E-ISSN 2581 - 7957
P-ISSN 2277 - 3916

CVR Journal of Science and Technology, Volume 19, December 2020
DOI: 10.32377/cvrjst 1906

Man Machine Interface Design using Hardware Programming for Performance Enhancement

Dr. T. Harinath[1] and Dr. K. Lal Kishore[2]
[1]Senior Manager, ECIL/Hyderabad, India
Email: harnath.t@gmail.com
[2]Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: lalkishorek@gmail.com

***Abstract*: Existing Embedded Systems are designed using a microprocessor. Design methodology of the embedded system depends on the architecture of the microprocessor used. Architecture of these embedded system designs have total dependence on the architecture of the microprocessor used. Study and analysis of various embedded system designs for the past thirty years revealed that there are several other design, development and manufacturing related issues with the microprocessor based designs like speed of processing, power consumption, excessive utilization of silicon, etc. Objective of this paper is to critically examine the design requirements for Man Machine Interface Logic, which is an embedded system and ascertain the issues in the existing designs. Hardware Programming, an innovative conceptual solution is proposed to overcome the issues observed by designing an embedded system without using any microprocessor. Design approach and design methodology based on Hardware Programming have been presented. Concepts of designing Man Machine Interface logic using Hardware Programming have been elaborated. Comparative Analysis is provided to ensure both the qualitative and quantitative improvement in performance. Performance is gauged in terms of both qualitative and quantitative parameters.**

***Index Terms:* Embedded System, Protocol Analysis, Architecture, Silicon Utilization, Latency, Man Machine Interface, Hardware Programming**

I. Introduction

A. Man Machine Interface

Man Machine Interface (MMI) logic is a subsystem used in an electronic system for incorporating control, status, system and storage operations. These operations facilitate user in selecting different sets of parameters among the available groups for carrying out different activities without any design changes, whether hardware or software [1], [2]. In other words, MMI logic is meant for incorporating programmability feature to an embedded system. MMI logic itself is an independent embedded system. Transfer of information to and from the system is through interfaces [3]. MMI has five types of interfaces. They are input interface, output interface, I/O interface, system interface and memory interface. Collection of control information is through input interface from input devices like keypad. Display of output status information is through output interface to output devices like Liquid Crystal Display (LCD), Light Emitting Diodes (LED). Bidirectional flow of information is through I/O interfaces using I/O devices like Universal Asynchronous Receiver Transmitter (UART). Non-volatile storage of control and configuration data is done non-volatile memory like Serial Peripheral Interface (SPI) Flash memory using storage interface. Passing on the user fed control data and collection of system related status information is through system bus. UART is used for system bus. SPI is used for memory interface. Design methodology for MMI design is similar to any other embedded design. The factors to be concentrated on while designing MMI with a microprocessor are: Processing capabilities, Memory requirements, Data transfer mechanism, Integration with rest of the system and Clock requirements [4], [5].

Study and Analysis of various embedded systems design for the past 30 years have lead to the categorization of embedded systems design into following types:

- Embedded System using Hard Processor
- Embedded System using Soft Processor
- Embedded System using Multi-Core Hard Processor
- Embedded System using Multiple Processors
- Embedded System using Micro Controller
- Embedded System using embedded hard processor on Field Programmable Gate Array (FPGA) and reconfigurable logic on the same FPGA
- Embedded System using Digital Signal Processor
- Combination of two or more of the above

The Trends being followed in latest embedded system design are:

- System Design with Multiple – Processors and Multi Core Processors [6], [7], [8].
- System Design with Micro Controllers and Peripherals on same chip
- Hardware/ Software Co-Design
- Heterogeneous Embedded System Design
- FPGA based Embedded System Design [9], [10], [11].
- Single Bit Processor based System [12].
- Distributed Co-operative Design Method [13], [14].
- Power – Delay Reduction, Memory Efficient Techniques
- Dynamic Profiling [15].

B. Issues in Existing Embedded Systems

Several issues have been observed in embedded system design with respect to the design, performance, reliability and productivity in industrial application perspective. A list of the issues is given below.

- Architectural Dependence
- Multiple Skill Set Requirement
- Sequential Execution of Instructions

E-ISSN 2581 - 7957
P-ISSN 2277 - 3916

CVR Journal of Science and Technology, Volume 19, December 2020
DOI: 10.32377/cvrjst 1906

- Speed of Processing
- Large Memory Requirement
- Additional Resources Requirement
- Stringent Clock Requirement
- Excessive Utilization of Silicon
- High Power Consumption
- Design Complexity
- Huge Cost of Development
- Production / Manufacturing related issues

C. *Possibilities of Replacing Microprocessor*

Design of an embedded system revolves around the architecture of the microprocessor used. Main activities of an embedded system are classified into three parts, namely receiving the data input, processing the data and sending the output data. Data input is collected from different input sources. Received data received is processed by the pre-programmed algorithms and results are stored in memory. These processed outputs are sent out to output devices. For example, in case of a software defined radio, base band signal is received through a Coder Decoder (CODEC), various algorithms like modulation, filtering, etc are run on the base band signal received. Samples of modulated and filtered Intermediate Frequency (IF) are sent out to the Radio Frequency (RF) circuitry for further processing. In order to effectively implement the above three functions, a microprocessor is used.

The functions performed by the microprocessor used in an embedded system are: Accessing and controlling peripherals, scheduling events, processing of interrupts, managing memory, receiving data, executing algorithm, processing data and sending data. Microprocessor plays a supporting role in an embedded system and an embedded system can be designed with same specifications using different types of microprocessors / microcontrollers / digital signal processors. Hence, the focus of embedded system must be on receiving the inputs, computation of the algorithms and transmitting the processed data to the external world. If all the activities being executed by the microprocessor are implemented with some other logic or mechanisms, Microprocessor can be successfully replaced or avoided in the design of an embedded system.

II. MMI Design using Hardware Programming based Design Methodology

Hardware Programming based design methodology is an innovative technique to design an embedded system without using any microprocessor, but realizing the logic as well as driver applications using hardware programming with Hardware Description Language (HDL) like VHDL or Verilog HDL [16], [17], [18], [19]. This design methodology is applied for MMI logic that can be used in software defined radio project. This can be used in any other systems as well, for example, networking applications, nuclear applications and so on, with very minor changes. Figure1 illustrates the application of MMI in a system. As presented in the diagram, P1 is input interface. P2a and P2b are output interfaces, P3 is I/O interface. P4 is system interface and P5 is Memory interface.

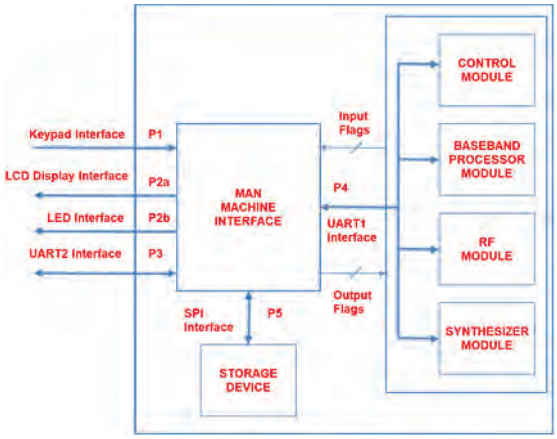

Figure1. Application of Man Machine Interface

A. *Identification of Activities*

Activities of MMI Logic to be designed are:

- ✓ To ascertain the data entered through Keypad
- ✓ To provide appropriate messages on LCD Display
- ✓ To restrict the user access through passwords management
- ✓ To collect the user configuration data through asynchronous serial interface
- ✓ To authenticate the data access through asynchronous serial interface
- ✓ To provide replies to user through asynchronous serial interface
- ✓ To store user configuration data in non-volatile memory through synchronous serial interface
- ✓ To configure the system based on the user configuration data through asynchronous serial interface as system bus
- ✓ To collect the status information of entire system using system bus
- ✓ To provide user friendly menu driven operations
- ✓ To facilitate for erasure or modification of the configuration data based on the user commands
- ✓ To prepare preset channels for user convenience (A preset channel is a group of parameters and this entire group can be selected with a single key stroke)

B. *Dividing System into Functional Modules*

In order to realize the functions identified above, logic of MMI module is divided into the following functional modules.

a) Clock generation logic
b) Keypad interface logic
c) LCD display interface logic
d) LED interface logic
e) Key processing logic
f) Serial peripheral interface logic
g) UART1 receiver logic
h) UART1 transmitter logic
i) UART1 driver logic
j) UART2 receiver logic
k) UART2 transmitter logic
l) UART2 driver logic
m) Passwords Management logic

Modules (g), (h) and (i) are integrated to form UART1 logic. Modules (j), (k) and (l) are integrated to form UART2 logic. MMI logic design using hardware programming is shown in Figure 2. As shown in the figure, modules are categorized into peripheral modules, algorithm modules and clock & control modules. All the modules are independent in working and do not require microprocessor for its working. Keypad Interface Module, UART1 Receiver and UART2 receiver are input peripheral modules, LCD interface, LED interface, UART1 Transmitter and UART2 transmitter modules are output peripheral modules. UART1 Driver, UART2 Driver module and Passwords Management Module are algorithm modules. SPI interface module is an I/O peripheral module. Clock generation module and Key processing logic work together as clock and control module. Key processing logic performs control operations (i.e. generating required control signals) and also does the processing of the keypad interface output signals.

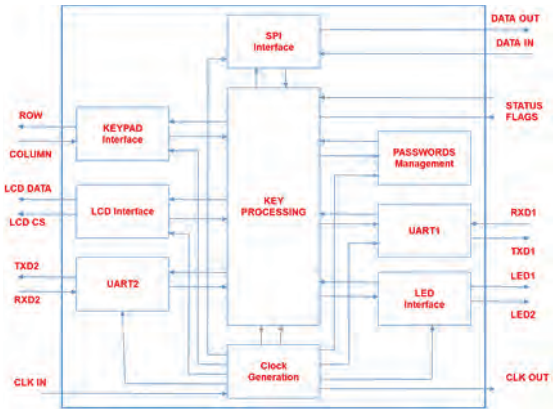

Figure 2. MMI Design using Hardware Programming

C. *Identifying the Events of Functional Modules*

Figure 3a through Figure 3h depict the events executed by various modules of MMI logic.

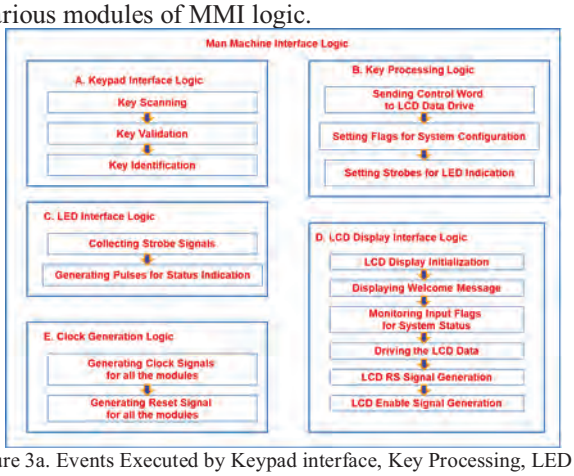

Figure 3a. Events Executed by Keypad interface, Key Processing, LED Interface, LCD Display Interface and Clock Generation Logics of MMI

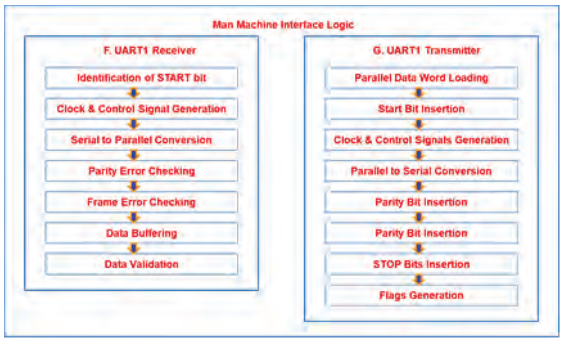

Figure 3b. Events Executed by UART1 Receiver & UART1 Transmitter Logics of MMI

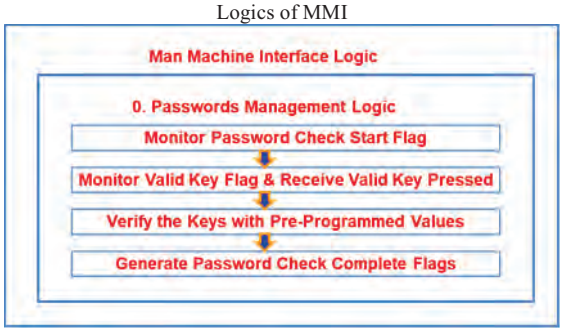

Figure 3c. Events Executed by Passwords Management

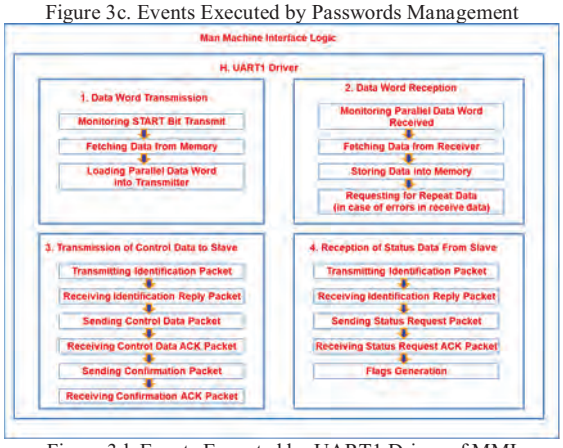

Figure 3d. Events Executed by UART1 Driver of MMI

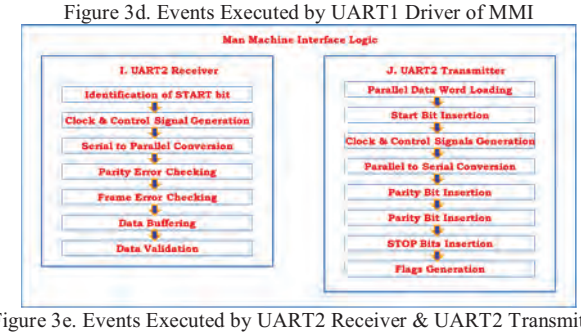

Figure 3e. Events Executed by UART2 Receiver & UART2 Transmitter Logics of MMI

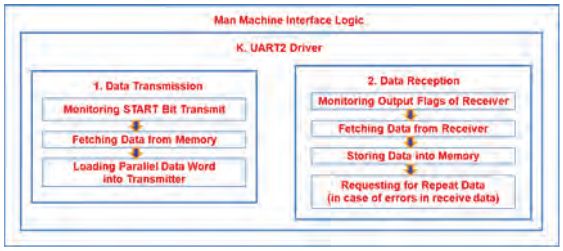

Figure 3f. Events Executed by UART2 Driver Logic of MMI

E-ISSN 2581 - 7957
P-ISSN 2277 - 3916

CVR Journal of Science and Technology, Volume 19, December 2020
DOI: 10.32377/cvrjst 1906

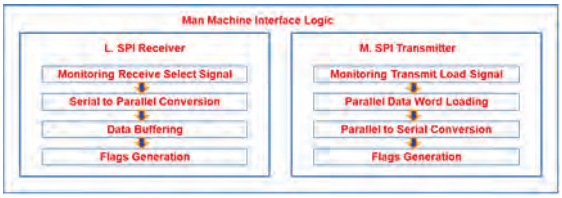

Figure 3g. Events Executed by SPI Receiver and SPI Transmitter Logics of MMI

Figure 3h. Events Executed by SPI Driver Logic of MMI

D. Analysis of Protocol

Protocol analysis is to be done for getting the particulars of sources & destinations of data, packet transfer information, and extraction & insertion of data in packet structures. MMI gets inputs from different sources. Key pressed information coming from keypad, Configuration data or algorithm related data from UART2, System related information from different slave devices through UART1, and Configuration data stored in non-volatile memory fetched through SPI interface are the various inputs to MMI logic. Processed data is given as output to multiple destinations. Messages related to control and status information are sent to LCD display. Status information is given to LEDs for continuous glowing or blinking of LEDs. Configuration data available in the system is given on demand through UART2. Reply messages to user queries are also given through UART2. Configuration data and control data are given to multiple slave devices of the system through system bus. Configuration data is loaded into non-volatile memory through SPI interface. Selective data is given to the appropriate modules inside the MMI logic for processing and computation of algorithms. Transfer of data from various sources to different destinations is done in many methods. Key press data is passed on in the form of logic levels on rows and columns. Different packet structures are used for Configuration UART and System bus UART. Configuration UART is for point to point communication. Figure4 shows the details of packets exchanged in the process of getting configuration data. System bus UART is used in point to multi point configuration. MMI is configured as the master and other system modules as slave devices. Master device only initiates communication. The details of packets exchanged for system bus communication to pass on control data / configuration data / algorithm related data / status information is shown in Figure5. Non-volatile memory access is through SPI interface, wherein MMI is master and Non-Volatile memory is slave. Different commands are sent by Master along with address to write data into memory, to read data from memory, to erase sectors and to bulk erase operations etc., Slave select is active, i.e., at logic level low till the communication between master and slave devices taking place.

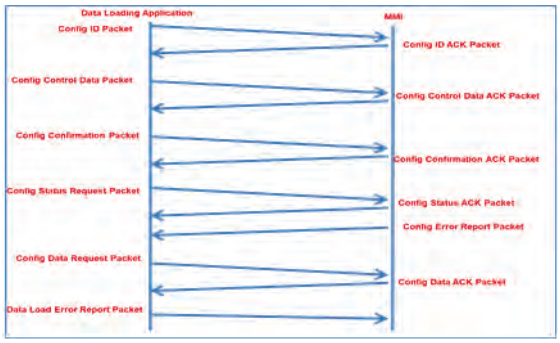

Figure 4. Data Loading into MMI

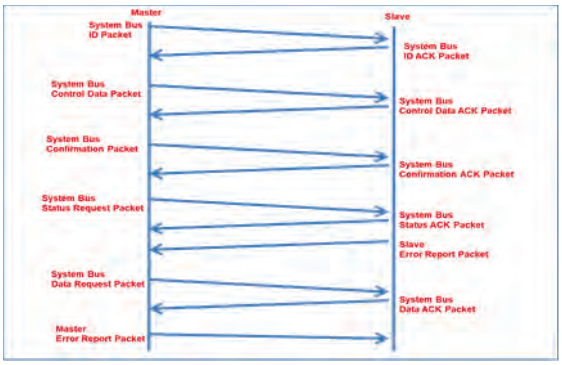

Figure 5. Flow of Packets through System Bus

E. Generation of Clock and Timing Signals

In order to ensure that all the events are executed at predictable time intervals, all the modules are designed to operate in synchronous mode. To avoid ambiguities, events are started or completed at known intervals of time. Single clock input is sufficient for MMI logic. All the other clock signals required for different modules are generated in clock generation module. A clock of 32.768 MHz clock is given as input. Some of the clocks required for internal operations of different modules are generated within those respective individual modules. Timing signals also are derived within those respective individual modules.

F. Receiving Data

The input data streams points must be analyzed for type of data, its availability, format and destinations for the data transfer between modules within MMI. Keypad interface logic provides valid key press output, corresponding row data and validation flag. This information is made available to key processing logic. UART1 receiver provides valid data output (in data words) and valid data flag to UART1 Driver. UART2 receiver generates valid data words received and valid data flag for UART2 Driver module. Receiver of SPI module generates parallel data output along with corresponding flags for the master driver of SPI to take up further processing of data packets.

G. Setting of Flags

Status of different operations is given in the form of setting Flags by individual modules. This will enable the concerned modules to initiate further activities. Key processing module performs control activities of MMI in addition to the processing of keys pressed to invoke certain events. Different operations selected through Keypad are: to allow data reception from UART2, to verify and authenticate the incoming data, to transfer algorithm related data from one segment to the other within the same non-volatile memory, to set configuration data for a selected operation of the system, to erase the configuration data, to ascertain status, to print control and status messages on LCD display etc. Keypad entries are not accepted during the process of transactions with non-volatile memory. Configuration data from UART2 is communicated through the exchange of a series of packets. Different flags are set to indicate the successful extraction of required data and information about errors in the received data. These flags are used to enable storage of data or to transmit data through system bus. UART1 data is analysed to monitor the status of different slave devices on system bus. Respective flags are set or reset accordingly. These flags are used for status indication on LCD display or LEDs. SPI Flags related to the status of data transfer and data erase operations being carried out with non-volatile memory are used for the purpose of displaying appropriate messages on LCD and also for initiating certain activities by the system.

H. Storage of Data

For performing data storage, points to be analysed are: data to be stored in Random Access Memory (RAM), data to be stored in Non-Volatile Random Access Memory (NVRAM), processing of NVRAM data, organization of RAM data and organization of NVRAM data. Configuration data, algorithm related data and preset channels data are stored in NVRAM. Initially with power ON or after reset operation, configuration and algorithm related data available in NVRAM is fetched and loaded into different segments of the RAM built with block RAMs of FPGA. Then flags are set accordingly to make system work as per the settings. Non-volatile memory is accessed through synchronous mode. Initially, configuration data is stored in RAM and after receiving the confirmation command from user, this data is dumped into NVRAM. Display messages are fixed and are stored as ASCII data in Read Only Memory designed with the look up tables of FPGA.

I. Processing of Data

Incoming data packets are processed to get the actual data. In MMI logic – keys, configuration and algorithm related data, stored data and status information are processed by different algorithm modules. Outcome of these algorithms are: To authorize the user in choosing parameters for different operations (through two-level passwords), to extract data for use in the system, to select the message to be displayed on LCD, to set / reset flags related to the status of slave devices of the system and so on.

J. Transferring the Processed Data

After processing the data, it should be made available at output peripheral modules and I/O peripheral modules. Data transfer through UART2 is done by exchange of different packets. These data packets contain a few keywords in addition to the actual data. UART2 transmitter will transmit data in serial fashion according to the protocol. Flags indicating the status regarding start bit transmit, transmit process in progress are monitored to avoid data overwrite and also to ensure error free transmission. Control data packets are shared with different modules of the system through UART1. As master slave configuration is adopted, transfer of data to slave devices will be done one after the other in a sequence with proper care. Similarly, status is collected from slave modules in a sequence by giving commands sequentially. With power on or with reset, status is obtained from slave devices by giving appropriate control commands. Later, exchange of control and status data is done depending on user demands through keypad. Flags of UART1 transmitter are continuously monitored by UART1 driver for smooth transfer of data. Data loading into non-volatile memory takes place through SPI interface depending on the user commands. Master driver of SPI interface logic arranges data as per the frame structure. Address and type of operations must be indicated clearly in their respective fields of the data frames. Slave select must be asserted low for the entire duration of data transfer.

K. Enabling / Disabling the Logic

To reduce power consumption, selective enabling / disabling of the logic is done in the following ways.

- To restrict unwanted operations by providing control signals to the concerned logics
- To shut down unused logic at any instant by supplying control signals to the concerned logics.
- To gate the clocks to ensure that clocks are not supplied to unused modules at any point of time.

III. Implementation of MMI Logic using Hardware Programming

XILINX Software ISE Design Suite 14.5 is used for various design activities of MMI like Design entry, Simulation, Implementation (Translate, Mapping and Place & Route), Programming File Generation, and Configuration of the Device. MMI logic is realized using SPARTAN 3AN XC3S1400AN device. The same MMI logic could be successfully ported on XC3S1600E device and different other devices of Virtex5 and Virtex6 families from XILINX and also on ALTERA devices like Cyclone II and Stratix Family FPGA devices. Device utilization summary of MMI logic is given in Figure 6. As shown in Figure 6, MMI logic consumed 1548 numbers of Slice Flip-Flops, 2520Nos of Occupied Slices and 4553Nos of 4-input LUTs and 16Nos of Block RAMs. Table-1 gives the summary of synthesis results of various modules used in MMI logic.

E-ISSN 2581 - 7957
P-ISSN 2277 - 3916

CVR Journal of Science and Technology, Volume 19, December 2020
DOI: 10.32377/cvrjst 1906

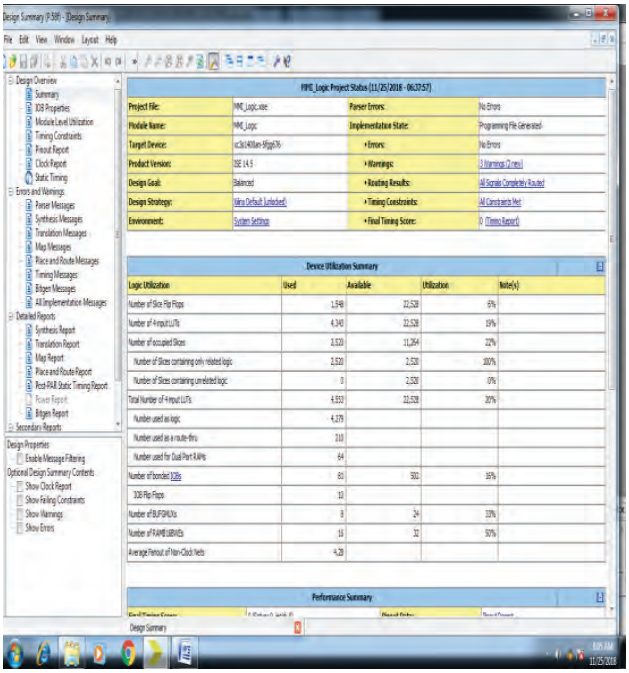

Figure 6. Device Utilization Summary for MMI

TABLE I.
SUMMARY OF SYNTHESIS RESULTS

Module	No of I/O pins	Occupied Slices	4-i/p LUTs	Block RAM	Power	
Clock Generation	12	15	19	-	65mW	
Keypad Interface	15	24	32	-	67mW	
Key Processing	137	370	701	-	71mW	
LCD Display Interface	27	99	195	-	66mW	
LED Interface	12	16	30	-	65mW	
UART1 Receiver	12	33	23	-	66mW	
UART1 Transmitter	14	24	40	-	65mW	
UART1 Driver	106	184	339	-	66mW	
UART1	88	245	401	-	70mW	
UART2 Receiver	12	53	33		68mW	
UART2 Transmitter	14	24	40	-	65mW	
UART2 Driver	83	553	1004	8	70mW	
UART2	65	641	1078	8	76mW	
SPI Interface	75	960	1824	8	71mW	
Passwords Management	74	246	468	-	68mW	
Man Machine Interface	81	2520	4553	16	106mW	

IV. BENEFITS OF MMI DESIGN USING HARDWARE PROGRAMMING

The benefits observed in MMI designed using Hardware Programming based design methodology are as given below [20].

- Focus is shifted to Interface, rather than the architecture of microprocessor
- Hardware Design Skills are sufficient
- Concurrent Execution of Events
- High Speed of Processing
- Low Latency of the System
- Reduction in Memory Requirements
- Elimination of Unwanted Additional Resources
- Single Clock Source is Sufficient
- Logical Resources are effectively utilized
- Reduced Power Consumption
- Simplified Design Methodology
- Low Cost of Development
- Effective Solutions for the issues being faced in Production / Manufacturing

Intel, Motorola family microprocessors and different families of DSPs from Analog Devices & Texas are considered for making comparison analysis explained in following sub-sections.

A. Reduction of Logical Resources

Traditional MMI logic based on microprocessor design requires Microprocessor, Program memory, RAM, Non-Volatile memory, UART Controller (configuration), SPI Controller (Non-Volatile memory), SPI Controller (system bus), Keypad read register, Keypad write register, LCD register, LCD display, LEDs, Keypad and Crystal oscillators to supply clock. All these components are needed for any microprocessor based design, be it a hard processor or a soft processor in an FPGA. If soft processor is used, logical resources of FPGA are consumed by the soft core of the processor. That soft core can cover some of the peripherals. If embedded processor (a built in component of FPGA) is used, then resources consumed within the FPGA are fixed, but resources available for logic realization are not consumed by the processor. Soft processors like Micro blaze, Nios, Pico blaze and ARM controllers etc. consume lot of logic resources that are more than one thousand slice registers and LUTs. These various devices used are actually meant for general purpose applications. Hence, customization with software programming is to be done to use them as per the user requirements of the system. Some of the features of peripheral devices as well as the processor may not be even used in some applications. Hence, those logic resources are not used and will go waste or sometimes might consume power. Hardware Programming is a customized design to effectively utilize the resources [21]. The peripherals are implemented as required and consume less logic resources. Unnecessary logic will not be used. Hardware programming model for MMI designed is a continuation of System on Chip concepts and consumes minimum logic resources as given below:

- FPGA, to implement all the logic on it

- Configuration memory for FPGA for storage of configuration data related to resources and interconnections among them. This memory can also be used to store configuration data of the system permanently
- Keypad to enter user commands
- LCD display to the display of messages
- LEDs for indicating status

The entire MMI logic implemented had consumed around 2500 Slices Registers only. Hence, there is a huge savings by virtue of customization due to the hardware programming design methodology.

B. Reduction of Clock Requirements

Processor based system is designed using built-in peripherals on the microprocessor chip and also using ASICs for some of the peripherals. These peripherals require clocks of some specific frequencies. Microprocessor also works with clock within certain range of frequency. Hence multiple clock sources are required. Whereas in the case of hardware programming based design, a single clock source is used to derive different clock sources required to make the logic modules operate at different frequencies appropriate to the transfer and computation of data through various interfaces. MMI has used only one clock of 32.768MHz. All the clock signals required for different modules are derived from the single clock source.

C. Reduction of Memory Requirements

Hardware Programming based embedded systems do not require any specific program memory and data memory as microprocessor is not used. Block RAMs of FPGAs are configured as synchronous DPRAMs for effective storage of required data. Hence very less number of memory locations are required. MMI logic has used only sixteen numbers of Block RAMs compared to Megabytes of memory required in case of processor based systems. This has lead to the improvement in execution time also.

D. Reduction of Power Consumption

As the number of components used is very less, the logic resources are very optimized and most importantly the unused logic at any time is shut down or disabled selectively, power consumption is drastically reduced. With the advent of latest technology programmable devices that have very thin silicon wafers, operating voltages have come down and hence the power consumption will still be reduced significantly. Data sheets of microprocessor indicate power consumption of microprocessor as a few hundreds of mill watts; MMI logic has consumed only a few tens of mill watts.

V. CONCLUSIONS

Hardware Programming based design methodology is used for the design of man machine interface logic and it can be considered as an alternative design methodology for embedded systems. As this design methodology does not require a microprocessor, the issues related to the microprocessor are overcome successfully. The design approach of modular structure makes the design simple and facilitates the designer in extending or modifying the design with ease. These Hardware Programming concepts can be extended for design of various systems being used in several industrial applications.

REFERENCES

[1] Wang Yue Sheng, Gu Xiao Lei, Wand Liang, “Design of Human Machine Interface of Metal Detector Machine based on µC/OS-III and emWin”, 34th Chinese Control Conference (CCC), 2015, DOI: 10.1109/ChiCC.2015.7261013.

[2] Khan G.N, Jin M, “Hardware Software Co-design of a Safety-Critical Embedded Computer System for an Automatic Endoscope”, Canadian Conference on Electrical and Computer Engineering (CCECE), 2002, DOI: 10.1109/CCECE.2002.1013019, pp 657-662.

[3] Ajay Kumar Singh, Ankitha Taneja, “Frequency allocation in software defined radio using smart cards”, Electronics for You, Oct-2008 Vol 40, No.10, pp-108.

[4] T. Harnath, K. Lal Kishore, “Development of Customized Interrupt Controller Logic”, International Journal on VLSI Design and Communication Systems, 2015, Vol 3, Issue 10, IJVDCS7951-248, pp 1446-1449

[5] Kristanian Blomquist, “Using nV SRAMs to super charge flash memory devices in data logging applications”, Embedded systems design, Dec-2008, Vol 21, No.12, pp-18.

[6] Gereon Fuhr, Seyit Halil Hamurcu, Diego Pal, Thomas Grass, Rainer L, “Automatic Energy-Minimized Hardware/ Software Partitioning for FPGA Accelerated MPSoCs”, IEEE Embedded Systems Letters (Early Access), 2019, DOI: 10.1109/LES.2019.2901224.

[7] Ionut Radoi, Florin Rastoceanu, Daniel-Tiberius Hritcu, “Data Transfer Methods in FPGA based Embedded Design for High Speed Data Processing Systems”, International Conference on Communications (COMM), 2018, DOI: 10.1109/ICComm.2018.8484792.

[8] Mahamudul Hassan, Sheikh Md Rabiul Islam, “Design and Implementation of Pre-Processing Chip for Brain Computer Interface Machine”, International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), 2019, DOI: 10.1109/ICREST.2019.8644230.

[9] Fateh Boutekkouk, “Soft Intellectual Properties (IPS) Integration for System on Chip (SOC) Design”, International Conference on Research Methodologies in Electronic Devices and Circuits, 2012, DOI: 02.EDC.2012.1.1, pp 101-106.

[10] Shaila S Math, Veerabhadrayya Math, “Design and Analysis of XILINX verified AMBA Bridge for SOC systems”, Lecture series in Computer Science, 2013, DOI: 03.LSCS.2013.2.535, Vol.2 pp 32-37.

[11] Parthasarathy T.R, Venkatakrishnan N, Balamurugan K, “Analysis of partitioning between ARM and FPGA on Performance Characteristics”, IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), 2012, DOI: 10.1109/ICACCCT.2012.6320745, pp 78-82.

[12] Jayasanthi Ranjith M.E, Muniraj N.J.R, “VLSI Implementation of Memory Efficient Single Bit Processor for Industrial Control Applications”, International Journal on Recent Trends in Engineering and Technology, 2010, DOI: 01.IJRTET.4.4.76, Vol 4, Issue 4, pp 29-31.

[13] Sikun Li, Zhihui Xiong, Tiejun Li, “Distributed Cooperative Design Method and Environment for Embedded System”, Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design, 2005, DOI: 10.1109/CSCWD.2005.194316, pp 956-960.

[14] Kopetz Hermann, "The Complexity Challenge in Embedded System Design", 11th IEEE International Symposium on Object Oriented Real Time Distributed Computing (ISORC), 2008, DOI: 10.1109/ISORC.2008.14, pp 3-12.
[15] Mahendra Vucha, Rajendra Patel, Arvind Rajawat, "Dynamic Profiling Methodology for Rescue Optimization in Heterogeneous Computing System", Elsevier Science and Technology, 2013, DOI: 03.elsevierst.2013.1.7, Vol 1, pp 41-46.
[16] T. Harnath, K. Lal Kishore, "Development of Customized Asynchronous Serial Transceiver", Proceedings of 48th CST Infrastructure, Computer Society of India, 2013, pp 41-46
[17] T. Harnath, K. Lal Kishore, "Development of Customized Synchronous Serial Transceiver", International Journal of VLSI and Embedded Systems, 2014, Vol 5, ISSN: 2249-6556, Article 06364, pp 1054-1060
[18] T. Harnath, K. Lal Kishore, "Development of Customized System Bus Transceiver", International Journal of VLSI and Embedded Systems, 2014, Vol 5, ISSN: 2249-6556, Article 06370, pp 1066-1073
[19] T. Harnath, K. Lal Kishore, "Development of Customized Input Output Processor", International Journal on VLSI Design and Communication Systems, 2014, Vol 2, Issue 10, IJVDCS3313-197, pp 1069-1074
[20] T. Harnath, K. Lal Kishore, "Hardware Programming to Improve Embedded System Performance", 33rd Indian Engineering Congress, 2018, Technical Volume, pp 352-358
[21] T. Harnath, K. Lal Kishore, "Hardware Programming as an alternative for Embedded System", International Journal on VLSI Design and Communication Systems, 2016, Vol 4, Issue 5, IJVDCS9724-70, pp 0357-0362