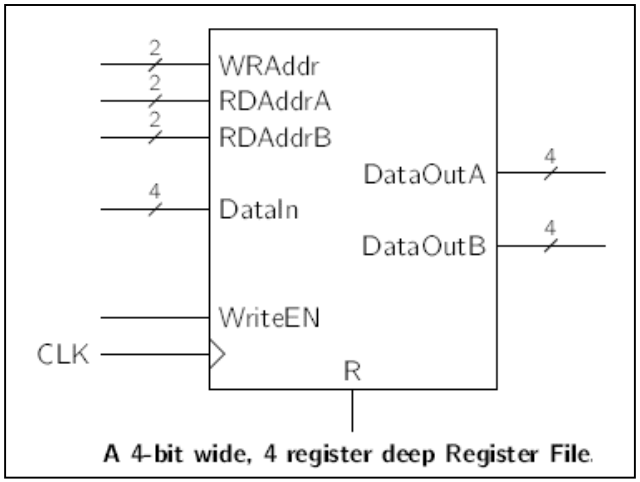


A 4-bit wide, 4 register deep Register File

Register Files
A 4-bit register file: all functionality

- A register file with all its functionality:
  1. Read: RDAddrW, RDAddrB, DataOutA, DataOutB,
  2. Reset: R
  3. Write: WRAddr, DataIn, WriteEn
  4. Disable write: WriteEn

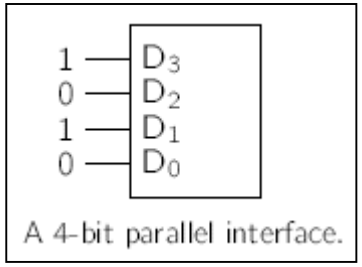
Address		Register
0	0	R0
0	1	R1
1	0	R2
1	1	R3



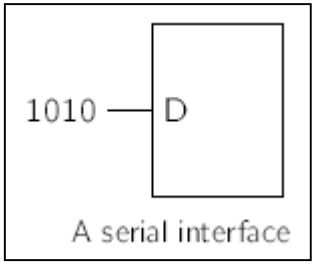
Shift registers

Shift Registers
Overview

- Shift registers are specialized registers used to convert data from one format to another
- Three types covered
  - Serial in/ serial out
  - Serial in/ parallel out
  - Parallel in/ serial out
- Why are shift registers important?
  - They are used in low-level communication interfaces in microcontrollers to shift data into or out of a device



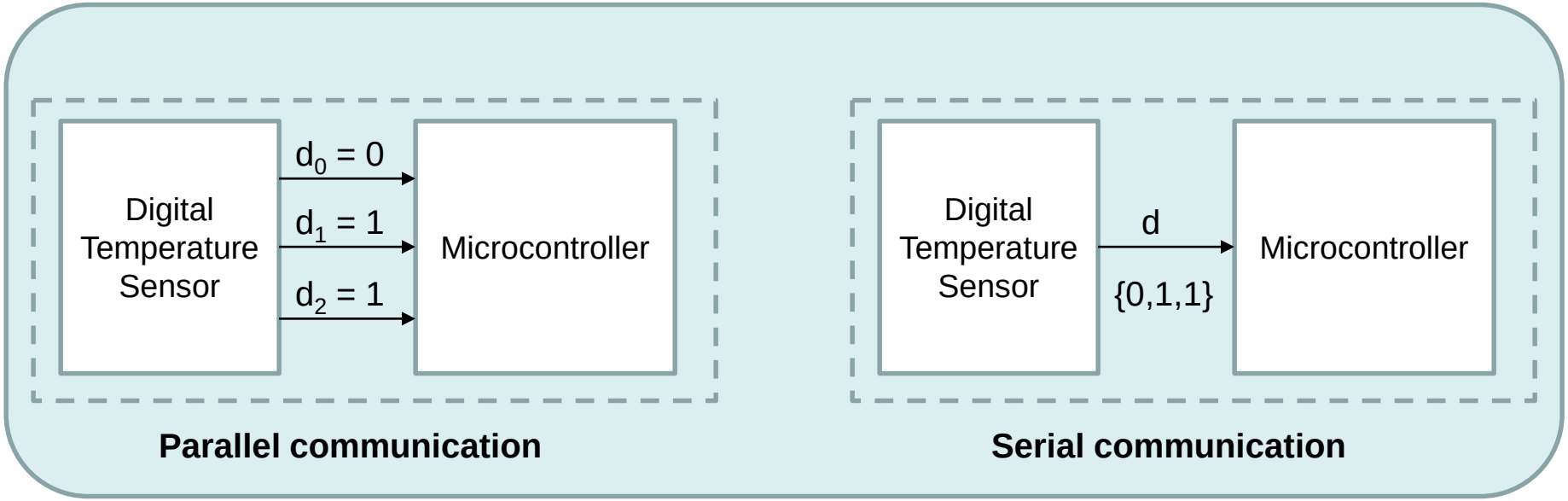
A 4-bit parallel interface.



A serial interface

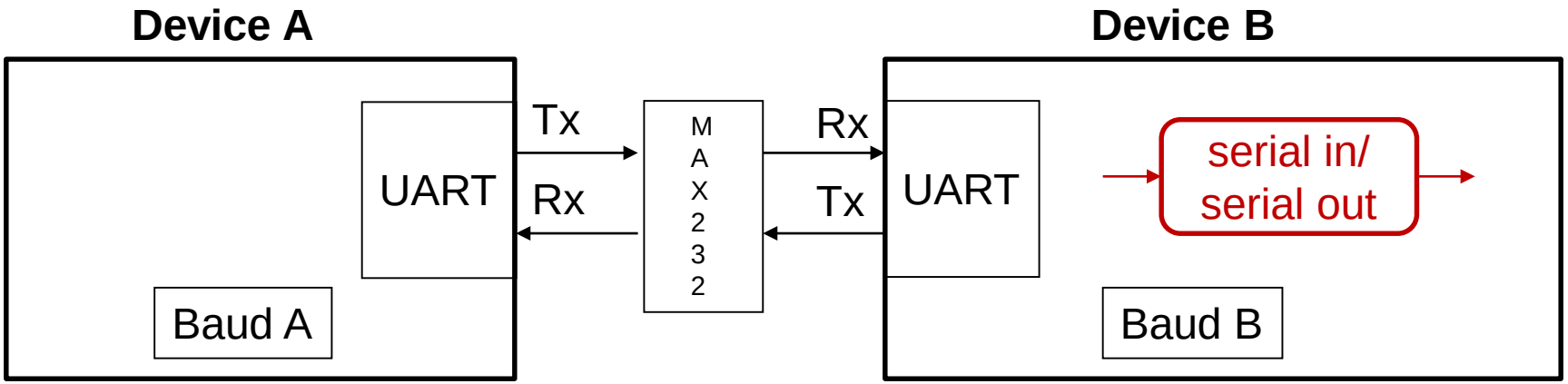
Shift Registers
Serial versus parallel interface

- In a parallel interface, n-bits of data are transferred at the same time
- In a serial interface, n-bits of data are transferred one bit at a time



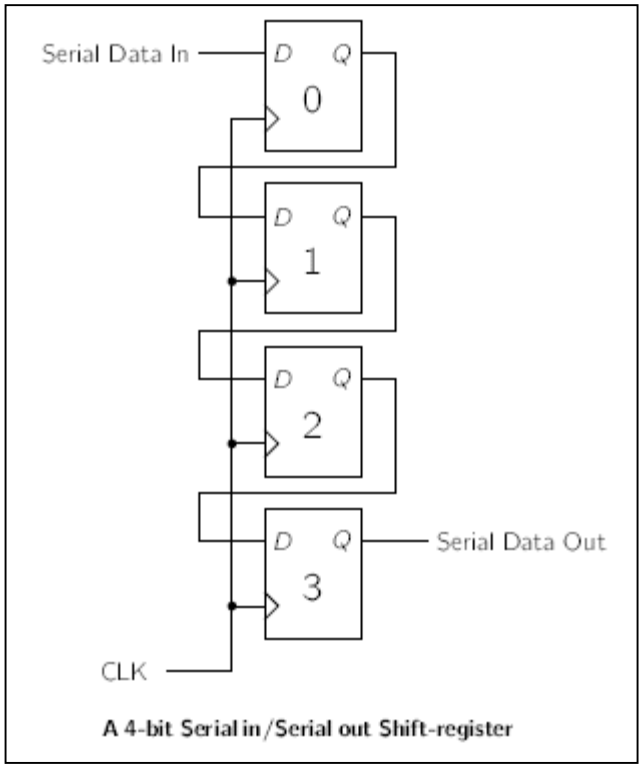
Shift Registers
Serial in/ serial out

- A serial in/ serial out shift register takes in n-bits of asynchronous serial data and outputs n-bits of synchronous serial data
- Example:
  - Device A exchanges serial data with device B asynchronously
  - Device B may used a serial in/ serial out register to convert an asynchronous serial data into a synchronous serial data



Shift Registers
Serial in/ serial out

- A n-bit serial in/ serial out shift register is made up of n D flip flops
- An output of a flip flop is fed into the input of the next flip flop
- On the ↑ of the CLK, data is shifted through the register



A 4-bit Serial in/Serial out Shift-register

Shift Registers
Serial in/ serial out: principle of operation

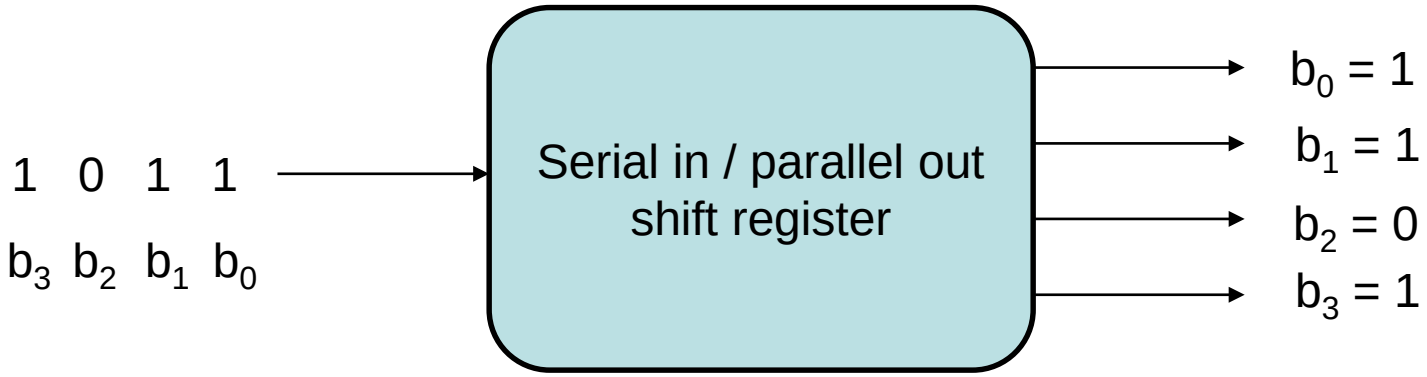
- Example: a 4-bit serial in/ serial out shift register
  - Consider a binary value of 1010 as the input
  - After every ↑ of the CLK, a new bit is shifted into the register
  - It takes 4 clock cycles for the first bit of the serial message to appear on the output

CLK	Q_0	Q_1	Q_2	Q_3	Output
0	0	0	0	0	
1	**0**	0	0	0	0
2	**1**	**0**	0	0	00
3	**0**	**1**	**0**	0	000
4	**1**	**0**	**1**	**0**	0000
5	0	**1**	**0**	**1**	00000
6	0	0	**1**	**0**	100000
7	0	0	0	**1**	0100000
8	0	0	0	0	10100000

The operation of a 4-bit Serial In/Serial Out Shift Register

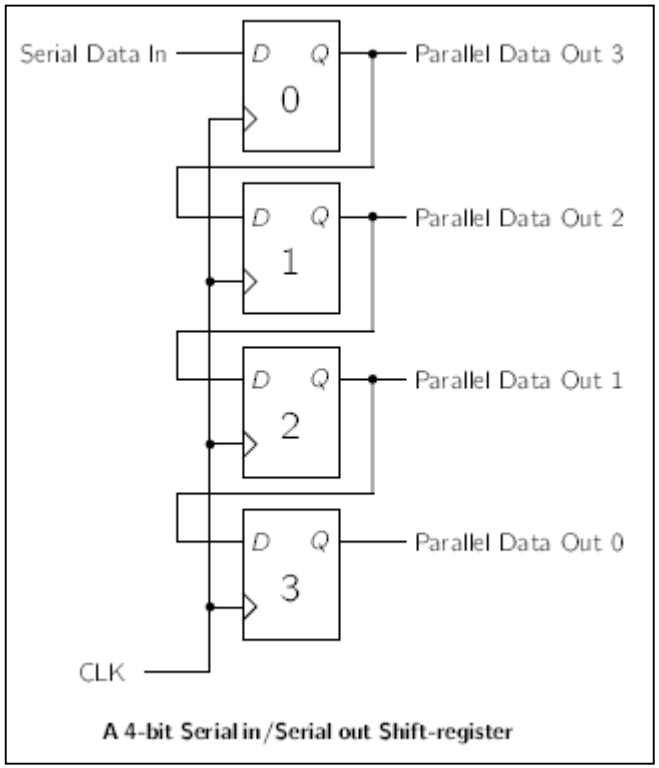
Shift Registers
Serial in/ parallel out

- A serial in/ parallel out shift register takes a n-bit serial number and converts it into a n-bit parallel interface



Shift Registers
Serial in/ parallel out

- Example: the circuit diagram for a 4-bit serial in/ parallel out shift register is made up of 4 flip flops
- It is similar to a serial in/ serial out shift register, where the data outputs are the Q outputs of the flip flops
- It is used to convert a serial input steam of binary data into a multiple wire parallel output within a embedded system



A 4-bit Serial in/Serial out Shift-register

Shift Registers
Serial in/ parallel out

- Example: a 4-bit serial in/ parallel out shift register showing the principle of operation
  - Consider a binary value of 1010 as the input
  - After every ↑ of the CLK, a new bit is shifted into the register
  - It takes 4 clock cycles for the full binary value to be present on the four Q outputs

CLK	Q_0	Q_1	Q_2	Q_3	Output
0	0	0	0	0	0000
1	**0**	0	0	0	0000
2	**1**	**0**	0	0	1000
3	**0**	**1**	**0**	0	0100
4	**1**	**0**	**1**	**0**	1010

The operation of a 4-bit Serial In/Parallel Out Shift Register

Questions
Register operations

Given two 2-bit registers (R0 and R1) (both registers have reset and enable pins), design a circuit which will implement the following register transfer operations. These operations synchronously transfer data between the registers based on a binary opcodes. A table of the opcodes and their transfer operation is given below.

Opcode		Operation	
F_1	F_0	Function	
0	0	F0	R1←0
0	1	F1	R1← $\overline{R1}$
1	0	F2	R0← $\overline{R0}$
1	1	F3	R1←R0

- F0: R1←0 (Clear R1 synchronously with the clock signal)
- F1: R1←$\overline{R1}$ (Complement R1)
- F2: R0←$\overline{R0}$ (Complement R0)
- F3: R1←R0 (Copy contents of R0 to R1)