Look-Up Tables (LUTs) for Multiple-Valued, Combinational Logic

Ali Sheikholeslami, Ryuji Yoshimura, and P. Glenn Gulak

Abstract

The use of Look-Up Tables (LUTs) is extended from binary to multiple-valued logic (MVL) circuits. A multiple-valued LUT can be implemented using both current-mode and voltage-mode techniques, reducing the transistor count to half compared to that of a binary implementation.

Two main applications for multiple-valued LUTs are multiple-valued FPGAs and intelligent memories. An FPGA uses a LUT as a generic logic block to provide programmability. In an intelligent memory, a multiple-valued LUT is added in the Y-decoder section to facilitate simple mathematical operations on the stored digits. An FFT operation is used as an example in this paper to illustrate how a multiple-valued LUT can be beneficial.

1. Introduction

Look-up tables are known in the literature [1] as a method of implementing an arbitrary binary logic function. A truth table for a general 2-input 1-output combinational logic function is shown in Figure 1. In a combinational logic expression, the output is uniquely determined by the current input bits. Therefore, a truth table like that shown in Figure 1(a) can fully characterize the function. A look-up table (LUT) is a direct implementation of the truth table. Figure 1(b) shows a LUT implementation for the truth table shown in Figure 1(a). There is a one-to-one correspondence between the rows of the truth table and the rows of the LUT. For any given input bit-pattern, only one of the paths from input f_i to output Z has all of its transistors ON.

Referring to Figure 1(b), the intermediate nodes of rows 0 and 2 have the same voltage independent of the logic values of A and B. The same is true for the intermediate nodes of rows 1 and 3. That means the right-most transistors of rows 0 and 1 can be shared with those of rows 2 and 4,

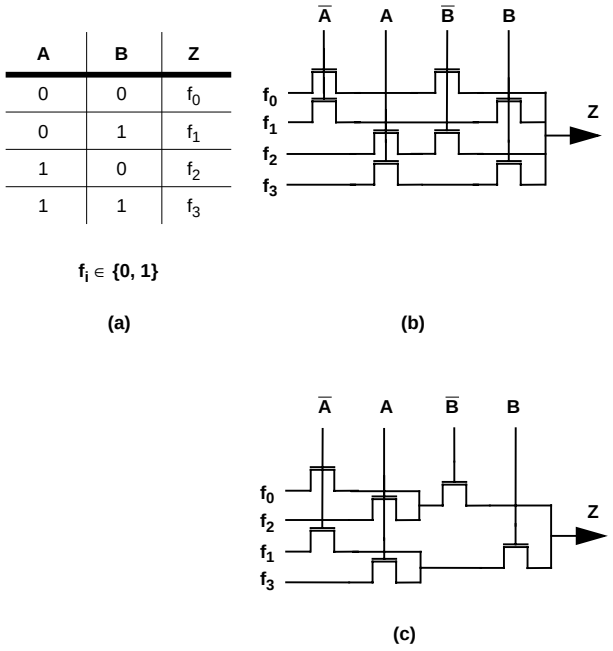

Figure 1. (a) A truth table for a general 2-input logic function, (b) A LUT implementation of the function, (c) A LUT implementation with reduced transistor count

respectively. The result is shown in Figure 1(c).

The ON path of a binary LUT carries binary information (i.e. 0 or 1) in the form of voltage or current signal to the output node. Elliott et al. [2] have shown that if proper timing is applied to a binary LUT, binary data can be transferred from one column of an array to a neighboring column thus accelerating mathematical computations. The main application of this is in an intelligent memory that could perform data mining, pattern recognition, or image processing. In this paper, we show that by using a multiple-valued LUT, a higher rate of data transfer (relative to a binary LUT) can be achieved with no area penalty. Section III of this paper proposes using a multiple-valued LUT in the Y-decoder section of a memory array to accelerate the computation of a Fast Fourier-Transform (FFT).

Ali Sheikholeslami (sheikh@eecg.toronto.edu) and P. Glenn Gulak (gulak@eecg.toronto.edu) are with the Department of Electrical and Computer Engineering, University of Toronto, Ontario, Canada, M5S 3G4. Phone (416) 978-1652, Fax (416) 971-2286.

Ryuji Yoshimura (ryuji@ele.eng.osaka-u.ac.jp) is with the Dept. of Electronics, Information Systems Eng., Osaka Univ., Suita 565, Japan. Phone (+81) 6-879-7781, Fax (+81) 6-897-7792. **This work was supported by Nortel and the Natural Sciences and Engineering Research Council of Canada.**

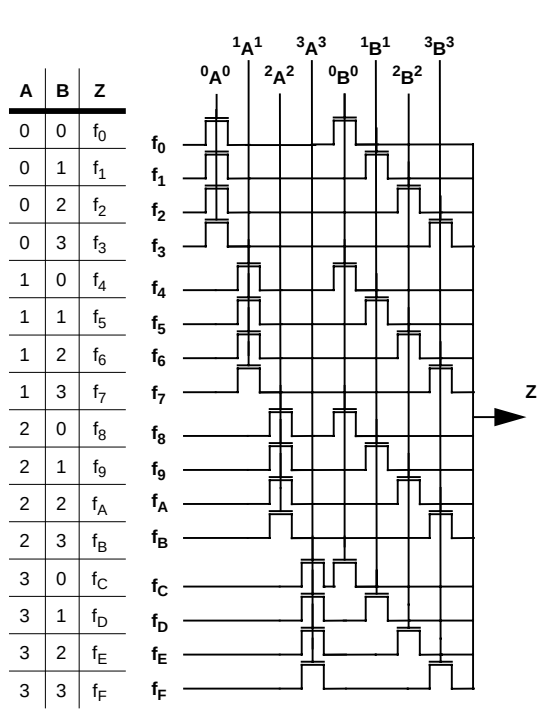

Figure 2. (a) A truth table for a general 2-input 4-valued logic function, (b) A LUT implementation of the function in (a). The delta literal ${}^{i}A^{i}$ is discussed in the text.

2. Multiple-Valued LUT

A general truth table for a 2-input 4-valued combinational logic function is shown in Figure 2(a). Parameters f_0 to f_F belong to {0, 1, 2, 3}. A CMOS implementation of the table is illustrated in Figure 2(b). The multiple-valued LUT is a direct extension of the binary LUT. Similar to the binary LUT, there is a one-to-one correspondence between the rows of the truth table and the rows of the LUT. ${}^{i}A^{i}$ refers to the *delta literal* of A, and is defined by:

$$ {}^{i}_{A}{}^{i} = \begin{cases} V_{DD} & \text{if } A = i \\ 0 & \text{if } A \neq i \end{cases} \qquad \text{(eq. 1)} $$

If the logical value of A and B are 0 and 2, for example, the pass transistors of row 2 turn on, connecting the output node to f_2 .

A delta literal generates a binary output that is connected to the gate of an NMOS transistor. Therefore, a pass transistor turns ON or OFF depending on the literal output being high or low, respectively.

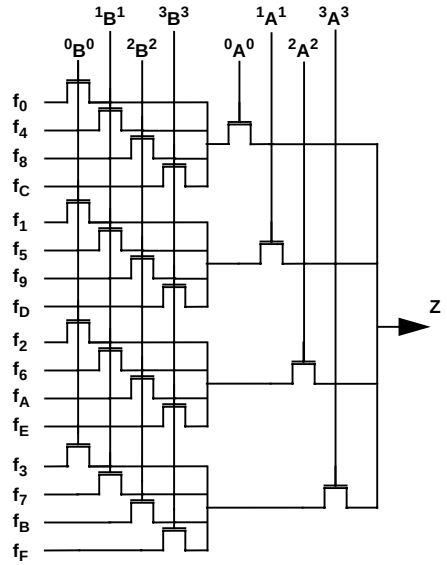

Figure 3. A LUT implementation for a general 2-input 4-valued logic function with reduced transistor count

The core of the LUT shown in Figure 2 uses 8 input wires and 32 transistors to implement a general 2-input 4-valued logic function. A binary LUT uses the same number of input wires but requires twice as many as transistors to implement the same function. Therefore, a 4-valued LUT requires less silicon area if the process technology is not metal limited.

The intermediate nodes of rows 0, 4, 8, and 12 have the same voltage independent of the logic values of A and B. The same is true for every fourth row of the LUT. Therefore, the right-most transistors of these rows can be shared without affecting the logic function of the LUT. Figure 3. shows such an implementation of the LUT with a row rearrangement. The transistor count for this implementation is 20.

Although we are using 4-valued logic in this section, the arguments and the results are easily extendible to higher radices. The block diagram of an MVL LUT with radix r is shown in Figure 4. The inputs to the LUT can be loaded from a memory or programmed directly in order to create a programmable multiple-valued logic function.

2.1 Voltage Mode

Table 1 shows the assigned voltage values for a 4-valued logic system. A maximum voltage of 1.8V is assigned to Logic 3, which is well below the full V_{DD} (=3.3V). The voltage gap between two consecutive digital values is 600mV. This is a sufficient voltage for a typical CMOS

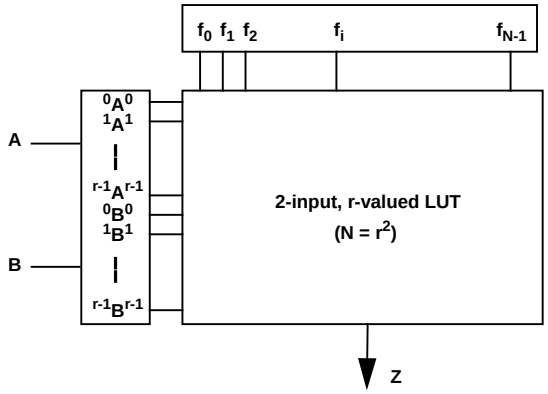

Figure 4. Block Diagram of a general 2-input r-valued LUT logic function

sense amplifier to discriminate between levels.

Logical Value	0	1	2	3
Voltage Value	0V	0.6V	1.2V	1.8V

Table 1: Voltage values assigned to 4-valued logic

The delta literals (Equation 1) generate a full V_{DD} instead of Logic 3 (1.8V) to drive the gate of an NMOS transistor. This guarantees the transistor to be fully ON independent of the logic value of the signal being carried. In other words, the voltage assigned to Logic 3 is not degraded by a threshold-voltage drop of the gate-to-source voltage of the NMOS transistor.

Figure 5 shows a CMOS implementation of the delta literal ${}^1A^1$. The circuit consists of two voltage-mode differential pairs that compare A with 0.3V (halfway between Logic 0 and 1) and 0.9V (halfway between Logic 1 and 2). The output of the two differential amplifiers NORed together to generate the final output for ${}^1A^1$.

Inputs f_0 to f_F can be implemented using a voltage divider technique [3]. As mentioned before, a reusable LUT

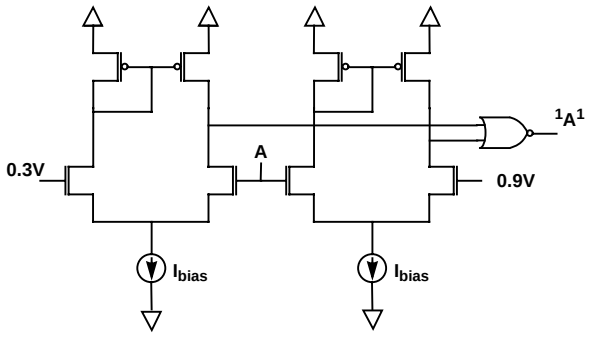

Figure 5. Circuit diagram of a voltage mode delta literal ${}^1A^1$

can be designed by making the circuits f_0 to f_F programmable or downloadable from a memory.

2.2 Current Mode

Table 2 shows the assigned current values for a 4-valued logic system. The maximum current (3μA) is assigned to logic 3.

Logical Value	0	1	2	3
Current Value	0μA	1μA	2μA	3μA

Table 2: Current values assigned to 4-valued logic

Using a 0.8μm technology, a 3μA current generates a voltage drop in the order of 100mV across the drain-source of each transistor. This does not affect the circuit performance unless the number of transistors in series exceeds 10.

A current-mode LUT is generally faster than a voltage-mode LUT. In both cases, only one path turns on depending on the logic values of A and B. However, a change in logic values of A and B requires less charge movement in the current-mode design since all internal nodes have relatively low voltages (i.e. no charging and discharging is required).

Figures 6 and 7 show two ways of implementing a delta literal (${}^1A^1$) in current mode. The input current in Figure 6 is sourced to the circuit and compared against two source currents (0.5μA and 1.5μA). If the input current lies between these two limits, the output node is pulled to VDD. Otherwise it is pulled to ground. Figure 7 shows a different implementation in which the input current is compared against a source and a sink current. This circuit eliminates the use of an inverter that was necessary in the previous implementation.

Inputs f_0 to f_F should be provided as current sources to the LUT. This is possible by using a current-mode multiple-valued memory [4] or by using a voltage-mode memory

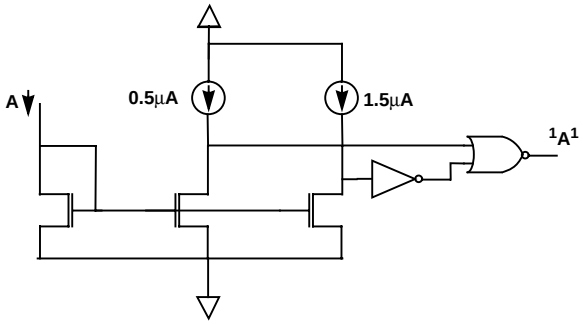

Figure 6. Circuit diagram of a current-mode delta literal ${}^1A^1$

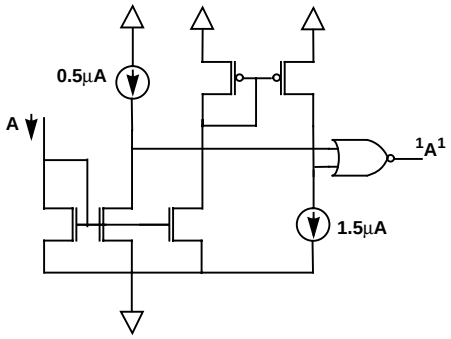

Figure 7. An alternative circuit diagram of a current-mode delta literal $^1A^1$

with a voltage to current conversion prior to the LUT.

3. Applications

A multiple-valued LUT offers the same function as a binary LUT but occupies less silicon area. Two applications that currently use binary LUTs are Field Programmable Gate Arrays (FPGAs)[1] and intelligent memories [2]. FPGAs provide instant implementing of logic circuits with negligible cost [1]. Intelligent memories are used to perform data mining, pattern recognition, or image processing [2]. Both of these applications can benefit from the advantages of a multiple-valued LUT.

3.1 Field Programmable Gate Arrays

Binary LUTs are widely used as the logic building blocks for FPGAs [1]. This is because an n-input LUT can easily be programmed to implement any Boolean function of n inputs. In other words, a LUT offers a simple hardware implementation for all combinational logic functions. The only difference between LUTs that implement various logic functions lies in the contents of the LUT register. Therefore, re-programming a LUT to implement a new function is as easy as loading new contents into the LUT register.

Using a multiple-valued LUT instead of a binary LUT saves silicon area due to the smaller transistor count, ignoring the peripheral circuitry. Moreover, a multiple-valued LUT seems the only reasonable choice for a logical building block of a multiple-valued FPGA [5].

3.2 Intelligent Memory

A binary LUT is usually used as the y-decoder of a DRAM. Ignoring the processing elements in Figure 8, the block diagram illustrates a standard memory in which a LUT selects one sense amplifier out of many (usually on the order of 1000) to be connected to the data pin. This is usually done by allocating a few address bits to the column select. The rest of the address bits are used to select a row (wordline) in the array. For example, 16 bits of address are required to uniquely select one bit out of 64kbits. Assuming

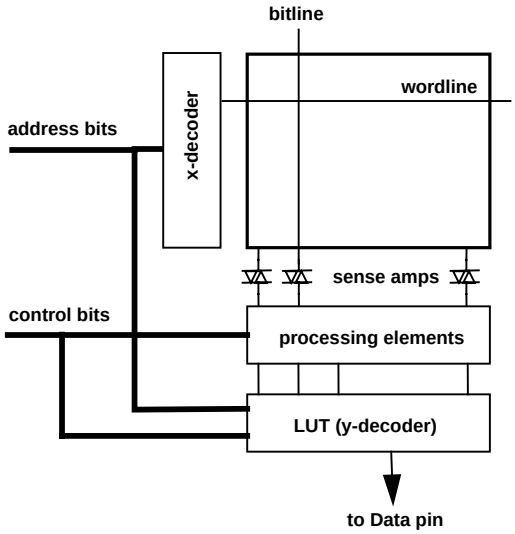

Figure 8. Block diagram of an intelligent memory

a memory array of 256 rows by 256 columns, 8 bits of address are allocated to the row select, and the other 8 bits to the column select. Once a bit is selected, it's value is put on the data pin for access by other chips (usually a microprocessor).

In a memory with 1000 columns, there are 1000 sense amplifiers that read the data of an entire row. Commercially, nowadays, 1, 8, or 16 of these sense amplifiers are accessed via the column select to transfer their data to 1, 8, or 16 data pins, respectively. The rest of the sense amplifiers are left unaccessed due to a limited number of data pins on each memory chip. This leads to a discrepancy between the internal and the available external bandwidth of a memory, also known as the von Neumann bottleneck [6]. For example, a memory chip with a 50ns access time and 1000 columns can provide a bandwidth of 20Gbits/sec internally while it could only provide a bandwidth of 20Mbits/sec externally through one data pin— a bandwidth reduction factor of 1000.

One way to overcome this bottleneck is to build processing elements close to the sense amplifiers inside the memory chip (refer to Figure 9) [2]. The processing elements reside below the sense amplifiers and at the top of the LUT. A processing element reads the data of the corresponding sense amplifier, performs a mathematical operation, and stores the final results in a specified memory location. Unlike the microprocessor, the processing elements utilize the full internal bandwidth of the memory. The only price for this is the silicon area occupied by the processing elements.

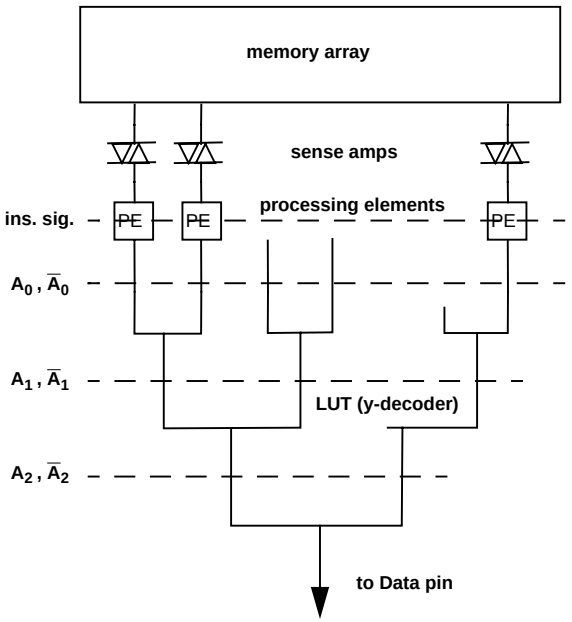

Figure 9. Block diagram illustrating the position of processing elements relative to the sense amplifiers and the LUT of an intelligent memory

LUTs provide cheap inter-column communication channels for the processing elements, since they already exist in the memory in the form of the y-decoder. Referring to Figure 1(c), a simultaneous high signal on A and $\overline{A}$ provides a communication path between the processing elements at f_0 and f_2, and the processing elements at f_1 and f_3, respectively.

Intelligent memories can be used to perform mathematical transforms such as a Fast Fourier Transform (FFT). An FFT maps a set of discrete time-domain data to a set of discrete frequency-domain data [7]. A 16-point FFT, for example, maps time-domain data points (x(0) to x(F)) to 16 frequency-domain data points (X(0) to X(F)). It is well known [7] that a 16-point FFT can be performed using a radix-2 or a radix-4 approach. In radix-2 approach, the input data are arranged in groups of 2, and the FFT operation is performed on 2 points at a time. The results of the first stage are then fed to the next stage as shown in Figure 10(a) until the final results are obtained. In radix-4 approach, the input data are arranged in groups of 4, and the FFT operation is performed on 4 points at a time as illustrated in Figure 10(b). Note that the radix-4 approach completes the 16-point FFT computations in 2 stages, as compared to 4 stages required in a radix-2 approach.

The structure of a binary LUT and a 4-valued LUT fits the computational structures of radix-2 and radix-4 FFT, respectively. In a binary LUT, a single address bit turns on

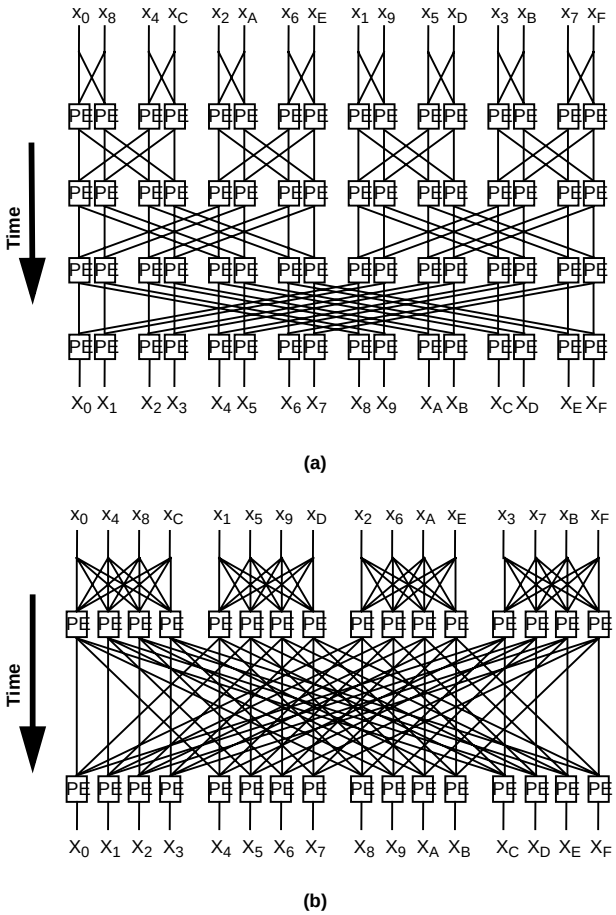

Figure 10. Flow graphs of (a) radix-2, 16-point FFT and (b) radix-4, 16-point FFT. Data flow is downward only. PEs and the LUT are reused at each stage to provide weighted sum of incoming signals. Branch weights are omitted for graph clarity [6]

a path between two neighboring processing elements, hence providing a communication path for a 2-point operation. In a 4-valued LUT, a single address digit turns on a path between four neighboring processing elements, hence providing a communication path for a 4-point operation. Moreover, as discussed in Section II, the path provided by a 4-valued LUT is capable of transferring 4-valued data among the processing elements. This allows using 4-valued processing elements with the same architectural complexity. Finally, the memory array in a radix-4 FFT can be replaced by a 4-valued memory array to reduce the number of memory fetches by a factor of 2.

In summary, the use of a 4-valued memory array, 4-valued processing elements, and a 4-valued LUT increases the density and speed of a radix-4 FFT implementation

compared to its binary counterpart. A 4-valued memory array cuts the number of memory fetches by a factor of 2. For example, a 32-bit wide data would only require sixteen 4-valued cells (i.e., 2bits/cell) instead of 32 binary cells. A 4-valued LUT uses a single wire (instead of 2 wires in binary) to carry 4-valued signals, hence reducing the interconnect area and power consumption.

Fast DCT would benefit from a similar application of these ideas. 2-D transforms on images would be implemented by calculating row transforms and then column transforms.

4. Summary

A voltage-mode and a current-mode implementations of a multiple-valued LUT is presented in this paper. A multiple-valued LUT can be used as a logic building block in a multiple-valued FPGA or as an inter-column communication channel in a multiple-valued intelligent memory. In both applications, the use of a multiple-valued LUT results in less silicon area and faster operation compared with designs using a binary LUT.

5. References

[1] T. S. Brown, R. Francis, J. Rose, and Z. Vranesic, "Field Programmable Gate Arrays", *Kluwer Academic Publishers*, 1992.

[2] D. G. Elliott, W. M. Snelgrove, and M. Stumm, "Computational Ram: A Memory-SIMD Hybrid and its Application to DSP", *Proceedings of the Custom Integrated Circuits Conference*, pp. 30.6.1-30.6.4, Boston, MA, May 1992.

[3] K. W. Current, "Memory Circuits for Multiple Valued Logic Voltage Signals", *Proc. of the 25th International Symposium on Multiple-Valued Logic*, pp. 52-57, 1995.

[4] E. Lee and P.G. Gulak, "Current-Mode Multivalued Dynamic MOS Memory with Error Correction", *IEE Electronics Letters*, Vol. 28, pp. 1067-1069, May 1992.

[5] Z. G. Vranesic, "The FPGA Challenge", to be presented at *the 28th International Symposium on Multiple-Valued Logic*, Fukuoka, Japan, May 1998.

[6] J. L. Hennessy and D. A. Patterson, "Computer Architecture, A Quantitative Approach", *Morgan Kaufmann Publishers Inc.*, 1990.

[7] A. V. Oppenheim and R. W. Schafer, "Digital Signal Processing", *Prentice-Hall Inc.*, 1975.