NCRISET-2017 e-ISSN: 2456-3463
International Journal of Innovations in Engineering and Science, Vol. 2, No.6, 2017
www.ijies.net

Implementation Of a RISC Based SOC For Handling And Packaging Industrial Process

Rupali V Bhange
Department of electronic and telecommunication
M. Tech (VLSI), GHRIET, Nagpur
Nagpur, India
rupalibhange94@gmail.com

Yogesh M Motey
Department of electronics and telecommunication
Assistant Prof. (VLSI), GHRIET, Nagpur
Nagpur, India
Yogesh.motey@gmail.com

Abstract—**In the era of industrialization, technological revolution reduced the intervention of humans to assist machinery. The privatization slogan has increased the competition among industries which leads to origin of new products/designs for automation. The paper presents, one such, design and implementation of 64-bit RISC processor on SOC for industry automation, mainly useful for packing. The design includes processor with BIST features; it is a mechanism that allows a machine to test itself. Here RISC processor is used to decrease the number of instructions and they execute the instruction at faster rate. Later the design is integrated on System on Chip (SoC) to obtain a single chip which reduces the overall power consumption. The design and synthesis is done by verilog and verified in Altera Quartus 11.0 RTL compiler, SoPC Builder, Nios II SBT Eclipse, and Modelsim 10.1c.**

*Keywords-**industrial automation, reduced instruction set computer, system-on-chip, handling and packaging, synthesis, simulation, verification, hardware description language***

I. INTRODUCTION

Industrial Automation is an important market segment which deals with semi or complete automation of the industrial manufacturing machinery and associated tools. Microcontrollers have been playing an important role in Industrial Automation markets for performing automated tasks. For high-end process automation, the usage of customized or highly specific Programmable Logic Controllers and x-86 or Power-PC based Single Board Computers is common. Traditional RISC based Chipsets and System-on-Chip products have focused exclusively on Consumer Electronics and related fields. However, there is a compelling reason for having customized RISC based Chipsets which can cater to market verticals like Industrial Automation, Portable Instrumentation, Portable Medical and Point-of-Sale.

In these market segments, the traditional approach has been to utilize a general purpose RISC chipset which is then interfaced with custom ASIC to achieve the intended functionality for an Industrial Automation or Retail Point of Sale kind of End-Product. This approach not only increases the BOM cost, but limits the scalability of the product for future generations and OEM/ODMs end up spending significant amount of their R&D efforts in building new products for these markets.

For example, let us consider the most common required interfaces for an Industrial Automation product i.e., ADC and CAN Interfaces. In traditional approaches, we would typically have the below mentioned configuration. Depending on the logic implemented inside the custom ASIC/FPGA, the Analog Front End may be a separate/discrete IC or maybe implemented inside the custom ASIC itself.

The above approach for designing products for Industrial Automation does not scale well and causes huge turn-around time for the Device Manufacturers whenever they are planning /thinking of introducing new products into the market.

II. MOTIVATION

In this section, we will explore the motivation and the need to integrate a peripheral like the PRU (Programmable Real-time Unit) within the System-On-Chip and later examine one specific use-case of the PRU i.e., Communication Sub-system for Industrial Automation.

For real-time applications such as Programmable Logic Controllers, Control Machinery, the requirement to respond to real-time events is a critical one. If we take any modern System-On-Chip, they typically consist of at least two levels of Interconnect/bus and also multiple levels of cache/internal/external memory which makes it difficult to respond to real-time events within tens of nanoseconds.

So this introduces a kind of designer's dilemma that while modern System-On-Chip offers the best connectivity/interfaces, cache and power requirements, at the same time cannot handle real-time events/processing withintens of nanoseconds. To overcome such performance issues, there has been one design argument that we need to integrate or merge Special Real-time Co-processors along with the regular RISC chipsets. This is analogous to integration of special purpose Audio/Video Co-processors to off-load the Audio/Video processing logic to custom real-time cores.

Another associated design challenge is that adding a real-time co-processor may not solve the real- time performance issue altogether. The real-time co-processor should also have necessary interfaces/operational infrastructure to be able to detect real-time events being triggered from the external world and they should be able to detect the occurrence of the real-time event within a few nanoseconds.

This leads us to the second design conclusion that these Real-time Co-processors should also have their own interrupt Infrastructure and access to the external world via some kind of I/O Pins or serial interfaces. Typically providing access to at least some of the peripherals would also make the real- time coprocessor much more flexible in terms of integration and general usage.

To summarize, the Real-time Coprocessor should meet/address the following requirements:

1. Have its own instruction execution unit independent of the RISC core

2. Should not utilize instruction pipeline cycles which will not guarantee real-time execution

3. Should have its own interrupt controller/mechanism to be able to detect real-time external events
4. Should have a separate instruction and data memory to ensure that the instruction fetch and data fetch are not multiplexed on the same lines
5. Should also have access to the external world via some General Purpose Pins or peripherals
6. Should preferably share the interconnect/external bus with the main RISC Core such that the real-time co-processor can also configure/control the memory mapped peripherals such as UART, PWM, SPI etc.

I. SOPC (SYSTEM-ON-A-PROGRAMMABLE-CHIP) BUILDER

SOPC builder is a powerful system development tools for creating systems based on processors, peripherals, interfaces and memories. SOPC Builder is implemented for the purpose of generating a complete system-on-a-programmable-chip (SOPC) by consuming less time than the accustomed integration methods [4].

Altera has SOPC Builder functionality built in Quartus II, which accordingly connects the soft-hardware components to construct a complete computer system that can be controlled on any of the FPGA chips and is also capable of producing interconnect logic automatically. It is outfitted with a library of built-in components such as a Nios II soft processor, memory controllers, interfaces, standard peripherals and custom peripherals.

In SOPC Builder, the system components are routed in a GUI (Graphical User Interface). The GUI is exclusive in configuring the soft-hardware components. It is a general-purpose tool for creating systems that includes a soft processor apart from the Nios II processor. It also contributes to writing software and system simulation.

A. *Core Functionalities of SOPC Builder*

• Describes the hardware of the system.
• Performs the system generation.
• Performs memory mapping for initiating the software development.
• Produces test bench to simulate the design.

B. *Architecture and Design of SOPC Builder*

Designs using SOPC Builder are generated to develop a top level HDL (hardware description language) file by connecting various modules together. These various modules are considered the building blocks for the SOPC Builder system [9]. For the connection of multiple components in the system, the modules use Avalon (Avalon switch interconnect) interfaces, such as memory-mapped, streaming and IRQ (interrupt request).

C. *SOPC Components*

The components in SOPC Builder are referred as hardware blocks of the system. They contain HDL descriptions of the hardware of the components and interfaces used for the hardware. They also contain the description of the parameters that determine the operation of the components (Fig. 1). The SOPC components are connected to the system interconnect fabric using the Avalon Memory-Mapped interface (Avalon MM) or the Avalon streaming interface (Avalon-ST) [4].

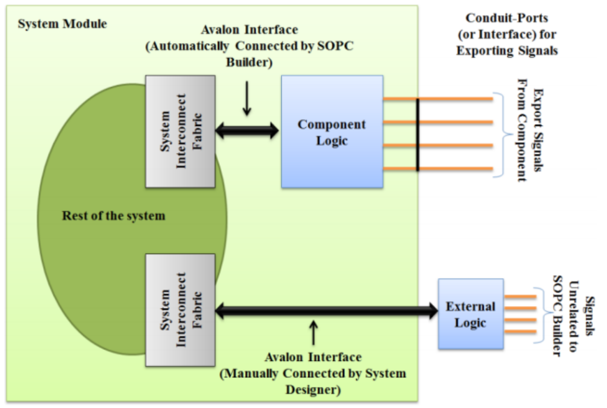
System Module
Avalon Interface (Automatically Connected by SOPC Builder)
Conduit-Ports (or Interface) for Exporting Signals
System Interconnect Fabric
Component Logic
Export Signals From Component
Rest of the system
System Interconnect Fabric
External Logic
Signals Unrelated to SOPC Builder
Avalon Interface (Manually Connected by System Designer)

Figure 1. SoPC Builder System

D. *Types of SOPC Components*

SOPC Builder's inbuilt components are
a. Static HDL Components
b. Generated HDL Components

c. Composed HDL Components
d. Custom Components
e. Third-Party Components
(a) Static Components: These are the components that accept the VHDL parameters. Examples: Address widths, Data widths and FIFO depths.
(b) Generated Components: These are the components whose hardware description language file is generated based on the value of its specified parameters. Example: A parameter that controls the number of interfaces.
(c) Composed Components: These are the components that are constructed from combinations of other components
(d) Custom Components: The design flow used to merge the custom components into the SOPC Builder is as follows:
1. Determine the interfaces required by the custom components.
2. Write the logic for each custom component.
3. Develop the custom components with the hardware description language files by using the component editor.
4. Represent the custom component in the system by an instance.
(e) Third-Party Components: These components are built by third parties. Components external to the SOPC Builder For the components that interfere to external logic or off-chip devices with Avalon- compatible signals outside the SOPC Builder system, the component files describe only the interface to the external logic. The connection of signals a the top-level of SOPC Builder to pins or logic defined outside the system is done manually.

E. SOPC Design Flow

The design flow of SOPC Builder is as follows (Fig. 2)
a. Add components by Component Editor.
b. Initiate Simulation of the system.
c. Develop the system design by adding components, IRQs and addresses.
d. Start the system generation.
e. Conduct system level simulation.
f. Compile the system design.
g. Download .sof file to an Altera FPGA.
h. Perform Testing.

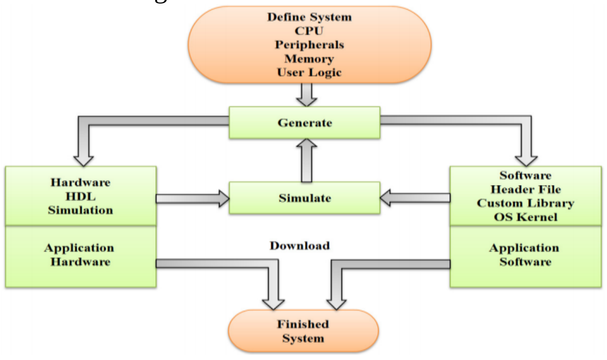

Figure 2. SoPC Design Flow

F. Avalon Switch Interconnect

The Avalon Memory-Mapped (Avalon-MM) interface has a bandwidth of steep structure generally used for interfacing the components of the system. The Avalon switch interconnects uses less logic but supplies absolute flexibility. This interconnect is a cross-connect fabric used for the purpose of switching and multiplexing.

The Avalon switch interconnect fabric is a combination of interconnect and logic resources. It is used for stocking the Avalon memory mapped master and slaves on the components. It is referred to a device having information about connections of the entire system and its components. It assures that connections between the master and slaves are routed precisely. This meets the requirements of the components.

The interconnect fabric permits the connection of an unlimited count of master components and slave components. These master and slave components can have a one-to-one connection, a one-to-many connection, a many-to-one connection, or a many-to-many connection. The system interconnect fabric is used to support the interfaces for the on-chip components of the system and interfaces for the off-chip devices of the system. In this interconnect fabric, master components and slave components with altered data widths are supported.

The system interconnect fabric also serves as a platform for the master and slave components running with several clock domains. It also supports master and slave components with several memory mapped ports.

The system implementation fabric for the Avalon-MM interfaces acts as a partial crossbar interconnects structure. The partial crossbar interconnect structure is a matrix with multiple inputs and multiple outputs. This interconnect structure arranges simultaneous paths between the master and slave components. In the Cyclone II FPGA, routing resources and synchronous logic constitute the system interconnect fabric.

G. Functionalities of System Interconnect Fabric

a. Decoding Address
b. Multiplexing Data path
c. Insertion of Wait State
d. Pipelined Read Transfers
e. Multi-master System Arbitration
f. Burst Adapters
g. Interrupts
h. Reset Distribution

H. Automated System Generation

SOPC Builder system integration tools automatically perform the process of configuration of the processor features; hence the hardware of the design is produced that is used to program an Altera device. The graphical user interfaces (GUI) help in structuring of Nios II systems with multiple peripherals and memory interfaces (Fig. 3).

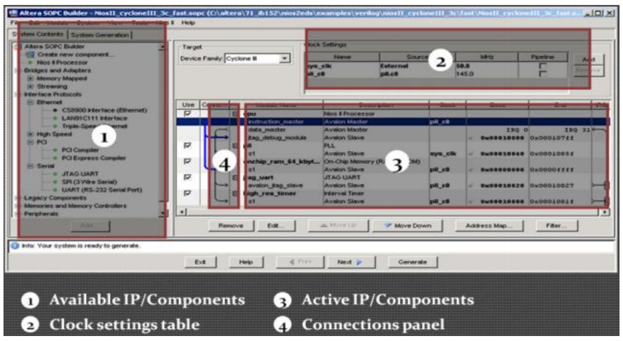

Figure 3 .SoPC Builder GUI

After system generation, you can download the design onto a board, and debug the software executing on the board.

II. NIOS II SYSTEM ON ALTERA'S DE2 BOARD

The Nios II processor and many other components such as standard peripherals and custom peripherals are used for the formation of a total system that can be integrated into a Nios II system on Altera's DE2 board [3]. The process of interfacing the Nios II processor and peripherals to the DE2 board chips is enabled on the Cyclone II FPGA chip. The interconnection network connecting these components in the FPGA chip is called the Avalon Switch Fabric (Fig. 4).

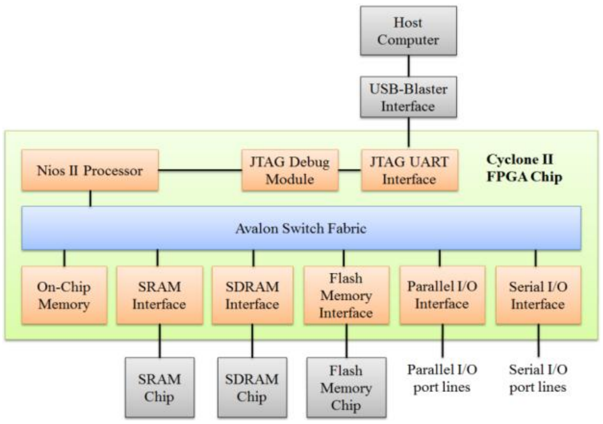

Figure 4. Nios II System on DE2

The memory blocks available in the Cyclone II FPGA chip serve as the on-chip memory for the Nios II processor. These memory blocks can be connected to the Nios II processor directly with the help of the Avalon network. The Input / Output interfaces are used for connecting I/O devices. A JTAG UART interface is used for the purpose of providing a Universal Serial Bus link between the Altera's DE2 board and the host computer to which the board is connected. This Universal Serial Bus link is called the USB-Blaster. The JTAG Debug module is used by the host computer to control the Nios II processor, downloading programs into the memory, starting and stopping execution. The memory chips such as SRAM and SDRAM can be connected by applicable interfaces. A hardware description language is used to define all the Nios II system components on the Cyclone II FPGA chip.

A. *Nios II Processor*

The Nios II processor is a configurable RISC processor [10]. Hardware structure includes:

a. Functional units of the Nios II architecture
b. Fundamentals of the hardware implementation

B. *Fundamentals of the hardware implementation*

The hardware implementation of Nios II architecture explains an instruction set. Any functional unit whose hardware is implemented can be programmed in software. Every hardware implementation has different objectives, for example, that the size of the core should be small and must yield high performance (Fig. 5). These features help the Nios II architecture adjust to many different applications. Implementation of the processor core specifically requires any of three trade-offs: more performance or less performance of a feature; inclusion of a feature in the core or exclusion of a feature; and implementation of the hardware and software programming of the features included in the core.

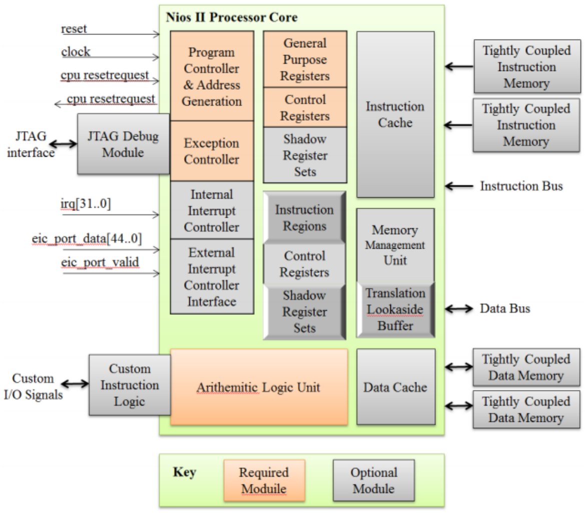

Figure 5. Nios II Core Block Diagram

C. *Nios II Processor Core Architecture*

The architecture of Nios II processor core consists of [3]

a. Register file

b. Arithmetic logic unit (ALU)

c. Interface to custom instruction logic
d. Exception controller
e. Interrupt controller

f. Instruction bus
g. Data bus
h. Memory management unit (MMU)
i. Memory protection unit (MPU)
j. Cache memories
k. Tightly-coupled memory interfaces for instructions and data
l. JTAG debug module.

D. Customizing Nios II Processor

Altera FPGAs offer to add new features in order to increase the performance of the Nios II processor. The advantage of customization is elimination of unnecessary processor features and peripherals to suite the hardware design in a smaller and lower-cost device. The following are the possibilities to program customized pins and logic resources available on the Altera DE2 board [1]:

a. Rearrangement of the pins on the chip is the best way to reduce the design of the board. For example, address and data pins can be moved for external SDRAM memory to cut the traces of the board in short.

b. The use of extra pins and logic resources on the chip is independent of the processor. Extra resources supply a few more extra gates and registers for the purpose of designing the board (Fig. 6). Also, these extra resources affect the complete system. Using extra pins and logic resources on the chip, the additional peripherals for the Nios II processor can be implemented. We can easily access the additional peripherals from the library of SOPC Builder, which are used for connecting the Nios II processor [3].

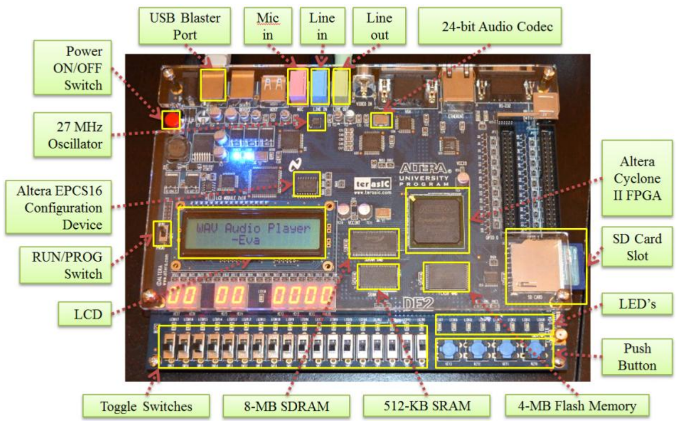

Figure 6 . Altera DE2 Board Components and Interfaces

E. Features of DE2 Board (Altera Cyclone II 2C35 FPGA)

Following is the list of features available on the DE2 Board [4]:

a. USB Blaster is inbuilt on the board mainly for programming purposes and for controlling API
b. JTAG Mode and AS Mode are supported
c. 8 MB (1M × 4 × 16) SDRAM
d. 1 MB Flash Memory
e. SD Card Port
f. 4 Push-buttons
g. 18 DPDT (Double Pole Double Throw) switches
h. 9 Green User LEDs
i. 18 Red User LEDs
j. 50 MHz Oscillator
k. 27 MHz Oscillator
l. 24-bit Audio CODEC including line-in/out, and mic-in jacks
m. VGA DAC
n. VGA out connector
o. TV Decoder
p. TV-in connector
q. Ethernet Controller
r. USB Controller as Host and Slave
s. RS-232 Transceiver and 9-pin connector
t. Mouse and keyboard connectors (PS/2)
u. IrDA transceiver
v. 2 x 40-pin Expansion Headers

III. PROPOSED METHODOLOGY

Fig.7. illustrates the process of material handling and packaging system, which shall be automated using the programmable logic controller. This is a step by step process on which corresponds to the input and output peripherals that are needed in programming the ladder diagram. Included in the automation is the placement of box, filling of materials, transferring, checking, and sealing of the final product, or item. The overall design is implemented using an experimental prototype.

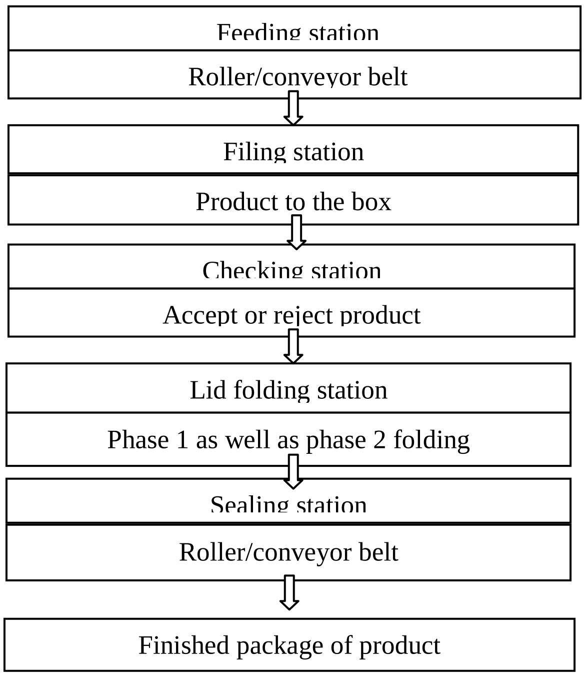

Figure 7. Diagrammatic structure of automated system

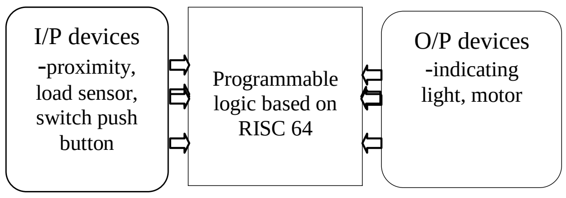
I/P devices
-proximity, load sensor, switch push button
Programmable logic based on RISC 64
O/P devices
-indicating light, motor

Figure 8. Conceptual Framework for Automation

Fig.8. shows the general PLC block diagram used by the researchers to implement the automation of the system. From the manual operation, a programmable logic controller is used to convert the system into an automated process. Input components such as sensors and switches are used to indicate the condition corresponding to the hardware flow diagram of the PLC project design. The programmable logic controller interfaced the system and provided the ladder diagram for the design. Output components such as motors and pneumatic cylinders are used to indicate the desired objective of the system. Also, fig.9. Show first phase of system. As shown, it consists of three stations namely box feeding, the filling, and checking stations. The 2”x2”x2” card-board box is used as the object to be transported. The researchers used a small marble with the same weight to act as a sample material for experimental testing the prototype.

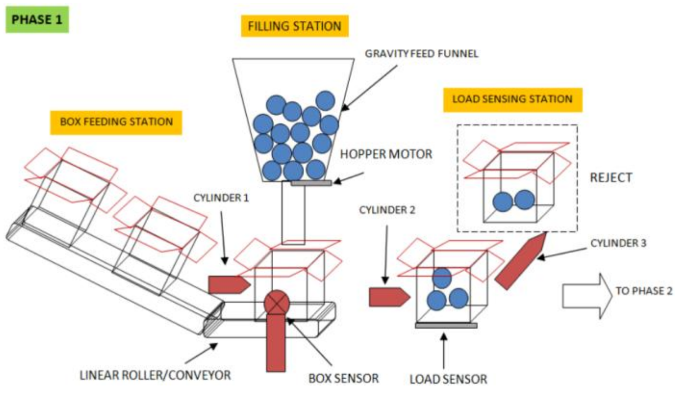

Figure 9. Flow for feeding, filling, and checking stations

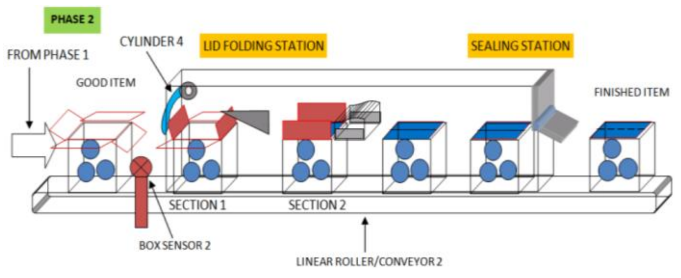

Figure 10. Flow for lid folding and sealing stations

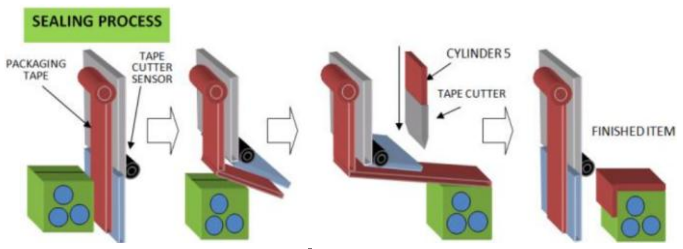

Figure 11. Sealing process station

Fig.10. shows the second phase of the design project. It is composed of two stations namely lid folding and sealing station. Fig.11. illustrates the detailed automated process for the packaging station. A 1-inch width packaging tape is used to seal the boxes in a single linear direction. A proximity sensor is used to detect the length of the tape that will seal the boxes. A tape cutter made of stainless steel is also used in this station. Below is the discussion of whole operation:

1. When the start button is pressed, box shall be push under the hopper by cylinder 1.
2. Box sensor 1 is provided as to sense the presence of box under the hopper. When a box is detected, hopper motor runs, thus; dropping marbles to the box (box is stationary).
3. A counting sensor is provided as to monitor the correct number of marbles being drop to the box.
4. After the desired numbers of balls is dropped, a load sensor activates as to determine if the loaded box is overload, under load, or exact load. Good items are pushed to the next station, and phase by cylinder 2, while the under load box and the over load is pushed away from the line by cylinder 3, which is to be checked manually.
5. Having a good item box or load triggers the conveyor 2, to conduct or start its operation. The approaching box is monitored by the box sensor 2.
6. When a box is sensed by box sensor 2, it shall activate cylinder 4, and folding the lid of the first section of the side of the box.
7. The second lid and section is folded, as the box moved on the specialized lid folding the obstacle, or as the box makes progress on the conveyor.
8. Having a partially closed box, and a continuous running conveyor, it passes to the sealing station which finishes the packaging process. At this station, a packaging tape, and a cutter is positioned on a flip type window as to allow the incoming box to pass beneath it.
9. The tape cutter sensor shall be triggered, if the flip type window is moved from upward to downward position. This shall activate and turned the tape cutter to move downward via cylinder 5, and cut the tape.
10. Having finished the required task, the system shall point out if another process is to be commenced as invoke by the operator.

11. The finished item shall be collected at the end of the line by the on-duty personnel.

IV. CONCLUSION

Today industrial automation software requirements include capability to implement applications involving widely distribute devices, high reuse of software components, formal verification that specifications are fulfilled. In this paper, an object oriented approach, the programming language together with a proper way to organize the inputs and outputs of FBs and supervisory control are proposed to implement industrial automation control systems to meet the new challenges of this field.

V. REFERENCES

[1] Altera Audio/Video Configuration Core for DE2-Series Boards. (July 2010). [Online]. Available: ftp://ftp.altera.com/

[2] SD card IP Core. Altera University Program Secure Data Card IP Core. (March 2009). [Online]. Available: ftp://ftp.altera.com (accessed September 19 -2012)

[3] Wolfson Electronics. (2004, April). Portable Internet Audio CODEC with Headphone Driver and Programmable Sample Rates. (WM8731

[10] My First NIOS II Software, Altera Corporation based on Altera Complete Design Suite Vision9.I, January 2010 [online] :http://www.altera.com/.

Rev3.4). [Online]. Available: https://instruct1.cit.cornell.edu/(accessed October 22-2012)

[4] Altera SOPC Builder User Guide. (2010, December). [Online]. Available: http://www.altera.com/literature/ug/ug_SOPC_builder.pdf

[5] Altera Embedded Peripherals IP Guide. (2011, June). [Online]. Available: http://www.altera.com/literature/ug/ug_embedded_ip.pdf

[6] S. Moslehpour, K. Jenab and B.S. Pabla, “Implementing a soft core NIOS II processor for VGA application,” International Journal of Engineering Research and Innovation. vol.4, no.2, pp. 12-26, 2012.

[7] S. Moslehpour, K. Jenab and S. Valiveti, “GPS time reception using Altera SOPC builder and Nios II: Application in train positioning,” International Journal of Industrial Engineering and Production Research, vol.23, no.1, pp. 13-21, 2012.

[8] S. Moslehpour, K. Jenab and B. K. Matcha, Design of the Nios II System for the Playing of Wave Files on an Altera DE2 Board, 2012.

[9] J. O. Hamblen and T. S. Hall, “Using system on a programmable chip technology to design embedded systems,” IJCA, vol.13, no.3, pp. 1-11, 2006.