CSD table of content and site map

An introduction to digital systems (1) (2) using VHDL and microcontrollers using C (3)

CSD content is discussed in lectures and labs and applied for designing projects. Similar materials and slides can be found in book chapters and webs all over the Internet.

NOTE: be aware that materials that are not from this CSD course will contain slightly different symbol diagrams, naming conventions and writing styles, and thus, we will rename and adapt them in order to be used in our CSD subject as study materials.

1.3.1.3. AND, NAND

1.3.1.4. OR, NOR

1.3.1.5. XOR, NXOR

1.3.1.6. Tri-state buffers and pin bi-directionality (explained in L9.1)

1.3.2. Equations in Boole's algebra

1.3.3. Truth table and canonical equations

1.2.3.1. Sum of minterms

1.2.3.2. Product of maxterms

1.3.4. Timing diagrams

1.4. Analysis concept map: methods for simple combinational circuits {*Circuit C* , P1 section A}, {*Circuits P*}, {*Circuit Q*}, {*Circuit K*}

1.4.1. Method I: Pen and paper analysis using boolean algebra [L1.3]

1.4.1.1. Properties of boolean functions

1.4.1.2. Duality principle, De Morgan's laws

1.4.1.3. Product of sums (PoS)

1.4.1.4. Sum of products (SoP)

1.4.1.5. Any type of equation or non-standard forms

1.4.2. Method II: Proteus simulation (virtual laboratory) of logic circuits for deducing truth tables [Lab1_1]

1.4.2.1. SPICE algorithms

1.4.2.1.1. Proteus ISIS from Labcenter Electronics (EETAC licence available) (intended for all CSD projects)

1.4.2.1.2. Optional. Multisim (from National Instruments, UPC license available)

1.4.2.2. Optional. Digital simulators

1.4.2.2.1. Hades: interactive simulation framework

1.4.2.2.2. Deeds: Digital electronics education and design suite

1.4.2.2.3. LogicWorks: is a schematic drawing and interactive digital simulation package for demonstrating logic design principles and practices within the education sector and industry.

1.4.2.3. Optional. Printed circuit board (PCB) design.

1.4.2.3.1. KiCad, our preferred tool for PCB.

1.4.2.3.2. Ultiboard (from National Instruments, UPC licence available), Fritzing, Altium Designer, Fusion 360, Proteus PCB Software, etc.

1.4.3. Method III: Analysis using synthesis and simulation EDA tools [Lab1_2]

1.4.4. Method IV: WolframAlpha numerical engine for calculating truth tables and deducing logic circuits [Lab1_1]

1.5. Design flow for inventing combinational circuits using VHDL [L1.4]

1.5.1. Specifications, design concept map

1.5.1.1. Symbol or entity

1.5.1.2. Truth table

1.5.1.3. Example timing diagram. The *Min_Pulse* concept

1.5.2. CSD design plans

1.5.2.1. **Plan A**: structural/equations single-file [Lab2] (P1 design section)

1.5.2.1.1. Canonical equations: sum of minterms or product of maxterms

1.5.2.1.2. Minimised equations: SoP or PoS

- Espresso heuristic algorithm: minilog.exe

- Truth table translated to Minilog text input format (.tbl)

- Minilog minimisation results: SoP or PoS logic equations and equation converter

- Don't-care inputs ("x" or "-")

- Incomplete functions: don't cares outputs [L2.5]

- Logic Friday and Karnaugh maps (not covered)

1.5.2.1.3. Only-NOR gates [L1.5]

1.5.2.1.4. Only-NAND gates [L1.5]

1.5.2.1.5. Any kind of nonstandard equation

1.5.2.2. **Plan B**: behavioural/algorithm single-file description [Lab2]

1.5.2.2.1. Direct translation of the truth table

1.5.2.2.2. Interpreting the truth table as a flowchart

1.5.2.2.3. Solving combinational circuits using C language and microcontrollers (covered in Chapter 3)

1.5.2.3. Plan C1: hierarchical designs (in CSD is used only in the FSM architecture in Chapter 2 [L6.1])

1.5.2.4. **Plan C2**: hierarchical design multiple-file using components and signals [Lab3]

1.5.2.4.1. Circuit expansion using components of the same kind

1.5.2.4.2. The method of decoders (MoD) [L3.3]

1.5.2.4.3. The method of multiplexers (MoM) [L3.3] [Lab3]

1.5.2.4.4. The method of ROM memory cells or RAM lookup tables (covered in Chapter 2)

1.5.3. Development [Lab2]

1.5.3.1. VHDL translation. Project location.

1.5.3.2. Project synthesis for a target PLD.

1.5.3.3. Target chip resource usage (logic elements)

1.5.3.4. RTL view schematic

1.5.3.5. Technology schematic

1.5.4. Test and verification (ideal/functional) [Lab2]

1.5.4.1. Test-bench fixture schematic and test-bench VHDL file

1.5.4.2. Stimulus process

1.5.4.3. Functional simulation and wave results discussion

1.5.5. Test and verification [L4.3] (target chip / technology) Lab4]

1.5.5.1. Gate-level simulation: propagation delay measurement

1.5.5.2. Timing analyser spreadsheet: worst-case scenario, longest propagation delay (t_P)

1.5.5.3. Calculating circuit's maximum speed (f_{MAX}). Millions of operations per second (Mops)

1.5.6. Prototyping and laboratory measurements

1.5.6.1. CPLD/FPGA laboratory training boards

1.5.6.1.1. Basic boards: DE10-Lite

1.5.6.1.2. Advanced boards

1.5.6.2. Laboratory instrumentation

1.5.6.2.1. Logic analyser

1.5.6.2.2. Pulse generator

1.5.6.2.3. Compact instrumentation

1.5.7. Project report and presentation [2] [L1.1]

1.5.7.1. Handwritten sketches (intended for all CSD projects)

1.5.7.2. English [3]

1.5.7.3. Presentation slides [2]

1.5.7.4. Correction grid and self-assessment (or guidelines/rubric on how to generate high quality reports)

1.6. Standard logic gates chips and circuits [L1.6]

1.6.1. Logic families

1.6.1.1. TTL, LS-TTL, HC, HCT, etc. (internal gate design not covered)

1.6.1.2. CMOS. How do NMOS/PMOS transistors work as ideal electronic switches?

1.6.2. Standard chip references for classic logic families

1.6.3. Electrical characteristics of logic gates

1.6.3.1. Voltage levels. What range is interpreted as '0' and as '1'?

1.6.3.2. Noise margin high (NMH), noise margin low (NML), logic family compatibility

1.6.3.3. Input-output transfer characteristic $V_O = f(V_I)$

1.6.3.4. Current and power consumption [L4.3]

1.6.3.5. Digital circuit propagation time, worst-case scenario: longest propagation delay [Lab4] [L4.3]

1.6.3.6. How fast is a digital circuit? [L4.3]

1.6.4. Input push-buttons and switches

1.6.4.1. Active-high button circuit

1.6.4.2. Active-low button circuit

1.6.5. Driving LED [L2.4]

1.6.5.1. LED biasing characteristics

1.6.5.2. Active-low and active-high outputs

1.6.5.3. Limiting resistor for worst-case scenario

1.6.6. Driving 7-segment displays [L2.4]

1.6.6.1. Common anode digits

1.6.6.2. Common cathode digits

1.6.6.3. Multiplexed operation using electronic transistor switches

1.7. Standard combinational logic circuits [L2.1]

1.7.1. Concept map

1.7.1.1. Truth table, symbol, timing diagram

1.7.1.2. Chip expansion, enable input

1.7.1.3. Examples of commercial chips

1.7.1.4. VHDL design plans

1.7.2. Multiplexer (*MUX_2^*n) or data selector [L2.1]

1.7.2.1. Description (specifications)

1.7.2.2. Design examples: {*MUX_8*} plan A, {*MUX_8*} plan B [Lab2], {*Dual_MUX_4*} plan A, {*Dual_MUX_4*} plan B

1.7.2.3. MUX expansion circuits: {*MUX_8*}, {*Dual_MUX_4*} plan C2; (optional) *MUX_8*} plan C1

1.7.2.4. Commercial chips

1.7.3. De-multiplexer (DeMUX_2^*n*) or data distributor [L2.2]

1.7.3.1. Description (specifications)

1.7.3.2. Design examples {*DeMUX_4*} plan A, {*DeMUX_8*} plan B

1.7.3.3. DeMUX expansion circuits: {*DeMUX_8*} plan C2

1.7.3.4. Commercial chips

1.7.3.5. Demonstration prototype *MUX-DeMUX*

1.7.4. Binary decoder [L2.3]

1.7.4.1. Description (specifications)

1.7.4.2. Design examples*:* {*Dec_3_8*} plan A, {*Dec_3_8*} plan B, *Dec_4_16* plan B,

1.7.4.3. Decoder expansion circuits: {*Dec_5_32*} plan C2)

1.7.4.4. Commercial chips

1.7.5. Binary encoders (priority high) [L2.3]

1.7.5.1. Description (specifications)

1.7.5.2. Design examples: *Enc_8_3*, {*Enc_10_4*} plan A, {*Enc_10_4*} plan B

1.7.5.3. Encoder expansion circuits: {*Enc_4_10*} plan C2

1.7.5.4. Commercial chips

1.7.6. Hexadecimal to 7-segment decoder [L2.4]

1.7.6.1. Basic decoder {*Hex_7seg_decoder*} plan A, {*Hex_7seg_decoder*} plan B

1.7.6.2. Control signals: blanking, ripple blanking, lamp test

1.7.6.3. Commercial chips

1.7.6.4. Design examples

1.8. Binary codes and code converters [L3.1]

1.8.1. Radix-2 (base 2) and radix-16 (hexadecimal) number systems

1.8.2. Gray code

1.8.3. BCD (binary-coded decimal)

1.8.4. One-hot and one-cold

1.8.5. Johnson

1.8.6. Keyboard symbols: ASCII, etc.

1.8.7. MORSE code

1.8.8. Code converter circuits

1.8.8.1. Bin_BCD_converter_6bit

1.8.8.2. Gray_Bin_converter_4bit

1.9. Binary (radix-2) arithmetic circuits

1.9.1. Addition [L3.1]

1.9.1.1. 1-bit full adder block: {*Adder_1bit* } plan A [L3.1], {*Adder_1bit*} plan B, {*Adder_1bit*} plan C2 MoM [Lab 3] and {*Adder_1bit*} plan C2 MoD

1.9.1.2. *n*-bit adders [L3.2]

1.9.1.2.1. Ripple-carry adder:{*Adder_4bit*}, {*Adder_8bit*} [Lab 3], {*Adder_16bit*} [Lab 4],

1.9.1.2.2. Carry-lookahead adder: {*Adder_4bit*}, {*Adder_16bit*} [Lab 4]

1.9.1.3. Commercial standard adder chips in classic technologies

1.9.2. Comparator

1.9.2.1. Expandable 1-bit comparator: {*Comp_1bit*} plan A [L3.1], {*Comp_1bit*} plan B, {*Comp_1bit*} using plan C2 MoM

1.9.2.2. *n*-bit comparators [L3.2]

1.9.2.2.1 {*Comp_4bit*} plan A, {*Comp_4bit*} and {*Comp_10bit*} plan C2

1.9.2.2.2 24-bit tree comparator (*Comp_24bit*)

1.9.2.3. Commercial standard comparator chips in classic technologies

1.9.3. Multiplier

1.9.3.1. 1-bit multiplier cell, {*Mult_1bit*} [L3.1]

1.9.3.2. Example of 8-bit multiplier {*Mult_8bit*} (optional {*Mult_8bit*} using plan B)

1.9.4. Counting the number of ones in a vector input (for example counting cars in parking slots) [L3.2]

1.9.4.1. Counting the number of ones in a 4-bit vector {*Ones_counter_4bit*}

1.9.4.2. *Counting the number of ones in a 8-bit vector* {*Ones_counter_8bit*} (P3)

1.9.5. Parity generators and checkers {*Parity_check_9bit*}

1.9.6. Radix-2 subtraction

1.9.6.1. 1-bit true subtractor cell (minuend, subtrahend, difference and borrows).

1.9.6.2. *n*-bit ripple subtractor {*Subtractor_8bit*}

1.10. Integer arithmetic [L4.1]

1.10.1. Integer number two's complement (2C) representation

1.10.2. Adder-subtractor (P4) {*Int_add_subt_8bit*}

1.10.3. Multiplier {*Int_mult_8bit*}

1.10.4. Comparator {*Int_Comp_8bit*}

1.11. Arithmetic Logic Unit {*ALU_4bit*} [L4.2]

1.11.1. Conceptual architecture

1.11.2. Bitwise logic operations (AND, OR, etc.)

1.11.3. Commercial standard arithmetic chips in classic technologies

1.12. Hardware description languages

1.12.1. VHDL (intended for all Ch1 and Ch2 CSD projects)

1.12.2. Verilog (not covered)

1.13. Target chips: programmable logic devices (PLD) [L4.3]

1.13.1. sPLD (ispGAL22V10) and programmable arrays to implement logic functions

1.13.2. CPLD: complex programmable logic device

1.13.3. FPGA: field programmable gate array. DE10-Lite, MAX II Micro Kit, DE2-115 boards

1.14. Electronic design automation (EDA) tools for synthesis

1.14.1. Lattice Semiconductor ispLEVER Classic / Diamond

1.14.2. Xilinx Vivado / ISE

1.14.3. Intel Quartus Prime (intended for all Ch1 and Ch2 CSD projects)

1.15. EDA tools for testing and verification

1.15.1. ALDEC Active HDL Lattice Edition

1.15.2. Xilinx ISim

1.15.3. Mentor Graphics ModelSim Intel FPGA Starter Edition (intended for all Ch1 and Ch2 CSD projects)

1.16. Additional materials and projects

1.16.1. Sample questionnaires and controls

1.16.2. Assignment collection: analysis and design.

1.16.3. Exams

Chapter II. **Sequential systems**

2.1. The concept of memory in digital circuits. Current state and next state [L5.1]

2.2. Latch circuits: asynchronous 1-bit memory cells

2.2.1 Latch RS (*RS_latch*).

2.2.1.1. Function table, state diagram, timing diagram

2.2.1. 2. How to design *RS_latch* using gates (structural plan A)

2.2.1.3. *RS_latch* used as push-button debouncing circuit

2.2.1.4. Commercial circuit

2.2.2. Transparent D latch (*D_latch*)

2.2.2.1. Function table, state diagram, timing diagram

2.2.2.2. Commercial circuit

2.3. Flip-flop. Synchronous 1-bit memory cell [L5.2]

2.3.1. The concept of CLK signal. Synchronicity

2.3.2. CLK distribution

2.3.3. CLK circuits

2.3.2.1. RC circuit

2.3.2.2. Integrated circuit 555 oscillator

2.3.2.3. Quartz crystal oscillator

2.3.4. The concept of clear direct (CD) or asynchronous reset [L5.2]

2.3.5. Power-ON reset and system initialisation[L5.2]

2.3.6. Standard flip-flops [L5.2]

2.3.6.1. RS flip-flop (*RS_FF*)

2.3.6.1.1. Deducing *RS_FF* from *RS_latch*, CLK's rising-edge detector

2.3.6.1.2. Function table, state diagram, timing diagram

2.3.6.1.3. Commercial chip

2.3.6.2. Data flip-flop (*D_FF*) [L5.3]

2.3.6.2.1. Deducing *D_FF* from *RS_FF*

2.3.6.2.2. Function table, state diagram, timing diagram

2.3.6.2.3. Commercial chip

2.3.6.3. JK flip-flop (*JK_FF*)

2.3.6.3.1. Function table, state diagram, timing diagram

2.3.6.3.2. Commercial chip

2.3.6.4. Toggle flip-flop (*T_FF*)

2.3.6.4.1. Function table, state diagram, timing diagram

2.3.6.4.2. Frequency divider by two

2.3.7. Timing parameters: Time CLK to output (t_{CO}). Set-up time (t_{SU}), hold time (t_H). Maximum CLK frequency (f_{CLKmax}).

2.3.8. Analysis of asynchronous and synchronous circuits based on flip-flops and logic (P5)
{*Circuit_async*} {*Circuit_Async4*} {*Circuit_Sync1*}

2.3.8.1. Method I: handwritten pen-and-paper analysis and discussion. Has the circuit any application? How many states does the circuit have?

2.3.8.2. Method II: using Proteus

2.3.8.3. Method III: using VHDL synthesis and simulation tools (plan C2 circuit)

2.4. Digital memory chips [L5.4]

2.4.1. Symbol, address and data

2.4.2. General architecture

2.4.3. Tri-state gates

2.4.4. How to make a wire or a cable bi-directional? Bus concept

2.4.5. ROM: read-only memory

2.4.5.1.VHDL circuit for describing ROM memory devices.

2.4.5.2.Method of ROM for implementing logic functions in VHDL, look-up tables (LUT)

2.4.6. (optional) RAM: random access memory

2.4.6.1. VHDL circuit (RAM_2^nxm, for example: *RAM_16x5*) and the idea of intellectual property (IP) specific circuits for a given target technology

2.4.6.2. Tri-state buffer in VHDL

2.5. Finite State Machine (FSM): Concept and design procedure [L6.1] [Lab 6]

2.5.1.Specifications

2.5.1.1. Function table, symbol, state diagram

2.5.1.2. CLK and CD circuits

2.5.1.3. Example of timing diagram

2.5.1.4. FSM systematic design procedure

2.5.2.Planning

2.5.2.1. Architecture: canonical, synchronous, plan C1: hierarchical, structural in a single VHDL file

2.5.2.2. State diagram, and FSM adaptation

2.5.2.3. State register: r-bit memory (*D_FF)*, state encoding (binary sequential, Gray, one-hot, etc.)

2.5.2.4. Output logic (CC2): truth table, behavioural interpretation: flowchart

2.5.2.5. Next state logic (CC1): truth table, behavioural interpretation: flowchart

2.5.3. Developing

2.5.3.1. VHDL translation, state enumeration, project location.

2.5.3.2. Synthesis project for a target chip. FSM encoding options

2.5.3.3. Target chip resource usage (*D_FF* registers)

2.5.3.4. RTL and technology view discussion

2.5.3.5. FSM state diagram discussion

2.5.4. Testing (functional)

2.5.4.1. Test-bench fixture schematic and test-bench VHDL file

2.5.4.2. CLK and other signals stimulus processes

2.5.4.3. Functional simulation and wave results discussion

2.5.4.3. Internal signals representation (*current_state*, *next_state*)

2.5.5. Testing (technology)

2.5.5.1. Gate-level (timing) simulation and results discussion

2.5.5.2. Propagation time CLK to output measurements (t_{CO})

2.5.5.3. Timing analyser spreadsheet and measurement of the maximum frequency of operation

2.5.6. Prototyping using FPGA training boards

2.6. Examples of FSM in VHDL: single-file (plan C1) projects [L6.2]

2.6.1. Designing flip-flops as FSM (two-state machines)

2.6.1.1. RS flip-flop (*RS_FF is JK_FF* never using J = 1; K = 1}

2.6.1.2. D-type (data) flip-flop {*D_FF*}

2.6.1.3. JK flip-flop {*JK_FF*}

2.6.1.4. T-type (toggle) flip-flop {*T_FF*}

2.6.2. Classroom luminaries control [Lab 6]

2.6.3. 16-key matrix keypad encoder (P6) {*Matrix_encoder_16key*}[L6.2]

2.6.4. Push-button de-bouncing filter {*Debouncing_filter*}

2.6.5. Traffic light controller {*Traffic_light_controller*}

2.6.6. Stepper motor controller

2.6.7. LED sequencer

2.6.8. Bicycle torch {*LED bicycle torch*}

2.6.9. Advanced matrix keypad (registered output with handshake signals)

2.7. Standard synchronous sequential systems as FSM [L7.1]

2.7.1. Concept map

2.7.2. Counters

2.7.2.1. Symbol, function table, modulo, timing diagram, state diagram, commercial chips

2.7.2.2. Control signals: count enable (CE), up and down (UD_L) or reversibility

2.7.2.3. Control signals: terminal count (TC) pulse

2.7.2.4. Output code:

2.7.2.4.1. Radix-2 (binary sequential)

2.7.2.4.2. BCD

2.7.2.4.3. One-hot or one-cold

2.7.2.4.4. Gray, Johnson, etc.

2.7.3. Design **plan X**: designing counters as FSM for small number of states and any output code

2.7.3.1. Synchronous 1-digit BCD counter {*Counter_BCD_1digit*} [L7.1]

2.7.3.2. *Counter_mod12* {*Counter_mod12*} [Lab 6] [Lab 7]

2.7.3.3. *Counter_Gray_3bit*

2.7.3.4. *Counter_Johnson_5bit*

2.7.4. Design **plan Y**: Designing large synchronous counters and registers using the VHDL arithmetic library and STD_LOGIC_VECTOR, single-file VHDL project. [L7.1]

2.7.4.1. Example: Counter modulo 12 {*Counter_mod12*} [Lab 7]

2.7.4.2. Example: *Counter_BCD_1digit* {*Counter_BCD_1digit*}

2.7.4.3. Example: The versatile counter modulo 16 radix-2 {***Counter_mod16***}. Additional control signal: parallel load (LD) or pre-setting output value

2.7.4.4. Example: *Counter_mod1M* (one million states)

2.7.5. Design **plan C2**: count truncation and count expansion [L7.3]. Concepts and chaining signals (terminal count - count enable). Hierarchical structures, standard components {*Counter_mod16*} and logic, VHDL multiple-file project

2.7.5.1. Counter modulo 12 {*Counter_mod12*}[Lab 7]

2.7.5.2. Counter modulo 10 {*Counter_BCD_1digit*} [L7.3]

2.7.5.3. 2-digit (modulo 100) BCD counter {*Counter_BCD_2digit*}

2.7.5.4. *Minutes_counter* {*Counter_BCD_mod60*}

2.7.5.5. Counter BCD module 24 {*Hour_counter*} (P7)

2.7.6. *n*-bit data register (Data_reg_nbit) [L7.2]

2.7.6.1. Symbol, function table, parallel load, timing diagram, state diagram, commercial chips

2.7.6.2. Plan Y {*Data_reg_4bit*}

2.7.6.3. Plan C2 using components (*Counter_mod16* or *Data_reg_4bit*)

2.7.7. *n*-bit shift register (Shift_reg_nbit) [L7.2]

2.7.7.1. Symbol, function table, parallel load, timing diagram, state diagram, commercial chips

2.7.7.2. Plan Y {*Shift_reg_4bit*}

2.7.7.3. Plan C2 using components (*Shift_reg_4bit*)

2.7.8. Barrel *p*_shift register *n*-bit. For example *Barrel_3_shift_reg_21* [D2.13]

2.7.8.1. Symbol, function table, parallel load, timing diagram

2.7.8.2. Circuit design using plan Y

2.8. Dedicated processors or subsystems [L8.1]

2.8.1. Architecture of an advanced digital system

2.8.2. Datapath (operational) unit, data input/output, status signals and flags

2.8.3. Control unit (FSM). Control signals, external inputs and outputs

2.8.4. CLK generator circuit [L8.2]

2.8.4.1. Frequency divider

2.8.4.2. Pulsed to square waveform converter using *T_FF*

2.8.4.3. Adaptable generic structure

2.8.5. Examples and applications of dedicated processors

2.8.5.1. MM:SS timer (P8) {*Timer_MMSS*}

2.8.5.1.1. The basic idea of a timer circuit

2.8.5.1.2. RC timer and re-triggerable functionality

2.8.5.1.3. Integrated circuit 555 timer

2.8.5.2. Serial multiplier {*Mult_4bit*}

2.8.5.3. Programmable timer

2.8.5.4. Serial {*Adder_4bit*}

2.8.5.5. Serial asynchronous receiver and transmitter subsystem {*UART_module*}

2.8.5.6. Pulse generator.

2.9. Additional materials and projects

2.9.1. Sample questionnaires and controls

2.9.2. Assignment collection: analysis and design.

2.9.3. Exam 2

Chapter III. **Microcontrollers**

3.1. Microcomputer architecture and basics [L9.1]

3.1.1. Microprocessor (µP)

3.1.2. Microcontroller (µC), RAM, ROM, I/O

3.1.3. Harvard and Von Neumann architectures

3.1.4. **PIC18F46K22** chip architecture. 8-bit µC from Microchip. The architecture of the PIC18F family (intended for all Chapter III projects)

3.1.5. (Optional) Microchip: PIC16F877A, ATmega8535, ATmega328P (Arduino UNO µC)

3.1.6. Low-level assembly and high-level C languages

3.2. (Optional) Other µC families and chips

3.2.1. Texas Instruments

3.2.2. Renesas

3.2.3. NXP

3.2.4. STMicroelectronics

3.2.5. Infineon (Cypress)

3.3. EDA tools for developing and testing

3.3.1. Microchip integrated development environment (IDE) MPLAB X (intended for all Ch3 CSD projects)

3.3.2. C compiler XC8 (intended for all Ch3 CSD projects)

3.3.3. Proteus VSM as the virtual laboratory simulation tool

3.4. Digital I/O. Combinational circuits in C

3.4.1. Specifications

3.4.1.1. Truth table, symbol, timing diagram

3.4.1.2. Symbol

3.4.1.3. Timing diagram

3.4.2. Planning [L9.2] [Lab 9]

3.4.2.1. Hardware schematic

3.4.2.1.1. Connecting switches and buttons

3.4.2.1.2. Oscillator (OSC) and reset (MCLR_L) circuits

3.4.2.1.3. Interfacing LED [L2.4]

3.4.2.1.4. Tri-state logic gates and wire/pin bi-directionality.

3.4.2.2. Software organisation: setup, main loop

3.4.2.2.1. Hardware-software diagram

3.4.2.2.2. RAM variables

3.4.2.2.3. *init_system()*: I/O port pin configuration data direction register TRIS.

3.4.2.2.4. *read_inputs()* [L9.3]

3.4.2.2.5. *truth_table(),* behavioural description, algorithm

3.4.2.2.6. *write_outputs()* [L9.4]

3.4.3. Development and testing

3.4.3.1. Proteus schematic capture: *pdsprj* file

3.4.3.2. MPLABX project for PIC18F46K22: C source file

3.4.3.3. Project compilation and chip configuration files: *hex*, *cof*

3.4.3.4. Proteus simulation and testing. Step by step debugging, watch variables window

3.4.3.5. Software/assembly execution time measurements

3.4.4. Example projects

3.4.4.1. Dual 4-channel multiplexer {*Dual_MUX_4*} [Lab9]

3.4.4.2. 1-digit BCD adder {*Adder_BCD_1digit*} (P9)

3.4.5. Prototyping and laboratory experimentation CSD_PICstick board

3.4.5.1. Chip programmer and program download MPLAB SNAP

3.4.5.2. In-circuit debugging: MPLAB SNAP

3.4.5.3. Measurements and characterisation (compact instrument VB8012)

3.4.6. (optional) Other prototyping boards (Explorer8, PICDEM2 plus, etc.)

3.5. FSM implementation in C language

3.5.1. Specifications [L10.1]

3.5.1.1. FSM concept adaptation to µC

3.5.1.2. CLK interface using interrupts

3.5.2. Planning (hardware and software planning as in 3.4.2)

3.5.2.1. Hardware-software diagram. RAM variables

3.5.2.2. Interrupts: interrupt service routine *ISR().* External event detection INT (CLK and push-button interface)

3.5.2.3. *output_logic()* and *write_outputs()*. Bitwise logic operations AND, OR, XOR, NOT, shift in C

3.5.2.4. *state_logic()* and *read_inputs()*. Truth table, equivalent behavioural flowcharts

3.5.3. Development and testing (as in 3.4.3)

3.5.4. Examples

3.5.4.1. 4-bit asynchronous serial transmitter [L10.2] (P10) {*Serial_transmitter*} (design phase#1)

3.5.4.2. 12-states Johnson sequencer {*Johnson_sequencer_mod12*}

3.5.4.3. 1-digit BCD counter (plan X) [Lab 10] {*Counter_BCD_1digit*}

3.5.4.4. Modulo 1572 binary radix-2 counter (plan Y) [Lab 10] {*Counter_mod1572*}

3.5.4.5. Traffic light controller

3.5.4.6. Stepper motor driver

3.6. Peripherals: LCD [L11]

3.6.1. LCD technology and controllers

3.6.2. LCD C libraries

3.6.3. Examples

3.6.3.1- 1-digit BCD adder with LCD display {*Adder_BCD_1digit_LCD*}

3.6.3.2. Serial transmitter with LCD display (P11) {*Serial_transmitter_LCD*} (design phase#2)

3.7. Dedicated processors in C (datapath, control unit) [L12.1]

3.7.1. Hardware-software diagram

3.7.2. Examples: Timer

3.7.2.1. Fixed-time timer with external CLK time-base [Lab 11] {*Timer*} (design phase#1)

3.7.2.2. Fixed-time timer with LCD [Lab 11] {*Timer_LCD*} (design phase#2)

3.7.3. Examples: event counter

3.7.3.1. Tachometer, speed meter, odometer

3.8. Peripherals: Timer0 (TMR0) [L12.2] and Timer2 (TMR2) [L12.2]

3.8.1. TMR0 hardware circuit and configuration registers

3.8.2. TMR2 hardware circuit and configuration registers

3.8.3. Examples: Timer (continuation)

3.8.3.1. Serial transmitter with TMR0 (P12) {*s_trans_LCD_TMR0*} (design phase#3)

3.8.3.2. Serial transmitter with TMR2 {*s_trans_LCD_TMR2*} (design phase#4)

3.8.3.3. Fixed-time timer with LCD and time-base TMR0 [Lab 11] {*Timer_LCD_TMR0*} (design phase#3)

3.8.3.4. Fixed-time timer with LCD and time-base TMR2 {*Timer_LCD_TMR2*} (design phase#4)

3.8.4. Other examples [L12.3]

3,8.4.1. 6-bit Johnson sequencer using TMR0 {*Johnson_sequencer_mod12_LCD_TMR*} (design phase#3)

3.8.4.2. Duty-cycle modulator: LED dimmer

3.9. Optional extended content. Other peripherals

3.9.1. Timer1 (TMR1)

3.9.2. A/D, EEPROM, USART, I2C, PWM, etc.

3.9.3. Port B interrupt on change (adapting matrix keypads)

3.10. Optional extended content. Other microcontrollers and microcomputers

3.10.1. PIC16F877, ATmega8535

3.10.2. Arduino. ATmega328P chip

3.10.3. Raspberry Pi.

3.11. Project examples, final bachelor thesis, research papers, books, etc. [L12.3]

3.12. Additional materials and projects

3.12.1. Sample questionnaires and controls

3.12.2. Assignment collection: analysis and design.

3.12.3. Exam 2
