Design and Verification of a Software Defined radio platform using Modelsim and Altera FPGA.

Barun Sharma,P.Nagaraju,Krishnamurthy Vaidyanathan.

Abstract-*:* The Altera Cyclone family of FPGA provides the ability to perform run time reconfiguration which is known as Dynamic Reconfiguration. Current technology allows designers to implement complete embedded computing systems on a single FPGA. Using an FPGA as the implementation platform introduces greater flexibility into the design process and allows a new approach to embedded system design. Since there is no cost to reprogramming an FPGA, system performance can be measured on-chip in the runtime environment and the system's architecture can be altered based on an evaluation of the data to meet design requirements.This paper concentrates on designing a reconfigurable platform which consists of reconfigurable coprocessors that can be used in various applications. The architecture that we have developed for the generic reconfigurable coprocessor is well suited for multiple application domains. It ties in to the sequential RISC processor based C programming framework where in the coprocessor performs specific functions, replacing a C function call. The methodology adopted to develop reconfigurable platform can be implemented for performing mathematical operation such as addition, subtraction and to perform complex operation such as Autocorrelation, it can also be used in Image Processing, Cryptography etc. As an example to illustrate this methodology reconfigurable coprocessor consisting of wrapper module along with controller module are designed to control external device like LCD. The architecture of the proposed methodology presented in this paper were prototyped using a Cyclone IV Starter Board (DEO Nano), which is based on Nios II Embedded Evaluation Kit.

Keywords*:* ALTERA FPGA,MODELSIM,LCD.

I.Introduction.

Embedded computing systems typically comprise both processors and dedicated logic modules to meet design specifications that include performance, area, power, and cost constraints. This has led researchers to investigate numerous issues that arise from Hardware/Software co design. Although older systems combined fixed processors and integrated circuits, current technology allows designers to combine both processors and dedicated logic to implement complete embedded computing systems as Systems-on-Chip (SoC) using either ASIC or FPGA platforms[1].

The twentieth century saw the explosion of hardware defined radio (HDR) as a means of communicating all forms of audible; visual, and machine-generated information over vast distances. Most radios are hardware defined with little or no software control; they are fixed in function for mostly consumer items for broadcast reception. They have a short life and are designed to be discarded and replaced [2].

- Barun Sharma currently pursuing Masters of Technology in Digital Communication in RV College of Engineering, Bangalore ,560059.Phone number-9019121052,Email id-Barunshrm@gmail.com.
- P.Nagaraju ,Associate Professor in Telecommunication Department in RV College of Engineering,Bangalore,560059.Email id-nagarajup@rvce.edu.in.
- Krishnamurthy Vaidyanathan , CEO of EI LABS INDIA PVT LTD.

Communications devices designed with application-specific integrated circuit (ASIC) technology suffer from one very significant limitation—the integrated circuits are not programmable. Therefore, deploying a new algorithm or an updated standard requires new hardware.

The solution of above big problems can be solved by using the software defined radio (SDR), which comprised of both software and hardware, it use a reprogrammable ability of field programmable gate array (FPGA) or any possessor to built an open architecture.

In this paper we prepare a design methodology to develop reconfigurable SDR platform using FPGA. The remainder of the paper can be categorized of following sections which are described as follows. Section 2 gives overview of SDR and basic definition related to reconfigurable platforms .Section 3 gives the descriptive view of various platforms that can be used to design Reconfigurable platform . Section 4 deals with the software packages required to implement the proposed flow of work. Section 5 contains the hardware requirement for the proposed work. Section 6 deals with general proposed methodology with the two examples to illustrate the importance of the design that has been proposed. Section 7 deals with results obtained and implementation of the proposed methodology using these software packages .Section 8 deals with conclusion.

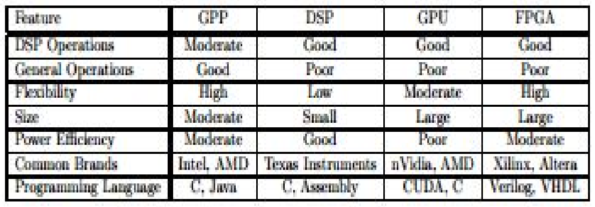
2. SDR Overview.

Software Defined Radio is a collection of hardware and software technologies that enable reconfigurable system architectures for wireless networks and user terminals. SDR provides an efficient and comparatively in-expensive solution to the problem of building multi-mode, multi-band, multi-functional wireless devices that can be enhanced using software upgrades. It is applicable across a wide range of areas within the wireless industry. Reconfigurability for an SDR can be "static" or "dynamic." Static reconfigurability is the ability to reconfigure equipment capability 'off-line' (e.g., at supply), or via a smart card. Pseudo-static reconfigurability is the ability to reconfigure equipment capability over-the-air (OTA), while at the same time leaving it un-changed during a call. Dynamic reconfigurability, on the other hand, is the ability to auto-re-configure equipment capability during a call. Fundamental differences exist in the manner in which systems using SDR technology are implemented in the commercial, government, and military wireless-market segments.

3. Platform defined for SDR.

Software-defined radios have been employed on numerous platforms, including general purpose microprocessors (GPP), digital signal processors (DSP), and graphics processing units (GPU). General purpose microprocessors, such as the Intel and AMD devices commonly found in personal computers, are not specialized for any particular application. Therefore, they are very flexible[2]. However, SDR systems using GPPs are often wasteful since these processors are designed for speed and generality rather than power efficiency or mathematical operations. Digital signal processors solve these two problems. DSPs, such as those manufactured by Texas Instruments, are specialized for performing mathematical operations and typically contain less hardware, increasing power efficiency. On the other hand, their narrow focus makes them potentially slow for other applications. Finally, graphics processing units employ massively parallel architectures that are optimized for vector manipulations and other graphical operations. Such parallel designs are well-suited for signal processing, but GPUs are still relatively difficult to program and they are extremely power hungry. A fourth option is the field-programmable gate array (FPGA). FPGAs are reconfigurable logic devices that enable highly parallel implementations of digital signal processing algorithms. They are becoming increasingly power efficient while simultaneously including specialized hardware for some DSP operations. In additional, they can be reprogrammed remotely. Due to these features, FPGAs make an ideal software-defined radio platform. Table 1 shows the comparison of microelectronics platform for SDR[11].

Table 1- Comparison of different SDR platforms.



Feature	GPP	DSP	GPU	FPGA
DSP Operations	Moderate	Good	Good	Good
General Operations	Good	Poor	Poor	Poor
Flexibility	High	Low	Moderate	High
Size	Moderate	Small	Large	Large
Power Efficiency	Moderate	Good	Poor	Moderate
Common Brands	Intel, AMD	Texas Instruments	nVidia, AMD	Xilinx, Altera
Programming Language	C, Java	C, Assembly	CUDA, C	Verilog, VHDL

4.Software Packages Required.

Basic software tools required to accomplish this project is as follows:

Modelsim- It is a verification and simulation tool for VHDL, Verilog, System Verilog, and mixed language designs.

Quartus 12.0-The software package by Altera enables the user to compile and synthesize the code for NIOS processor.

Qsys- Altera's Qsys system integration tool, which is used to design digital hardware systems that contain components such as processors, memories, input/output interfaces, timers, and the like. The Qsys tool allows a designer to choose the components that are desired in the system by selecting these components in a graphical user interface. It then automatically generates the hardware system that connects all of the components together.

Altera Monitor Programme: The designed circuit has to be downloaded into the FPGA device on a DE-series board. This can be done by using the Programmer Tool in the Quartus II software. However, it can be done using a simpler approach by using the Altera Monitor Program, which provides a simple means for downloading the circuit into the FPGA as well as running the application programs. Programs can be written in C/C++ .

5. Hardware kit required.

Basic hardware kit required to carry out the implementation is ALTERA DE series FPGA boards[18]. As there are many series of boards that is specified in Table 2 . Among these boards DEO-NANO board is used with specific family design as CYCLONE IVE EP42CE22F17C6[14].

Table 2- DE series FPGA device names.

Board	Device Name
DE0	Cyclone III EP3C16F484C6
DE0-Nano	Cyclone IVE EP4CE22F17C6
DE1	Cyclone II EP2C20F484C7
DE2	Cyclone II EP2C35F672C6
DE2-70	Cyclone II EP2C70F896C6
DE2-115	Cyclone IVE EP4CE115F29C7

The DE0-Nano board introduces a compact-sized PGA development platform suited for to a wide range of portable design projects, such as robots and mobile projects. The DE0-Nano is ideal for use with embedded soft processors—it features a powerful Altera Cyclone IV FPGA (with 22,320 logic elements), 32 MB of SDRAM, 2 Kb EEPROM, and a 16 Mb serial configuration memory device. For connecting to real-world sensors the DE0-Nano includes a National Semiconductor 8-channel 12-bit A/D converter, and it also features an Analog Devices 13-bit, 3-axis accelerometer device. The DE0-Nano board includes a built-in USB Blaster for FPGA programming, and the board can be powered either from this USB port or by an external power source. The board includes expansion headers that can be used to attach various Terasic daughter cards or other devices, such as motors and actuators. Inputs and outputs include 2 pushbuttons, 8 user LEDs and a set of 4 dip-switches[10].Fig 1 gives the descriptive view of Nano board.

Fig 1- Deo Nano board

6. Proposed Methodology.

The methodology can be broadly divided in two categories that are hardware system definition and software system definition .Hardware system synthesis can be implemented and designed using software package Qsys and Quartus . On the other hand software system synthesis can be implemented using Altera Monitor Programme .

Design of the specified methodology contains one processor with storage memory along with buffers and parallel to serial converter. For verification and testing external device is required. Fig 2 gives the descriptive view of predefined methodology.

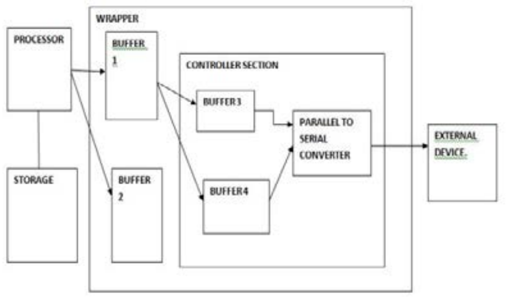


Fig 2 – Defined Methodology

In Fig 2 defined processor can be any processor but according to user point of view and due to fast processing speed preferred processor is NIOS processor. This methodology can be used to carry out any mathematical operation like calculating Autocorrelation, simple binary operation can be performed easily and we can reconfigure the defined system as per our requirements. It can also be used in cryptography and many more technologies. In this section focus will be more on illustrating the use of methodology with two suitable examples.

6.1 Performing mathematical operation.

As in the methodology defined there is one processor defined with one storage memory. Let us take a suitable example to perform Autocorrelation operation using this. Suppose buffer 1 defined in the above consist of one matrix of (2X2) and buffer 2 consist of other matrix of (2X2). Now to perform autocorrelation one has to perform matrix multiplication.

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \qquad \text{................(1)}$$

Information from buffer 1 is passed to buffer 3 and information from buffer 2 to buffer 4. Buffer 3 and buffer 4 forwards the existing matrix in them to multiplier which is defined as parallel to serial converter in our case, and after multiplication the required output which consist of matrix C as defined in equation 1 is displayed on external device that can be LCD which is compatible to display defined result or one can choose any suitable display source.

6.2 Cryptography.

The defined methodology can also be used in cryptography and can be used to transfer secured data in terms of any media types .Let us take an example of transferring a secured image. Consider buffer 3 and buffer 4 consist of two images, in which buffer 3 consist of real image and buffer 4 with duplicate. As we have defined our structure as reconfigurable on the software side so we are removing the buffer 1 and buffer 2 and connect buffer3 and buffer 4 directly to processor such as NIOS. At the encoder side we multiply these two images and transfer and at decoder side these two are decoded and original image can be extracted. Image multiplication can be done as shown in Fig 3.As we can see in the figure defined that two pixel values a(1,1) and a(1,2) multiplies with b(1,2) and b(2,2) to achieve the resultant pixel value . Similarly all the resultant image can be obtained by multiplication pixel values in matrix form.

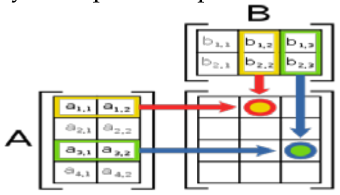


Fig 3 – Image Multiplication Example

In this case memory are used to store pixel values which are stored in the code which can be in C or C++ and that code can be dumped in to the storage memory and NIOS processor instantiate the code and pixel values when the system is executed .

The example gives the simple theoretical view of our proposed design and how the memories are utilized and how the data are dumped in to hardware kit. The proposed methodology varies according to the application.

7. Implementation and results.

To explain the proposed methodology reconfigurable platform has been designed to control LCD device as an example. Results will vary according to the application.

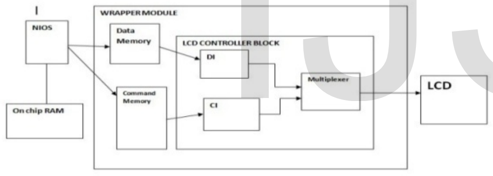


Fig 4 – Reconfigurable platform to control LCD.

In the proposed work LCD display that has been used is of company ODM. The defined modules in this are data memory (buffer 1), command memory (buffer 2), data interpreter (buffer 3), command interpreter (buffer 4), multiplexer (parallel to serial converter) and a external device LCD. Fig 4 gives the descriptive view of proposed design for LCD (character display) control.

Proposed work can broadly be classified systematically as follows:

7.1 Simulation and verification using Modelsim.

The modules defined in this application are data memory, command memory, data interpreter, command interpreter. These modules are coded and executed separately to create a design flow. Data memory consists of data to be displayed on the LCD and command memory consists of commands that are to be executed to initialize the LCD. Both Command and Data memory are 64x8 user-defined modules with 64 address lines each of 8 bit width. After coding each module separately, a wrapper module is created where each defined module is instantiated and inter-module connections are made. This wrapper module is then simulated on ModelSim, in which desired inputs are forced and outputs are observed. The output waveforms thus obtained are verified manually.

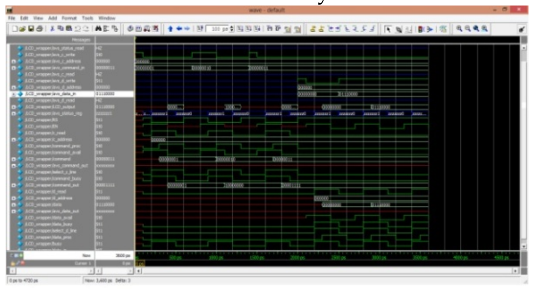


Fig 5- Wrapper module simulation

7.2 Compilation and simulation using Quartus.

The System on Programmable Chip building tool Qsys from Altera is used to implement the system. The Data Interpreter and Command Interpreter modules, which are cascaded together along with the Data Multiplexer as the LCD Controller block, is defined as an Avalon Memory Mapped Slave. The architecture of this Controller block includes a Status Register in which the LSB signifies 'Busy' status of this module. The Data Memory and Command Memory blocks are also defined as Avalon Memory Mapped Slaves since these modules are directly connected to the Avalon bus, and thus to the NIOS II processor. This definition is done by creating a "New Component" in Qsys and ensuring that the signals and registers defined in the Verilog code match their suitable roles in the system.

Once these modules have been defined as Avalon MM Slaves, the other components of the system are included viz. the NIOS II processor, On-Chip RAM (8K), JTAG UART and a General Purpose I/O controller along with the Reconfigurable platform components. The necessary clock connections, Avalon MM Master to Avalon MM Slave connections, Interrupt vector address and interrupt network connections and export of conduits to the surface-level so that internal connections may be customized are then made. Once the connections are made in Qsys, base addresses must be assigned to the memory mapped slaves, which is done by choosing the "Assign Base address" option.

After the system thus created above is saved, the .qsys, .sopcinfo, .qip files are generated by the software. These files are to be included in the Quartus project.

In Altera Quartus 12.0, the defined modules are included in the project directory along with the .qip file generated in the previous step. After this step, the top level Verilog module must be written. This will contain declaration of all signals that are input to or output from the FPGA along with the instantiation of the system designed in Qsys. The conduits in the system must be connected to their respective pins or other modules, as the case may be. Once this top level module has been saved and included in the project directory, it must be compiled. After compilation RTL logic is obtained which gives the descriptive view of our code written in Verilog which is as follows:

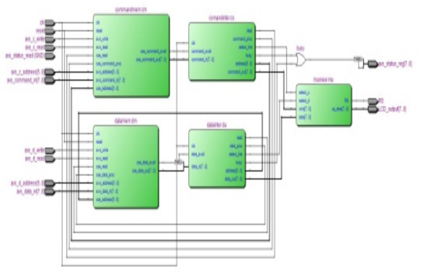


Fig 6- RTL view of wrapper.

Fig 7 gives the descriptive view of the qsys file generated after compilation of the top level design which is .sof file which contain other files also like .qip,.sopcinfo etc .
Modules generated by qsys are of two types:
Verilog Module.
VHDL module.

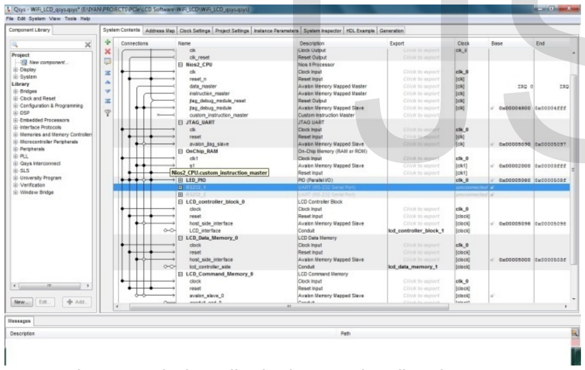
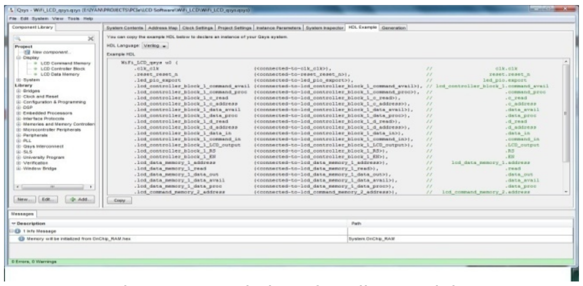


Fig 7 – Window displaying Design flow in Qsys.

The Qsys tool generates a Verilog module that defines the desired Nios II system. Normally, the Nios II module generated by the Qsys tool is likely to be a part of a larger design. However, in the case of our simple example there is no other circuitry needed. All we need to do is instantiate the Nios II system in our top-level Verilog or VHDL module, and connect inputs and outputs of the parallel I/O ports, as well as the clock and reset inputs, to the appropriate pins on the FPGA device. Fig 8 gives the information received from verilog module.



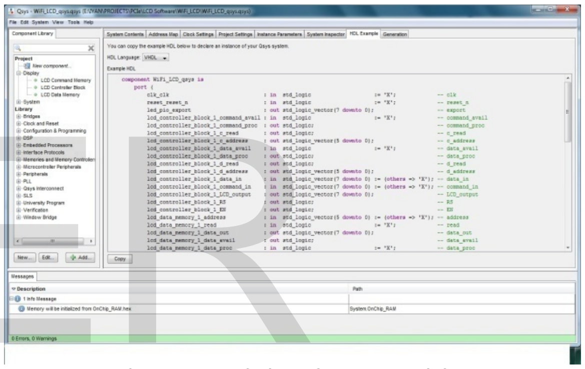
You can copy the example HDL below to declare an instance of your Qsys system.

HDL Language: Verilog

```
    WiFi_LCD_qsys u0 (
        .clk_clk                              (<connected-to-clk_clk>),                              //                     clk.clk
        .reset_reset_n                        (<connected-to-reset_reset_n>),                        //                   reset.reset_n
        .led_pio_export                       (<connected-to-led_pio_export>),                       //                 led_pio.export
        .lcd_controller_block_1_command_avail (<connected-to-lcd_controller_block_1_command_avail>), // lcd_controller_block_1.command_avail
        .lcd_controller_block_1_command_proc  (<connected-to-lcd_controller_block_1_command_proc>),  //                        .command_proc
        .lcd_controller_block_1_c_read        (<connected-to-lcd_controller_block_1_c_read>),        //                        .c_read
        .lcd_controller_block_1_c_address     (<connected-to-lcd_controller_block_1_c_address>),     //                        .c_address
        .lcd_controller_block_1_data_avail    (<connected-to-lcd_controller_block_1_data_avail>),    //                        .data_avail
        .lcd_controller_block_1_data_proc     (<connected-to-lcd_controller_block_1_data_proc>),     //                        .data_proc
        .lcd_controller_block_1_d_read        (<connected-to-lcd_controller_block_1_d_read>),        //                        .d_read
        .lcd_controller_block_1_d_address     (<connected-to-lcd_controller_block_1_d_address>),     //                        .d_address
        .lcd_controller_block_1_data_in       (<connected-to-lcd_controller_block_1_data_in>),       //                        .data_in
        .lcd_controller_block_1_command_in    (<connected-to-lcd_controller_block_1_command_in>),    //                        .command_in
        .lcd_controller_block_1_LCD_output    (<connected-to-lcd_controller_block_1_LCD_output>),    //                        .LCD_output
        .lcd_controller_block_1_RS            (<connected-to-lcd_controller_block_1_RS>),            //                        .RS
        .lcd_controller_block_1_EN            (<connected-to-lcd_controller_block_1_EN>),            //                        .EN
        .lcd_data_memory_1_address            (<connected-to-lcd_data_memory_1_address>),            //      lcd_data_memory_1.address
        .lcd_data_memory_1_read               (<connected-to-lcd_data_memory_1_read>),               //                        .read
        .lcd_data_memory_1_data_out           (<connected-to-lcd_data_memory_1_data_out>),           //                        .data_out
        .lcd_data_memory_1_data_avail         (<connected-to-lcd_data_memory_1_data_avail>),         //                        .data_avail
        .lcd_data_memory_1_data_proc          (<connected-to-lcd_data_memory_1_data_proc>),          //                        .data_proc
        .lcd_command_memory_2_address         (<connected-to-lcd_command_memory_2_address>),         //   lcd_command_memory_2.address
```

Fig 8 –Instantiation of Verilog module

As we know Qsys tool always generates Verilog modules, which can then be used in designs specified using either Verilog or VHDL languages. Fig 9 shows a top-level VHDL module that instantiates the Nios II system. After defining the hardware system next step is to define software system. For controlling LCD proper C code is written which consist of command memory instruction and description of the of the method involve to control LCD. In brief Register select (Rs), Enable (E) and Read write (R/W) instruction is used to control LCD.



You can copy the example HDL below to declare an instance of your Qsys system.

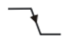
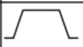
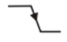
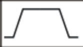
HDL Language: VHDL

```
    component WiFi_LCD_qsys is
        port (
            clk_clk                              : in  std_logic                    := 'X';             -- clk
            reset_reset_n                        : in  std_logic                    := 'X';             -- reset_n
            led_pio_export                       : out std_logic_vector(7 downto 0);                    -- export
            lcd_controller_block_1_command_avail : in  std_logic                    := 'X';             -- command_avail
            lcd_controller_block_1_command_proc  : out std_logic;                                       -- command_proc
            lcd_controller_block_1_c_read        : out std_logic;                                       -- c_read
            lcd_controller_block_1_c_address     : out std_logic_vector(5 downto 0);                    -- c_address
            lcd_controller_block_1_data_avail    : in  std_logic                    := 'X';             -- data_avail
            lcd_controller_block_1_data_proc     : out std_logic;                                       -- data_proc
            lcd_controller_block_1_d_read        : out std_logic;                                       -- d_read
            lcd_controller_block_1_d_address     : out std_logic_vector(5 downto 0);                    -- d_address
            lcd_controller_block_1_data_in       : in  std_logic_vector(7 downto 0) := (others => 'X'); -- data_in
            lcd_controller_block_1_command_in    : in  std_logic_vector(7 downto 0) := (others => 'X'); -- command_in
            lcd_controller_block_1_LCD_output    : out std_logic_vector(7 downto 0);                    -- LCD_output
            lcd_controller_block_1_RS            : out std_logic;                                       -- RS
            lcd_controller_block_1_EN            : out std_logic;                                       -- EN
            lcd_data_memory_1_address            : in  std_logic_vector(5 downto 0) := (others => 'X'); -- address
            lcd_data_memory_1_read               : in  std_logic                    := 'X';             -- read
            lcd_data_memory_1_data_out           : out std_logic_vector(7 downto 0);                    -- data_out
            lcd_data_memory_1_data_avail         : out std_logic;                                       -- data_avail
            lcd_data_memory_1_data_proc          : in  std_logic                    := 'X';             -- data_proc
```

Fig 9- Instantiation of VHDL module.

The descriptive way of running LCD is shown in Table 3 which is coded in C ,Wrapper module in verilog and for verification purpose in Modelsim also. Altera Monitor Programme is used to dump the C file we obtain after converting .sopcinfo file from Qsys to .elf file which contains C code.

Table 3 – Relation between the operation and combination of RS and R/W.

RS	R/W	E	OPERATION
0	0		Write instruction code
0	1		Read busy flag and address counter
1	0		Write data
1	1		Read data

NIOS processor defined play an important role by connecting on chip ram with specific design and the overall design is controlled by NIOS processor. To check the output on Deo Nano board which consists of Altera FPGA with cyclone IV device pin assignment is done in Quartus package which gives the description that outputs are assigned to particular GPIO pins. Descriptive view of pin assignment in Quartus 12.0 is shown in Fig 10.

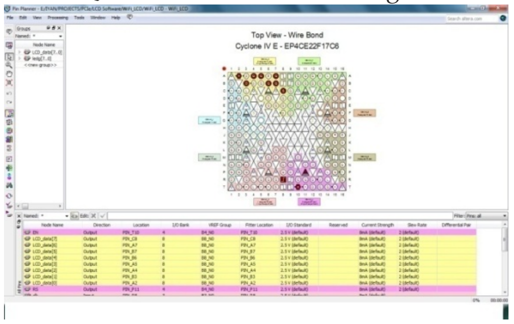


Fig 10 – Descriptive View of pin Assignment in Quartus .

To check the flow design and output in a defined way and to check the output waveform Signal Tapper is used which gives the flow with description according to the clock pulse. All the modules are exported and Hardware kit is specified before running dumping the programme.

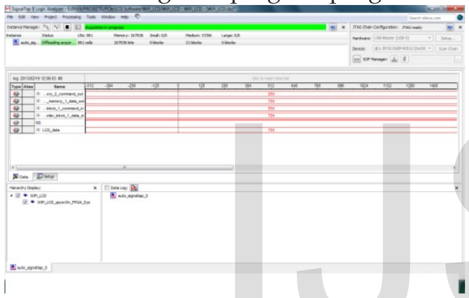


Fig 11 – Signal Tapper output.

After checking the logic in signal tapper .sof file is burn on to the Deo Nano board and LCD (20X4) is controlled using our reconfigurable platform as shown in Fig 12. One can display according to there requirements by changing the code without changing the hardware connection and system created. One can control any character LCD using this reconfigurable platform.

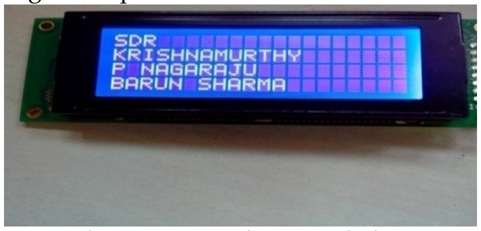


Figure 12 –LCD Character Display

8. Conclusion.

In this designed work, an efficient short cycle design flow has been proposed. With this design flow, the designer could implement his design models originally written as verilog codes or vhdl codes. As well as, this approach reduces the time required to complete the hardware implementation. It will give the beginner designer, for instance the student, a better and easy understanding of how different design parts behave using his/her written verilog or vhdl codes. The designed system consisting of coprocessors that can be used as a reconfigurable platform and one can control any external computing devices like LCD, DAC, WIFI module etc, using this prototype. Innovative design enhancements from engineering community, coupled with process improvements from FPGA vendors can play a crucial role in producing fast, small and efficient communications systems that can be used in commercial arenas.

References

[1] P. G. Burns, "Software Defined Radio for 3 G", Artech House – England, 2003.
[2] " Software-Defined Radio ", Wipro Technologies, 2002.
[3] Grégory, E.N, M.S and François. V, "Transaction Level Modeling of SCA Compliant Software Defined Radio Waveforms and Platforms PIM/PSM" Design, Automation & Test in EuropeConference & Exhibition, 16-20 April 2007.
[4] Hikmat N. Abdullah and Alejandro Valenzuela "A Joint Matlab/FPGA Design of AM Receiver for Teaching Purposes". EMNT2008 conference, Munich University of Technology,Germany, 2008.
[5] J.Mitola III, " Software radios: Survey,critical evaluation and future directions ," Aerospace and electronic systems magazines ,IEEE ,vol.8,no.4,pp.25-36,APR 1993.
[6] J.E.cooley,"A day in the life of rf spectrum,"Master'sthesis,Massachusetts Institute of Technology,2005.
[7] Alexander Loula, Open BTS installation and Configuration guide v0.1,2009-05-25.
[8] R.Vidano," Speakeasy II-an IPT approach to software programmable radio development ," MILCOM 97 Proceedings,vol.3,pp.1212-1215,Nov.1997.
[9] J,Chapin,V.Bose , " The Vanu Software Radio System," in 2002 Software Defined Radio Technical Conference,San Diego, 2002 .
[10] DEO Nano User Manual-World leading FPGA based products and design service.
[11] Steven J. Olivieri,"Modular FPGA based Software defined Radio for Cubestas", thesis,Worcester Polytechnic Institute,May 2011.
[12] Nios II Processors, Altera Corp., http://www.altera.com/nios.
[13] Avalon Bus Specification, Altera Corp., Reference Manual, Rev. 2.3, 2003.
[14] Quartus II Handbook, Volume 4: SOPC Builder, Altera Corp., Handbook, Rev. 5.4, 2005.
[15] J.-M. Daveau ang G. F. Marchioro, T. B. Ismail and A. A. Jerraya, "Protocol selection and interface generation for HW–SW codesign", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 5, No. 1, pp 136–4.
[16] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computer, Vol. 35, No. 1, pp 70–78, January 2002.
[17] D. Benitez, "A Quantitative Understanding of the performance of the Reconfigurable coprocessors",Proceedings 2002.
[18] Altera Product Catalog version 12.0.