Hardware Implementation of Densely Packed Decimal Encoding

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Manish Kumar Gupta
(10606068)

Department of Computer Science and Engineering

National Institute of Technology Rourkela

May 2010

Hardware Implementation of Densely Packed Decimal Encoding

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Manish Kumar Gupta

under the guidance of

Dr. Ashok Kumar Turuk

Department of Computer Science and Engineering

National Institute of Technology Rourkela

May 2010

National Institute of Technology Rourkela

CERFTIFICATE

This is to certify that the thesis entitled, "**HARDWARE IMPLEMENTATION OF DENSELY PACKED DECIMAL ENCODING**" submitted by **MANISH KUMAR GUPTA** in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in COMPUTER SCIENCE AND ENGINEERING at the National Institute of Technology, Rourkela is an authentic work carried out by him under my supervision and guidance.

PLACE: NIT Rourkela

DATE:

Dr. Ashok Kumar Turuk

Associate Professor

National Institute of Technology

Rourkela – 769008

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude and indebtedness to Dr. Ashok Kumar Turuk, Department of Computer Science and Engineering, N.I.T. Rourkela for introducing the present topic and for his inspiring guidance, valuable suggestions and support throughout this project work.

I am very much thankful to Electronics Department and members of ESD lab for providing me all the equipments and facilities in the Embedded Systems Lab to carry out my implementation work successfully. I am also thankful to all my professors, lecturers and members of the Computer Science Department for their generous help in various ways for the completion of the thesis work.

I thank my family members for encouraging me at every stage of this project work. Last but not least, my sincere thanks to all my friends who have patiently extended all sorts of help for accomplishing this undertaking.

Manish Kumar Gupta

Table of Contents

List of Figures

ABSTRACT

Binary Coded Decimal (BCD) in which four bits are used for each decimal digit is a widely used encoding for decimal data .Decimal arithmetic and shifting are simplified by using operands in this form, and both rounding to a specified number of digits and conversions to or from characters are trivial. For the storage and simple manipulation of decimal data, BCD remains an appropriate encoding to use. In some situations, however, a more compact representation offers significant advantages. Decimal floating-point numbers in a compact form can be used to implement the requirements of the IEEE 854 standard and meet the increasing demands for decimal arithmetic in applications.

An efficient encoding scheme for decimal data is described by Chen and Ho.Chen Ho encoding is a lossless compression of three decimal digits coded in BCD into 10 bits using an algorithm which can be applied or reversed using only simple Boolean operations. Densely Packed Decimal (DPD) is an refinement of the Chen ho encoding. It gives the same compression and speed advantages but is not limited to multiples of three digits. The DPD encoding allows arbitrary-length decimal numbers to be coded efficiently while keeping decimal digit boundaries accessible. This results in efficient decimal arithmetic and makes the efficient and optimized use of available resources such as storage or hardware registers.

This thesis embodies the work done to implement the Densely Packed Decimal (DPD) encoding on hardware using digilent board containing VIRTEX-II Pro FPGA.

Chapter 1

Introduction

Motivation
Objective

Chapter 1

1 Introduction

In the real world people mostly use decimal arithmetic ; however present day digital electronics is based on binary signals hence digital computers perform only binary arithmetic. Recently in applications such as financial and commercial, decimal arithmetic at machine level has received many interests. Decimal notations are strongly being used for machine input-output and this gives a clear indication that users prefer to use decimal arithmetic. This preference is because of the applications that require very precise calculation which can not be supported by binary arithmetic as decimal fractions are not supported very well here as far as precision is concerned. For example, the value of 0.1 in binary requires infinite recurring binary fractions. In contrast, a decimal number system can represent 0.1 precisely. The software implementation of decimal arithmetic eliminates these conversion errors, but it is typically 100 to 1000 times slower than binary arithmetic [2].

The most common encoding that is widely used for decimal data is Binary Coded Decimal Encoding (BCD) in which a single decimal digit is represented by four bits. The use of operands in this encoding simplifies the decimal arithmetic and shifting, and both rounding to a specified number of digits and conversions too or from characters are trivial.BCD has always remained an appropriate encoding to use for the storage and simple manipulation of decimal data. However, in some situations it is advantageous to use a more compact form of representation. For example, only 32 BCD digits can be stored in a 128- bit hardware register, but in the same space 33 decimal digits along with a sign and a 4-digit exponent can be held if a more efficient representation is used.[1]

1.1 Motivation

As the demand of use of decimal arithmetic in various computing application is increasing the use of decimal floating point number in compact form can be a appropriate solution. Specifically, decimal addition, subtraction, shifting, rounding, and conversions to character form are significantly simplified by the preservation of decimal digit boundaries.

Chen and Ho [5] described a scheme for encoding decimal data which is quite efficient. The Chen-Ho encoding, as it is called, compresses three decimal digits into 10 bits with very little waste, giving a 17% more compact encoding than BCD(which uses 12 bits to store three decimal digits).It uses a Huffman code[4],with most significant bits selecting various digit combinations. The main advantage of Chen-Ho encoding over binary representation in 10 bits is that only simple Boolean operations are needed for conversion to or from BCD; multiplication and divisions are not required..

Another encoding technique proposed by M. F. Cowlishaw [1], Densely Packed Decimal (DPD), uses an equivalent encoding to the Chen-Ho scheme, but it is an improvement over Chen-Ho encoding and hence has further advantages. The main advantage over Chen-Ho encoding is that it is not restricted to the fact that for compression the decimal digits should be in multiple of three, which is the primary requirement of Chen-Ho encoding.

The cost of deploying hardware is decreasing day by day and the significant hardware issues like operation speed and storage efficiency attracts the attention of hardware designers to add a decimal arithmetic unit to CPUs to perform decimal calculations. The recent developments in embedded systems technology and FPGA solutions have also motivated the use of a separate hardware system for applications involving fast operation speeds.

Now a days various hardware platforms are available that provide control of both logic and software code by a soft file. Because of this, the low cost of design maintenance and degree of design reuse is greatly enhanced. Xilinx, Altera are some of the platform providers.

The platforms available like Digilent board, Altera Kit etc. contain FPGA to carry out design, implementation, and real time simulation of hardware. Softwares like the Xilinx ISE system is an integrated design environment that consists of a set of programs to create (capture), simulate and implement digital designs in an FPGA target device.

It offers an attractive and easy to use GUI along with online help. By providing integrated tools for HDL entry, synthesis, implementation, and verification in a free downloadable environment, ISE 10.1 helps users rapidly achieve design goals while reducing overall project cost.

1.2 Objective

- The objectives of this project is to implement the Densely Packed Decimal Encoding, proposed by M. F. Cowlishaw [1], on hardware (using Virtex-II Pro FPGA Platform) and simulate the hardware design using available hardware and software platforms .

The thesis contains the implementation details of Densely Packed Decimal (DPD) encoding using hardware description language VHDL, Xilinx ISE WebPACK 10.1 for simulation, design, and implementation on FPGA of target device ,Xilinx ChipScope Pro for debugging and output and Digilent Board containing VIRTEX-II Pro FPGA as target device.

Chapter 2

Literature Survey

Binary Coded Decimal Encoding
Chen-Ho Encoding
Densely Packed Decimal Encoding

Chapter 2

2 Literature Survey

2.1 Binary Coded Decimal Encoding

To BCD-encode a decimal number using the common encoding, each digit is encoded using the four-bit binary bit pattern for each digit. For example, the number 127 would be:

000100100111

Since most computers store data in eight-bit bytes, there are two common ways of storing four bit BCD digits in those bytes ,each digit is stored in one byte, and the other four bits are then set to all zeros, all ones (as in EBCDIC code) or to 1011(as in the ASCII code) two digits are stored in each byte .

A widely used variation of the two-digits-per-byte encoding is called "packed BCD", where numbers end with a sign 'digit', for which the preferred values are 1100 for + and 1101 for −. In packed BCD the number 127 would be represented as the bytes 00010010 01111100, and −127 as 0001001001111101.

While BCD does not make optimal use of storage (about 1/6 of the available memory is not used in packed BCD), conversion to ASCII, EBCDIC, or the various encodings of Unicode is trivial, as no arithmetic operations are required. More dense packings of BCD exist; these avoid the storage penalty and also need no arithmetic operations for common conversions.

Unlike pure binary encodings large numbers can easily be displayed by splitting up the nibbles and sending each to a different character with the logic for each display being a simple mapping

function. Converting from pure binary to decimal for display is much harder involving integer multiplication or divide operations. The BIOS in PCs usually keeps the date and time in BCD format, probably for historical reasons (it avoided the need for binary to ASCII conversion).

If a decimal digit requires four bits, then three decimal digits require 12 bits. However, since $2^{10}>10^3$, if three decimal digits are encoded together then only 10 bits are needed. Two such encodings are *Chen-Ho encoding* [5] and *Densely Packed Decimal* [1]. The latter has the advantage that subsets of the encoding encode two digits in the optimal 7 bits and one digit in 4 bits, as in regular BCD.

2.2 Chen-Ho Encoding

The Chen-Ho encoding[5] allows three decimal digits to be represented in ten binary bits, which may have up to 1,024 possible different values, and which therefore can encode the 1,000 possibilities for three digits with only a little waste.

The advantage that is incurred from Chen–Ho encoding over a straightforward binary representation in 10 bits is that only simple Boolean operations are needed for conversion to or from BCD; multiplications and divisions are not required. This encoding also has the advantage over variable length schemes, because its fixed-length mapping allows simpler encoding and decoding in either hardware or software.

The Chen–Ho scheme works satisfactorily when decimal numbers have lengths which are multiples of three decimal digits, as this encoding packs three digits into 10 bits with little waste. It is less satisfactory for arbitrary lengths, however, because either digits must be wasted or more than one encoding has to be used to represent the various digits of the number.

2.3 Densely Packed Decimal Encoding

Densely Packed Decimal Encoding proposed by M. F. Cowlishaw [1] is an improvement over Chen-Ho encoding. It uses the coding scheme equivalent to the Chen-Ho but instead of using Huffman-code it uses a fresh arrangement of bits that gives it further advantages over the Chen–Ho scheme. The advantages of DPD over Chen-Ho encoding can be listed as follows:[1]

1. The encoding of decimal digit is not restricted to the fact that the number of decimal digits always be a multiple of three. It can encode arbitrary number of decimal digits. One or two decimal digits are compressed into the optimal four or seven bits respectively.
2. The encoded decimal numbers can be expanded into a longer field simply by padding with zero bits; re-encoding is not necessary. While Chen-Ho encoding requires a re-encoding instead of simple padding if an encoded two digits is expanded into three digit field.
3. When numbers in the range 0 through 79 are encoded by this scheme they have the same right-aligned encoding as in BCD. While in Chen-Ho encoding only the numbers 0 through 7 remains same as in BCD.

These advantages make the new encoding a better choice than Chen-Ho encoding for both hardware and software representations of decimal numbers.
Here are some examples of encoding in BCD , Chen-Ho and Densely Packed Decimal[1]:

Decimal	BCD	Chen-Ho	Densely Packed
005	0000 0000 0101	000 000 0101	000 000 0101
009	0000 0000 1001	110 000 0001	000 000 1001
055	0000 0101 0101	000 010 1101	000 101 0101
099	0000 1001 1001	111 000 1001	000 101 1111
555	0101 0101 0101	010 110 1101	101 101 0101
999	1001 1001 1001	111 111 1001	001 111 1111

Fig 1 : Examples of BCD, Chen-Ho and DPD Encoding

Details of the encoding

The DPD encoding, categorizes each of the three digits as follows:

- Small ,when in the range 0-7 and requires 3 bits.
- Large, when in the range 8-9 and requiring one bit.

The most significant bit of each BCD digit is 0 for small values, and 1 for the large values.

The possible combinations of these ranges are then [1] :

Number of small digits	Possibilities	Number of bits required for digits	Number of bits to indicate combination
All	51.2%	3+3+3	1
Two	38.4%	3+3+1	3
One	9.6%	3+1+1	5
Zero	0.8%	1+1+1	7(only 5 needed)

Chapter 3

Implementation

Design of DPD System

System Platforms

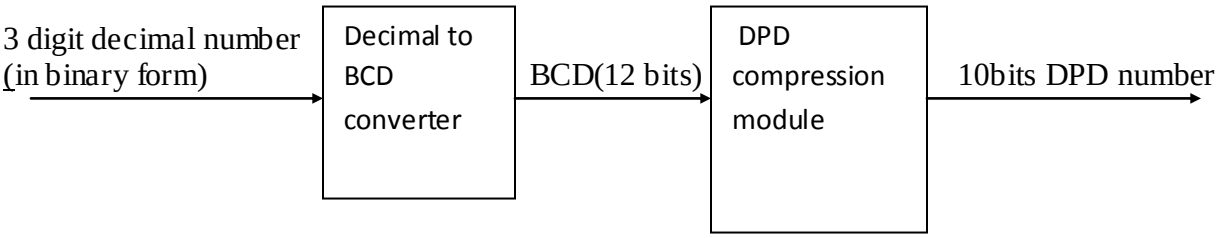
Chapter 3

3 Implementation

3.1 Design of DPD System

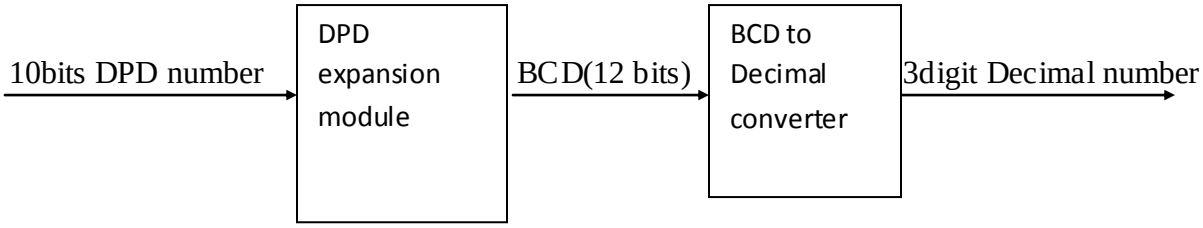
The compression and expansion block of Densely Packed decimal encoding system is designed as to implement the DPD encoding which is proposed by M. F. Cowlishaw [1]:

Compression Block



For compression , three digit decimal number is passed as input to the compressing block where Decimal to BCD converter converts the binary number in BCD encoded number of 12 bits.This 12 bit BCD number is then fed to DPD compression module which finally compresses the number and encode it in 10 bits densely packed decimal form.

Expansion Block



For expansion the 10 bits DPD number is now the input to the expansion block where first it is expanded to BCD form by DPD expansion module.The 12 bits BCD output is then converted to decimal number in simple binary format by BCD to decimal converter.

3.1.1 Decimal(Binary) to BCD converter

Before the decimal number, which is in simple binary format, is compressed using DPD encoding it is converted into BCD form, since DPD works upon the decimal number encoded in BCD. For conversion to BCD, a BCD converter module in VHDL is developed that takes 10 bits decimal number in binary format and converts into 12 bits of BCD output.

The Binary to BCD converter works in following steps [10]:

1. The binary number is shifted to left by 1 bit.
2. Three columns are devised hundreds, tens and units of 4 bit each from left to right. As the binary number is shifted to left one by one ,after 8 shifts the number is in hundreds, tens and units column with most significant bit in hundreds column.
3. If the value of number in any of the BCD column is 5 or greater, 3 is added to that BCD column.
4. Step 1 is performed again till every bit of binary number is shifted.

For adding three , a add-3 module is created which maps the value of column(when greater than 5) to their added result(result after adding three).
In the BCD to Binary converter module the add-3 module is used as a component and is shown as Fig 2 in the block diagram of binary to BCD converter Fig 3.

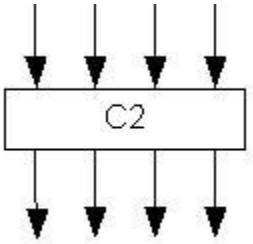


Fig 2 Add-3 module for binary to BCD converter

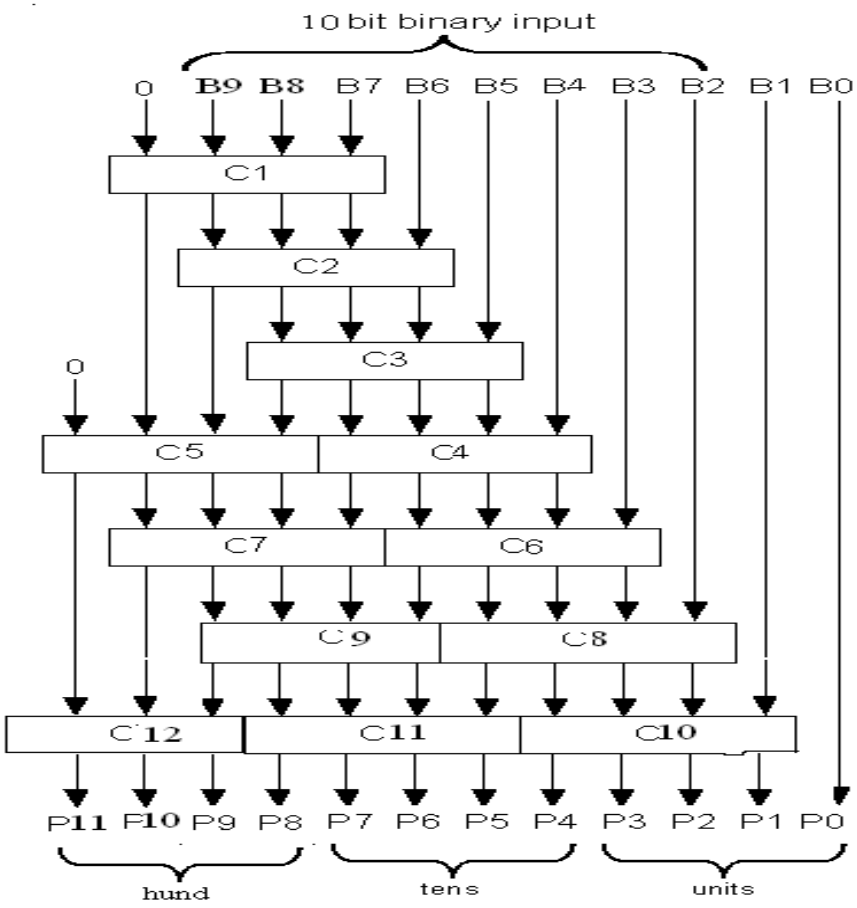


Fig 3 : Block Diagram of Binary to BCD Converter

3.1.2 Densely Packed Decimal Compression Module

Once the decimal number is converted to BCD format ,it can be compressed using DPD encoding.The encoding circuit is a simple combinational logic involving 12 inputs and 10 outputs and is given by following expressions in VHDL [1] :

```
p <= ((NOT a) AND b) OR (a AND j AND (NOT i)) OR (a AND f AND i AND (NOT e));
q<= ((NOT a) AND c) OR (a AND k AND (NOT i)) OR (a AND g AND i AND (NOT e));
r<= d;
s<=  ((NOT e) AND f AND (NOT( a AND i))) OR ( (NOT a) AND (NOT i) AND e AND j)
OR ( e AND i );
```

t<= ((NOT e) AND g AND (NOT (a AND i))) OR ((NOT a) AND (NOT i) AND e AND k) OR (a AND i);

u<= h;

v<= a OR e OR i;

w<= a OR (e AND i) OR ((NOT e) AND j AND (NOT i));

x<= e OR (a AND i) OR ((NOT a) AND k AND (NOT i));

y<= m;

where a,b,c,d,e,f,g,h,I,j,k,m represents 12 bits of input BCD number and p,q,r,s,t,u,v,w,x,y signifies 10 bits of output DPD decocded number.

3.1.3 Densely Packed Decimal Expansion Module

The expansion or decoder circuit is also a simple combinational logic involving boolean operations.The input to the expansion module is 10 bits DPD decoded number and output is 12 bits BCD number.The module is written in VHDL and contains following expressions [1]:

a<= (v AND w) AND ((NOT x) OR (NOT s) OR (s AND t));

b<= p AND ((NOT v) OR (NOT w) OR (s AND (NOT t) AND x));

c<= q AND ((NOT v) OR (NOT w) OR (s AND (NOT t) AND x));

d<= r;

e<= v AND (((NOT w) AND x) OR (((NOT t) OR s) AND w AND x));

f<= (s AND ((NOT v) OR ((NOT x) AND v))) OR (p AND (NOT s) AND t AND v AND w AND x);

g<= (t AND ((NOT v) OR ((NOT x) AND v))) OR (q AND (NOT s) AND t AND v AND w AND x);

h<= u;

i<= v AND (((NOT w) AND (NOT x)) OR (w AND x AND (s OR t)));

j<= ((NOT v) AND w) OR (s AND v AND (NOT w) AND x) OR (p AND v AND w AND ((NOT x)OR ((NOT s) AND (NOT t))));

k<= ((NOT v) AND x) OR (t AND v AND (NOT w) AND x) OR (q AND v AND w AND ((NOT x)OR ((NOT s) AND (NOT t))));

m<= y;

3.1.4 BCD to Binary Converter

After the decoding is over by DPD expansion module, the number is in BCD format and needs to be converted to decimal binary form.This conversion is carried out by BCD to binary converter which is developed in VHDL as a module that takes 12 bits BCD number as input and produces 10 bits decimal number in binary as output.

The converter is developed from the idea that a BCD number is comprised of hundreds, tenth and unit for a three digit number.Suppose the three digit number is represented in BCD as:

(a b c d) (e f g h) (i j k l)

H T O

For conversion to decimal number ,say , N

N = 100H + 10T + O

To achieve this value of 'N' following steps are followed:

1. The number 'H' is shifted to left three times

   H --> 8H ……..(1)

2. The number 'H' is again shifted to left one time

   H --> 2H……(2)

3. Adding (1) , (2) and number 'T' we have,

   10H + T

4. Shifting (10H + T) to left three times we get,

   (10H + T) - -> 8(10H + T) ……(3)

5. Shifting (10H + T) to left one time

   (10H + T) - -> 2(10H + T)……(4)

6. Adding (3) , (4) and number 'O' we have the final result.

   80H + 8T

   20H + 2T

   + O

   N = **100H + 10T + O**

3.2 System Platforms

The DPD system designed as described above is implemented using both software and hardware platforms.Thus it can be said that the DPD system is a hardware-software codesign. The hardware and software platforms are well described in the following subsections.

3.2.1 Hardware Platform: Virtex-II Pro Platform

The Virtex-II Pro™/Virtex-II Pro X Platform FPGA solution is the most technically sophisticated silicon and software product development in the history of the programmable logic industry.Leading teams from top embedded systems companies worked together with Xilinx software teams to develop the systems software and IP solutions that enabled the new system architecture paradigm. The result is the first Platform FPGA solution capable of implementing high performance system-on-a-chip designs previously the exclusive domain of custom ASICs, yet with the flexibility and low development cost of programmable logic. The Virtex-II Pro/Virtex-II Pro X family marks the first paradigm change from programmable logic to programmable systems, with profound implications for leading-edge system architectures in networking applications, deeply embedded systems, and digital signal processing systems. It allows custom user-defined system architectures to be synthesized, next-generation connectivity standards to be seamlessly bridged, and complex hardware and software systems to be co-developed rapidly with in system debug at system speeds. Together, these capabilities usher in the next programmable logic revolution [7].

Each of the larger devices of the family incorporates one or two small yet powerful IBM® PowerPC™ 405 processor cores, each capable of more than 300 MHz clock frequency and 420 Dhrystone MIPS [7]. The PowerPC 405 cores are fully embedded within the FPGA fabric, where all processor nodes are controlled by the FPGA routing resources. This provides the utmost architectural capability, where complex applications may be efficiently divided between high-speed logic implementation and high-flexibility software implementations. The Virtex-II Pro/Virtex-II Pro X products are based on the most advanced FPGA fabric available. The Virtex-II Pro/Virtex-II Pro X family is the first FPGA family to incorporate both serial transceiver technology and a hard processor core within a general-purpose FPGA device.

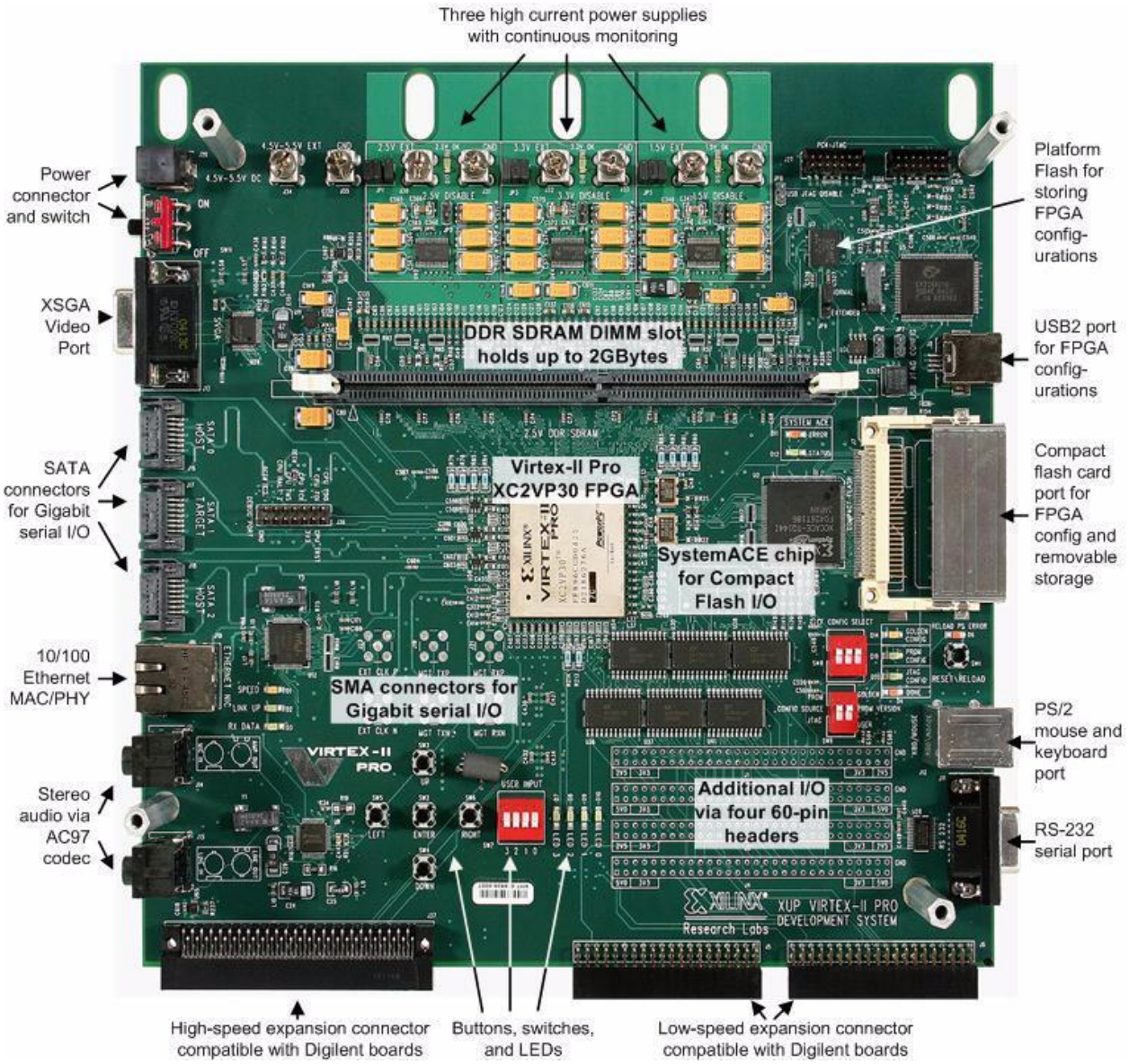


Fig 4 : XUP Virtex-II Pro Development System Board Photo

It is significant that the embedded systems enabled by Virtex-II Pro solutions are "all-soft," in that both logic and software code are controlled by a soft data file. Because of this, the low cost of design maintenance and degree of design reuse is greatly enhanced.

Some of the salient features of Virtex-II pro development system board are shown in Fig 4.

3.2.2 Software Platform: Xilinx ISE WebPACK 10.1

The Xilinx ISE system is an integrated design environment that consists of a set of programs to create (capture) , simulate and implement digital designs in an FPGA or CPLD target device. It offers an attractive and easy to use GUI along with online help.

ISE WebPACK 10.1 software offers a complete front-to-back FPGA design solution allowing users to immediately begin projects. By providing integrated tools for HDL entry, synthesis, implementation, and verification in a free downloadable environment, ISE 10.1 helps users rapidly achieve design goals while reducing overall project cost. The various features of Xilinx WebPACK are described below:

Xilinx ISE Simulator

Key features are:-

1. Smart compile technology to realize faster run times
2. Tcl command console to easily transition from the ISE software graphical user interface to a command line environment.
3. Expanded integrated Timing Closure Environment to allow cross probing between timing analyzer, constraints editor and floorplan viewer for ease in design exploration of Virtex series designs
4. New power optimization in Xilinx Synthesis Technology (XST) and placement, together with improvements in routing, deliver an average of 10 percent lower dynamic power for the Spartan-3 generation of FPGAs.

Xilinx CORE Generator

The CORE Generator is a design tool that delivers parameterized Intellectual Property (IP) optimized for Xilinx FPGAs. The Core Generator provides the following ready-made functions which include:

- FIFOs and memories
- Adders and Subtractors

- FIR filters
- FFTs
- Standard bus interfaces such as PCI and PCI-X,

Xilinx ChipScope PRO

ChipScope Pro tool inserts logic analyzer, bus analyzer, and virtual I/O low-profile software cores directly into the design, allowing the programmer to view any internal signal or node, including embedded hard or soft processors. Signals are captured at or near operating system speed and brought out through the programming interface, freeing up pins for the design. Captured signals can then be analyzed through the included ChipScope Pro Logic Analyzer.

ChipScope Pro Key Features [9]:

- Analyze any internal FPGA signal, including embedded processor busses
- Inserts low-profile, configurable software cores either during design capture, or after synthesis
- All ChipScope Pro cores are available through the Xilinx CORE Generator System
- Enhancements to the Virtex-5 System Monitor console make it easier to access on-chip temperature, voltage, and external sensor
- Change probe points without re-synthesizing

The modules presented in Section 3.1 are developed in VHDL to carry out the implementation of Densely Packed Decimal (DPD) Encoding, proposed by M. F. Cowlishaw [1], using the software and hardware platform discussed and presented in Section 3.2. In the next Section we will see the simulation and results of design developed for the implementation.

Chapter 4

Simulation and Results

Simulation Setup

Results and Analysis

Chapter 4

4 Simulation and Results

This section presents the simulation and results of the Densely Packed Decimal Encoding System when implemented using hardware and software platforms as discussed in section 3 . In the following subsection we will see the simulation setup that includes steps and procedures to carry out the simulation and finally the results are analyzed.

4.1 Simulation Setup

Once the all VHDL modules are ready ,they should be simulated before they are put in real hardware. We can create a test bench waveform from the Project→New Source menu of ISE and it will assist in setting up the simulation.
The DPD encoder module when simulated using test bench with 999 as the BCD input , the simulation obtained is as shown in Fig 5.
The simulation result as obtained on DPD encoding is given as input to DPD decoder and the simulation obtained is as shown in Fig. 6.

Once we simulate our design and feel it is function correctly, then we can move on to generating the data needed to actually program the target device with our design.

First of all,create the UCF file taking help from the user manual of the target device.. The UCF file is an ASCII file specifying constraints on the logical design.The UCF file is required to communicate with our design which is implemented on target device using switches, push buttons , LED's etc. We create this file and enter our constraints in the file with a text editor. We can also use the Xilinx Constraints Editor to create constraints within a UCF file. These constraints affect how the logical design is implemented in the target device.

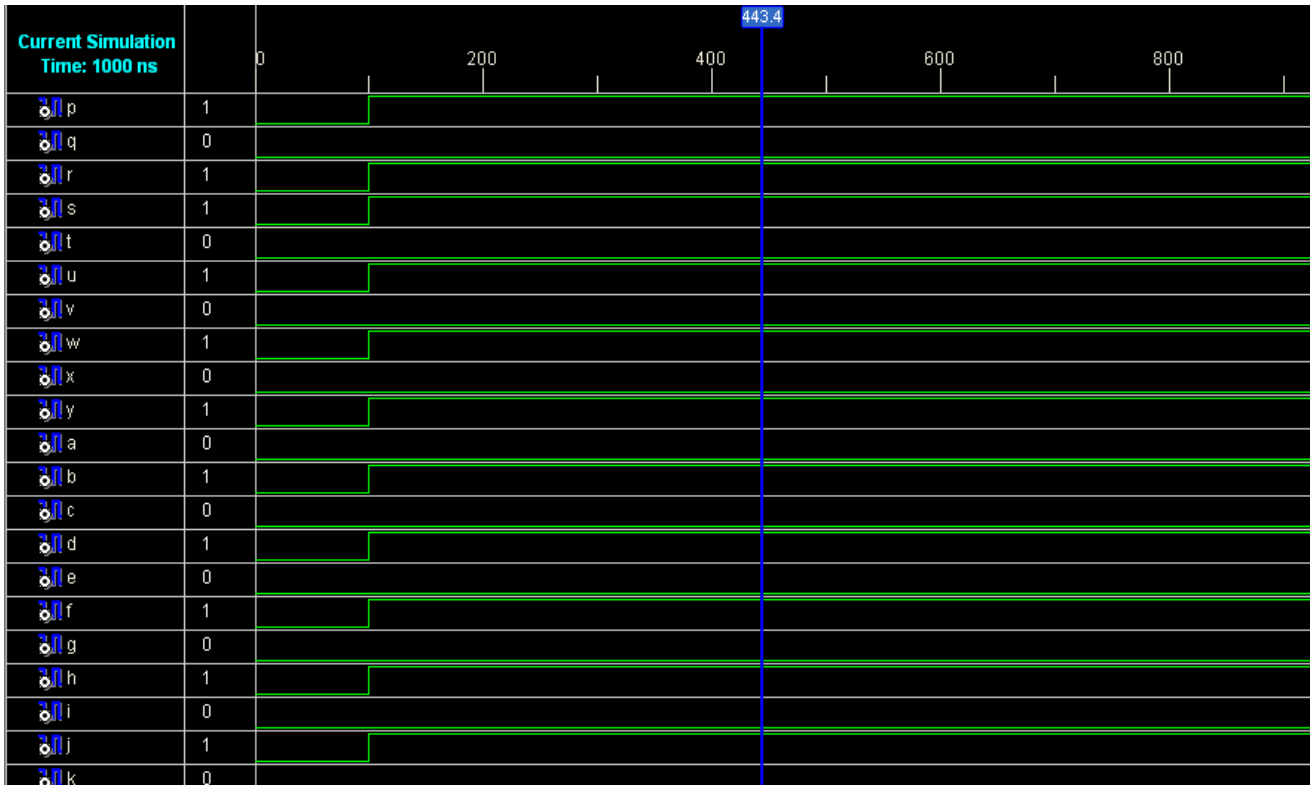


Fig 5 : Test Bench Simulation of DPD encoding with 999 as BCD input

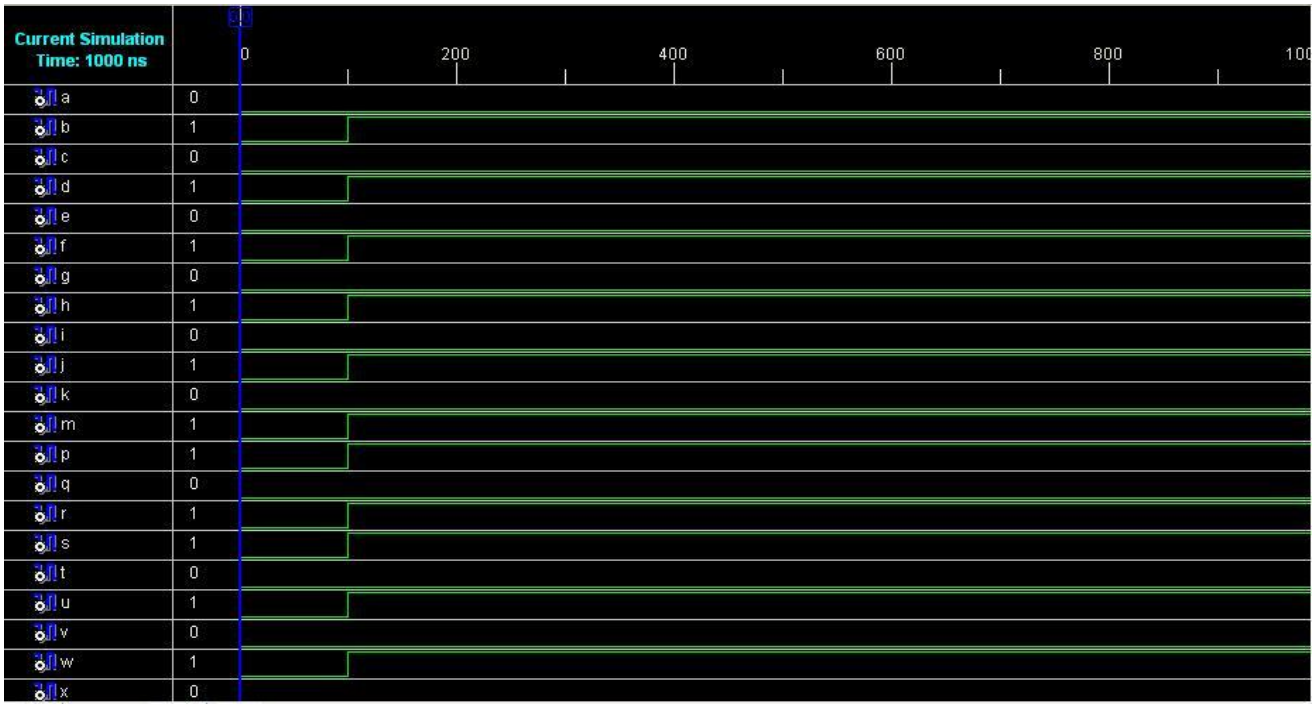


Fig 6 : Test Bench Simulation of DPD decoding with 999 as BCD output

The UCF file for our design contains following constraints:

```
NET "CLK" LOC ="AJ15 ";
NET "reset" LOC= "AC11";
NET "load"   LOC= "AD11";
NET "enable" LOC= "AF8";
```

We use Xilinx ChipScope PRO to debug and see the results, so we need to create a chipscope file with .cdc extension. This can be done by following steps:

- Right click on the top module of the design intended for verification or debugging, and select new source. Then select **ChipScope Definition and Connection File.**
- Give an appropriate file name and follow the directions. A new file (with file name as given) will be created in the source window under the project hierarchy.
- Double click on this new source file which causes the chipscope window to pop up. Follow the directions given and check the required settings. Select the number of **trigger ports** and their respective **widths** depending on the design requirement. Trigger ports would enable the the start of simulation later on when the design is implemented on target device.
- On Capture Parameters, uncheck the **Data Same As Trigger** option. Define **Data Width** as total width of signals that will be displayed on waveform for debugging. **Data Depth** defines the number of samples of run need to be displayed.
- On Net Connections tab, when Modify **Connections** is clicked, a window will pop up. Select the appropriate signals from the list of nets in the popped up window and make connections to the respective clock, trigger and data signals.
- Once all the connections are made press **OK**.Then press **Return to Project Navigator** and **Save Project** changes.

Now we have all the necessary files available for implementation on target device. It's time to go for synthesis and programming flow.

The first step is **synthesis.** This is the process of implementing the VHDL into logic gates. Once that is done, the next few steps are device and device vendor specific steps of **translating** the gates into physical implementations of the gate functionality. For example, our VHDL might synthesize a 2-input AND gate, but the technology only has 2-input NOR gates. These steps will

implement the AND gate as two NAND gates, for example. Once this technology **mapping** is done the design can be placed into the target device and **routed**. Finally, **generating programming file** will create a *bit file* that will actually program the device with the design.

Now double-click on the **Analyze Design Using Chipscope** under process window. This will open main window of chip-scope pro.

First of all ,click on JTAG chain to detect available JTAG connection to target devices attached. A new menu window appears, describing the detected device. Select the target Device, configure it using the generated program *bit file* and click OK. On click of OK ;the trigger window, the data window and the console window appear in the main window. Click on **FILE -> Import** and select the chipscope file that we had created as .cdc extension. Importing the file will change the name of signal as per our program in the trigger and waveform window.

Now change the trigger signals using target device switches, push buttons as mapped by UCF file and run the simulation .The changed waveform will appear as influenced by change in trigger signals.

4.2 Results and Analysis

The design when simulated and implemented on the target device , the device utilization results obtained are summarized in Fig 7

The DPD encoding of decimal number eighty (80) is simulated and the result obtained on chipscope is as shown in Fig 8.

The DPD decoding of encoded decimal number 80 is simulated and the result obtained on chipscope is as shown in Fig 9.

FinalDPD Project Status (04/15/2010 - 11:39:20)			
Project File:	FinalDPD.ise	**Current State:**	Programming File Generated
Module Name:	Encoding	• **Errors:**	No Errors
Target Device:	xc2vp30-7ff896	• **Warnings:**	6 Warnings
Product Version:	Standalone Programming Tools 10.1	• **Routing Results:**	All Signals Completely Routed
Design Goal:	Balanced	• **Timing Constraints:**	All Constraints Met
Design Strategy:	Xilinx Default (unlocked)	• **Final Timing Score:**	0 (Timing Report)

FinalDPD Partition Summary
No partition information was found.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	306	27,392	1%
Number of 4 input LUTs	353	27,392	1%
Logic Distribution			
Number of occupied Slices	337	13,696	2%
Number of Slices containing only related logic	337	337	100%
Number of Slices containing unrelated logic	0	337	0%
Total Number of 4 input LUTs	404	27,392	1%
Number used as logic	267		
Number used as a route-thru	51		
Number used as Shift registers	86		
Number of bonded IOBs	24	556	4%
IOB Flip Flops	6		
Number of RAMB16s	1	136	1%
Number of BUFGMUXs	2	16	12%
Number of BSCANs	1	1	100%
Number of RPM macros	14		

Fig 7 : Design Summary of the DPD system(encoding) when implemented on target device

Fig 8. Chipscope result of DPD Encoding decimal number 80.

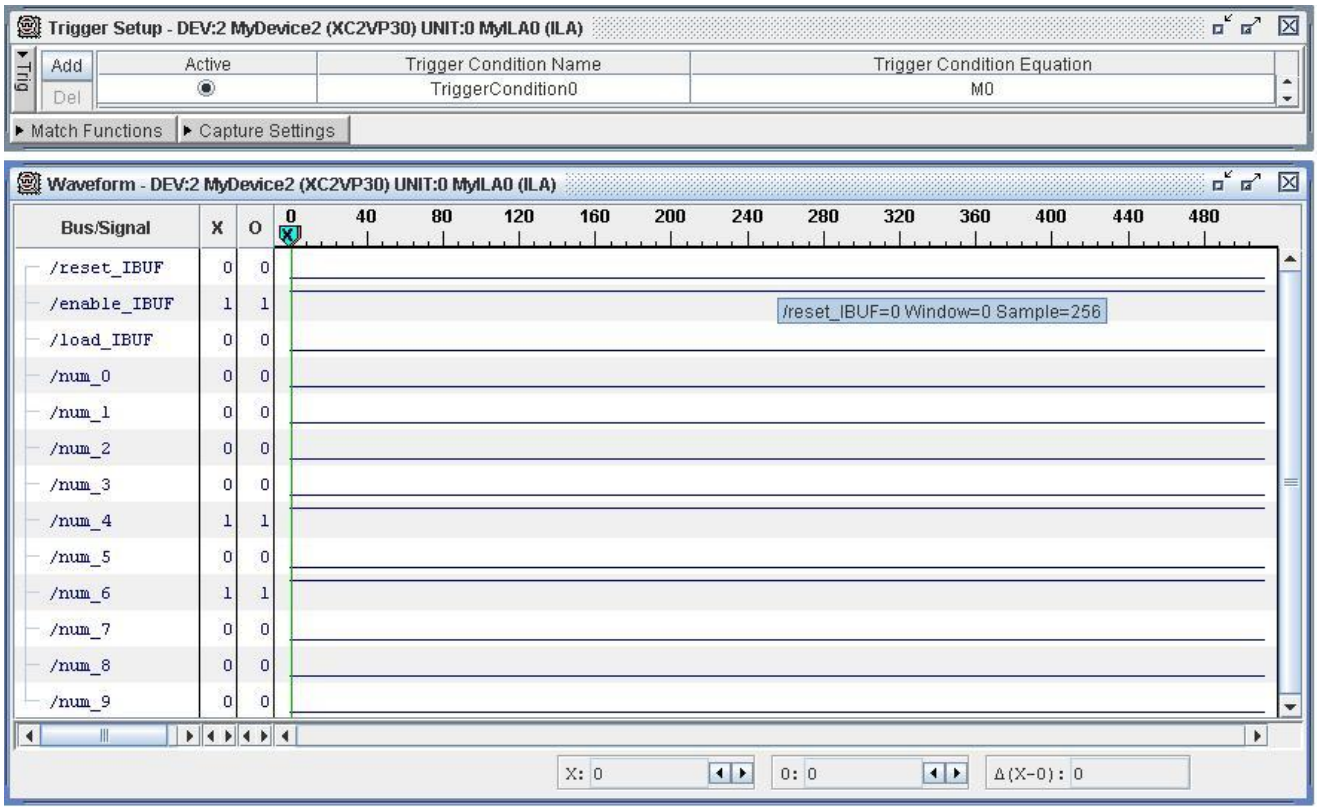


Fig 9. Chipscope result of DPD decoding of decimal number 80

Chapter 5

Conclusions and Future Work

Chapter 5

5 Conclusions and Future Work

The thesis covers the implementation of Densely Packed decimal encoding and decoding, the encoding technique proposed by M. F. Cowlishaw [1], on a Virtex–II Pro development platform. The DPD system developed successfully performs compression and expansion of decimal numbers.

The future work may consists of providing runtime input to system which can be done using EDK.Using the FPGA editor and Floor planner of the ISE the system can be made more efficient in terms of space acquired when deployed on a real hardware chip.

References

1. Cowlishaw, M. F., "Densely packed decimal encoding". IEEE Proceedings Computers and Digital Techniques (Institution of Electrical Engineers), 149 (3),pp.102-104,May 2002

2. Cowlishaw, M. F. , "Decimal Floating-Point: Algorithm for Computers", Proceedings of the 16th IEEE Symposium on Computer Arithmetic, pp.104-111, June 2003

3. Kaivani, A., Alhosseini A., Gorgin, S. and Fazlali M."Reversible Implementation of Densely-Packed-Decimal Converter to and from Binary-Coded-Decimal Format Using in IEEE-754R", pp.273-276, 9th International Conference on Information Technology (ICIT'06), 2006.

4. HUFFMAN, D. A.: "A method for the construction of minimum-Redundancy codes", Proc.IRE, 40,(9),pp.1098–1101,1952.

5. CHEN, T. C., and HO, T.: "Storage-efficient representation of decimal data", CACM, 18,(1),pp.49–52,1975.

6. Cowlishaw, M. F.,"Summary of Densely Packed Decimal encoding". *http://speleotrove.com/decimal/DPDecimal.html.*

7. Virtex-II Pro and Virtex-II Pro X FPGA User Guide, UG012 (v4.2) 5 November 2007.

8. Xilinx University Program,Virtex-II Pro Development System, Hardware Reference Manual UG069 (v1.2) July 21, 2009

9. "ChipScope Pro and the Serial I/O Toolkit", *http://www.xilinx.com/tools/cspro.htm*

10. "Binary to BCD converter", *http://www.engr.udayton.edu/faculty/jloomis/ece314/notes/devices/binary_to_BCD/bin_to_BCD.html*