Aalto University
School of Electrical
Engineering

ELEC-E3540 Digital Microelectronics II Introduction

Vishnu Unnikrishnan
vishnu.unnikrishnan@aalto.fi

04.03.2019

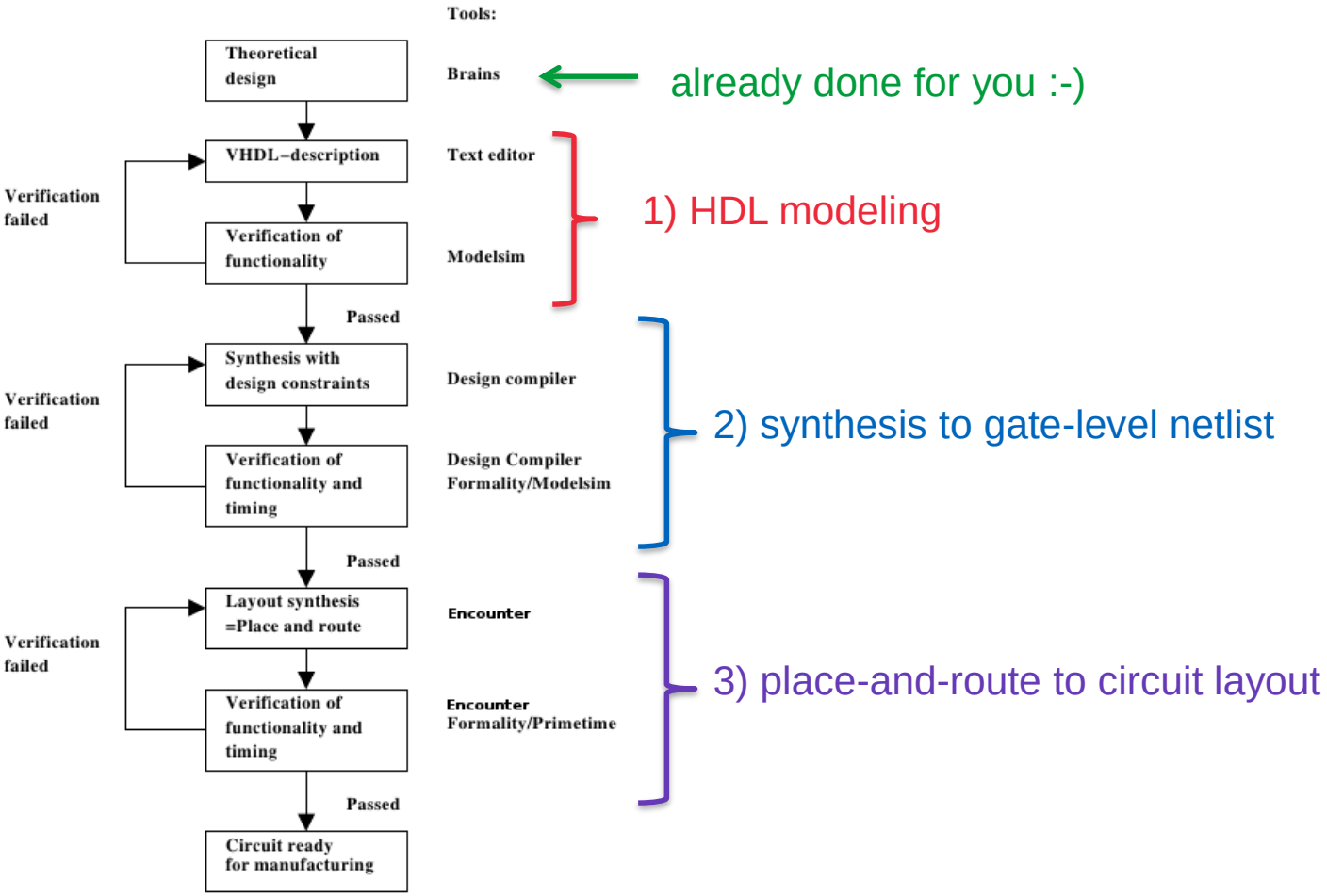
Course staff

- Main teacher: **Vishnu Unnikrishnan**
  - vishnu.unnikrishnan@aalto.fi, room 2189
- Assistant teachers:
  - **Ilia Kempi**
    - ilia.kempi@aalto.fi, room 2190
  - **Cheung Tze (Dicky)**
    - tze.cheung@aalto.fi, room 2186
- ELE department is located in TUAS building, 2nd floor

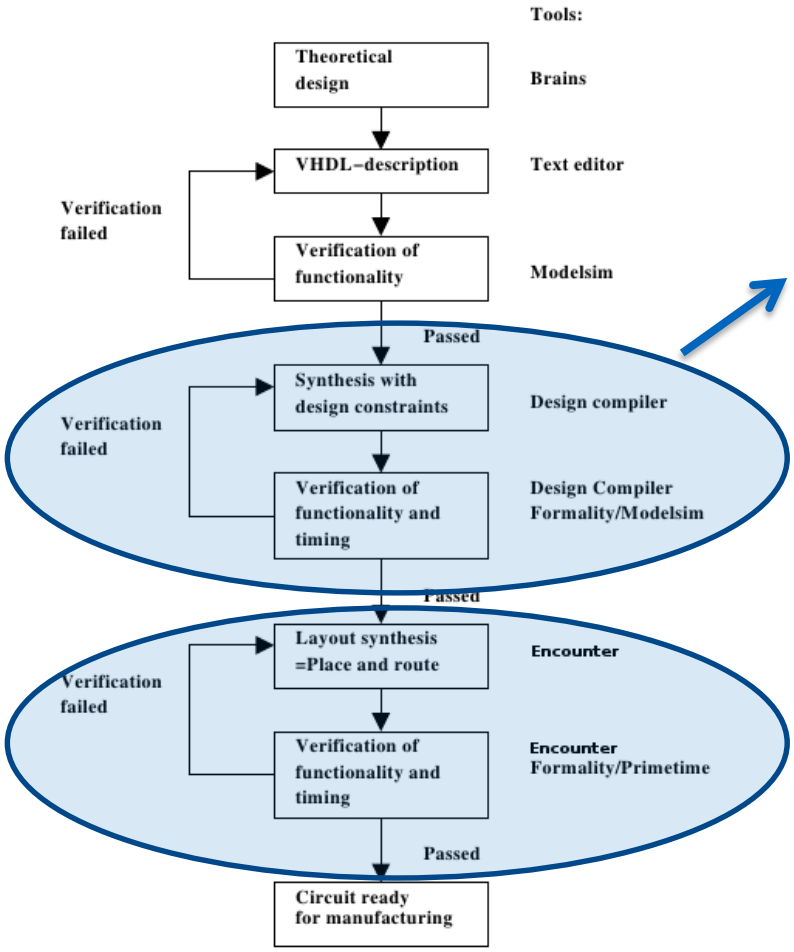
Course objective

- The main objective is to **learn to implement digital circuits on higher abstraction level than the transistor**
  - Modeling of complex functions/algorithms or entire systems with **hardware description language (HDL)**
  - Translation into gate-level netlist and circuit layout with automated **synthesis** and **place-and-route** software tools

Standard digital design flow



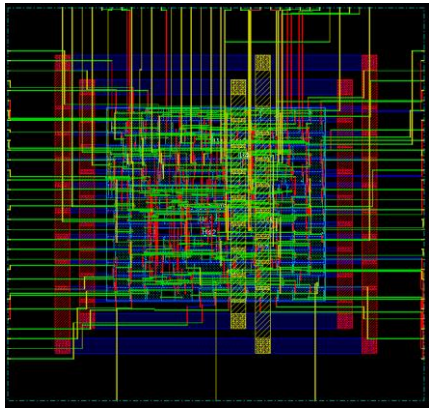
Standard digital design flow



VERILOG NETLIST

```
HS65_LH_DFPRQX9 gray_counter_m_generate_MSB_count_reg ( .D(
      gray_counter_m_generate_MSB_N0), .CP(osc_inm), .RN(n445), .Q(
      gray_counter_m_gray_count_16_) );
HS65_LH_MUX21X4 U247 ( .D0(generate_piso_onehotv_1_), .D1(
      generate_piso_onehotv_32_), .S0(n452), .Z(n216) );
HS65_LH_IVX7 U262 ( .A(gray_counter_p_gray_count_0_), .Z(
      gray_counter_p_generate_parity_N0) );
HS65_LH_IVX7 U263 ( .A(gray_counter_m_gray_count_0_), .Z(
      gray_counter_m_generate_parity_N0) );
HS65_LH_NAND2X4 U265 ( .A(n461), .B(n456), .Z(n439) );
```

LAYOUT



Course structure

- This is a self-learning course: there will be only one lecture besides this one
- Of course, help will be provided upon request

- Material:
  - Course book: Peter J. Ashenden, “The designer’s guide to VHDL”, 3rd edition
  - Slides, tutorials, instructions, etc. available in MyCourses

Course structure

- **Six mandatory exercises**
  - will help you to learn the basics of VHDL coding
  - support the design assignment
  - grade = pass/fail

- **Design assignment**: implementation of PIC16F84A microcontroller
  - learn the complete design flow of a complex digital system (VHDL + synthesis + place-and-route)
  - final course grade = design assignment grade

Six mandatory exercises

- Topics given in MyCourses
- For each exercise, a “pre-exercise task” will help you to get prepared
  - just for your own use, no need to return it
- Exercises are *completed* in computer class during the two-hour exercise sessions
  - teacher and/or assistants will be there to provide help
- How to “return” an exercise: **show the code, testbench and working simulation to the teacher or assistant**
  - he will mark the exercise as completed, IF it is correct
  - no need to return the code “physically”

Design assignment

- Perform the whole digital IC implementation flow of part of the PIC16F84A microcontroller
  - VHDL modeling + synthesis + place-and-route
- PIC chosen because of its simple structure, and because assembler compiler is available
- Nevertheless, learning its functionality is not very straightforward, **so start studying immediately!**
  - datasheet available in MyCourses

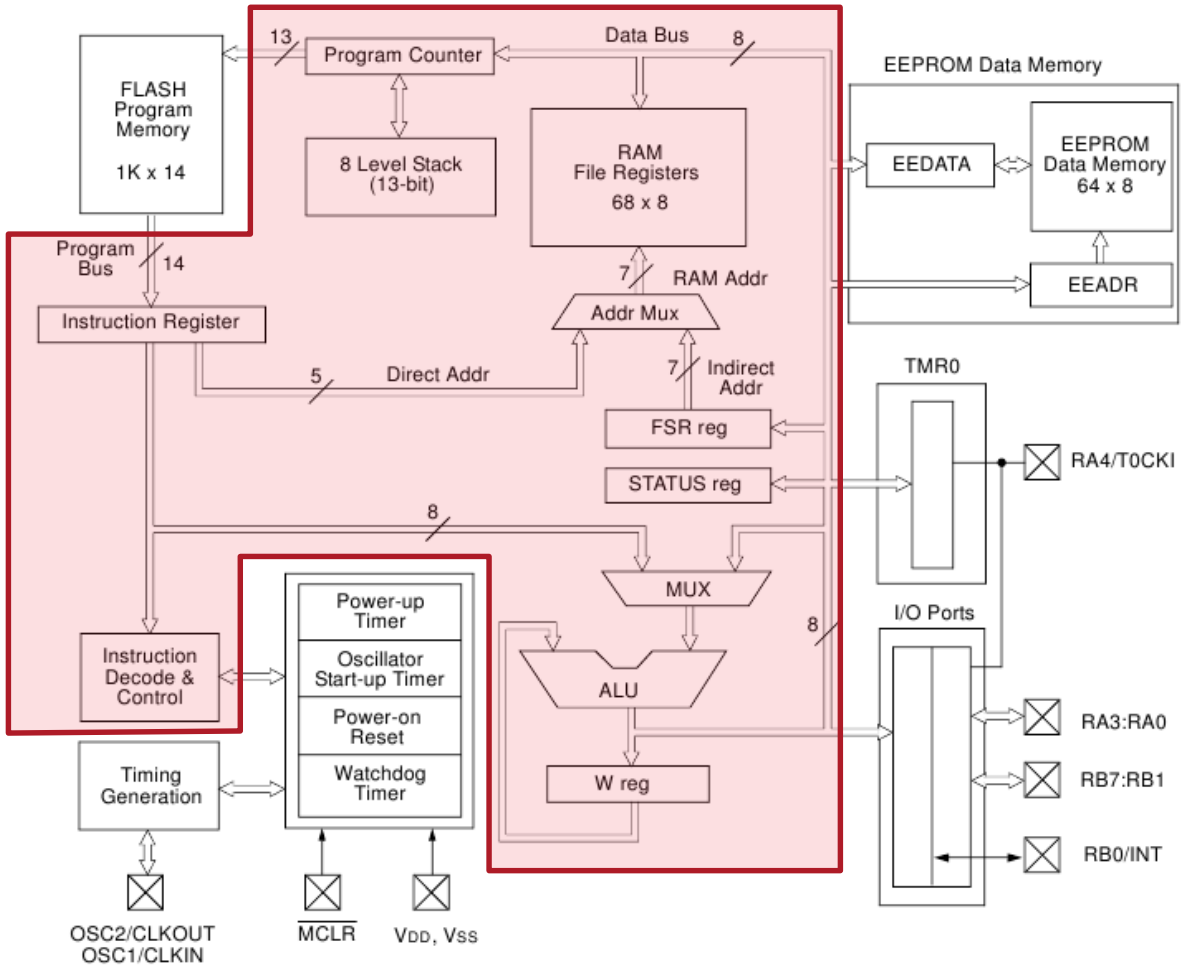
Design assignment

- Course will be graded based on **study diary** and documentation of the design
  - the study diary should document and describe the phases of the design flow, difficulties encountered and how they were solved

- Things to be graded:
  - Quality of the code, clear structure, commented, easy to read.
  - Gained understanding of the subject. This should be visible in your study diary.
  - 100% functionality is not required to pass, but you should show that you have tried your best and learned something.

The PIC16F84A

part to be implemented



The PIC16F84A instruction set

instructions NOT to be implemented

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
~~DECFSZ~~	~~f, d~~	~~Decrement f, Skip if 0~~	1 (2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
~~INCFSZ~~	~~f, d~~	~~Increment f, Skip if 0~~	1 (2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
~~BTFSC~~	~~f, b~~	~~Bit Test f, Skip if Clear~~	1 (2)	01 10bb bfff ffff		3
~~BTFSS~~	~~f, b~~	~~Bit Test f, Skip if Set~~	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
~~CALL~~	~~k~~	~~Call subroutine~~	2	10 0kkk kkkk kkkk		
~~CLRWDT~~	~~-~~	~~Clear Watchdog Timer~~	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
~~GOTO~~	~~k~~	~~Go to address~~	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
~~RETFIE~~	~~-~~	~~Return from interrupt~~	2	00 0000 0000 1001		
~~RETLW~~	~~k~~	~~Return with literal in W~~	2	11 01xx kkkk kkkk		
~~RETURN~~	~~-~~	~~Return from Subroutine~~	2	00 0000 0000 1000		
~~SLEEP~~	~~-~~	~~Go into standby mode~~	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Stages of command execution

1) IFetch: Fetch instruction from program memory and decode it.
2) Mread: Read operand from memory, if required.
3) Execute: Perform operation.
4) Mwrite: Increment PC, write data to memory or register.

Execution cycle of an instruction

IFtch	Mread	Exec	Mwrite

- Every instruction can be divided in “stages”. Maximum number is four, since PIC datasheet describes execution in max four clock cycles.
- Only Mwrite is strictly synchronous operation, but in order to make things easier, advice is to implement the steps with a synchronous state machine.
- Every command does not require every step.

Software tools

- All required software tools are available on ELE department's computing machine (VSPACE)
  - connection through X2Go-client or SSH
  - computer account required

- Connection accessible only from Aalto network (e.g., computer classrooms)

Software tools

- For coding VHDL, feel free to use any text editor you like
  - gedit, kwrite, kate, emacs, gvim, ...
  - Modelsim's own text editor is also an option, even though it's not very good

- However, whatever text editor you choose, **please learn to use it efficiently!**
  - identation settings, (un)comment multiple lines of code, etc.
  - keyboard shortcuts for most used commands

Course schedule & rules

- Total **8 exercise sessions** are scheduled
  - rationale: 1 session/exercise + 2 extra
- Purpose of exercise sessions:
  - main time to **ask for help**
  - only time to **“return” completed exercises**
    - returning outside exercise sessions not allowed
- If questions outside exercise sessions are absolutely necessary, come to meet in person
  - do **not** send emails, unless the answer is as simple as yes/no

Course schedule & rules

- Exercises can be time-consuming, so exercise session times are **not sufficient**
  - you must work also independently between the sessions

- Exercises must be returned **in order**
  - not possible to e.g. return exercise 2 before 1
- There are **deadlines** for exercises 4, 5, 6
  - see schedule in MyCourses
  - each late returned exercise will cause a cumulative penalty of -1 in the final course grade!

How to pass

1. Complete all **six exercises** and get them accepted by the teacher or assistant

2. Complete the **design assignment** and submit it via MyCourses

- Firm deadline for everything: **31st May 2018**