FPGA tutorial

Lecture 6

02.12.2020

Jochen Steinmann

III. Physikalisches Institut B

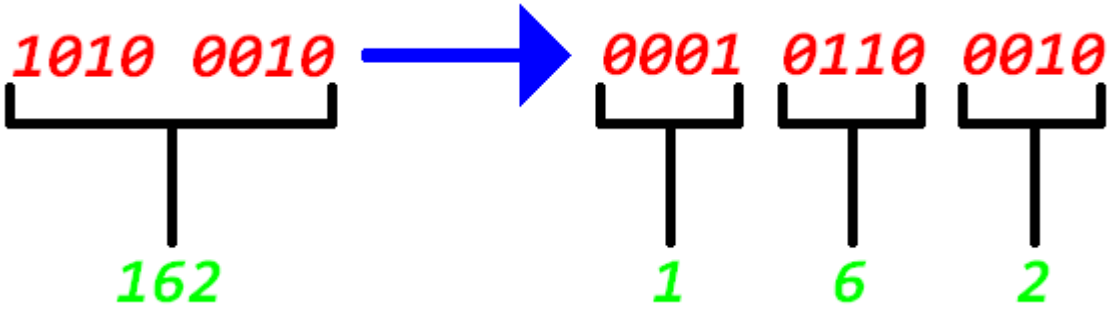
RWTH AACHEN UNIVERSITY

Think hardware!


Solution

BCD

- Example:



We would like to have this for 4 digits → Up to 9999

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 **14** shifts are done, it's finished.
   Evaluate each column for the BCD values
4. Go to step 1.

9999 = 10 0111 0000 1111

BCD - Example

100's	10's	1's	Binary	Operation	
			1010 0010		162
		1	010 0010	<< #1	
		10	10 0010	<< #2	
		101	0 0010	<< #3	
		1000	0 0010	add 3	1 cycle
	1	0000	0010	<< #4	
	10	0000	010	<< #5	
	100	0000	10	<< #6	
	1000	0001	0	<< #7	
	1011	0001	0	add 3	1 cycle
1	0110	0010		<< #8	
1	6	2			

BCD decoder

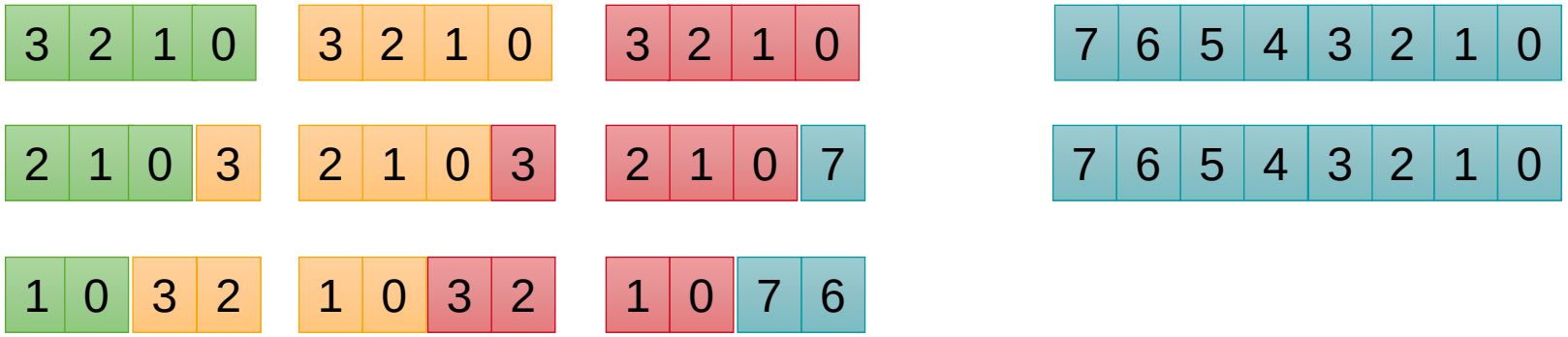
also possible with std_logic_vector
=> see moodle

```
architecture behavior of bcd is

begin
    process(i_slv_binary)
        -- temporarly variables
        variable huns : unsigned ( 3 downto 0) := (others => '0');
        variable tens : unsigned ( 3 downto 0) := (others => '0');
        variable ones : unsigned ( 3 downto 0) := (others => '0');
    begin
        -- reset all outputs
        huns := (others => '0');
        tens := (others => '0');
        ones := (others => '0');

        for i in 7 downto 0 loop

            if ( huns >= 5 ) then
                huns := huns + 3;
            end if;
            if ( tens >= 5 ) then
                tens := tens + 3;
            end if;
            if ( ones >= 5 ) then
                ones := ones + 3;
            end if;

            huns := huns(2 downto 0) & tens(3);
            tens := tens(2 downto 0) & ones(3);
            ones := ones(2 downto 0) & i_slv_binary(i);

            -- report "ones " & integer'image( to_integer(ones) );

        end loop;

        o_slv_decimal <=  std_logic_vector(huns) & std_logic_vector(tens) & std_logic_vector(ones);

    end process;
end behavior;
```

```
entity bcd is
--    GENERIC (
--    );
    port(
        i_slv_binary   :   in std_logic_vector( 7     downto 0);
        o_slv_decimal  :  out std_logic_vector( 3*4-1 downto 0)
    );
end bcd;
```

What is happening inside VHDL

```
huns := huns(2 downto 0) & tens(3);
tens := tens(2 downto 0) & ones(3);
ones := ones(2 downto 0) & i_slv_binary(i);
```



III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY


NEW

VHDL

III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY

Arrays

First attempt to use some more storage

- Up to now, we have handled „just“ 8 bit at once

- But let us assume, that we get every 2ns 10 bit from our ADC
- We would like to store some of these samples for further processing
  - e.g. averaging, some fancy algorithms etc.

- Note that arrays are usually implemented using
  - gates and flip-flops
  - not ROM's and RAM's

III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY

Defining our own type

VHDL allows defining your own type

- Today we define a global type, which has to be encapsulated in a package

```
library ieee;
use ieee.std_logic_1164.all;

package pkg is
    type slv8_array_t is array (natural range <>) of std_logic_vector(7 downto 0);
end package;

package body pkg is
end package body;

library ieee;
use ieee.std_logic_1164.all;

library work;
use work.pkg.all;
```

Unconstrained array definition
Must be constrained when we use it

Natural is positive integer

Use our own package

Library work is our current workspace.

Using arrays

Quite similar to std_logic_vector

```
entity MultiBitShift is
    port(
        i_sl_CLK     :  in  std_logic;                      -- clock
        i_sl_en      :  in  std_logic;                      -- enable
        i_slv_data   :  in  std_logic_vector(7 downto 0);   -- data input
        o_slv_shift  :  out slv8_array_t(7 downto 0)        -- last X samples
    );
end MultiBitShift;
```

```
architecture behavior of MultiBitShift is
    signal shift_reg : slv8_array_t(7 downto 0) := (others=>(others=>'0'));
```

Initialisation with 0

Arrays in gtwave

- Not supported by VCD file
- But we can change to **ghw** output of GHDL, there it is supported
  - The makefile, I‘ve provided has alredy this changes included.

III. Physikalisches Institut B
RWTH AACHEN UNIVERSITY

When to use which FPGA

Use the one, which fits your requests and money

How to select the proper FPGA

A incomplete guide

- Vendor
  - Is the vendor given by e.g. the collaboration
  - Does some of your collegues have experience with a certain vendor
  - Which license do we need to program the FPGA (is there a free license?)
- Interfaces, do we need some special interfaces
  - PCIe, PCI, High Speed, Memory etc.
- Speed
  - Which signal speed do we have to expect?
- Timing
  - What are the timing requests?
  - How many bits at which time?
- Amount of logic
  - What do we have to implement?
- PCB layout ?
  - How about the constrains from this?

+ many other constrains

How to select the proper FPGA?

- For sure, the largest FPGA would fullfill all our needs ….
  - But in most cases out of budget!

Select a FPGA, which is **slightly** larger than your estimate!

Let‘s have a look at XILINX

A vendor, which is quite common in HEP

45nm	28nm	20nm	16nm
SPARTAN 6	VIRTEX 7 KINTEX 7 ARTIX 7 SPARTAN 7	VIRTEX UltraSCALE KINTEX UltraSCALE	VIRTEX UltraSCALE+ KINTEX UltraSCALE+

Xilinx

Artix 7

Artix-7 FPGAs

30€

		Transceiver Optimization at the Lowest Cost and Highest DSP Bandwidth (1.0V, 0.95V, 0.9V)							
	Part Number	XC7A12T	XC7A15T	XC7A25T	XC7A35T	XC7A50T	XC7A75T	XC7A100T	XC7A200T
Logic Resources	Logic Cells	12,800	16,640	23,360	33,280	52,160	75,520	101,440	215,360
	Slices	2,000	2,600	3,650	5,200	8,150	11,800	15,850	33,650
	CLB Flip-Flops	16,000	20,800	29,200	41,600	65,200	94,400	126,800	269,200
Memory Resources	Maximum Distributed RAM (Kb)	171	200	313	400	600	892	1,188	2,888
	Block RAM/FIFO w/ ECC (36 Kb each)	20	25	45	50	75	105	135	365
	Total Block RAM (Kb)	720	900	1,620	1,800	2,700	3,780	4,860	13,140
Clock Resources	CMTs (1 MMCM + 1 PLL)	3	5	3	5	5	6	6	10
I/O Resources	Maximum Single-Ended I/O	150	250	150	250	250	300	300	500
	Maximum Differential I/O Pairs	72	120	72	120	120	144	144	240
Embedded Hard IP Resources	DSP Slices	40	45	80	90	120	180	240	740
	PCIe® Gen2[1]	1	1	1	1	1	1	1	1
	Analog Mixed Signal (AMS) / XADC	1	1	1	1	1	1	1	1
	Configuration AES / HMAC Blocks	1	1	1	1	1	1	1	1
	GTP Transceivers (6.6 Gb/s Max Rate)[2]	2	4	4	4	4	8	8	16
Speed Grades	Commercial Temp (C)	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2
	Extended Temp (E)	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3	-2L, -3
	Industrial Temp (I)	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L	-1, -2, -1L

	Package[3], [4]	Dimensions (mm)	Ball Pitch (mm)	**Available User I/O**: 3.3V SelectIO™ HR I/O (GTP Transceivers)							
	CPG236	10 x 10	0.5		106 (2)		106 (2)	106 (2)			
	CPG238	10 x 10	0.5	112 (2)		112 (2)					
	CSG324	15 x 15	0.8		210 (0)		210 (0)	210 (0)	210 (0)	210 (0)	
	CSG325	15 x 15	0.8	150 (2)	150 (4)	150 (4)	150 (4)	150 (4)			
	FTG256	17 x 17	1.0		170 (0)		170 (0)	170 (0)	170 (0)	170 (0)	
	SBG484	19 x 19	0.8								285 (4)
Footprint Compatible	FGG484[5]	23 x 23	1.0		250 (4)		250 (4)	250 (4)	285 (4)	285 (4)	
	FBG484[5]	23 x 23	1.0								285 (4)
Footprint Compatible	FGG676[6]	27 x 27	1.0						300 (8)	300 (8)	
	FBG676[6]	27 x 27	1.0								400 (8)
	FFG1156	35 x 35	1.0								500 (16)

Notes:

Xilinx

about 1000,00 € per piece

UltraScale +

	Device Name	KU3P	KU5P	KU9P	KU11P	KU13P	KU15P	KU19P
Logic	System Logic Cells (K)	356	475	600	653	747	1,143	1,843
	CLB Flip-Flops (K)	325	434	548	597	683	1,045	1,685
	CLB LUTs (K)	163	217	274	299	341	523	842
Memory	Max. Distributed RAM (Mb)	4.7	6.1	8.8	9.1	11.3	9.8	11.6
	Total Block RAM (Mb)	12.7	16.9	32.1	21.1	26.2	34.6	60.8
	UltraRAM (Mb)	13.5	18.0	0	22.5	31.5	36.0	81.0
Clocking	Clock Mgmt Tiles (CMTs)	4	4	4	8	4	11	9
Integrated IP	DSP Slices	1,368	1,824	2,520	2,928	3,528	1,968	1,080
	PCIE4 (PCIe® Gen3 x16)	1	1	0	4	0	5	0
	PCIE4C (PCIe® Gen3 x16 / Gen4 x8 /CCIX)	0	0	0	0	0	0	3
	150G Interlaken	0	0	0	1	0	4	0
	100G Ethernet w/ KR4 RS-FEC	0	1	0	2	0	4	1
I/O	Max. Single-Ended HD I/Os	96	96	96	96	96	96	72
	Max. Single-Ended HP I/Os	208	208	208	416	208	572	468
	GTH 16.3Gb/s Transceivers	0	0	28	32	28	44	0
	GTY 32.75Gb/s Transceivers	16	16	0	20	0	32	32
Speed Grades	Extended[1]	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3
	Industrial	-1 -1L -2	-1 -1L -2	-1 -1L -2	-1 -1L -2	-1 -1L -2	-1 -1L -2	-1 -1L -2
	Footprint[2, 3] / Dimensions (mm)	HD I/O, HP I/O, GTH 16.3Gb/s, GTY 32.75Gb/s						
Footprint compatible with 20nm UltraScale Devices with same footprint identifier	B784[4] / 23x23[5]	96, 208, 0, 16	96, 208, 0, 16					
	A676[4] / 27x27	48, 208, 0, 16	48, 208, 0, 16					
	B676 / 27x27	72, 208, 0, 16	72, 208, 0, 16					
	D900[4] / 31x31	96, 208, 0, 16	96, 208, 0, 16		96, 312, 16, 0			
	E900 / 31x31			96, 208, 28, 0		96, 208, 28, 0		
	A1156[4] / 35x35				48, 416, 20, 8		48, 468, 20, 8	
	E1517 / 40x40				96, 416, 32, 20		96, 416, 32, 24	
	A1760 / 42.5x42.5						96, 416, 44, 32	
	E1760 / 42.5x42.5						96, 572, 32, 24	
	J1760 / 42.5x42.5							72, 468, 0, 32
	B2104 / 47.5x47.5							72, 468, 0, 32

Kintex® UltraScale+™ FPGAs

ZYNQ

Zynq®-7000 SoC Family

		Cost-Optimized Devices						Mid-Range Devices			
	Device Name	Z-7007S	Z-7012S	Z-7014S	Z-7010	Z-7015	Z-7020	Z-7030	Z-7035	Z-7045	Z-7100
	Part Number	XC7Z007S	XC7Z012S	XC7Z014S	XC7Z010	XC7Z015	XC7Z020	XC7Z030	XC7Z035	XC7Z045	XC7Z100
Processing System (PS)	Processor Core	Single-Core Arm® Cortex®A9 MPCore™ Up to 766MHz			Dual-Core Arm Cortex-A9 MPCore Up to 866MHz			Dual-Core Arm Cortex-A9 MPCore Up to 1GHz[1]			
	Processor Extensions	NEON™ SIMD Engine and Single/Double Precision Floating Point Unit per processor									
	L1 Cache	32KB Instruction, 32KB Data per processor									
	L2 Cache	512KB									
	On-Chip Memory	256KB									
	External Memory Support[2]	DDR3, DDR3L, DDR2, LPDDR2									
	External Static Memory Support[2]	2x Quad-SPI, NAND, NOR									
	DMA Channels	8 (4 dedicated to PL)									
	Peripherals	2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO									
	Peripherals w/ built-in DMA[2]	2x USB 2.0 (OTG), 2x Tri-mode Gigabit Ethernet, 2x SD/SDIO									
	Security[3]	RSA Authentication of First Stage Boot Loader, AES and SHA 256b Decryption and Authentication for Secure Boot									
	Processing System to Programmable Logic Interface Ports (Primary Interfaces & Interrupts Only)	2x AXI 32b Master, 2x AXI 32b Slave 4x AXI 64b/32b Memory AXI 64b ACP 16 Interrupts									
Programmable Logic (PL)	7 Series PL Equivalent	Artix®-7	Artix-7	Artix-7	Artix-7	Artix-7	Artix-7	Kintex®-7	Kintex-7	Kintex-7	Kintex-7
	Logic Cells	23K	55K	65K	28K	74K	85K	125K	275K	350K	444K
	Look-Up Tables (LUTs)	14,400	34,400	40,600	17,600	46,200	53,200	78,600	171,900	218,600	277,400
	Flip-Flops	28,800	68,800	81,200	35,200	92,400	106,400	157,200	343,800	437,200	554,800
	Total Block RAM (# 36Kb Blocks)	1.8Mb (50)	2.5Mb (72)	3.8Mb (107)	2.1Mb (60)	3.3Mb (95)	4.9Mb (140)	9.3Mb (265)	17.6Mb (500)	19.2Mb (545)	26.5Mb (755)
	DSP Slices	66	120	170	80	160	220	400	900	900	2,020
	PCI Express®	—	Gen2 x4	—	—	Gen2 x4	—	Gen2 x4	Gen2 x8	Gen2 x8	Gen2 x8
	Analog Mixed Signal (AMS) / XADC[2]	2x 12 bit, MSPS ADCs with up to 17 Differential Inputs									
	Security[3]	AES & SHA 256b Decryption & Authentication for Secure Programmable Logic Config									
	Speed Grades: Commercial	-1			-1			-1			-1
	Speed Grades: Extended	-2			-2,-3			-2,-3			-2
	Speed Grades: Industrial	-1, -2			-1, -2, -1L			-1, -2, -2L			-1, -2, -2L

Notes:
1. 1 GHz processor frequency is available only for -3 speed grades for devices in flip-chip packages. Please see the data sheet for more details.
2. Z-7007S and Z-7010 in CLG225 have restrictions on PS peripherals, memory interfaces, and I/Os. Please refer to the Technical Reference Manual for more details.
3. Security block is shared by the Processing System and the Programmable Logic.

Lattice

MachXO2

MachXO2 & LatticeXP2 Series - Bridging and I/O Expansion FPGAs

Features			MachXO2™									LatticeXP2™				
Device			LCMXO2-256	LCMXO2-640	LCMXO2-640U	LCMXO2-1200	LCMXO2-1200U	LCMXO2-2000	LCMXO2-2000U	LCMXO2-4000	LCMXO2-7000	LFXP2-5E	LFXP2-8E	LFXP2-17E	LFXP2-30E	LFXP2-40E
LUTs			256	640	640	1280	1280	2112	2112	4320	6864	5 k	8 k	17 k	29 k	40 k
EBR SRAM	# of Blocks		0	2	7	7	8	8	10	10	26	9	12	15	21	48
kbits			0	18	64	64	74	74	92	92	240	166	221	276	387	885
Distrib. RAM	kbits		2	5	5	10	10	16	16	34	54	10	18	35	56	83
UFM	kbits		0	24	64	64	80	80	96	96	256					
sysDSP™ Blocks	18x18 Blocks											3	4	5	7	8
	Multipliers											12	16	20	28	32
PLL + DLL					1+2				2+2			2+0		4+0		
DDR Support					DDR 266, DDR2 266, LPDDR266							DDR/2 400				
Configuration Memory			Internal Flash									Internal Flash				
Dual Boot[4]			✓									✓				
Bit-stream Encryption												✓				
Embedded Function Blocks			I²C (2), SPI (1), Timer (1)													
Core Vcc	1.2 V		ZE & HE									✓				
	1.8 - 3.3 V															
	2.5 - 3.3 V		HC									HC				
Temp.	C		✓									✓				
	I		✓									✓				
	AEC-Q100												✓			
0.4 mm Spacing																
WLCSP	25	2.5 x 2.5 mm				18			18							
	49[2]	3.2 x 3.2 mm						38								
ucBGA	64	4 x 4 mm	44													

Summary

The selection of the FPGA depends a lot on the application!

Exercises

Counter and BCD

- In order to test the BCD, we have simply put in a for loop with the values
- Change the for loop to a 8 bit counter!
  - Use the BCD module
  - Use the counter module

Practise connecting and using multiple modules!

Multi Bit Shift Register

- Same as the shift register (lecture 2)
- BUT: instead of shifting a 1 in a circle – we want to store the latest 8 input values