FPGA tutorial

Lecture 3

Wednesday 09.09.2015 – 14:00

Jochen Steinmann

Physics Institute III B

RWTH AACHEN UNIVERSITY

XILINX®

3rd Project - Summary

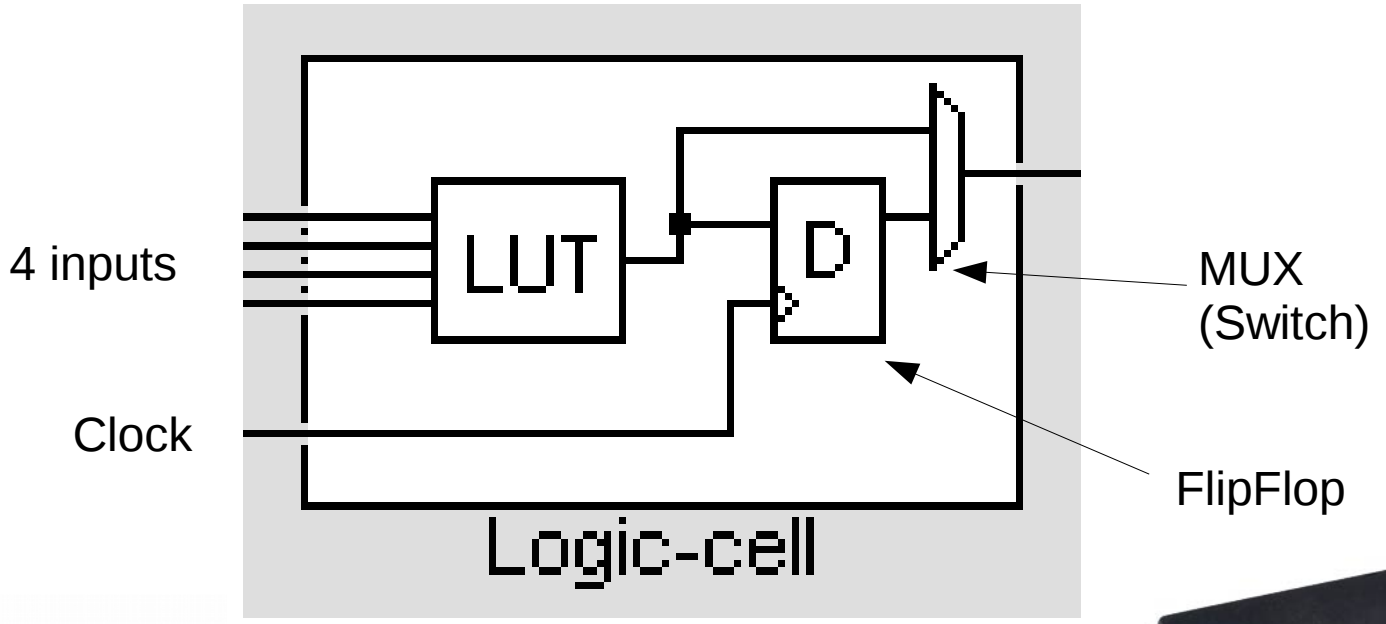
- Counter
  - CLK Clock input
  - DIR Up / Down
  - EN enable
  - RST Reset (synchronous)
    will be executed on next clock event
  - VAL 16bit wide **register** for counting value

3rd – Project – Solution

- In the skeleton I used switches for everything
  - also for clock!
  - gives opportunity to see, what happens when

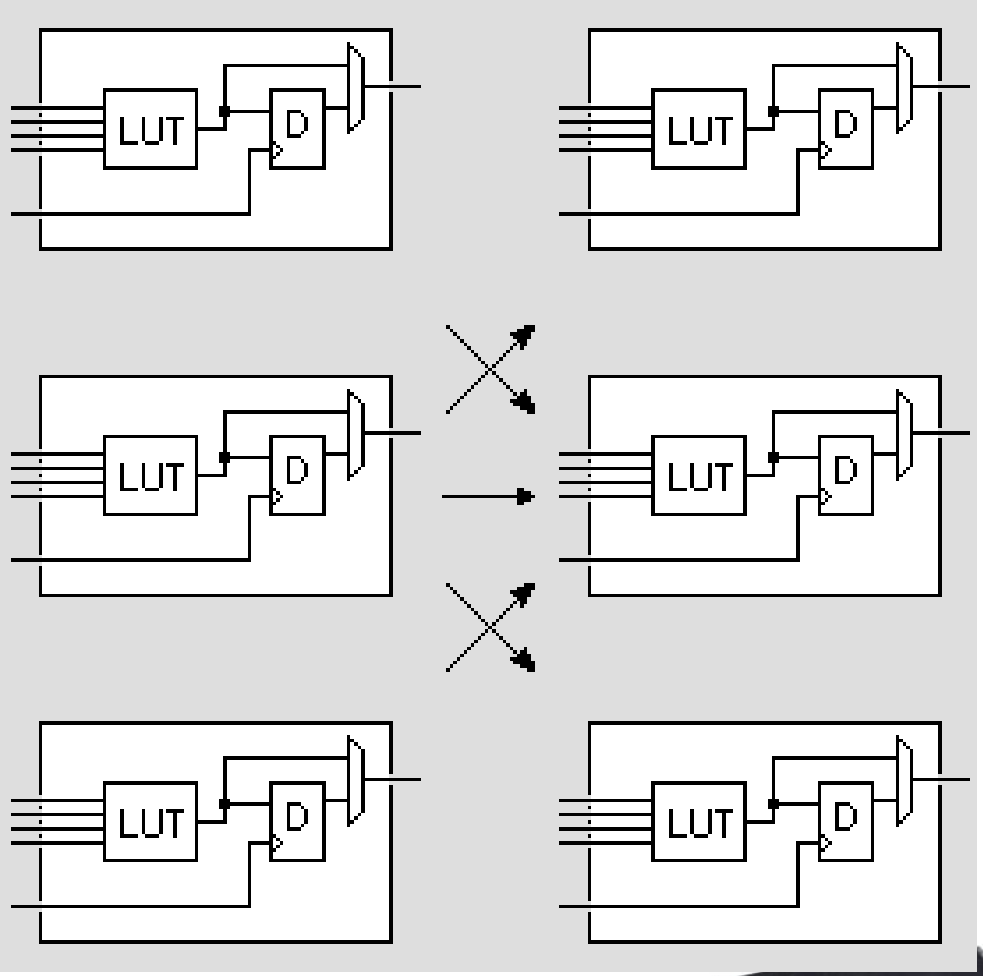
FPGA – Implemented Logic

- Almost all logic can be reduced to use ANDs, ORs and FlipFlops
- FPGA uses “logic cells” for implementation
- Up to 4 / 6 inputs can go to one cell



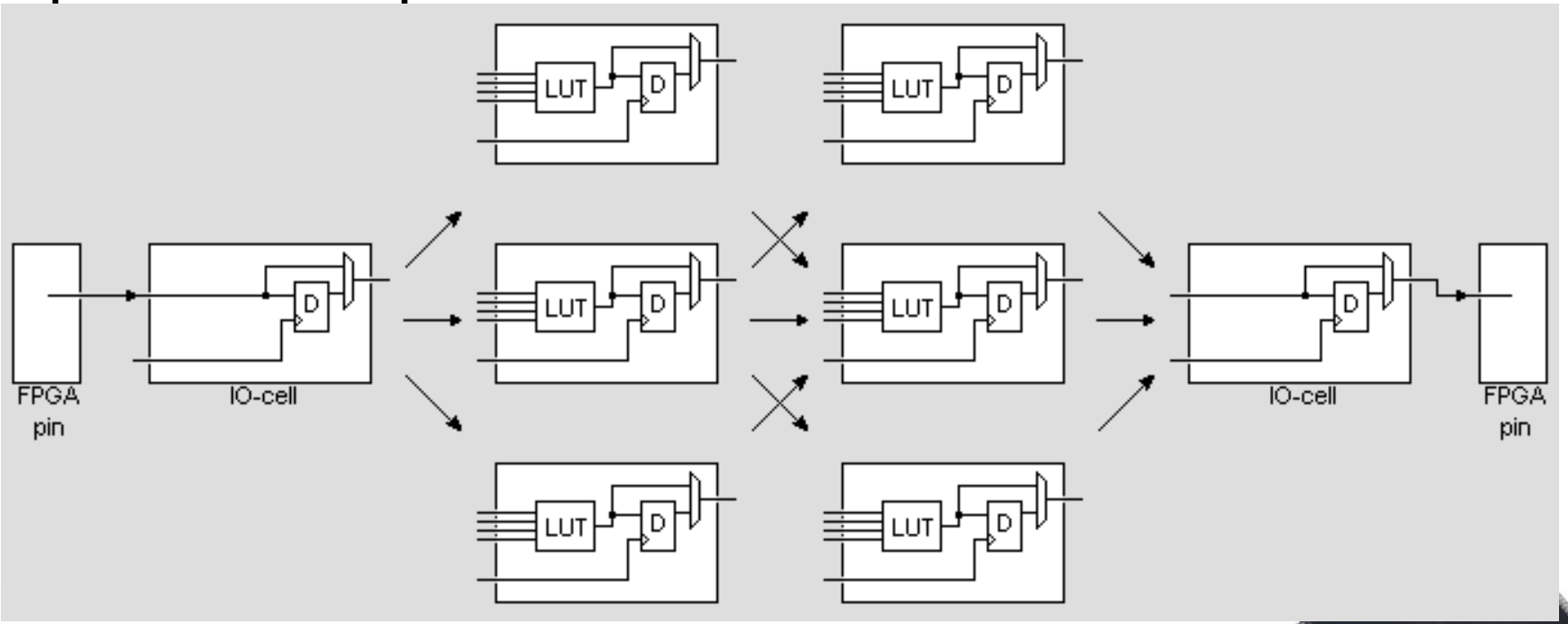
FPGA – Implemented Logic

- logic cells can be connected within each other
- inputs and content of LUT is generated by configuration
  → VIVADO
- FPGA on BASYS3
  - 33k logic cells



FPGA – input / output

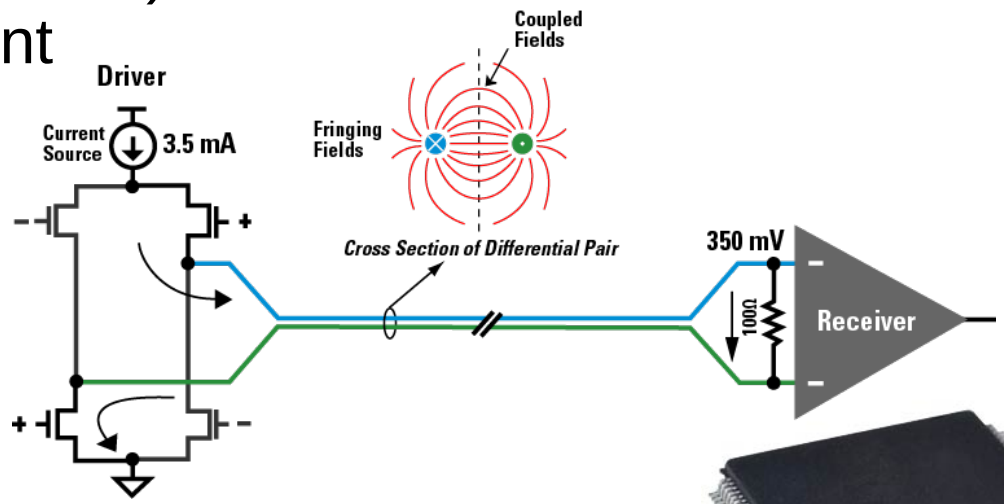
- IO – cells have
  - FlipFlops
  - Pull – Up / Pull Down
  - Input and Output Driver



FPGA – Output Levels

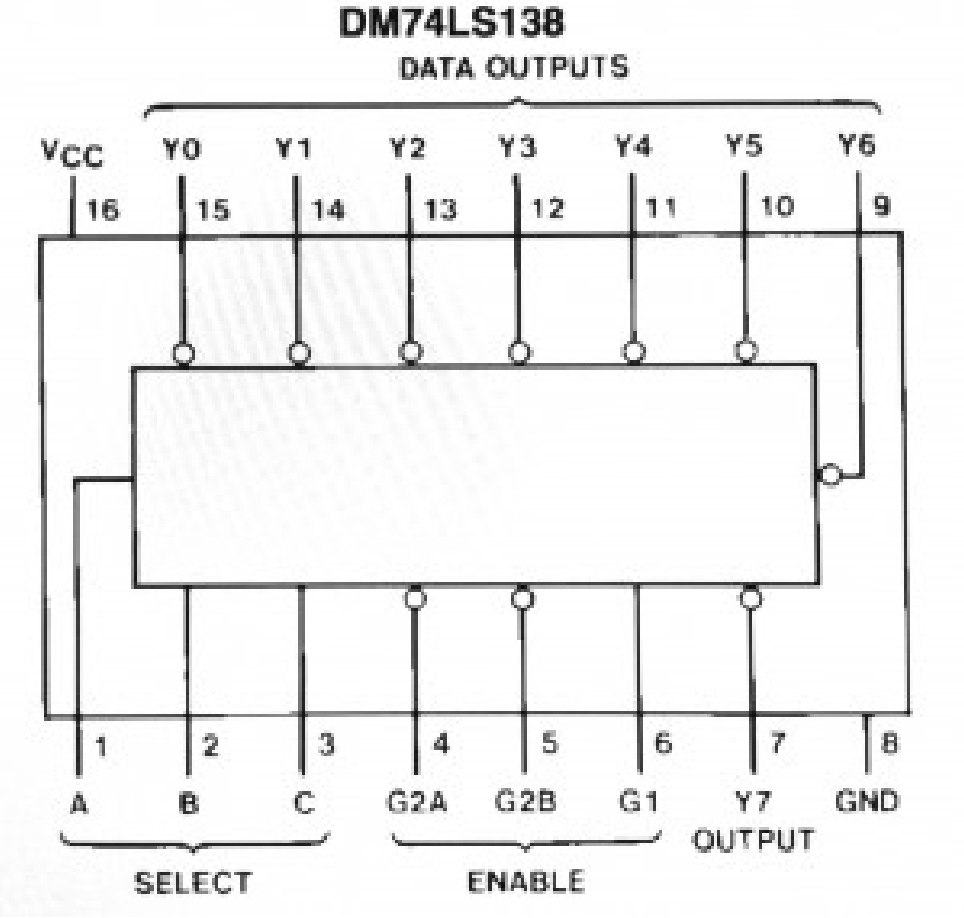
- Slow Signals
  - Single ended
    - LVCMOS, LVTTL (0 - 3V3)
- Fast Signals
  - Differential Signals
    - LVDS (CM 1V25 AC 400mV) sometimes using current source

Distortions affect only the common mode
Voltage difference is unaffected
Mostly using twisted pairs



Talking to many devices

- Address decoder
- selects only one device
  3 out of 8
- 2 different types
  - one hot (1)
  - one cold (0)
- all others inverted
  logic level



one famous type:
74 HC 138

Logic table of 74HC138

FUNCTION TABLE

ENABLE INPUTS			SELECT INPUTS			OUTPUTS							
G1	$\overline{G2A}$	$\overline{G2B}$	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	L	H	H	H	H	H	L	H	H	H	H
H	L	L	H	L	L	H	H	H	H	L	H	H	H
H	L	L	H	L	H	H	H	H	H	H	L	H	H
H	L	L	H	H	L	H	H	H	H	H	H	L	H
H	L	L	H	H	H	H	H	H	H	H	H	H	L

disabled

Need of an address decoder?

- Independent:
  - outside FPGA
  - inside FPGA
- Different logic blocks (chips) have to be addressed.
  - ensure that only one is active and talks with the master
  - if multiple devices (slave) are active, may cause short circuits and lot of trouble

4th – Project

- Address encoder
  - if talking to multiple chips, you have to ensure, that only one is selected at the same time!
  - Two options
    - one hot → selected chip 1, others 0
    - one cold → selected chip 0, others 1
  - Input
    - 2 bit (0-3)
  - Output
    - 4 bit

Hint:

you can use a bus to access elements of a bus

a[b] → will access the bit b of a

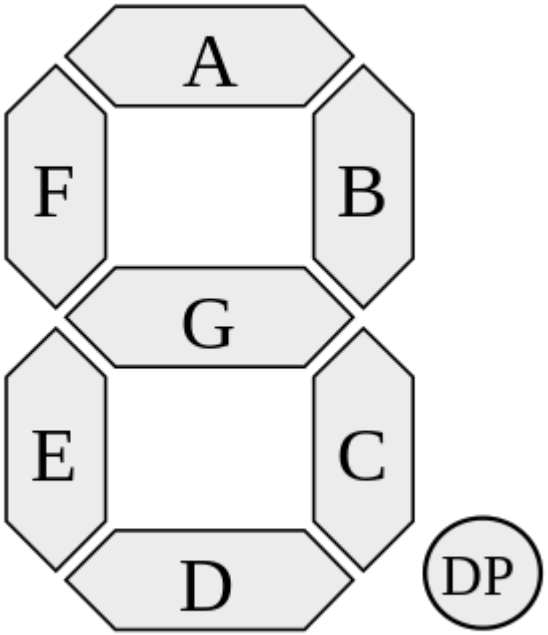
Output to humans

- There are many ways how an FPGA device can communicate with the “operator”
  - High energy physik
    - mostly remote access → e.g. Network etc.
  - Consumer devices
    - need graphical menues
      - LEDs → easy
      - **7-Segment displays → more complex, but still simple**
      - LCDs → too complex for this course

Difference

- Humans use **decimal** system
- FPGAs and computers use **binary** system

- Need to convert between different worlds
  - 1001 → 9

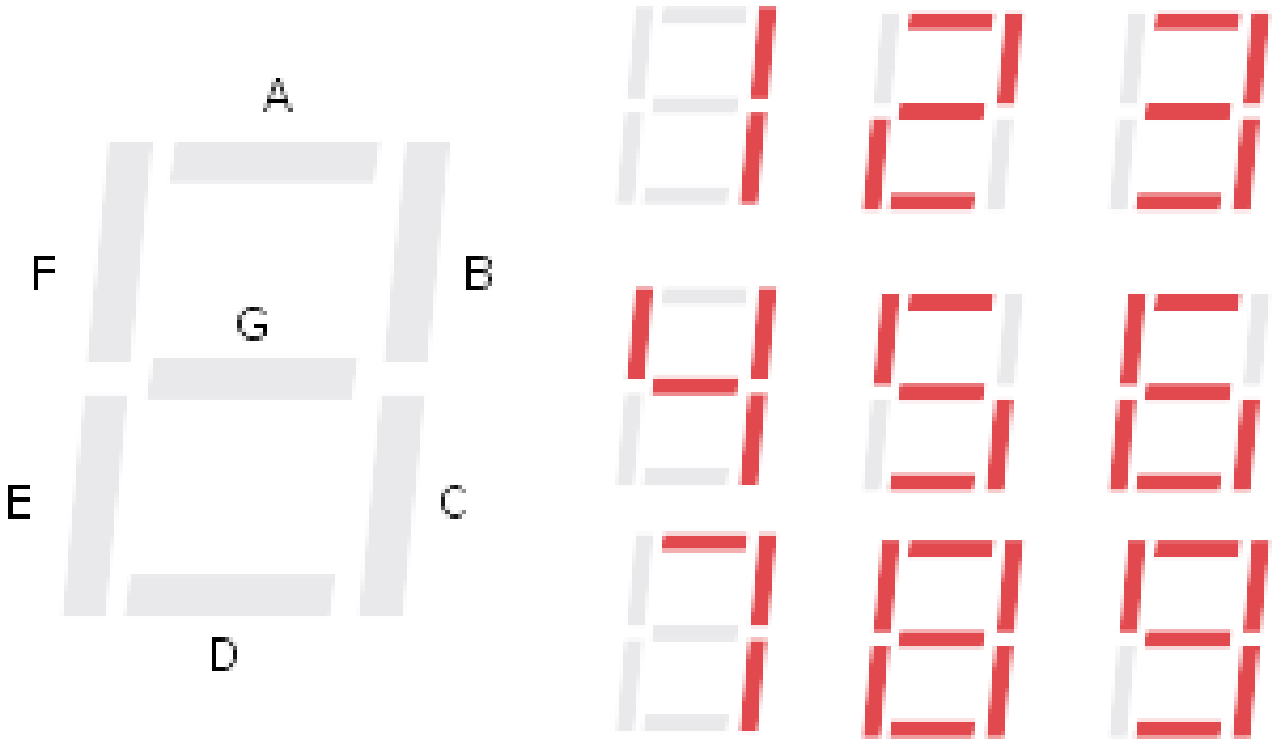


1st 7 segment lookup table

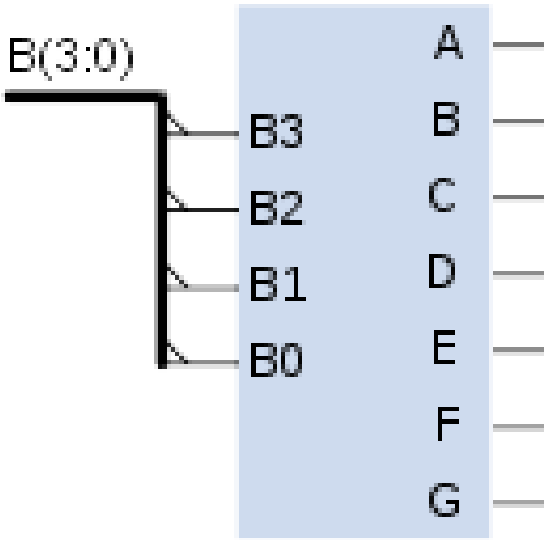
5th - Project

- Heading forwards to display numbers on the 7-segement display
- Lookup table for 7 segment display
  - 7 segments can display 10 values 0 – 9
  - depending on the input different outputs have to be switched on
- Inputs 4 Bit → sw3 – sw0
- Outputs 8 Bit → 7 Segments

5th – Project



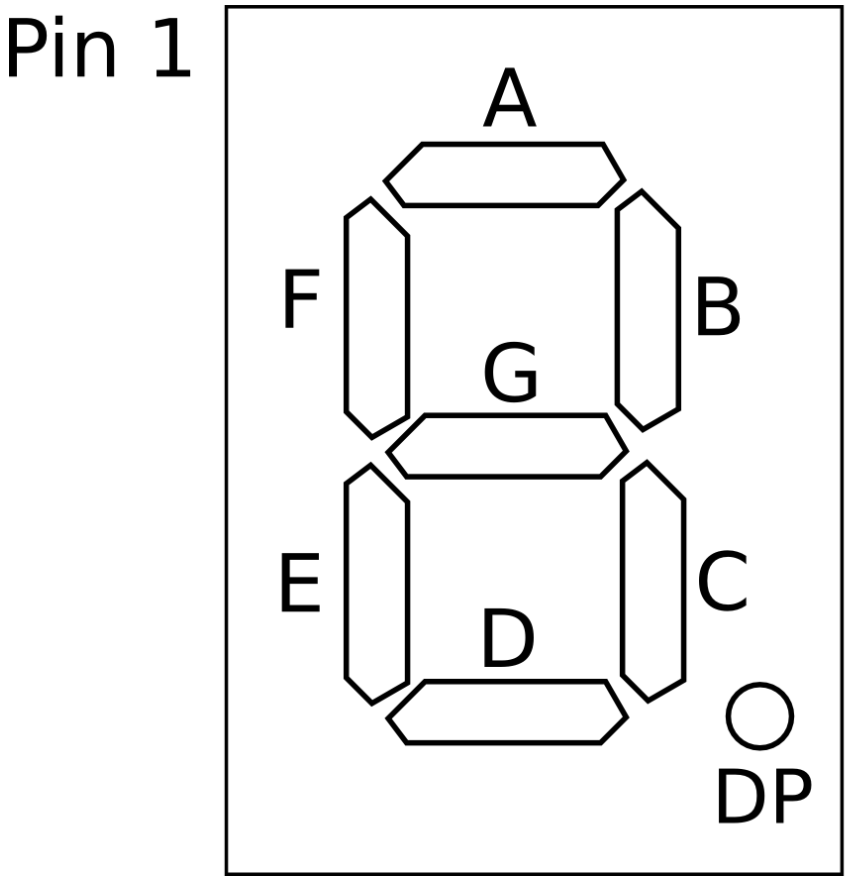
An un-illuminated seven-segment display, and nine illumination patterns corresponding to decimal digits

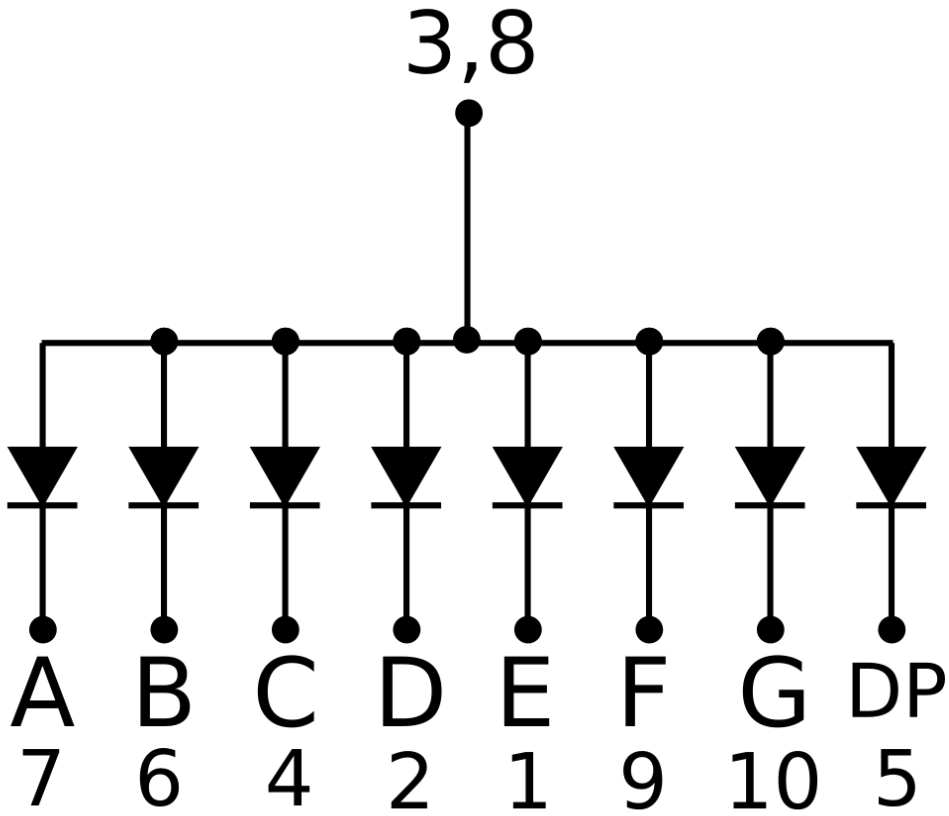


one Bus

segments = 'b0000011;
A + B on

5th - Project





Verilog – CASE

- simple case structure

```
reg [1:0] address;
case (address)
    2'b00 : begin
        statement1;
    end
    2'b01, 2'b10 : statement2;
    default : statement3;
endcase
```

either you implement all possible combination, or you use a default Otherwise the logic is not fully determined.

6th - Project

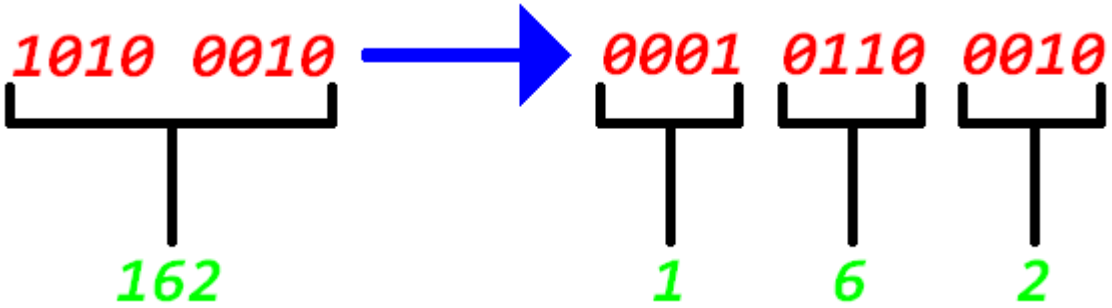
- Problem:
  - FPGA uses binary notation
  - Human uses decimal notation
  - have to split our number into digits
    - ones
    - tens
    - hundrets

use 8 bit input → 0 … 255

Binary Coded Decimal
4 Bit for each digit

BCD

- Example:



BCD Algorithm

1. If any column (1000's, 100's, 10's, 1's) is 5 or greater add 3 to that column
2. Shift all #'s to the left 1 position
3. If 8 shifts are done, it's finished.
   Evaluate each column for the BCD values
4. Go to step 1.

http://www.eng.utah.edu/~nmcdonal/Tutorials/BCDTutorial/BCDConversion.html

BCD – Pseudo Code

```
for(i=0; i<8; i++) {
        //check all columns for >= 5
        for each column {
                if (column >= 5)
                        column += 3;

        //shift all binary digits left 1
        Hundreds <<= 1;
        Hundreds[0] = Tens[3];
        Tens << = 1;
        Tens[0] = Ones[3];
        Ones << = 1;
        Ones[0] = Binary[7];
        Binary <<= 1;
}
```


100's	10's	1's	Binary	Operation
			1010 0010	
		1	010 0010	<< #1
		10	10 0010	<< #2
		101	0 0010	<< #3
		1000		add 3
	1	0000	0010	<< #4
	10	0000	010	<< #5
	100	0000	10	<< #6
	1000	0001	0	<< #7
	1011			add 3
1	0110	0010		<< #8

162

1 6 2

Verilog – for loop

- define somewhere an integer i

  for(i=start; i<=stop; i=i+1) begin

  end
- Difference to C/C++ or PCs
  - due to synthesis loops runtime is zero
  - **loops don't need a clock!**
  - just to make the source code look a bit nicer :-)

Verilog - shift

- Multiply and divide by 2
  - value << position → shift to left → * 2
  - value >> position → shift to right → / 2
- if you shift out of memory, the bit is lost

ToDo

4) address decoder
   (de-) select one bit of 8
   3 bit input
   3 to 8 (one cold)

5) 7 segment lookup table
   map 4 input bits to the 7 segments, which should light up

6) Binary coded decimal
   display 4 bits as 0 … 9 on the 7 segments