University of Kent | School of Engineering and Digital Arts

EDA ENGINEERING ASSIGNMENT COVER SHEET

Title:	CAD 1 Introduction to VHDL design experiment (Group 41)
Module:	EL568 Digital Implementation
Work Setter:	Dr P Assimakopoulos
Assessment:	This assignment contributes 18.0% of the module.
Deadline:	Submission deadline: 23:59 pm Friday Week 22 Marked work return date: Friday Week E1 Submit: On-Line
Intended Learning Outcomes:	This assignment contributes to the assessment of the EL568 learning outcomes: *8.1 demonstrate the necessary skills to model digital components using VHDL;*
Feedback method:	Feedback will be via: Model solution on Moodle; Verbal comments in class; Email message. Further feedback will be available on request.

You are reminded of the University rules on Academic Integrity as stated in the student handbook. Examples of conduct regarded as a breach of these regulations include:

- *Plagiarism: reproducing in any work submitted for assessment or review (for example, examination answers, essays, project reports, dissertations or theses) any material derived from work authored by another, without clearly acknowledging the source*
- *Duplication of Material*
- *Conspiring with others to reproduce the work of others without proper acknowledgement, including knowingly permitting work to be copied by another student*
- *Falsification of data/evidence.*

Please note that IET accreditation requires that when coursework contributes 30% or more of a module then credit cannot be obtained if either your coursework or examination scores are more than 10% below the module pass mark. This is irrespective of your module overall mark.

%	**Marking Criteria**
>70	The work is very clearly presented and structured to show the method and analytical, or design process. Assumptions are stated and values are given to appropriate accuracy, significant figures, & correct units are used. High standard engineering practice is demonstrated (e.g. use of Nearest Preferred Component values). Comparisons, judgements and critical assessments are made where appropriate. Understanding beyond the provided teaching materials may be demonstrated.
60-69	Good presentation and structure with little or no ambiguity or omitted stages. Correct values are obtained, though not necessarily to appropriate significant figures, units are provided – though not always using appropriate engineering prefix (e.g. pF compared to F), required results are obtained, but no appropriate comment is made.
50-59	Acceptable presentation and method, though some stages may lack detail or contain errors. Results may be incorrect though method is broadly appropriate. Units may be missing or occasionally incorrect. Some sections may be missing.
40-49	A significant amount of the required work is submitted, though there may be errors and presentation not be clear in places. There may be errors in calculation but methodology is broadly appropriate.
0-39	Only a small amount of the required work is submitted, structure and presentation are very unclear and difficult to follow. Many errors in calculations, and little evidence of understanding.

Design and Implementation of a Direct Digital Synthesiser (DDS) Circuit

Introduction

The aim of this laboratory is to build and test, using Modelsim/ISE Simulator (ISim), a VHDL model of a Direct Digital Synthesiser (DDS) circuit which can them be implement on an FPGA board by using the Xilinx ISE 14.7 software tools to map the design onto the Xilinx Spartan-6 NEXYS-3 development board. The laboratory aims to test your understanding of digital design, VHDL and the tools and techniques necessary to map a VHDL model onto hardware.

Laboratory Outputs

The components of this laboratory to be assessed are:

- Documented VHDL models and testbench simulations of the first task's circuits and timing logic (Timing Generation, Debounce Circuit, Phase Accumulator).

[5 marks]

- Documented VHDL models and testbench simulations of the WAVE Gen and DAC2 interface circuits.

[4 marks]

- A documented VHDL model of the complete circuit (DDS, DAC2 Interface and control switches) i.e. including components and top level.

[4 marks]

- A Xilinx Place and Route Summary containing details of the logical resources that have been used, demonstrating that a valid .bit file has successfully been generated.

[2 marks]

- A brief report summarising what has been achieved.

[5 marks]

You should attempt the first task in Weeks 16 and 18 of the Laboratory and the remaining parts in Weeks 19 and 20. All files will need to be submitted to Moodle for final verification together with the report

Background

Direct Digital Synthesis (DDS) is a well-known and well used technique for generating arbitrary periodic waveforms. It is used as a building block in many applications in communications and instrumentation. The circuit to be designed in this experiment is a basic version. It is possible to enhance the quality and performance of this circuit using a number of well-known digital signal processing techniques. You may encounter some of these in other modules on your course. Figure 1 is a Top Level symbol of the circuit you will be building.

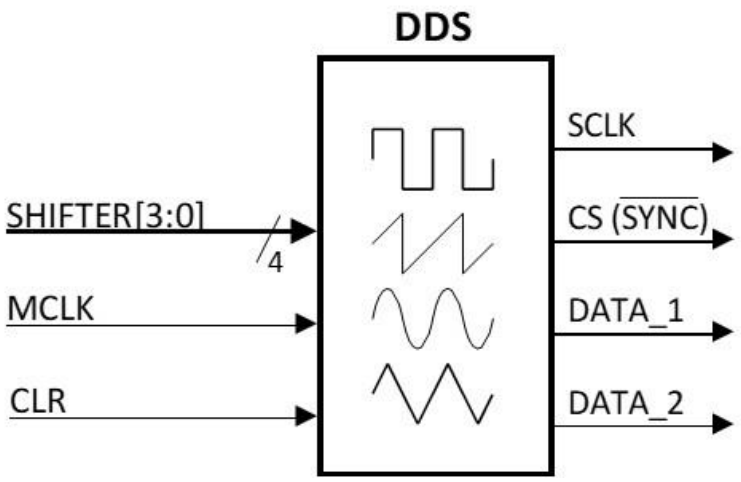


Figure 1. Top-Level Symbol of the DDS Circuit

A block diagram of this circuit to be designed is shown in Figure 2. The system will be built using a number of components that need to be designed and tested individually before connecting them together in a top level description. These are:

- The DDS Phase Accumulator. (Weeks 16 and 18)
- The Debounce and Timing Circuits (Weeks 16 and 18)
- The DA2 (DAC) converter circuit interface. (Weeks 19 and 20)
- The Waveform Generation Circuit. (Weeks 19 and 20)
- The completed Circuit (Weeks 19 and 20)

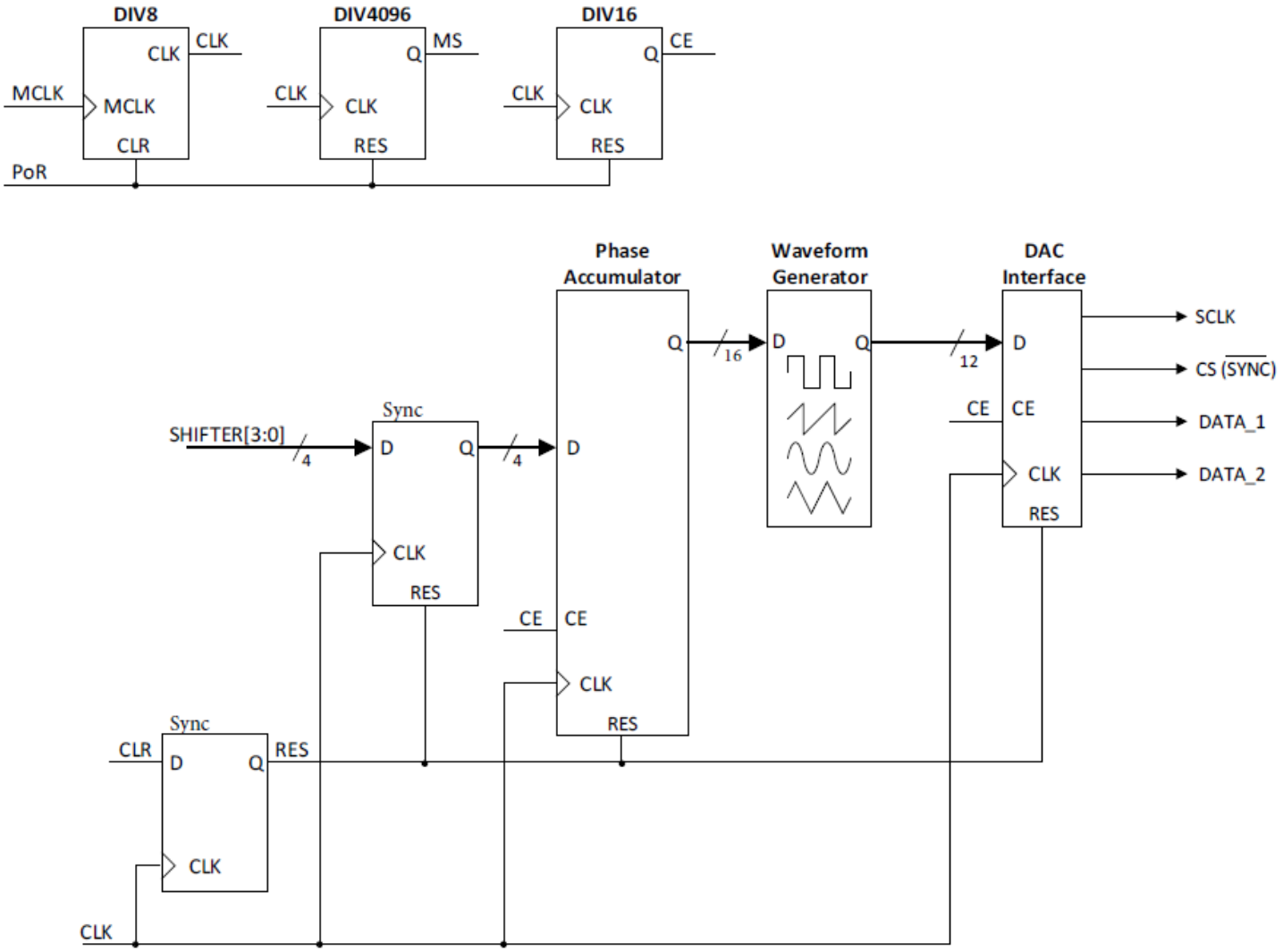


Figure 2. Conceptual Block Diagram of the DDS Waveform Generator Circuit

CAD1 Laboratory: First Task - Weeks 16 and 18

By the end of Week 18 session you should expect to have developed and tested the following VHDL models of the major circuit components needed to construct the DDS waveform generator.

Resets

There are two reset signals in this design. One (**CLR**) is a manual reset connected to a pushbutton, which will be a signal in the entity as shown in Figure 1. The other (**PoR**) is a Power-on Reset. This is a control that is generated by the hardware internally. It starts the FPGA circuit running and loads initial values. It is not a VHDL signal – its effect is implemented by the preset values in your VHDL circuit. For example:

```
signal clk_i : std_logic := '1' ; -- The signal will be one at Power-on Reset
signal count : integer := 0 ; -- The counter will be reset at Power-on Reset
```

Timing Generation (`DIV8, DIV16, DIV4096`).

These 3 components can be found in Figure 2. The Nexys-3 board has a 100 MHz crystal clock. In this design the internal logic on the FPGA will use a 12.5 MHz clock. This is achieved by building a "divide by 8" counter (called **DIV8**) which produces the internal **`CLK`** signal from the Master clock input (**`MCLK`**) which is connected to Pin **V10** of the FPGA on the development board.

Hint: draw the timings diagram on a sheet of paper before you start any coding.

Two further timing circuits are needed: **DIV16** which produces a *pulse*, one **`CLK`** long *every* 16 **`CLK`** clock cycles (called **`CE`** in the block diagram) and **DIV4096** which also produces *a pulse*, one CLK long every 4096 **`CLK`** cycles (called **`MS`** in the block diagram). Both pulse generators have a synchronous reset (**`RES`**).

Begin this laboratory by building a VHDL model and a test bench for *each* of these three circuits to verify that they work as expected. **Hint: These circuits are very similar in structure and you should be able to re-use much of the code you develop.**

The Debounce Circuit (DEBOUNCE) Implemented by SYNC

Four slider switches (SW7 – SW4) will be used to program the output frequency of the phase accumulator (Pins T5, V8, U8, N8 respectively). To ensure proper operation of the circuit it is advisable to build a debounce circuit to ensure that there are no asynchronous glitches on these signals. Another debounce circuit is also required for Reset input which is connected to a push- button on the Nexys-3 board (Pin V2). The schematic shown in Fig 3 uses a Sync circuit rather than a proper debounce circuit (a simpler version..).

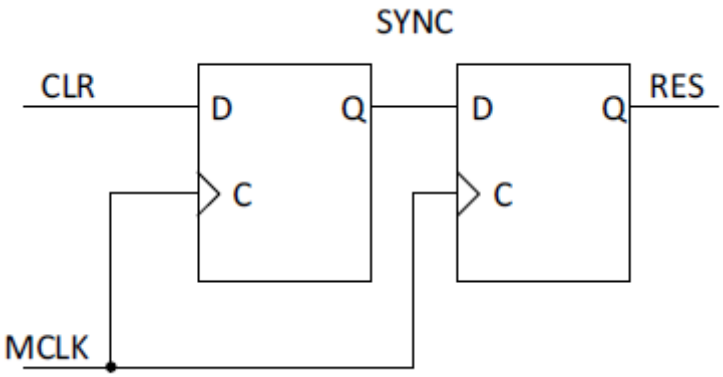


Figure 3. A Sync Circuit for RESET Button and Slider switches

The Phase Accumulator (PHASE_ACCU) Circuit

The Phase Accumulator is built using an adder and a register with a synchronous reset **RES** as shown in Figure 4. On the rising edge of the clock, if **CE** = 1 the output of the adder will be stored in the 16-bit register. The adder is used to accumulate the previous value of the accumulator with the 4-bit input **D**. The output of the register will be used as the input into the waveform generator.

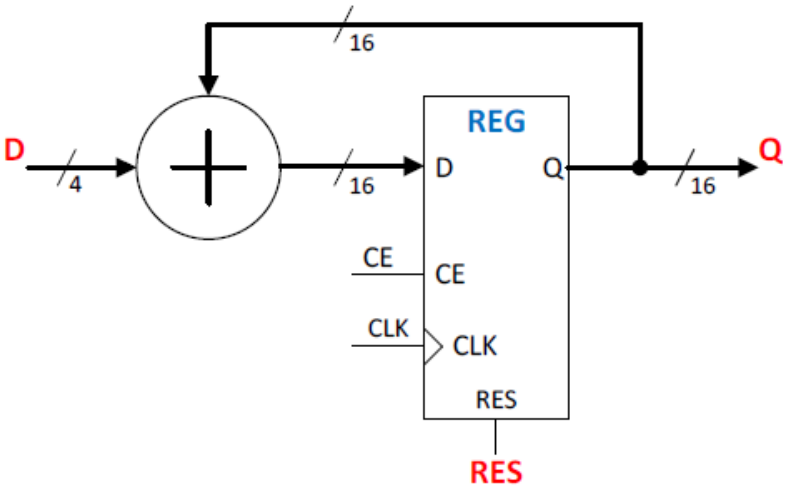


Figure 4. The PHASE_ACCU circuit

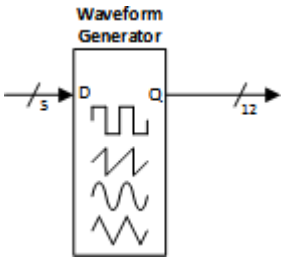
Build a VHDL model of this circuit and call it PHASE_ACCU. Devise a simple test-bench to test the output of the circuit when `D = "0011"`. What type of waveform do you expect to see?

Every **CLK** clock cycle, the input word is added to the previous value stored in the register (i.e accumulated). This will continue until the Accumulator overflows at which point the process begins again. When looking at the MSBs of the Accumulator output a sawtooth waveform is produced. The frequency of this waveform will be determined by the value input into the accumulator (referred to as the Phase Increment).

By using simple combinational logic it is possible to convert the output of the Phase Accumulator into any arbitrary waveform. The most common is a sine or cosine wave although triangle, square and any arbitrary shape is also possible.

CAD1 Laboratory: Second Task - Weeks 19 and 20.

The Waveform Generator Circuit (**WAVE_GEN**)



Waveform Generator Circuit

Each laboratory group will be asked to generate a **unique** waveform generation circuit.

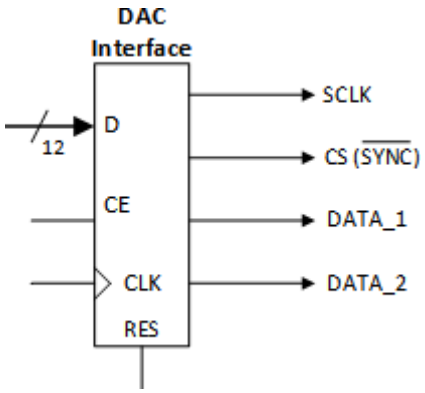
The circuit has a **5-bit** input and will take 5 of the 16 output bits from the **PHASE_ACCU** circuit and use them to generate a 12-bit output that will be used as an input to the **DAC_INTERFACE** circuit. Which output bits should it take? The selection of which bits to use is done in the Port Mapping for this component in the DDS top level design.

The best method for generating an arbitrary waveform is by means of a lookup table (known as a LUT). In this experiment, you have 5 bits input, which will define *how many* entries there are in the table and 12 bits output which defines *how big* each entry is.

You need to create a look up table with maps the input entries to a corresponding output value for the waveform which is available in your Moodle submission folder.

Excel is a useful tool for generating such data tables. Bear in mind that waves are symmetrical about 0, but the output must always be a positive number.

The DAC Interface Circuit (`DAC_INTERFACE`)



DAC Interface Circuit

It is necessary to convert the digital output of the waveform generator into an analogue signal. To generate an analog signal it is necessary to use a Digital-to-Analog Converter (DAC). Neither the Xilinx chip nor the Nexys-3 board has a dedicated DAC so it is necessary to use an extra peripheral component. This is called a PMOD DA2 and is available from the Technical Support Centre. PMOD is a 6-pin or 12-pin interface standard used by Digilent (and increasingly by other companies) to connect additional functions/circuits to the main experimental board. The Nexys-3 board has 4 Pmod ports. Please connect your PMOD to Port JA1. The DA2 Pmod actually has two Digital-to-Analog converters on it. Although this experiment only needs one, past years' groups have damaged some of the DAC devices so o/p 1 may not work, so you should program both outputs to generate the same data. The circuit that will be used to perform this conversion is the DAC121S101 which is manufactured by Texas Instruments. The data sheet for this peripheral component is available in the CAD1 Laboratory folder on Moodle. **It is important that you look at this data sheet before implementing your design. Please read it**.

The interface connection for the Pmod is shown in Figure 5. The pinout configuration is shown in Table 1.

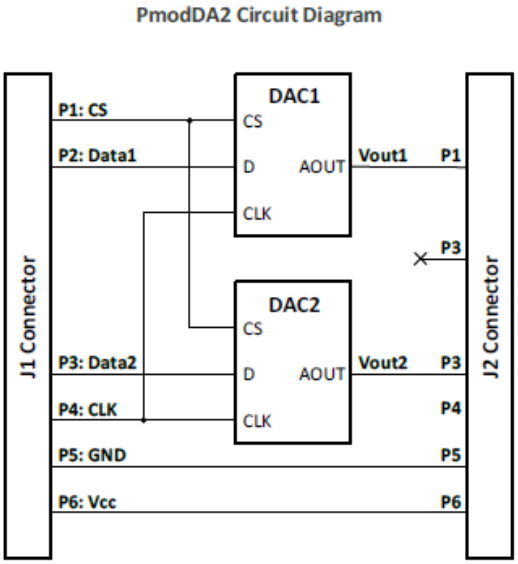


Figure 5. PmodDA2 Interface Circuit.

Pin No	Signal	Description
1	~SYNC (CS)	Chip Select (and Data Sync)
2	DINA	Data in for Channel A
3	DINB	Data in for Channel B
4	SCLK	Serial Clock
5	GND	Ground
6	VCC	Power (3.3V or 5V)

Table 1. Pin-out of PmodDA2 peripheral

The timing waveform for this circuit has been reproduced here for reference (See Figure 6). The waveforms shown are the inputs to the PmodDA2 (J1 connector in Figure 5) so your circuit needs to generate them. The SCLK runs at the same rate as your internal clock. The timing on Figure 6 is for the DA2 reading data, therefore it uses the falling edge. As you are writing data, you must use the rising edge. The DATA is expected to be 16 bits long, but the output from the Wave Generator is only 12 bits. You will need to read the documentation for the PModDA2 to find out what the other 4 bits are meant to be and whether they should appear before or after your wave data. Consider how you would build this circuit and develop a block diagram before starting to code it in VHDL. Make sure you have checked the timing constraints shown in this diagram and that you have fulfilled them. **Show your diagram to a demonstrator to ensure that the design is suitable before coding and testing in VHDL.**

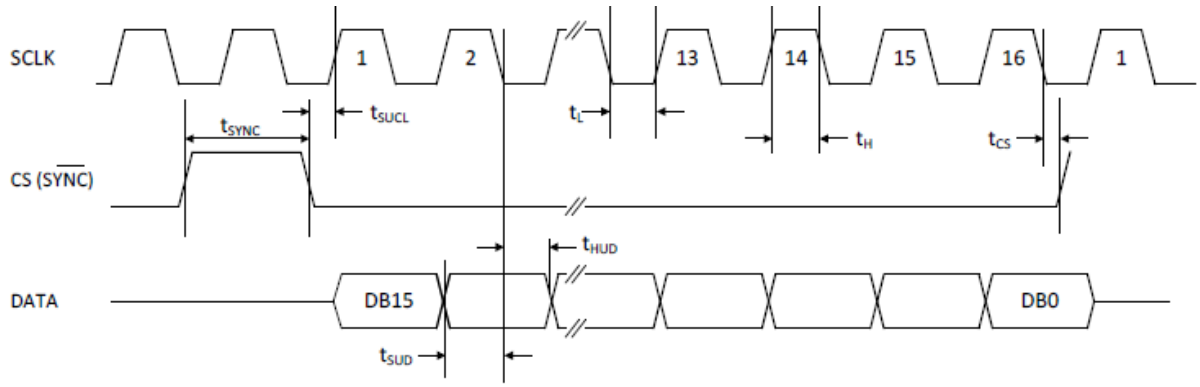


Figure 6. Timing Diagram for DAC Interface.

Build a final test bench to confirm the operation of your complete circuit which should now contain all of the components in Figure 2 which you have now designed.

Implementation

Once you have completed the ModelSim design phase you are ready to implement this circuit on the Xilinx FPGA. The separate introduction to Xilinx ISE in Week 14 told you how to do this. At the end of the process you will have generated data describing how much logic your circuits have generated. Be sure to include this data with your laboratory report. Remember to use the User Constraint File provided on Moodle (**Nexys3.ucf**) to ensure that your logic signals are connected to the right pins on the FPGA development board.

Once you have an error free synthesised circuit you are able to download it onto the board and use an *oscilloscope* to check the output waveform. This last part is not assessed but you will you will be able to do this once the school opens by getting one of the Spartan 6 boards on a short loan from TSC. If you have any doubts at this stage **ASK THE DEMONSTRATOR to confirm that you have done everything correctly before proceeding with the download. These devices can be easily damaged.**

Do NOT connect the DA2 board until you have verified that the output from the FPGA (JA1 to JA4) is correct.

Submission Checklist

At the end of this experiment you should submit the following in a zip file to Moodle:

- All VHDL files for the DDS circuit together with their respective Test Benches. Remember to format and comment your code. Marks will be lost for uncommented code.
- Screen plots of Modelsim Waveform outputs demonstrating the operation of the component circuits
- A Xilinx project directory containing the DDS project and all of the log files generated by the synthesis process describing the device usage.
- A valid `.bit` file. All files will be tested again when the work is marked.
- A brief report summarising what has been achieved.

These data should by zipped and submitted to Moodle by Friday 23:59 Week 22.

References

[1] **FPGA Prototyping by VHDL Examples (Xilinx Spartan™ -3 Version), Pong P. Chu, Wiley, 2008. ISBN 978-0-470-18531-5**

[2] **DAC121S101 12-bit MicroPower Digital-to-Analog Converter (Texas Instruments)**

[3] **Digilent PmodDA2 Reference Manual.**

[4] **Digilent Nexys3 Reference Manual.**

[5] **EL568 VHDL Lecture Notes**

[6] **An Introduction to Digital Logic Booklet**