International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 4, No. 3, November 2015, pp. 192~200
ISSN: 2089-4864

An Efficient approach for Design and Testing of FPGA Programming using LabVIEW

B. Naresh Kumar Reddy, N. Suresh
Dept. of Electronics and Communication Engineering, NIT Goa, India

Article Info

Article history:

Received Jun 26, 2015
Revised Sep 9, 2015
Accepted Sep 28, 2015

Keyword:

LabVIEW software
FPGA Board

ABSTRACT

Programming of Field Programmable Gate Arrays (FPGAs) has long been the domain of engineers with VHDL or Verilog expertise. FPGA's have caught the attention of algorithm developers and communication researchers, who want to use FPGAs to instantiate systems or implement DSP algorithms. These efforts however, are often stifled by the complexities of programming FPGAs. RTL programming in either VHDL or Verilog is generally not a high level of abstraction needed to represent the world of signal flow graphs and complex signal processing algorithms. This paper describes the FPGA Programs using Graphical Language rather than Verilog, VHDL with the help of LabVIEW and features of the LabVIEW FPGA environment.

Corresponding Author:

B. Naresh Kumar Reddy,
Departement of Electronics and Communication Engineering,
NIT Goa, India.
Email: naresh.klu@gmail.com

1. INTRODUCTION

A. Introduction to FPGA:

Field Programmable Gate Arrays pervasively known as FPGA is an option for usage of advanced rationale in frameworks [1]. They are pre-assembled chips that might be customized electrically to actualize any advanced configuration. The main static memory-based FPGA (ordinarily termed as SRAM based FPGA) isintroduced. This construction modelling took into consideration both logic and interconnection arrangement utilizing a series of design bits. Xilinx introduced the cluster of configure logic blocks (CLB‘s) with I/O, Which hold 64 CLB‘s & 58 I/O in First modern Commercial FPGA‘s, FPGAs have become colossally in many-sided quality [2]. Now a days advanced FPGA can hold roughly 0.33 million rationale pieces and 1100 I/O. The fundamental building design of FPGA comprises of three real parts: programmable rationale pieces, which actualize the rationale capacities, programmable directing (interconnects) to execute these capacities and I/O closes to Make off-chip associations A Design of FPGA architecture is shown in figure1.

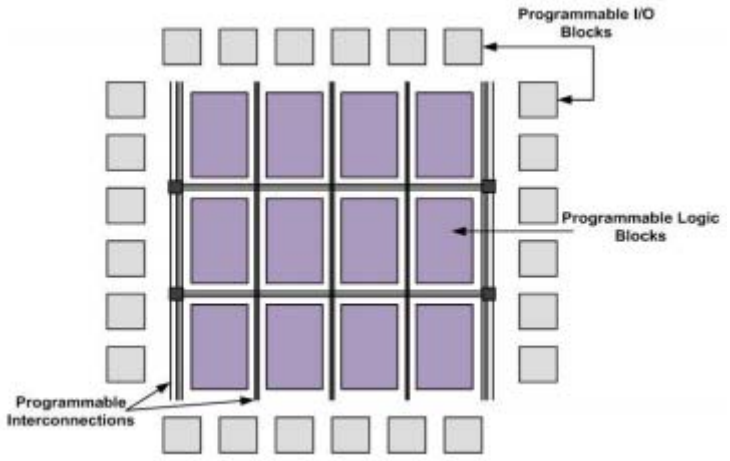

Figure 1. FPGA Architecture

1) Programmable Logic:

FPGA consists of programmable logic block, which is used for essential processing and storing elements used in various computerized frameworks. The Fundamental element in programmable logic block holds several types of reconfigurable combinational logic like flip-flops, latches in order to reduce area and delay cost. There are also advanced FPGAs, which consist of heterogeneous mixture of different blocks such as dedicated memory blocks, multiplexers etc. and each of them are used for specific functionality. Designmemory is utilized by entire logic block to control capacity of every component inside the block.

2) Programmable Interconnect:

By programming the FPGAs, we can give connections among various logic blocks and I/O blocks to finish a client characterized outline. Each FPGA consists of components like pass transistors, multiplexers and tri-state buffers [2]. Most part of the pass transistors and multiplexers are used to interface logic elements in logic cluster, while each among three are used for more worldwide directing structures. Some of the worldwide steering structures, which are used as a part of FPGAs are island style, cellular, bus based and registered architectures

3) Programmable I/O:

Programmable I/O means a media or mean to interface logic blocks and routing architectures to variety of outer segments in FPGA. The logic circuitry and I/O pad present in FPGA forms are also in I/O cell. These cells are present in critical segment of the FPGA and expanded over 40% of FPGAs zone. The most challenging concern among Programmable I/O block is that there is a great diversity among reference and supply voltage standards. A standout amongst the most critical choices in I/O structural planning configuration is the determination of models that will be backed. This includes painstakingly made exchange off's on the grounds that, dissimilar to Look Up Tables, which can actualize any advanced capacities, I/O cells can for the most part execute the voltage guidelines chose by planners [3]. Silicon region needed for I/O cells will be essentially increased for supporting expansive number of measures and moreover to increase large number of gauges pin capacitance may increase the number of pins, which will restrain execution.

4) Hardware Description Language (HDL):

Hardware description Languages (HDL) includes VHDL, Verilog, Systemc and Handle-C.Most of the tie we use Handle C for FPGA programming. VHDL and Verilog are developed for industry measures. HDL's have numerous sellers offering recreation and synthesis tools [4]. Behavioural, RTL and structural levels of depiction might be utilized between alterably in these dialects. Sytem C is used for displaying framework level behaviour and have C++ based libraries. As primary language of System C is C++, software processes can be more effectively demonstrated when compared to conventional HDL, even though System C is increasing their development but does not reach the development of VHDL or Verilog synthesis products. Handel-C requires the originator to unequivocally depict parallel handling squares inside a procedure. It incorporates characteristics for between methodology correspondences.

In Ref. [3] Alastair M. Smith describes about the applications of geometric programming configuration of homogeneous FPGA architectures and constructs on an expanding group of work concerned with demonstrating reconfigurable architectures and presents a full region and postponement model of a Reconfigurable Devices.

B. Introduction to LabVIEW:

National Instruments provides LabVIEW software (Laboratory Virtual Instrumentation Engineering Workbench), which provides a platform and development environment using visual programming language. LabVIEW programming is perfect for any estimation or control framework, and the heart of the NI outline stage. Coordinating all the apparatuses that specialists and researchers need to assemble for an extensive variety of uses in drastically less time, LabVIEW is an advanced environment for critical thinking, quickened gains, and constant development [5], LabVIEW has two sections: the front panel and functional block diagram. Functional block diagram is a programming area and front panel provides an interface to develop. By establishing the relations between front panel and the Functional blocks, applications are developed.

LabVIEW programs are termed as Virtual Instruments, on the other hand VI. LabVIEW holds an exhaustive set of instruments for procuring dissecting, showing, what's more putting away information and also provide instruments which are used to troubleshoot the code.

When LabVIEW opens, it shows two windows if one wants to write any program in LabVIEW, first in that program, operation is found out, that operation we draw as Graphical Diagram in Functional block diagram window then we can give inputs (control) and output (indicator) in front panel, after assigning, we have to connect through wire from inputs (control) to function and function to output (Indicator). LabVIEW provides an environment for programming, which is used for undergraduate engineering training [6]. It also offers help for data acquisition hardware, multitasking, inherent libraries and basic meaning of client interfaces and is generally utilized within expert building.

2. FPGA PROGRAMMING

FPGA stands for "Field Programmable Gate Array". FPGA essentially consists of large array of gates which are programmable and can be reconfigured anytime anywhere. "Large array of gates" is an oversimplified description of FPGA [7]. FPGA is to be sure considerably more perplexing than basic show of Gates. At the same time the fact is, there are numerous doors inside the FPGA, which could be self-assertively associated together to make a circuit of your decision. FPGAs are fabricated by organizations like Xilinx, Altera, Actel and so on. FPGA‘s are in a broad sense like CPLD‘s yet CPLD‘s are little in size and capacity contrasted with FPGA.

Verilog is a Hardware Description Language (HDL) which could be utilized to portray advanced circuits in a text based way. We have to compose our system for FPGA utilizing a HDL like Verilog. Before HDL‘s were famous, specialists made use of everything with schematics. Which are radiantly simple with little outlines, yet are excruciatingly unmanageable for an expansive design.

Example: Priority encoder In FPGA Programming Using Verilog, VHDL

Priority encoder is a circuit that converts encoded inputs to the binary form. The binary representation of original number from priority encoder circuit represent from zero to most significant bit. By acting on highest priority request they control interrupt requests.

An 8-bit priority encoder is circuit which is used for converting an encoded input to a binary representation.

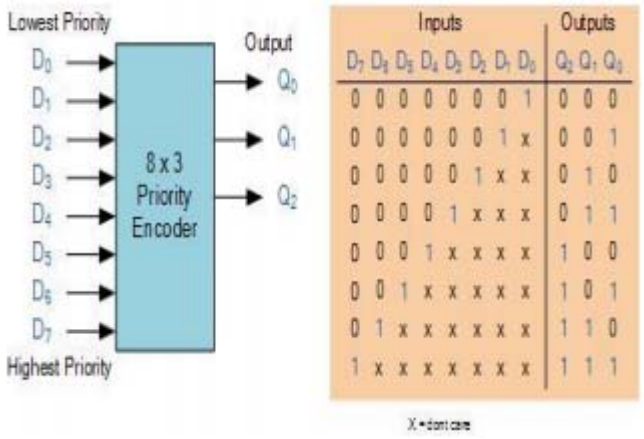

Figure 2. Basic Diagram & Truth Table

Depending on the no of data input lines the digital encoder produce 2, 3, 4 bit output lines. An n-bit encoder circuit has 2n input lines and n-bit output and include configurations like 4-to-2, 8-to-3 and 16-to-4 line. A binary equivalent of input value _1 'is generated by the encoder as output. The binary equivalent thus generated is available to encode either in decimal or hexadecimal input pattern as (binary coded decimal) BCD bit.

A) VHDL CODE:

```
Entity priority-encoder_8-3 is
Port (a: in logic_vector (7 down to 0);
b: out  logic_ vector (2 down to 0));
end priority-encoder_8-3;
Architecture Behavioral of priority_encoder_8_3 is
begin
Process (a)
begin
if a(0)='1' then b<="000";
elseif a(1)='1' then b<="001"; elseif
a(2)='1' then b<="010"; elseif
a(3)='1' then b<="011"; elseif
a(4)='1' then b<="100"; elseif
a(5)='1' then b<="101"; elseif
a(6)='1' then b<="110"; elseif
a(7)='1' then b<="111"; else null;
end if;
end process;
```

B) Verilog Code:

```
module pri (a,b);
input [7:0] a; output
[2:0] b; reg [2:0]
b;
always@(a)
begin
if (a[0]) b<= 3'b000;
else if (a[1]) b <= 3'b001;
else if (a[2]) b <= 3'b010;
else if (a[3]) b <= 3'b011;
else if (a[4]) b <= 3'b100;
else if (a[5]) b <= 3'b101;
else if (a[6]) b <= 3'b110;
```

```
else if (a[7]) b <= 3'b111;
else
b <= 3'bxxx;
end
end module
```

In this example, I am explaining Verilog Code in Xilinx as shown as Figure 3

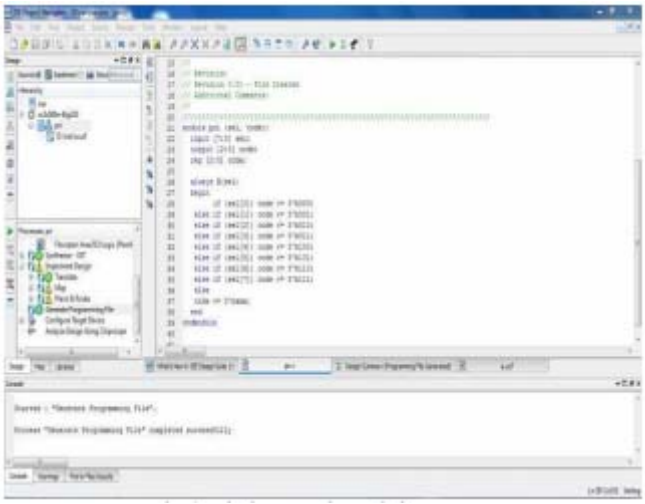

Figure 3. Priority Encoder code in XILINX

After writing the code in Xilinx to synthesize the problem it shows errors/warning (If in this program having errors/warnings) or Running (it contains perfect code). After competition of synthesize by click on View RTL Schematics as shown as Figure 4.

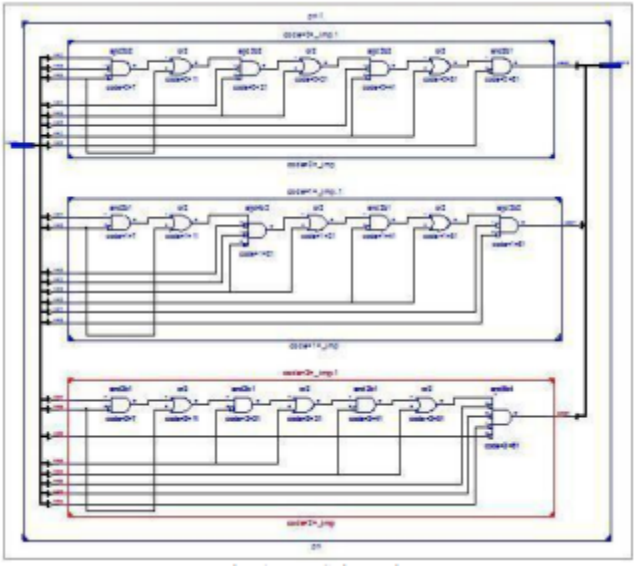

Figure 4. RTL Schematics

After checking the errors we can execute the program in Modelsim output waveforms as shown as Figure 5.

Figure 5. Output Waveforms

If one wants program Execute in hardware kit (Spartan 3E FPGA Starter kit, Spartan6 FPGA Kit) you have to create UCF file by using generating Programming file. User Constrain Files are American Standard Code for Information Interchange (ASCII) files specifying constraints on the logical design. You can create these files and enter your I/O interfaces with any text editor Based on hardware kit. If you want Complete System of Testing FPGA explained [8]. One also uses the Constraints Editor to create constraints within UCF files. These constraints affect how the logical design is implemented in the target device. These Files are used to override constraints specified during design entry

The Xilinx software still uses "last constraint wins" much same as HDL/NCF/UCF/PCF processing. Presently, the UCF files are handled with the request in which they are added to the Task (either in the Project Navigator or via Tcl command), and it has no bearing on timestamps or the order in which the documentation were adjusted, automatic generation of VHDL code [9], UML diagrams explained how synthesize in VHDL [10]

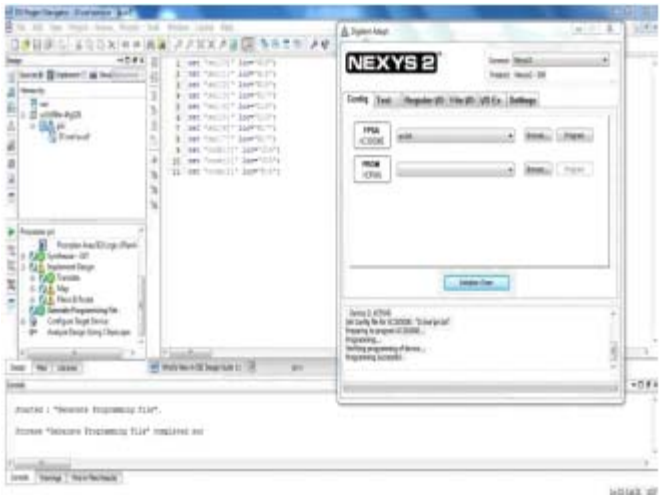

Figure 6. Loading UCF File

After creating UCF file, we have to connect hardware kit. Then upload the program using DIGILENT software. Based on code Program will be executed

3. PROBLEM DESCRIPTION

In this research we are going to proposed the FPGA Programs using Graphical Language rather than Verilog, VHDL with the help of LabVIEW.

Based on following points we will execute FPGA program in LabVIEW.

i) Launch LabVIEW software

ii) Draw Graphical diagram in Block Diagram Window

iii) Insert DAQ Assistants for input (Acquire signal) and output (generating Signal)

iv) Select input port lines and output port lines
v) DAQ Assistant gives single input and single output
With the help of Index array an build array we will design more than one input and one output
vi) Competition of Graphical diagram connect FPGA Kit
vii) Execute the program

LabVIEW programs are also called as Virtual Instruments, or VIs, because it seems & operation like as physical element or original element, such as oscilloscopes and multimeter's (based on input oscilloscope will change). LabVIEW provides a complete set of tools for analysing, displaying, and storing data and for troubleshooting the code.

When Launch LabVIEW, initially shows ―Getting Started‖ window. As shown as Figure 7.

Figure 7. Lab VIEW Getting Started Window

To create a new VI, select Blank VI (i.e. LabVIEW programs stored in VI) or to create a new LabVIEW project and select Empty project.

On clicking the blank VI it shows two windows one is front panel window and the other is block diagram window. Front panel is the user interface component and the block diagram shows the functionality of program.

Example: Priority encoder In FPGA Programming Using Graphical Language

When we launch LabVIEW. we have to insert DAQ Assistant in Block diagram window.

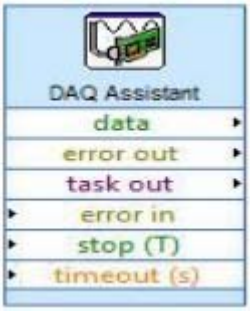

Figure 8. DAQ Assistant

By keeping this Function on the block diagram, a new task is created by DAQ Assistant, and for continuous measurement or generation a While loop is placed around DAQ Assistant.
To make the task globally accessible from any application, you must convert the Express VI to an NI-DAQ task saved in MAX [11-12].

You can generate NI-DAQmx API code from a DAQ Assistant Express VI. Right click on the DAQ Assistant Express VI and select generate NI-DAQmx Code from the shortcut menu to generate both configuration and example code for the task.
In our Example Priority encoder has 8 inputs and 3 outputs then we can use Index array and Build array.
INDEX ARRAY: It contains n-dimensional array. If ndimensional array contains no elements then sub array present in INDEX ARRAY returns the default value of the defined data type. The number of index inputs in the array matches the number of dimensions in n-dimensional array.

Index array Function Contains n-Dimension array, index acts as controls and element or sub array acts as indicator. In our Example 8-bit parity encoder then we can set 8-Dimension array we are giving Index (0 to 7), Then automatically it generates 8 elements or sub arrays.
BUILD ARRAY: It has only input available upon the placement of function. To add input to the node make a right click and select the option Add Input from the menu if you wire control references of different classes to this function.

The Build Array function contain Element and Array acts as Controls appended array acts as Indicator. In our example 8-bit parity encoder gives 3 outputs then we can set 3 elements in Build array it Appended array connects to the DAQ Assistant2 data.

After constructing index array and Build array in block diagram window to draw Graphical diagram. In priority encoder total graphical diagram in Block Diagram window and LabVIEW as shown as figure.

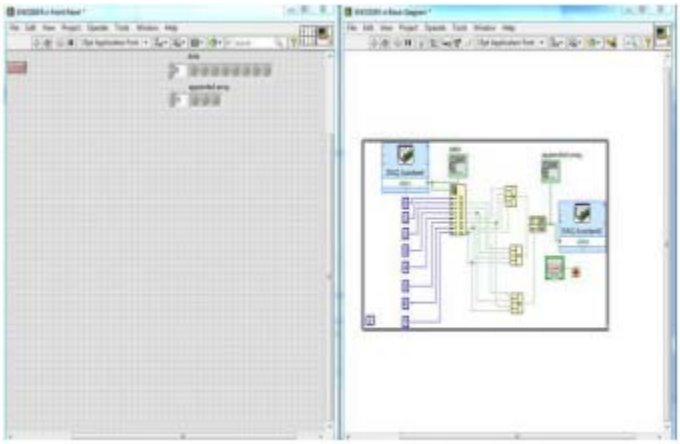

Figure 9. Priority encoder Graphical diagram in LabVIEW

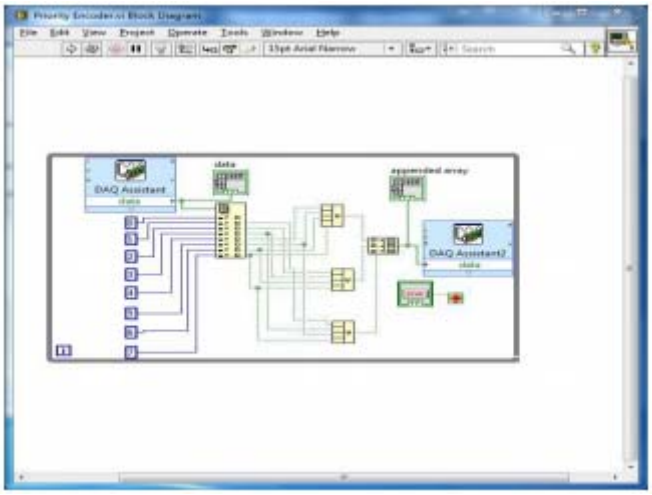

Figure 10. Priority encoder Graphical diagram in Window

4. RESULT

When we do FPGA Programming in Lab VIEW rather than Verilog or VHDL Based on Procedure easily we get results rather than Traditional programming Language in priority encoder after competition of Graphical Diagram we have to connect FPGA Kit and Run the Lab VIEW, it shows input and output in FPGA Kit and Front panel Window. Result in LabVIEW as shown as figure 11 in LabVIEW.

5. CONCLUSION

LabVIEW FPGA side-steps the need for VHDL or Verilog knowledge and allows novices and experts alike to take advantage of FPGA hardware. LabVIEW FPGA employs G programming and provides a high level of abstraction for translating signal processing algorithms to code that can run on hardware. The environment provides power debug and compilation features to helps ease FPGA application development.

REFERENCES

[1] S. Belkacemi, K. Benkrid, A. Benkrid, "Efficient FPGA hardware development: A multi-language approach", *Journal of Systems Architecture*, 53, (2007), 184–209.
[2] The Xilinx HDL Homepage. Available from: http://www.xilinx.com/labs/lava/index.htm.
[3] George A, Alastair M. Smith, Member IEEE,. Constantinides, Senior Member, IEEE, and Peter Y.K. Cheung, Senior Member, IEEE, "FPGA Architecture Optimization Using Geometric Programming", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 29, No. 8, August 2010.
[4] VHDL Language guide, http://ece.wpi.edu/~wrm/Courses/EE3810/geninfo/Welcome%20%20the%20VHDL%20Language.pdf.
[5] Brunini, Marilza A. Lemos Danilo M. Galdenoro Botura Jr. Marcio A. Marques Luiz Carlos Rosa, "Virtual Instrumentation: A Practical Approach to Control and Supervision Process", 2011 International Conference on Computer Science and Network Technology 978-1-4577-1587-7/11/$26.00 ©2011 IEEE.
[6] B.M. Dunkin and T.L. Schwartz, "Facilitating interdisciplinary hands-on learning using LabVIEW", *Int. J. Eng. Educ*., vol. 16, no. 3, pp. 218–227, 2000.
[7] Aiwu Ruan, Bairui Jie, Li Wan, Junhao Yang, Chuanyin Xiang, Zujian Zhu, Yu Wang, "A bitstream readback-based automatic functional test and diagnosis method for Xilinx FPGAs", *Microelectronics Reliability* (2014).
[8] Ignacio Bravo, Alfredo Gardel, Beatriz Perez, Jose Luis Lázaro, Jorge García, David Salido, "A new approach to evaluating internal Xilinx FPGA resources", *Journal of Systems Architecture*, 57, (2011), 749–760 .
[9] P. Martín a,∗, E. Buenoa, Fco. J. Rodríguez a, O. Machadoa, B. Vuksanovic, "An FPGA-based approach to the automatic generation of VHDL code for industrial control systems applications: A case study of MSOGIs implementation", *Mathematics and Computers in Simulation*, 91 (2013), 178–192.
[10] Stephen K. Wood, David H. Akehurst, Oleg Uzenkov, W. Gareth J. Howells, and Klaus D. McDonald-Maier, "A Model-Driven Development Approach to Mapping UML State Diagrams to Synthesizable VHDL". *IEEE Transactions on Computers*, Vol. 57, No. 10, October 2008.
[11] Ho Mann, H.G. Essel, N. Kurz, R.S. Mayer, W. Ott, D. Schall, "The new data acquisition system at GSI", *IEEE Trans. Nucl. Sci*., vol. 43, no. 1, pp. 132–135, Feb. 1996
[12] H Pichlik and R. Jamal, *LabVIEW Applications and Solutions*. Englewood Cliffs, NJ: Prentice-Hall, 1999, see also http://www.ni.com/