Embedded and Ambient System Laboratory

Syllabus for FPGA measurements 1. 2. 3.

Measurement 1: Introduction to Verilog, to LOGSYS Spartan 3E board and the task

Measurement 2: Design tasks of Calc_1, Calc_2, Calc_3 and Calc_4

Measurement 3: Design task of Calc_5.

Place of measurements:

Department of Measurements and Information Systems

FPGA Laboratory, Building I, wing E, Lab. IE321.

Measurement 1:

Introduction

The goal of this measurement is to introduce the design of a complex digital systems, starting from specification, ending with final verification and documentation.

During the design **Verilog HDL** (Hardware Description Language) will be used for the specification and an FPGA (Filed Programmable Gate Array) device will be used, as the implementation platform. Each functional block will be described using **high level language constructs**. Functional blocks will be connected using **structural description**, building hierarchical design.

Task to be implemented: Simple Calculator

The task is to design and implement a simple calculator, using the Logsys Spartan 3E FPGA development board. The calculator will perform the four basic arithmetic operations, and will use 4 bit wide operands. Results will be displayed on the development board displays (LEDs, and 7 segment display).

During the 3 scheduled lab different versions of the simple calculator will be designed, which assists you to understand the design of complex digital systems, and also helps to try out the different peripherals of the development board.

Documentation of the work must be prepared about the last two versions of the calculator. Documentation must inlcude:

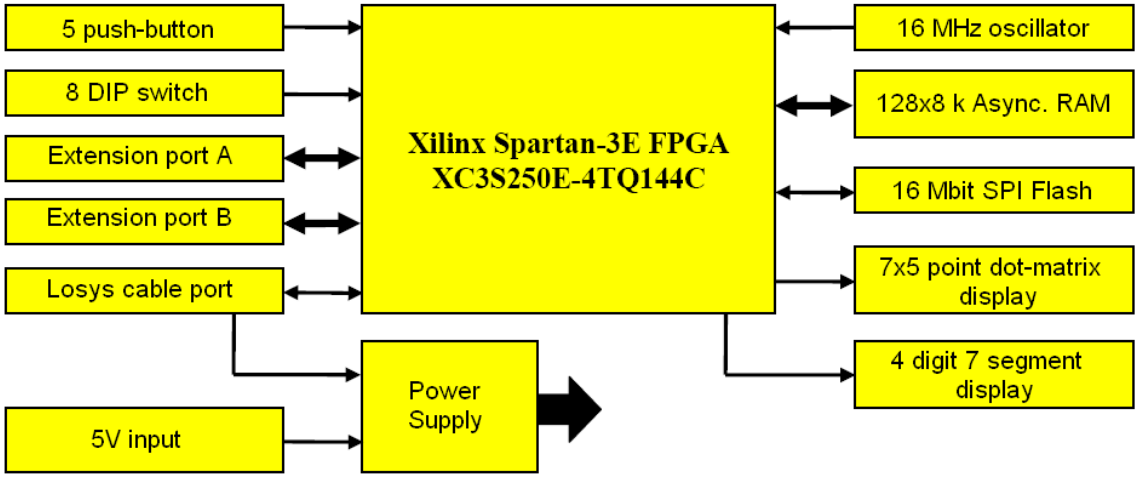
- *Short task description*
- *Optionaly block diagram*
- *Full Verilog source code*
- *Simulation screen-shoots*
- *Short description of the implemented modules.*

The general specification of the calculator:

- Input data:
  - 4 bit wide operands, using the development board switches. (2 x 4 pcs.)
  - Inputs are positive integer numbers
- Operations
  - Addition
  - Substraction
  - Multiplication
  - Division (In the first design XOR will be used for simplicity)
- Operations can be selected using four buttons.
- Result display
  - LEDs, in binary form
  - 7 segment display, decimal and hexadecimal format.

The LOGSYS Spartan-3E development board.

The Logsys Spartan-3E development board is a simple and cost effective development board, designed for beginner HDL designers, mainly used in education. The following image depicts the internal structure and peripherals of the board.



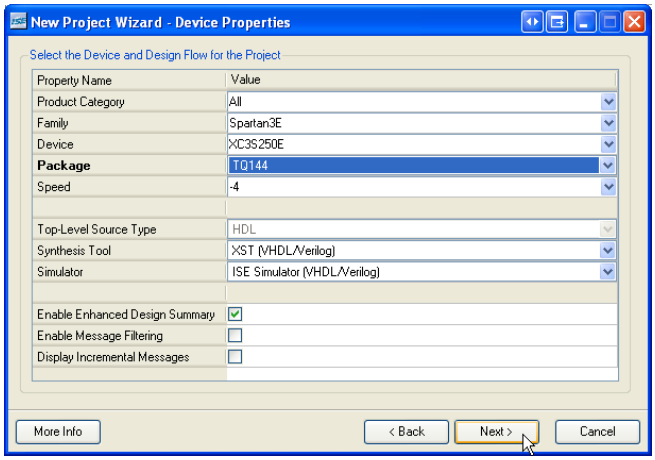
Detailed specification is available (in Hungarian):

http://logsys.mit.bme.hu/download/LOGSYS_SP3E_FPGA_Board.pdf

Webpage of the Logsys system: (Drivers/programs)

http://logsys.mit.bme.hu/

Important! The projects must be compiled using the following setup:



Peripherals being used

LEDs

The Logsys Spartan-3E includes 8 LEDs, marked with LD7 (left most) … LD0 (right most). Pin numbers can be found in the following table, and indicated on the PCB also. LEDs must be driven using active high volatge levels.

LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA pin	P43	P50	P51	P52	P53	P54	P58	P59

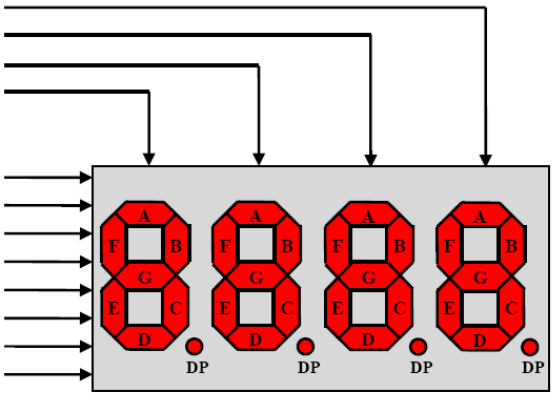
Seven segment display.

The following picture summarizes the connection pattern of the 4 digits 7 segment display of the Logsys Spartan-3E development board. Digits are numbered as DIG3 (left most)..DIG0 (right most).

- **All seven segment signals are active low**. (Both digit and segment signals)
- **Segment driver and dot matrix display line driver signals are common, which might cause aritfacts on the unhandled dot-matrix display.**

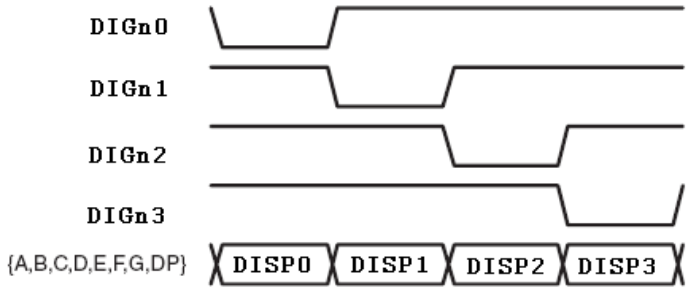
Digit select	FPGA pin
DIGn0	P40
DIGn1	P35
DIGn2	P26
DIGn3	P22

Segmen select	FPGA pin
SEGn0 (A)	P16
SEGn1 (B)	P17
SEGn2 (C)	P20
SEGn3 (D)	P21
SEGn4 (E)	P31
SEGn5 (F)	P32
SEGn6 (G)	P33
DP	P34



All four digits can be used in **Time Division Multiplexing** mode, while each digit has a common segment driver.

Time Division Multiplex (TDM) driving of the 7 segment displays. DIGnX selects the actual digits, while SEGnY (A, B, C, D, E, F, G, DP) asserts the active segment lines of the required number.



At first the right most digit is selected, by driving DIGn0 signal low. (Active low drive, DIGn0 = 0, DIGn1=DIGn2=DIGn3= 1). Meanwhile segment driver signals (A-G, DP) are controlled to display the corresponding character. (Eg. To display an "0" character, A=B=C=D=E=F=0, G=1, active low drive)

After that the system will select the second digit (DIGn0 = 1, DIGn1=0, DIGn2= DIGn3= 1), meanwhile driving the segment driver signals accordingly. And so on…

Four digits must be selected in a cyclical manner. The TDM frequency must be selected to be higher than human eye's inertia. Usual multiplexing frequency is cca. 500 Hz. (Multiplexing frequency is limited by the circuit to cca. 10 kHz).

DIP Switches

The development board contains 8 DIP Switch. The pinout can be found in the following table. (Leftmost switch is marked as 0, right most is 7.)

Switch	7	6	5	4	3	2	1	0
FPGA pin	P47	P48	P69	P78	P84	P89	P95	P101

Buttons:

A LOGSYS Spartan-3E development board has 5 buttons. The leftmost button is BTN3. The rightmost button is called as RST, although it is similar to the others, has no dedicated reset circuit, but mainly used to issue manual RESET to the tested system.

Button	BTN3	BTN2	BTN1	BTN0	RST
FPGA pin	P12	P24	P36	P38	P41

Clock circuit:

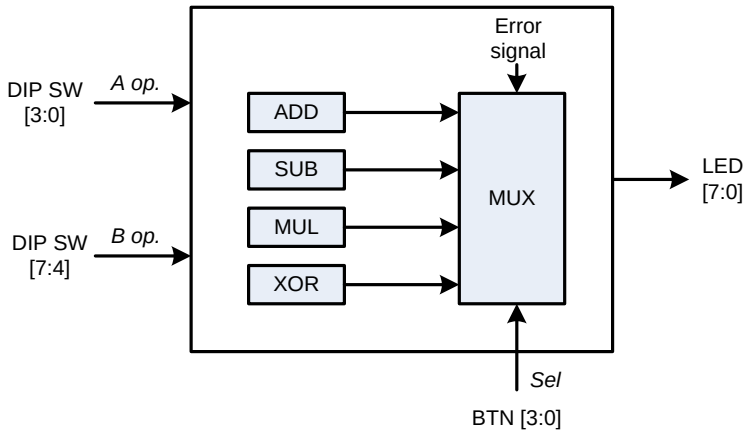
Two clock sources are available on the board. Both will be used in our design

- 16 MHz fixed oscillator. Pin number: P56
- LOGSYS Development port CLK line, used in USRT data transfer: P129

Measurement 2:

Design task 1 : CALC_1 circuit:

Four bit wide input operands are set using the dip switches, the operation is selected by pressing one of the four buttons. Results are displayed on the eight LEDs in binary format. Operations are: addition, subtraction, multiplication and exclusive or (**XOR).** Error condition occurs when the result of the subtraction is a negative number. *Error is displayed by turning all LEDs on*



Remark: All four operations can be realized at different abstraction level of hardver description.

1. A trivial approach would be to define a table/array using embedded memory. While we have only 2 simple 4 bit operands, it would need a 256 deep 8 bit wide memory. Let us forget this solution. (Imagine the memory requirements in case of 16 bit operands!)
2. We can define each operation using a **bit level** description. E.g. a simple one bit adder circuit consist of AND/OR and XOR gates. This is also not the best way, while the synthesis tool able to built up the circuit, but can not always recognize the real content of the logic from this low level description, and this might results in an un-optimal mapping to the FPGA primitives.
3. Each operation can be described using **high level operators.** (like + , - ,*, ^) This simplifies the implementation, and results in easy-to-read code. As we will see division operation can not be synthesized, therefore algorithmical description is needed.

Task: Design the CALC_1 calculator. Source code must be verified using ISE Simulator. If simulation results are correct, implement the design and download to the FPGA. Check functionality on the development board too.

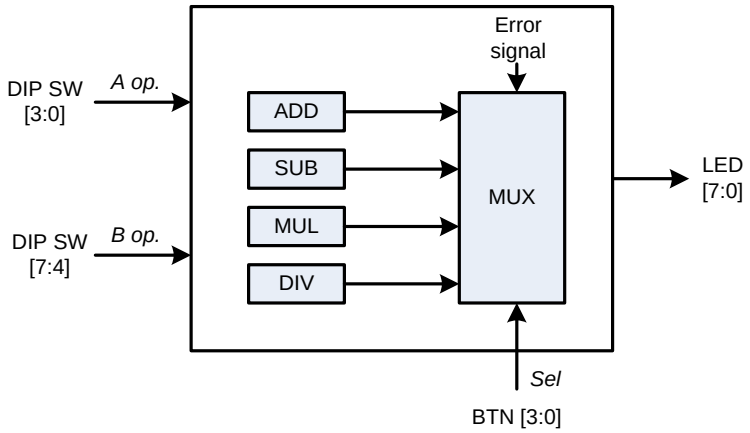
Important! All designed modules must be globally synchronous.

All Flip-flop must get the same clock, *always* block must use the same clock's rising edge.

Please use reset for flip-flops.

Design task 2: CALC_2 circuit:

Let us modify the previous circuit, by replacing the XOR operation with division. Also extend the error handling with division by zero.



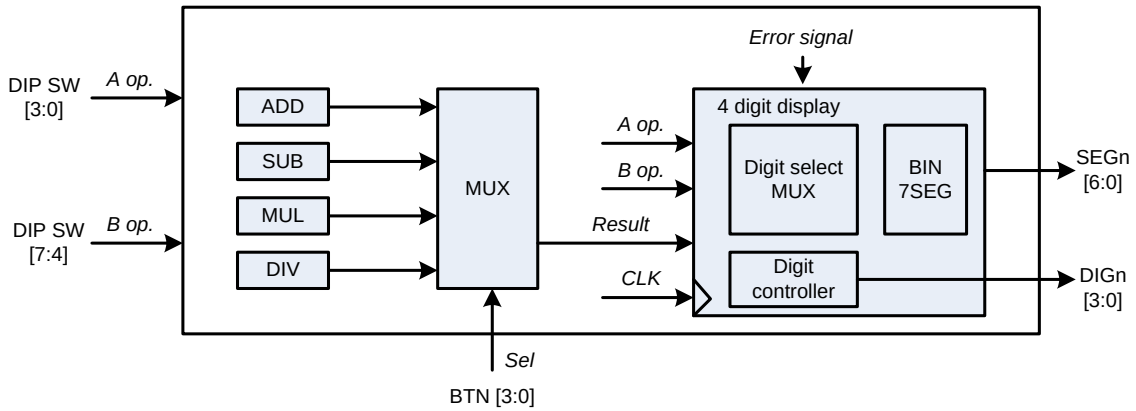
Binary division can be implemented in many different ways:

1. We can build a fully combinatorial design, as we would execute the division on paper.
2. We can handle all possible scenarios with a big “case” condition or as mentioned in the former task description, using a 256 entry look-up-table..
3. Alternatively sequential implementation is also possible. (Preferred) Divisor is subtracted from dividend as many times as the result is bigger than divisor, meanwhile counting the cycles. Quotient is equal to the cycle count.

Design task 3: CALC_3 circuit:

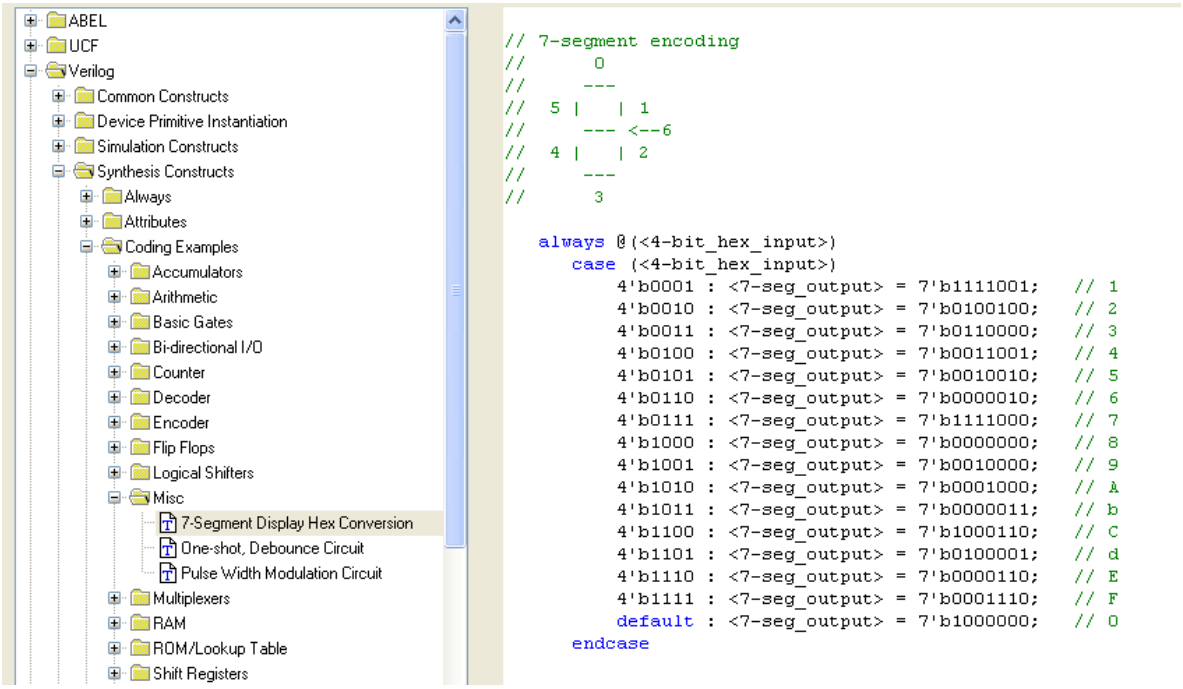
CALC_2 Calculator is fully functional, but the usage is quite inconvenient, while results appear only on the LEDs. Let us extend the functionality with a **numerical** (seven-segment) **display.**

- The first two digits should display the two input operands, while the last two digits should be used for displaying the results in **hexadecimal format.**
- Last two digits displays should remain blank when no operation is selected.
- If error condition occurs, last two digits should display *„EE"* indicating error.



Note: BIN7SEG module is available in the development environment under:

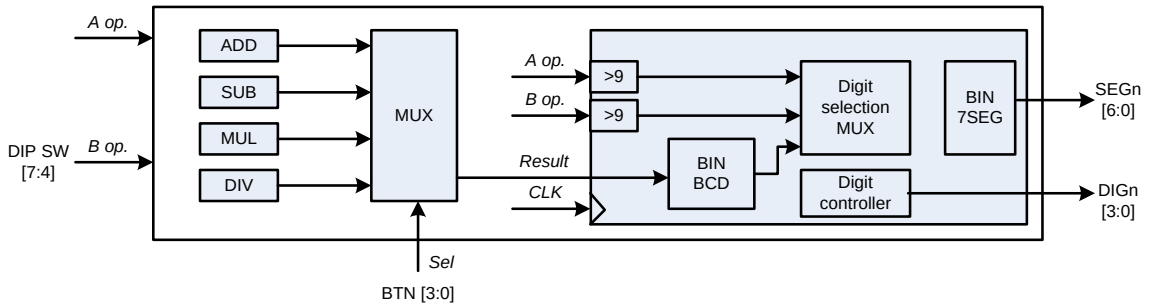
Edit Menu > Language Templates…  **. Verilog > Synthesis Constructs> Coding Examples > Misc > 7-Segment Display Hex.**



Design task 4: CALC_4 circuit - *Should appear in the final documentation*

CALC_3 Calculator is very handy for engineers, who understand hexadecimal numbers, but traditional users might be messed up. Modify the CALC_3 circuit in that way, that it **accepts and displays numbers only in decimal format.**

- Only input numbers in range 0 .. 9 should be accepted; otherwise the corresponding digit should display „E" indicating error condition.
- By limiting the input numbers to decimal, the biggest number to be displayed is 9 x 9= 81, which allows us to display the result on –the rightmost- two digits.
- CALC_4 should include a binary-to-BCD unit to convert the result in usual decimal –human readable- format.



(The usual (Sub <0, DIV B op==0) error handling is not depicted here.)

Remark: Binary-to-BCD is not a simple operation. Operation can not synthesized directly by most of the synthesiers.

Measurement 3:

Design task 5: CALC_5 circuit - *Should appear in the final documentation*

The CALC_5 calculator has many differences compared to CALC_4 circuit:

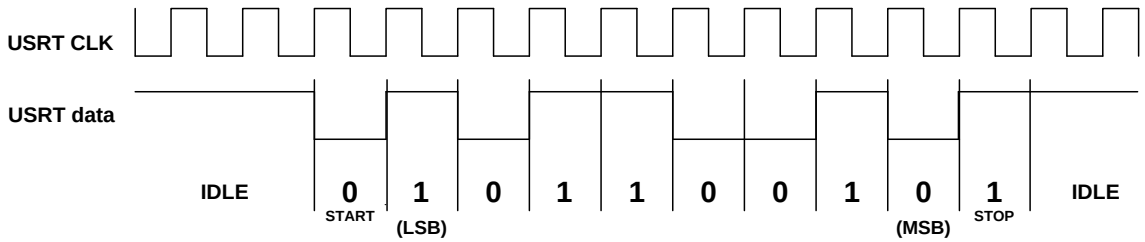
- Input operands and operations are received through standard USRT (Universal Synchronous Receive/Transmit) communication interface from a PC terminal window. .
  - Both operands might be one or two digits (Max. input number is 99)
  - ENTER is validating each input. An input sequence would be:
    - Operand A: One or two digit decimal number
    - ENTER
    - Operand B: One or two digit decimal number
    - ENTER
    - Operation code: ”+” ”-” ”*” ”/”
    - ENTER
  - ESC aborts current calculation, and resets the calculator. (Waiting for Op. A)
  - Note: In case of two digits operands, the transmission order is tens – units
  -
- The result is displayed on four digit seven-segment display.

The USRT receiver should use the standard UART framing, with the following settings:

- 8 Data bits
- No parity
- 1 stop bit

The USRT cable shifts data on the rising edge of the USRT_CLK, therefore ideal position for sampling data is at the falling edge USRT_CLK. Please remember, that **fully synchronous design should be created**, therefore USRT_CLK can not be connected directly to any flip-flop clock input. Falling edge of the USRT_CLK should be detected with a dedicated circuit running from the main 16 MHz clock input.

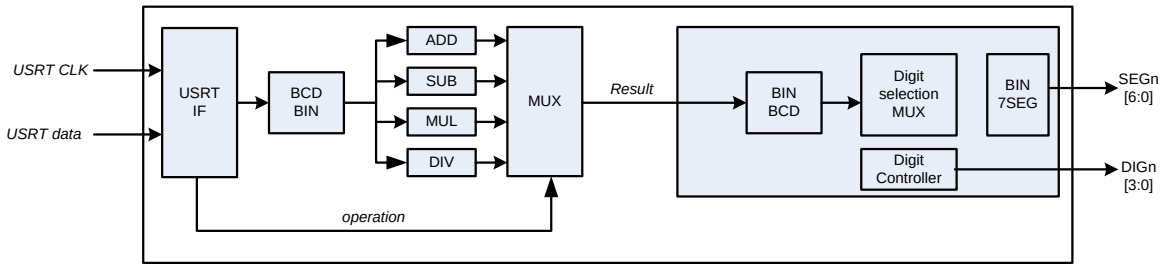
The following figure depicts the standard USRT framing.



The required ASCII codes:

Character	**ASCII code**
0 – 9	0x30 – 0x39
+	0x2B
-	0x2D
*	0x2A
/	0x2F
ENTER	Can be modified at Logsys GUI \r: 0x0D (carriage return) \n: 0x0A (new line) \r \n: first: 0x0D, after: 0x0A
ESC	0x1B

Reccomended architecture:



USRT pinout:

- USRT_CLK: LOGSYS development port CLK line (P129)
- USRT RX (input data): LOGSYS cable MOSI (P120)
- USRT TX (ouput data): LOGSYS cable MISO (P143)

Test questions:

1. What are the benefits of using high level HDL languages?
2. What does hierarchical design means?
3. What is the difference between structural and behavioral description?
4. What does non-blocking statement means?
5. Define in Verilog a four bit wide register with asynchronous active high reset!
6. What is Time Division Multiplexing?
7. What kinds of resources are available in FPGA devices?