ENERGY-EFFICIENT AND PARAMETERIZED DESIGNS FOR FAST FOURIER TRANSFORM ON FPGAS*

Seonil Choi[1], *Gokul Govindu*[1†], *Ju-Wook Jang*[2‡], *Viktor K. Prasanna*[1]

[1]Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089
{seonilch+govindu+prasanna}@usc.edu

[2]Electronic Engineering
Sogang University
Seoul, Korea
jjang@sogang.ac.kr

ABSTRACT

In this paper, we develop energy efficient designs for the Fast Fourier Transform (FFT) on FPGAs. Architectures for FFT on FPGAs are designed by investigating and applying techniques for minimizing the energy dissipation. Architectural parameters such as degrees of vertical and horizontal parallelism are identified and a design domain is created through a combination of design choices. We determine design trade-offs using high-level performance estimation to obtain energy-efficient designs. We implemented a set of parametrized designs having parallelism, radix and choice of storage types as parameters, on Xilinx Virtex-II FPGA to verify the estimates. Our designs dissipate 57% to 78% less energy than the optimized designs from the Xilinx library. In terms of a comprehensive metric such as EAT (Energy-Area-Time), our designs offer performance improvements of 3-13x over the Xilinx designs.

1. INTRODUCTION

Characteristic features like customizability and high processing power and DSP oriented features like embedded multipliers and RAMs have made FPGAs an attractive option for implementing signal processing applications. Traditionally the performance metrics for signal processing have been latency and throughput. However energy efficiency has become increasingly important with the proliferation of portable, mobile devices. One such energy conscious application is software-defined radio (SDR) [4]. A FPGA-based system is a very viable solution for SDR's requirement of adaptivity and high computational power.

In this paper, we present energy-efficient FFT designs on FPGAs. The FFT is the compute-intensive portion of broadband beamforming applications such as those generally used in SDR and sensor networks. We investigate design techniques for minimizing the energy dissipated by FPGAs and apply the techniques for designing architectures and algorithms for FFT. We identify the architectural parameters that characterize the FFT designs and which affect the energy dissipation of the designs. A high level energy performance model is developed using these parameters. This model is used to determine design trade-offs, estimate the energy efficiency and arrive at energy-efficient designs. A parametrized architecture is designed, so that by selecting appropriate parameter values, the architecture of a complete design can be easily synthesized. Our parameterized design has more flexibility than a soft IP core, because it exploits the degrees of parallelism and throughput to a greater extent. We implemented and simulated a set of designs on Xilinx Virtex-II FPGA using Xilinx ISE tools to obtain energy dissipation values. We also compare the latencies, the area, and energy dissipations of our designs with the Xilinx library based designs. We use both estimated values (based on the model) and actual values (based on the implemented designs) in our comparisons. These comparisons show that our designs can provide significant reductions in not only latency but also energy dissipation. We thus provide a parametrized architecture and high-level model for fast estimation and implementation of energy efficient FFT designs.

The remainder of this paper is organized as follows. Section 2.1 presents design techniques for minimizing the energy dissipation in FPGAs. Our parameterized FFT architectures are presented in Section 2.2. Section 3 shows our performance estimation and the implementations of the selected designs on FPGAs. We also compare our designs with Xilinx library based designs. Finally, Section 4 concludes the paper.

2. ENERGY-EFFICIENT DESIGN FOR FAST FOURIER TRANSFORM

In this section, we briefly discuss techniques that can be applied to FPGA-based designs to obtain energy efficiency. Then we present our energy-efficient, parametrized architectures for FFT on FPGAs, inculcating the aforementioned design techniques.

2.1. Energy-Efficient Design Techniques

In literature, there are many low-level power management techniques that lead to energy savings when applied to FPGA design [7, 9]. One such technique is clock gating, which is used to disable parts of the device that are not in use during the computation. In the Xilinx Virtex-II family of FPGAs, clock gating can be realized by using primitives such as BUFGCE for dynamically driving a clock tree only when the corresponding logic is used. For example, FFT computation has many complex number multipliers to perform twiddle factor computations (multiplication followed by addition/subtraction). Because of the nature of the FFT algorithm, more than a quarter of the twiddle factors are 1, -1, j, or $-j$ and their computation can be bypassed. Thus, the implementation of twiddle factor computation can exploit clock gating to disable the

*This work is supported by the DARPA Power Aware Computing and Communication Program under contract F33615-C-00-1633 monitored by Wright Patterson Air Force Base and in part by the National Science Foundation under award No. 99000613.

†Gokul Govindu's work is supported by Satyam Computer Services.

‡Ju-wook Jang's work is supported by LG Yonam Foundation.

unnecessary computation blocks. Choosing energy-efficient *bindings* is another technique. A binding is a mapping of a computation to an FPGA component. The ability to choose the proper binding is due to the existence of several configurations for the same computation. For example, since FFT has a high data storage requirement, different bindings of the storage elements affect energy dissipation significantly. There are three possible bindings for storage elements in Virtex-II devices based on the number of entries: registers, slice based RAM (SRAM), and embedded Block RAM (BRAM). For large storage elements (those with more than 48 entries) BRAM shows an advantage in power dissipation over other implementations. A designer can analyze the trade-offs that arise from various bindings based on the design requirements. A pipelined architecture might be chosen since many digital signal processing applications process a stream of data. For these applications with regular data flow, pipelining increases throughput. But pipelining might increase power dissipation since all logic in the design is continuously active. Since throughput is maximized, eventually, the energy dissipation is reduced. Moreover since interconnect accounts for considerable power dissipation, the degree of parallelism and the depth of pipelining which might increase interconnect and thus energy dissipation, have to analyzed before implementation. Another technique is algorithm selection**.** A given application can be mapped onto FPGAs differently by selecting different algorithms. For example in implementing the FFT, the choice of a radix-4 based algorithm significantly reduces the number of complex multiplications that would otherwise be needed if a radix-2 based algorithm were used. Thus the trade-offs between different algorithms and architectures should be analyzed to achieve energy-efficient designs. More techniques are discussed in [2].

2.2. Designs for Fast Fourier Transform

For FFT designs, we use the well known Cooley-Tukey method. The calculation of an N-point FFT requires $O(N)$ operations for each of its $\log_2(N)$ stages, so the total computation required is $O(N \log_2 N)$ [6].

Due to the fact that, in practice, FFTs often process a stream of data, a pipelined architecture has been chosen. The N-point FFT design is based on the radix-4 algorithm. While there are many design parameters, we identify the parameters that determine the FFT architecture and eventually affect the energy dissipation. The parameterization is the key of our design since we explore design space based on the parameters for energy efficiency. There are five design parameters that characterize an N-point FFT designs: 1) the problem size (N), 2) the degree of horizontal parallelism (H_p), 3) the degree of vertical parallelism (V_p), 4) the binding for storage element, and 5) the precision of data. The horizontal parallelism determines how many radix-4 stages are used in parallel ($1 \leq H_p \leq \log_4 N$). Vertical parallelism determines the number of inputs being computed in parallel. Using the radix-4 algorithm, up to 4 inputs can be operated on in parallel. We have considered five basic building blocks described in [2]: radix-4, data buffer, data path permutation, parallel-to-serial/serial-to-parallel mux, and twiddle factor computation. Each individual block is parametrized, so that a complete design for any N can be obtained from combinations of the basic blocks:

Radix-4 butterfly (`R4`): This block performs a set of additions and subtractions with 16 adders/subtracters. It takes four inputs and produces four outputs in parallel. Each input data has real and imaginary components. The complex number multiplication for 1, -1, j, or $-j$ is implemented by remapping the inputs data path and using adders / subtracters.

Twiddle factor computation (`TW`): This block performs the complex number multiplication of the data with twiddle factors. The twiddle factors are obtained from a sine/cosine lookup table. Bypassing the multiplication when the value of twiddle factors is 1, -1, j, or $-j$ can reduce computation and thus energy (by disabling the multipliers). This block contains 4 multipliers, 2 adders/subtracters and two sign inverters.

Data buffer (`DB`): This block consists of two RAMs having N/V_p entries each. Data is written into one and read from the other RAM simultaneously. The read and write operations are switched after every N inputs. The data write and read addresses are at different strides determined by the architecture. For example in a $N = 16$, single input case, writing is done sequentially and reading is at strides of four.

Data path permutation (`PER`): In the parallel architectures ($V_p = 4$) , after computation of each stage, the data paths need to be permuted so that data can be accessed in parallel and in the correct order by the next stage. Dependencies occur due to stride accesses requiring data from either same locations or same RAMs. Figure 1 shows the permutation of the first stage for the 16-point FFT. On the first clock cycle, four data are stored in the first entry of each `DB` in parallel (See Figure 1 (a)). On the second clock cycle, another four data are stored in the second entry of each `DB` with one location being permuted (See Figure 1 (b)). On the third and fourth clock cycles, the operation is performed in the same manner and the final result is shown in Figure 1 (c). Note that the four data, a_0, a_4, a_8, and a_{12}, are stored in different `DBs` so that the radix-4 computation can be performed in parallel. The permutation occurs at every stage in the same manner.

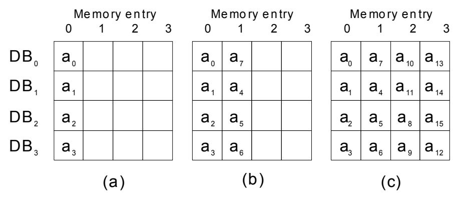

Fig. 1. Data permutation for `DB` at the first stage for 16-point FFT (clock cycle (a) $t = i$ (b) $t = i + 1$, and (c) $t = i + 3$)

Parallel-to-serial/serial-to-parallel mux (`PS/SP`): This block is used when the data is fed into the Radix-4 block in parallel and fed out in serial in the serial architecture ($V_p < 4$). While the radix-4 module operates on four data in parallel, the rest of architecture is flexible. Thus, to match the data rate, a parallel-to-serial mux before the radix-4 module and a serial-to-parallel mux after the radix-4 module are required.

For example, a 16-point FFT algorithm has 2 radix-4 stages. In the design, we can use one or two radix-4 blocks ($H_p = 1, 2$) depending on the sharing of the radix-4 block resource. If $H_p = 1$, one radix-4 block is used and is shared by the first and second stages. Thus a feedback datapath is necessary which decreases the throughput of the design. Figure 2 (a) shows an architecture for $N = 16$ where $V_p = 1, H_p = 2$. Figure 2 (b) shows a fully parallel architecture when $V_p = 4, H_p = 2$. This design has 12 data buffers, two radix-4 blocks, and 3 twiddle computation blocks. We also develop the associated algorithm for various architectures. Figure 3 describes the parallel algorithm for the architecture in Figure 2 (b). The variable `P` is used for horizontal

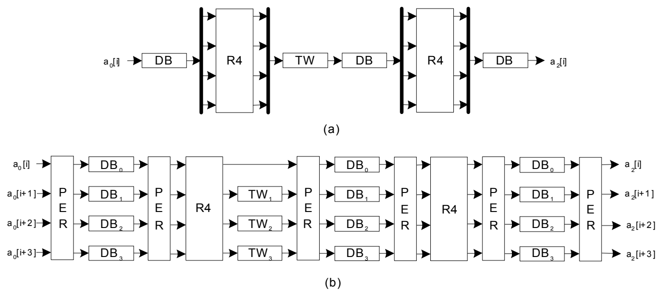

Fig. 2. Architectures for 16-point FFT (a) $(H_p, V_p) = (2, 1)$ and (b) $(H_p, V_p) = (2, 4)$

parallelism ($H_p = \log_4 N = 2$) and there are four unrolled do parallel loops ($V_p = 4$). `Quad`, `Dist`, `P`, `L`, and `R` are used for indexing data buffer in parallel.

```
Quad = N/4; Dist = N/4;
for P=0 to $\log_4$N-1 do parallel (note: horizontal parallelism, $H_p$=$\log_4$N)
  for K = 0 to $4^p$-1 do
    L = 4*K*Quad/$4^p$; R = L+Quad/$4^p$-1;
    $IdxTw_1$ = K; $IdxTw_2$ = 2*K; $IdxTw_3$ = 3*K;
    $Tw_1$ = w[$IdxTw_1$]; $Tw_2$ = w[$IdxTw_2$]; $Tw_3$ = w[$IdxTw_3$];
    for J= L to R do
      do parallel (note: vertical parallelism, $V_p$=4)
        $a_{p+1}$[J] = $a_p$[J]+$a_p$[J+Dist/$4^p$]+$a_p$[J+2*Dist/$4^p$]+$a_p$[J+3*Dist/$4^p$];
        $a_{p+1}$[J+Dist/$4^p$] = $a_p$[J]-j*$a_p$[J+Dist/$4^p$]-$a_p$[J+2*Dist/$4^p$]+j*$a_p$[J+3*Dist/$4^p$];
        $a_{p+1}$[J+2*Dist/$4^p$] = $a_p$[J]-$a_p$[J+Dist/$4^p$]+$a_p$[J+2*Dist/$4^p$]+j*$a_p$[J+3*Dist/$4^p$];
        $a_{p+1}$[J+3*Dist/$4^p$] = $a_p$[J]+j*$a_p$[J+Dist/$4^p$]-$a_p$*[J+2*Dist/$4^p$]-j*$a_p$[J+3*Dist/$4^p$];
      end parallel
      do parallel
        $a_{p+1}$[J] = $a_{p+1}$[J];
        $a_{p+1}$[J+Dist/$4^p$] = $Tw_1$*$a_{p+1}$[J+Dist/$4^p$];
        $a_{p+1}$[J+2*Dist/$4^p$] = $Tw_2$*$a_{p+1}$[J+2*Dist/$4^p$];
        $a_{p+1}$[J+3*Dist/$4^p$] = $Tw_3$ *$a_{p+1}$[J+3*Dist/$4^p$];
      end parallel
    end for
  end for
end parallel for
```

Fig. 3. Algorithm used for the architecture in Figure 2 (b)

3. PERFORMANCE ESTIMATION AND DESIGN SYNTHESIS

Since the architecture is parameterized, we can generated all possible designs by varying the parameter values. However, rather than implementing and simulating all designs, we define the high-level model using the techniques in [1] and conduct performance estimation and design trade-offs. Then the chosen candidate designs are implemented. Our target device is Virtex-II FPGA (speed grade -5) which is a high-performance, platform FPGA from Xilinx [8]. We have chosen the XC2V1500 for small FFT designs ($N \leq 64$) and XC2V3000 for large FFT designs ($N > 64$). These devices have 48 and 96 18×18-bit embedded multipliers, respectively.

3.1. Energy Performance Estimation

In FPGA designs with streams of data, throughput is an important factor in energy dissipation. Thus, in our pipelined design, the energy equation is $E = P/Th$, where P is the average power dissipation and Th is the throughput of the design. Note that $1/Th = L$ can be considered the *effective latency* of the design. The effective latency accounts for the benefits of overlapping computations in pipelining. Based on the architecture and algorithm in Section 2.2, it can be shown that the equation to calculate the latency (L), of computing an N-point, radix-4 FFT is: $L = N \log_4 N/(V_p \times H_p)$, where L is in cycles. To convert this latency to seconds, we merely divide by the clock frequency. We also know the types of FPGA components (multipliers, registers, etc.) and the amounts of each type of component that are used by for five basic building block. We obtain the power function for each basic building block using the techniques in [1]. We sum the average power dissipation of each blocks to estimate the total power dissipation. Since power is energy divided by latency, and we have earlier calculated latency, we can multiply the power by the latency to estimate the energy used in executing the algorithm. The power functions for the data buffer, the radix-4 block, the data path permutation, parallel-to-serial/serial-to-parallel mux, and the twiddle computation block are P_{DB}, P_{R4}, P_{PER}, $P_{PS/SP}$ and P_{TW}, respectively, where $P_{DB} = 1.23N + 35.44$ (mW) using SRAM, $P_{DB} = 0.0156N + 79.65$ (mW) using BRAM, $P_{R4} = 142.84$ (mW), $P_{PER} = 54.09$ (mW), $P_{PS/SP} = 13.52$ (mW), $P_{TW} = 0.0054N + 183.57$ (mW) using Block SRAM, $P_{TW} = 0.4879N + 157.74$ (mW) using SRAM, and $P_{IO} = 44$ (mW). Thus, the energy can be estimated as

$$\begin{aligned} E &= L \cdot \{V_p(H_p+1)P_{DB} + 2mH_pP_{PER} + (H_p-1)P_{R4} \\ &\quad +2s(Hp-1)P_{PS/SP} + t_v t_h P_{TW} + 2V_pP_{IO}\} \end{aligned} \qquad (1)$$

m is the number of the data path permultation block ($m = 1$ when $V_p = 4$, otherwise $m = 0$), s is the number of parallel-to-serial/serial-to-parallel muxes ($s = 1$ when $H_p = 1$, otherwise $s = 0$). $t_v t_h$ is the number of twiddle computation blocks ($t_v = V_p - 1$ when $V_p = 4$, otherwise $t_v = V_p$; $t_h = H_p - 1$ when $H_p = \log_4 N$, otherwise $t_h = H_p$).

We apply the parameter values to Equation 1 to compare the energy dissipations of various problem sizes. The clock speed of FPGA-based designs varies based on the likely speed achievable after place and route results. Figure 4 shows the energy and area estimates for various design points when computing a 256-point FFT. It clearly shows that the BRAM designs are better than the SRAM based in both metrics. The BRAM based designs use BRAMs for data buffer and phase lookup table since they are more energy-efficient for $N > 64$. The precision of each data is 16 bits. We assume a clock frequency of 100MHz since the implemented designs are run at a clock frequency of 100Mhz. We choose the minimal energy dissipation by selecting different parameter values ($H_p = 4$ and $V_p = \log_4 N$, BRAM based). It shows that parallelism increases the energy efficiency of the FFT design despite increasing the area requirement.

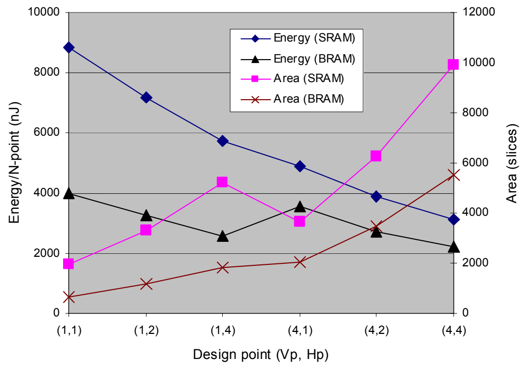

Fig. 4. Energy and area estimates for various designs for $N = 256$

Table 1. Performance comparion with Xilinx library based designs.

Problem	Xilinx (100MHz)					Our designs (100MHz)											Area	Time	E	EAT
size(n)	A(slice)	T(usec)	Em(nJ)	EAT	E/AT	Vp	Hp	Binding	A(slice)	T(usec)	Eest(nJ)	Em((nJ)	Error	EAT	E/AT	(inc.)	(dec.)	(dec.)	(dec.)
16	1362	0.16	179.68	0.04	0.83	1	2	SRAM	1171	0.16	65.40	76.99	15%	0.014	0.41	0.86x	1.0x	57%	2.71x
						4	2	SRAM	2390	0.04	63.55	75.18	15%	0.007	0.79	1.75x	4.0x	58%	5.45x
64	1079	1.92	1785.60	3.70	0.87	1	3	SRAM	2266	0.64	552.39	493.32	12%	0.72	0.79	2.10x	3.0x	72%	5.17x
						1	3	BRAM	1613	0.64	464.24	390.40	19%	0.40	0.38	1.49x	3.0x	78%	9.18x
						4	3	SRAM	5690	0.16	393.86	418.68	6%	0.38	0.46	5.27x	12.0x	77%	9.70x
						4	3	BRAM	4193	0.16	403.17	400.41	1%	0.27	0.60	3.89x	12.0x	78%	13.77x
256	1303	7.68	6927.36	69.32	0.69	1	4	BRAM	2050	2.56	2582.17	2223.1	16%	11.67	0.42	1.57x	3.0x	68%	5.94x
						4	4	BRAM	5624	0.64	2203.22	1971.3	12%	7.10	0.55	4.32x	12.0x	72%	9.77x
1024	1557	30.72	34283.5	1639.8	0.72	1	5	BRAM	2744	10.24	14963.5	13739.4	9%	386.06	0.50	1.76x	3.0x	60%	4.25x
						4	5	BRAM	6673	2.56	11424.7	9204.2	20%	157.23	0.54	4.29x	12.0x	73%	10.43

Eest is the estimated energy. Em is the measured energy from the synthesized designs. The unit of EAT is 1E-9. Inc. stands for increment. Dec. stands for decrement.

3.2. Performance of Synthesized Designs

All designs are parameterized based on N, Hp, Vp, binding, and precision as described in Section 2.2 and are implemented after coding in VHDL. Based on the performance estimation, we identified several designs for various problem sizes. We fixed the precision as 16 bits. These designs were synthesized using XST (Xilinx Synthesis Technology) in Xilinx ISE 4.1i [8]. The place-and-route file (.ncd file) was obtained for Virtex-II XC2V1500 and XC2V3000. The input test vectors for the simulation are randomly generated and the average switching activity is 50%. Mentor Graphics ModelSim 5.5e was used to simulate the designs and generate simulation results (.vcd file). These two files are then provided to the Xilinx XPower tool to evaluate the average power dissipation. The energy dissipation values are obtained by multiplying the average power by the latency. We observed that the error of energy estimation (see Table 1) is below 20% for the energy dissipation. The source of error mostly is due to the control logic since the estimation does not include it. If the estimates of designs are within 20% error range, the synthesized and simulated values need to be compared. Note BRAM is chosen for architectures for $N > 64$ since BRAM can store large data relatively using less energy than SRAM.

We also use FFT designs from the Xilinx library to compare with our designs. From the results in Table 1, our designs dissipates 57% to 78% less energy for various problem sizes. It is clear that our designs can perform the computations both faster and more energy-efficiently than the Xilinx based designs. If we use a comprehensive metric, EAT (Energy-Area-Time) and energy / (area $\times$ latency) (E/AT), our designs offer a performance improvement of 3-13x and 10-56%, respectively, compared to the Xilinx designs. Note that the E/AT metric is the average power density of the design.

Energy efficiency is achieved using pipelining, appropriate disabling of the compute blocks and choosing efficient memory bindings. Pipelining increases the throughput and decreases the effective latency. Disabling twiddle factor computation blocks reduces the amount of energy dissipated by more than 30%. Also, selecting a radix-4 algorithm as the basic building block reduces the number of complex multiplications in the design. Choosing bindings such that the data buffers and sine/cosine lookup tables are implemented using SRAM for $N < 64$ or BRAM for $N \geq 64$ increases the energy efficiency. We also observed that while parallelism increases the throughput and eventually the energy efficiency, the energy used by the interconnect in FPGAs significantly increases. For example, the design of $(V_p, H_p) = (4, 1)$ dissipates 20% more energy than the design of $(V_p, H_p) = (1, 4)$ for $N = 256$ while the former has 2 times higher throughput. It is because the former uses more interconnects and memory elements.

4. CONCLUSION

To develop energy-efficient designs for FFT, we discussed energy-efficient design techniques and a parametrized FFT architecture. Performance estimation and design trade-offs were performed to identify energy-efficient designs. We observed that the interconnects dissipate significant amount of energy in a parallel architecture. We are working on modeling interconnect energy for performance estimation and understanding design trade-offs.

5. REFERENCES

[1] S. Choi, J.-W. Jang, S. Mohanty, and V. K. Prasanna, "Domain-Specific Modeling for Rapid System-Wide Energy Estimation of Reconfigurable Architectures," *Engineering of Reconfigurable Systems and Algorithms*, 2002.

[2] S. Choi, Ronald Scrofano, V. K. Prasanna, and J.-W. Jang, "Energy Efficient Signal processing using FPGAs" *to appear in ACM Field Programmable Gate Array*, 2003.

[3] S. Choi, "Domain Specific Modeling and Energy Efficient Designs for Signal Processing Kernels using FPGAs," Doctoral Dissertation, in preparation, 2003.

[4] C. Dick, "The Platform FPGA: Enabling the Software Radio", *Software Defined Radio Technical Conference and Product Exposition* November 2002.

[5] P. Master and P. M. Athanas, "Reconfigurable Computing Offers Options For 3G," *Wireless Systems Design,* pp. 20-23, 1999.

[6] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 1989.

[7] L. Shang, A. Kaviani, and K. Bathala, "Dynamic Power Consumption in Virtex-II FPGA Family," *International Symposium on Field Programmable Gate Arrays*, 2002.

[8] Xilinx Application Note, Virtex-II Series and Xilinx ISE 4.1i Design Environment, http://www.xilinx.com, 2001.

[9] G. Yeap, *Practical Low Power Digital VLSI Design*, Kluwer Academic Publishers, 1998.