Transactions Briefs

Board-Level Multiterminal Net Assignment for the Partial Cross-Bar Architecture

Xiaoyu Song, William N. N. Hung, Alan Mishchenko, Malgorzata Chrzanowska-Jeske, Andrew Kennings, and Alan Coppola

***Abstract*—This paper presents a satisfiability-based method for solving the board-level multiterminal net routing problem in the digital design of clos-folded field-programmable gate array (FPGA) based logic emulation systems. The approach transforms the FPGA board-level routing task into a Boolean equation. Any assignment of input variables that satisfies the equation specifies a valid routing. We use two of the fastest Boolean satisfiability (SAT) solvers: *Chaff* and *DLMSAT* to perform our experiments. Empirical results show that the method is time-efficient and applicable to large layout problem instances.**

***Index Terms*—Digital design, layout, logic emulation, satisfiability, very large scale integration (VLSI).**

I. Introduction

There has been an ever increasing interest in computing engines based on field-programmable gate arrays (FPGAs) [1]. These engines make possible high-speed reconfigurable prototyping [11] and emulation systems [5]. There are two major steps in using FPGAs for prototyping or logic emulation. First, the large design is partitioned such that each subcircuit can fit into the FPGAs available on the hardware platform [11]. Second, the board-level routing problem (BLRP) is performed to connect signals between the FPGAs [6], [7]. In hardware platforms such as the realizer [2] and the enterprise emulation [4] systems, the set of FPGAs are interconnected by field-programmable interconnect chips (FPICs) using a partial crossbar architecture (also referred to as a clos-folded network [2]). In this architecture, the pins of the FPGAs are divided into N subsets, where N is the number of FPICs. All pins belonging to the same subset number in different FPGAs are connected to the same FPIC. In such a system, any circuit I/Os will have to go through an FPIC to reach the FPGAs. For this purpose, a number of pins on each FPIC are reserved for I/Os. An example of the partial crossbar architecture is illustrated in Fig. 1, showing three FPGAs and two FPICs.

In this paper, we specifically consider the BLRP. The BLRP has been previously studied in [6]–[8], and [10] and various algorithms proposed, but no experimental results for large problems were reported. We present a new satisfiability-based methodology for solving the BLRP in partial crossbar architectures. Unlike previous heuristics, our algorithm is a complete method and will find a routing solution if it exists.

The remainder of this paper is organized as follows. In Section II, we will define the problem formally and review the related work. In Section III, a novel approach based on satisfiability is presented. Experimental results are presented in Section IV. Section V concludes the paper.

Manuscript received February 1, 2002; revised June 10, 2002.

X. Song, A. Mishchenko, M. Chrzanowska-Jeske and W. N. N. Hung are with the Department of Electrical and Computer Engineering, Portland State University, Portland, OR 97201 USA (e-mail: William.N.Hung@intel.com).

A. Kennings is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1 Canada.

A. Coppola is with Cypress Semiconductor, Beaverton, OR 97008 USA.

Digital Object Identifier 10.1109/TVLSI.2003.812322

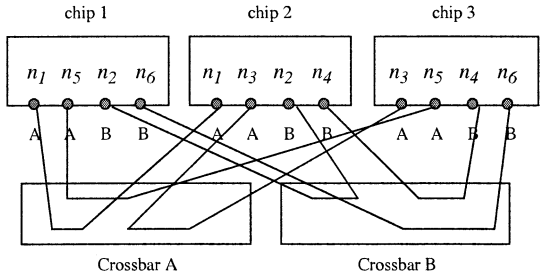

Fig. 1. An instance of 2BLRP (2, 2, 4, 8).

II. Problem Definition and Related Work

A. Problem Formulation

The FPGAs are referred to as *chips*. Let us assume that all chips are identical and the interconnection crossbars are used only to connect to the chips but not to each other. Let F be a set of p identical FPGA chips, numbered by $1, 2, \ldots, p$. A chip has a set of I/O ports. I/O ports of each chip are *evenly* divided into k groups of size m : $S_1, S_2, \ldots, S_{\mathrm{k}}$. We assign a distinct type for each group, S_{i}, $i = 1, \ldots, k$. We use labels: A, B, C, ..., to represent their types. An I/O port in a group S_{i} possesses the type of S_{i}. In other words, we say I/O ports of each chip are evenly divided into k groups of size m such that (i) type(S_1), type(S_2), ..., type(S_{k}) are pair-wise distinct; (ii) size(S_1) = size(S_2) = ... = size(S_{k}) = m.

In the following, m is the number of the pins having the same type in each chip, k is the number of types in each chip which represents the number of crossbars, and p is the number of FPGA chips. Pins of each chip are evenly routed to k crossbar switches using N nets. A 2BLRP with parameters m, k, p and N is denoted by 2BLRP(m, k, p, N). An instance of the 2BLRP(2, 2, 3, 6) is shown in Fig. 1, where $m = 2$, $k = 2$, $p = 3$, and $N = 6$.

A *multiterminal board level routing problem* (*BLRP*) is defined as follows. Given a set of multiterminal interchip nets $M = \{n_1, n_2, \ldots, n_{\mathrm{N}}\}$, where $n_{\mathrm{t}} = \{i_1, \ldots, i_s\}$, $i_g \neq i_{\mathrm{h}}$, i_{g}, $i_h \in \{1, \ldots, p\}$, $g, h \in \{1, \ldots, s\}$, $t = 1, 2, \ldots, N$, find an assignment of M to I/O ports of F such that, for each net $n_{\mathrm{t}} = \{i_1, \ldots, i_{\mathrm{s}}\}$, type($i_{\mathrm{g}}$) = type($i_{\mathrm{h}}$), $i_{\mathrm{g}} \neq i_{\mathrm{h}}$, $i_{\mathrm{g}}, i_{\mathrm{h}} \in \{1, \ldots, p\}$, $g, h \in \{1, \ldots, s\}$, and $t = 1, 2, \ldots, N$.

B. Previous Work

Some heuristics were proposed for solving the BLRP in [2] and [3]. Optimal algorithms for board-level routing, when all nets are two-terminal nets and the I/O-pin subset size is even, were proposed independently by Chan and Schlag [10] and Mak and Wong [7]. An $O(N^2)$-time algorithm for solving any two-terminal BLRP was presented in [7], where N is the number of nets. An I/O pin capacity constraint was proposed to assure the existence of a solution. The algorithm was based on the iterative computation of Euler circuits in graphs of BLRPs. They also proved that the multiterminal routing problem is NP complete. In [1616], satisfiability formulation was used in island-style symmetric FPGA routing.

III. Solving BLRP(M, K, P, N) via Satisfiability

A. SAT Formulation

In order to reduce BLRP to the satisfiability formula, it is necessary to encode the problem by introducing *Boolean variables* and formulating Boolean constraints in terms of *SAT clauses*. Given the problem with N multiterminal nets and k types of I/O ports, it is reasonable to introduce $N \times k$ Boolean variables to encode the problem. Each variable represents the possibility to route the given net using one of k possible pin types. If the variable is 1, the routing includes the given possibility; if the variable is 0, this possibility is not used.

To express the constraints, consider the set of requirements for a feasible solution. The requirements are of two types 1) the covering constraints and 2) the closure constraints. The first group of constraints ensures that each net is routed at least once. The second group ensures that in 2a) no net is routed more than once, and that in 2b) for each chip and each pin type, the number of associated nets does not exceed the number of available pins.

1) Covering Constraints: For each net n, there is only one covering constraint. This constraint relates all the variables associated with this net in the following way:

$$x_n^1 \vee x_n^2 \vee \cdots \vee x_n^k = 1.$$

As discussed above, this formula means that the net can be routed using any pin type.

2) Closure Constraints: The first type of closure constraints requires that each net was routed no more than once. In other words, among each pair of variables ($x_{\mathrm{n}}^{\mathrm{i}}$, $x_{\mathrm{n}}^{\mathrm{j}}$), at least one is assigned to zero

$$\overline{x_n^i} \vee \overline{x_n^j} = 1, \quad \forall i, \quad j \in \{1, 2, \ldots, k\}, \quad i \neq j.$$

Let $x_{ab-n_i}^k$ be a Boolean variable representing a net n_i connecting chips a and b using pin type k. We need to introduce the set of variables $X_{\mathrm{c}}^{\mathrm{k}}$ corresponding to nets connecting chip c with other chips using the pin type k. More formally, we have

$$X_c^k = \left\{ x_{ab-n_1}^k \middle| (a = c) \vee (b = c) \right\}.$$

The second type of closure constraints combines all variables $X_{\mathrm{c}}^{\mathrm{k}}$ that correspond to all nets connecting chip c with other chips using the pin type k. Among variables belonging to $X_{\mathrm{c}}^{\mathrm{k}}$ there should be no more than m variables equal to 1. In other words, in every group of $m+1$ variables belonging to $X_{\mathrm{c}}^{\mathrm{k}}$, there should be at least one variable equal to 0.

To formulate these constraints, it is necessary to create the set $X_{\mathrm{c}}^{\mathrm{k}}$ for each chip and each pin type, and next take all possible combinations of $m+1$ variables in the negative polarity. The illustration of this type of constraints is given below.

3) An Example: We use the example in Fig. 1 to show our SAT formulation. The variables are introduced as follows. There are 12 variables totally. The solution shown in Fig. 1 corresponds to the following assignment:

Chips 1 and 2: $x_{12-n_1}^A = 1,\ x_{12-n_1}^B = 0,\ x_{12-n_2}^A = 0,\ x_{12-n_2}^B = 1$

Chips 2 and 3: $x_{23-n_3}^A = 1,\ x_{23-n_3}^B = 0,\ x_{23-n_4}^A = 0,\ x_{23-n_4}^B = 1$

Chips 1 and 3: $x_{13-n_5}^A = 1,\ x_{13-n_5}^B = 0,\ x_{13-n_6}^A = 0,\ x_{13-n_6}^B = 1$

The constraints can be expressed as follows.

The covering constraints for all nets:

$$x_{12-n_1}^A \vee x_{12-n_1}^B = 1; \quad x_{12-n_2}^A \vee x_{12-n_2}^B = 1;$$
$$x_{23-n_3}^A \vee x_{23-n_3}^B = 1; \quad x_{23-n_4}^A \vee x_{23-n_4}^B = 1;$$
$$x_{13-n_5}^A \vee x_{13-n_5}^B = 1; \quad x_{13-n_6}^A \vee x_{13-n_6}^B = 1.$$

The first set of closure constraints for all nets:

$$\overline{x_{12-n_1}^A} \vee \overline{x_{12-n_1}^B} = 1; \quad \overline{x_{12-n_2}^A} \vee \overline{x_{12-n_2}^B} = 1;$$
$$\overline{x_{23-n_3}^A} \vee \overline{x_{23-n_3}^B} = 1; \quad \overline{x_{23-n_4}^A} \vee \overline{x_{23-n_4}^B} = 1;$$
$$\overline{x_{13-n_5}^A} \vee \overline{x_{13-n_5}^B} = 1; \quad \overline{x_{13-n_6}^A} \vee \overline{x_{13-n_6}^B} = 1.$$

The second set of closure constraints on chip 1 with pin type A, $X_c^k = \{x_{12-n_1}^A, x_{12-n_2}^A, x_{13-n_5}^A, x_{13-n_6}^A\}$

$$\overline{x_{12-n_1}^A} \vee \overline{x_{12-n_2}^A} \vee \overline{x_{13-n_5}^A} = 1; \ \overline{x_{12-n_1}^A} \vee \overline{x_{12-n_2}^A} \vee \overline{x_{13-n_6}^A} = 1;$$
$$\overline{x_{12-n_1}^A} \vee \overline{x_{13-n_5}^A} \vee \overline{x_{13-n_6}^A} = 1; \ \overline{x_{12-n_2}^A} \vee \overline{x_{13-n_5}^A} \vee \overline{x_{13-n_6}^A} = 1.$$

Similarly, the other closure constraints can be obtained for chips 1, 2, and 3 with types A and B, respectively, and each of them contains four constraints. It is easy to verify that the assignments of variables corresponding to the solution in Fig. 1 satisfy the given set of constraints.

4) Complexity: The bottleneck of the proposed approach is in the number of the closure constraints of the second type. It grows exponentially with the number of nets involving the given chip. However, it is polynomial for small m. Assuming that all pins of the chips are used, there are $z = k^* m$ nets connecting each chip. Then, the total number of clauses can be approximated as follows:

$$N \times 1 + N \times \frac{k \times (k-1)}{2} + p \times k \times \binom{k \times m}{m+1}.$$

IV. Experimental Results

We initially attempted to use a method based on Boolean decision diagrams (BDDS) to solve the Boolean equations representing BLRP. However, The number of variables needed to represent the routing problem is measured in hundreds and often thousands. The BDD-based implementation failed due to the explosize BDD size for problems with large number of Boolean variables. Thus, we could not solve even the smallest examples out of those that are listed in the experimental results section below.

We subsequently formulated the routing constraints as CNF formulas that were checked by SAT solvers. We used two of the fastest solvers [13] from the numerous available SAT solvers: *Chaff* [14] and *DLMSAT* [12] (also known as DLM), which are complete and incomplete solvers, respectively. Chaff employs a conflict resolution, conflict clause addition, and nonchronological backtracking scheme similar to *GRASP* [13], but employs watch lists to speed up its execution. *DLM* is a discrete Lagrange-multiplier-based global-search method for solving satisfiability problems. In contrast to clause weight schemes that rely only on the weights of violated constraints to escape from local minima, *DLM* uses the value of an objective function to provide further guidance. The dynamic shift in emphasis between the objective and the constraints is the key of Lagrangian methods. One of the major advantages of *DLM* method is that it has very few algorithmic parameters to be tuned by users and the search procedure can be made deterministic. *DLM* often performs as one of the best existing methods and can achieve an order-of-magnitude speedup for some problems.

Our test results presented in Table I are of great importance since we have the first experimental results for the problem studied in [6]. Our programs and results are available online [15]. *Benchmark* is the name of a generated benchmark. P is the number of chips (FPGAs). K is the number of pin types (FPICs). M is the number of pins of each type. N is the number of nets. Max is the largest number of terminals (size) of a net. Ave is the average size of all nets. *Pin%* is the average pin utilization. *Vars* is the number of variables needed to encode the problem for the *SAT* solver. *Clauses* is the number of clauses given to the *SAT* solver. *Literals* is the number of positive and negative polarity literals in all clauses. *Prep* is the time needed to transform the

TABLE I
EXPERIMENTAL RESULTS

Benchmark	P	K	M	N	Max	Ave	Pin%	Vars	Clauses	Literals	Prep	Routability	DLM	Chaff
p020_k5_m2_n07	20	5	2	49	7	4.0	98	245	12039	35725	0	yes	0.03	0.68
p020_k5_m2_n08	20	5	2	52	8	3.8	99	260	12147	36025	0	yes	0.04	1.31
p020_k5_m2_n06	20	5	2	59	6	3.3	97	295	12149	35975	0	yes	0.03	2.01
p020_k5_m2_n05	20	5	2	64	5	3.1	99	320	12279	36325	0	yes	0.02	0.22
p020_k5_m2_n04	20	5	2	76	4	2.6	99	380	12411	36625	0	yes	0.02	0.16
p020_k5_m3_n14	20	5	3	55	13	5.4	99	275	133855	534375	0.09	yes	2.37	85.88
p020_k5_m3_n11	20	5	3	60	11	5.0	100	300	135340	540220	0.1	yes	0.43	100.71
p020_k5_m3_n09	20	5	3	66	9	4.5	99	330	132051	526950	0.09	yes	0.39	2.28
p020_k5_m3_n08	20	5	3	68	8	4.4	100	340	132898	530300	0.09	yes	0.32	0.33
p020_k5_m3_n07	20	5	3	77	7	3.9	100	385	132997	530525	0.09	yes	0.50	10.29
p020_k5_m3_n06	20	5	3	88	6	3.4	100	440	134218	535200	0.09	yes	0.35	83.91
p020_k5_m3_n05	20	5	3	99	5	3.0	99	495	134339	535475	0.1	yes	0.28	69.36
p020_k5_m3_n04	20	5	3	114	4	2.6	99	570	134504	535850	0.09	yes	0.30	0.99
p020_k3_m3_n08	20	3	3	36	8	5.0	100	108	7536	29892	0	yes	0.04	0.01
p020_k4_m3_n10	20	4	3	30	10	6.0	75	120	21110	84080	0	no	N/A	0.08
p020_k4_m3_n08	20	4	3	67	8	3.6	100	268	39409	156832	0.02	yes	0.11	0.19
p020_k5_m3_n15	20	5	3	30	15	8.0	80	150	91620	365910	0	no	N/A	0.01
p020_k7_m3_n08	20	7	3	105	8	4.0	100	735	811055	3240125	0.62	yes	3.59	207.79
p050_k5_m3_n07	50	5	3	150	7	4.0	80	750	244720	975030	0	no	N/A	0.01
p050_k5_m3_n08	50	5	3	171	8	4.4	100	855	337956	1348575	0.25	yes	2.85	109.56
p050_k6_m3_n08	50	6	3	212	8	4.2	99	1272	911222	3638952	0.65	yes	9.54	109.20
p050_k7_m3_n08	50	7	3	241	8	4.3	99	1687	2070897	8274189	1.57	yes	23.97	1453.00
p100_k5_m3_n08	100	5	3	344	8	4.4	100	1720	684464	2731320	0.48	yes	10.54	112.62
p150_k5_m3_n08	150	5	3	552	8	4.1	100	2760	1024647	4088100	0.72	yes	7.31	69.64
p200_k3_m3_n45	200	3	3	45	45	24.0	60	135	15843	63057	0	no	N/A	0.01
p200_k4_m3_n55	200	4	3	50	55	28.0	58	200	70646	281984	0	no	N/A	0.01
p200_k5_m3_n55	200	5	3	90	55	28.0	84	450	862730	3449210	0	no	N/A	0.01
p200_k5_m3_n08	200	5	3	717	8	4.2	100	3585	1368537	5460525	1.1	yes	14.6	91.11

problem into a *SAT* instance. *DLM* and *Chaff* are the times needed to solve the problem by *DLM* and *zChaff* SAT solvers, respectively. The runtime is in seconds on 850-MHz Pentium III with 1-GB RDRAM (only a small part of memory has been used). The run time can be improved by fine-tuning the *SAT* solver parameters. All solutions have been automatically verified using a built-in verifier.

For the same parameters of P, K, and M, we increase N (number of nets) gradually. It is pretty clear that the number of variables, clauses and literals generated in our satisfiability formulation also increases with N. However, the time it takes for the *DLM* SAT solver does not necessarily increase with the rising N. The same phenomenon can be observed for the number of chips (P). Increasing P (and N) results in more variables, clauses, and literals. But the time it takes to solve the problem does not necessarily increase with P. Since *DLM* transforms the routing problem into discrete Lagrangian domain, the time it takes to solve the problem is not directly proportional to the number of constraint clauses. We have tested two sets of problems: $M = 2$ and $M = 3$. It takes a longer time to solve the problem for $M = 3$ than for $M = 2$. This is understandable as more pins implies more complexity for the problem. For the majority of cases, *Chaff* takes longer to compute than *DLM*. This does not necessarily mean that *Chaff* is in general slower than *DLM*. There are a lot of other parameters that could be tuned by these SAT checker implementations to speed up the runtime for particular types of problems. In addition, *Chaff* is a complete *SAT* checker that is able to confirm unroutability. This is very important as heuristic methods and local searches (like *DLM*) run forever under those circumstances.

V. CONCLUSION

We have studied a satisfiability-based method for solving the board-level multiterminal net routing problem in partial crossbar FPGA based logic emulation systems. Our approach transformed the FPGA routing task into a Boolean equation. If the problem is not satisfiable, a feasible routing does not exist. Experimental results demonstrate its time-efficiency and applicability to large layout problem instances.

ACKNOWLEDGMENT

The authors would like to thank W. K. Mak for helpful discussions.

REFERENCES

[1] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field Programmable Gate Arrays*. Norwell, MA: Kluwer, 1992.
[2] J. Varghese, M. Butts, and J. Baatcheller, "An efficient logic emulation system," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 171–174, 1993.
[3] M. Slimane-Kadi, D. Brasen, and G. Saucier, "A fast FPGA prototyping system that uses inexpensive high-performance FPIC," in *Proc. ACM/SIGDA Int. Workshop Field-Programmable gate Arrays*, 1994.
[4] L. Maliniak, "Multiplexing enhances hardware emulation," *Electron. Design*, pp. 76–78, 1992.
[5] S. Walters, "Computer-aided prototyping for ASIC-based systems," *IEEE Design Test*, pp. 4–10, 1991.
[6] W. K. Mak and D. F. Wong, "Board-level multi-terminal net routing for fpga-based logic emulation," in *Proc. Int. Conf. Computer-Aided Design*, 1995, pp. 339–344.

[7] ——, "On optimal board-level routing for FPGA-based logic emulation," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 282–189, Mar. 1997.
[8] S. Lin, Y. Lin, and T. Hwang, "Net assignment for the FPGA-based logic emulation system in the folded-clos network structure," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 316–320, Mar. 1997.
[9] N. C. Chou, L. T. Liu, C. K. Cheng, W. J. Dai, and R. Lindelof, "Circuit partitioning for huge logic emulation systems," in *Proc. Design Automation Conf.*, 1994, pp. 244–249.
[10] P. K. Chan and M. D. F. Schlag, "Architectural tradeoffs in field-programmable-device based computing systems," in *Proc. IEEE Workshop on FPGAs for Custom Computing Mach.*, Apr. 1993, pp. 152–161.
[11] M. Gokhale, W. Holmes, A. Kopser, S. Lucas, R. Minnich, and D. Sweely, "Building and using a highly parallel programmable logic arrays," *Comput.*, vol. 24, pp. 81–89, Jan. 1991.
[12] Y. Shang and B. W. Wah, "A Discrete lagrangian-based global-search method for solving satisfiability problems," *J. Global Optim.*, vol. 12, no. 1, pp. 61–99, 1998.
[13] M. N. Velev, "Effective use of boolean satisfiability procedures in the formal verification of superscalar and VLIW microprocessors," in *Proc. ACM/IEEE Design Automation Conf.*, Las Vegas, NV, June 18–22, 2001, pp. 226–231.
[14] M. W. Moskewicz, C. F. Madigan, Y. Zhou, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *Proc. Design Automation Conf.*, June 18–22, 2001, pp. 530–535.
[15] W. N. N. Hung, , http://www.ece.pdx.edu/~whung/BLRP, Aug. 2002.
[16] R. G. Wood and R. A. Rutenbar, "FPGA routing and routability estimation via Boolean satisfiability," *IEEE Trans. VLSI Syst.*, vol. 16, pp. 222–231, June 1998.

Analysis of Buck Converters for On-Chip Integration With a Dual Supply Voltage Microprocessor

Volkan Kursun, Siva G. Narendra, Vivek K. De, and Eby G. Friedman

***Abstract*—An analysis of an on-chip buck converter is presented in this paper. A high switching frequency is the key design parameter that simultaneously permits monolithic integration and high efficiency. A model of the parasitic impedances of a buck converter is developed. With this model, a design space is determined that allows integration of active and passive devices on the same die for a target technology. An efficiency of 88.4% at a switching frequency of 477 MHz is demonstrated for a voltage conversion from 1.2–0.9 volts while supplying 9.5 A average current. The area occupied by the buck converter is 12.6 mm^2 assuming an 80-nm CMOS technology. An estimate of the efficiency is shown to be within 2.4% of simulation at the target design point. Full integration of a high-efficiency buck converter on the same die with a dual-$V_{\rm DD}$ microprocessor is demonstrated to be feasible.**

***Index Terms*—Buck converter, dc-dc converter, dual supply voltage, high efficiency, integrated inductors, low power, low voltage, modeling of dc–dc converters, monolithic dc–dc conversion, multiple supply voltages, power supply, supply voltage scaling, switching dc–dc converters, voltage regulator.**

Manuscript received July 30, 2002; revised October 19, 2002. This work was supported in part by a Grant from Intel Corporation.

V. Kursun and E. G. Friedman are with the Electrical and Computer Engineering Department, University of Rochester, Rochester, NY 14627-0231 USA.

S. G. Narendra and V. K. De are with the Microprocessor Research Laboratories, Intel Corporation, Hillsboro, OR 97124 USA.

Digital Object Identifier 10.1109/TVLSI.2003.812289

I. Introduction

Decreasing the power dissipation and current demand of high-performance microprocessors are the two primary reasons for implementing a dual-$V_{\rm DD}$ microprocessor [1]. Due to the quadratic dependence of the dynamic switching power and the more than linear dependence of the subthreshold and gate oxide leakage power on the supply voltage, power dissipation is significantly reduced when portions of a microprocessor operate at a lower voltage level. A linear relationship exists between the current demand and power consumption of a microprocessor. Reducing the maximum power consumption, therefore, reduces the maximum current required by a microprocessor, thereby decreasing the number of power and ground pads on a microprocessor die. In order to maximize this reduction in current, the lower voltage supply of a dual-$V_{\rm DD}$ microprocessor should be integrated on the same die with the microprocessor. Moreover, in order to fully exploit expected reductions in power and current, the energy overhead of an integrated dc–dc converter to produce a second voltage level must be minimized.

Buck converters are popular due to the high efficiency and good output voltage regulation characteristics of these circuits [2]–[5]. In single power-supply microprocessors, the primary power supply is typically an external (nonintegrated) buck converter. In a dual-$V_{\rm DD}$ microprocessor, the choices are either a second external dc–dc converter, or a monolithic (both active and passive devices on the same die as the load) dc–dc converter.

In a typical nonintegrated switching dc–dc converter, significant energy is dissipated by the parasitic impedances of the interconnect among the nonintegrated devices (the filter inductor, filter capacitor, power transistors, and pulse width modulation circuitry) [3]. Moreover, the integrated active devices of a pulsewidth modulation circuit are typically fabricated in an old technology with poor parasitic impedance characteristics.

Integrating a dc–dc converter with a microprocessor can potentially lower the parasitic losses as the interconnect between (and within) the dc–dc converter and the microprocessor is reduced. Additional energy savings can be realized by utilizing advanced deep submicrometer fabrication technologies with lower parasitic impedances. The efficiency attainable with a monolithic dc–dc converter, therefore, is higher than a nonintegrated dc–dc converter.

Fabrication of a monolithic switching dc–dc converter, however, imposes a challenge as the on-chip integration of inductive and capacitive devices is required for energy storage and output signal filtering. Integrated capacitors and inductors above certain values are not acceptable due to the tight area constraints that exist within high performance microprocessor integrated circuits (ICs). Another significant issue with integrated inductors is the poor parasitic impedance characteristics which can degrade the efficiency of a voltage regulator. The value, physical size, and parasitic impedances of the passive devices required to implement a buck converter, however, are reduced with increasing switching frequency [2]–[4]. Integrated capacitors of small value (used for decoupling and constrained by the available area on the microprocessor die) are available in high-performance microprocessors [6]. Furthermore, with the use of magnetic materials, a new integrated microinductor technology with relatively small parasitic impedances and higher cutoff frequencies (over 3 GHz) has recently been reported [7]. Therefore, employing switching frequencies higher than the typical switching frequency range found in conventional dc–dc converters permits the on-chip integration of active and passive devices of a buck converter onto the same die as a high-performance microprocessor.