Low Power Design for ASIC Cores*

ALVAR DEAN^a, DAVID GARRETT^{b,†}, MIRCEA R. STAN^{b,‡}
and SEBASTIAN VENTRONE^a

^aIBM Microelectronics, Essex Junction, VT 05452;

^bECE Dept., University of Virginia, Charlottesville, VA 22903

(Received 20 June 2000; In final form 3 August 2000)

A semicustom ASIC design methodology is used to develop a low power DSP core for mobile (battery powered) applications. Different low power design techniques are used, including dual voltage, low power library elements, accurate power reporting, pseudo-microcode, transition-once logic, clock gating, and others.

Keywords: Low power design; ASIC; Processor core

I. INTRODUCTION

Low power design is mainly driven by the need to contain power dissipation (hence reduce packaging and cooling costs) for high-performance systems at one end of the applications spectrum and by the desire to reduce power consumption (hence reduce size, weight and increase battery life) for portable applications at the other end. This paper presents several low power techniques used in the design of an ASIC DSP core for portable applications. Both dynamic power (in active mode), and static power (in standby mode), are critical and need to be addressed for battery-operated devices.

I.1. Portable Applications

The portable, battery-operated marketplace is dominated by embedded applications with customer specific software run from a Read-Only Memory (ROM). The challenge for the designers is to balance the desired performance requirements of current compute intensive applications with the need to reduce cost, form factor and increase battery life. Dynamic power is strongly dependent on the switching activity, hence the power can be reduced by tailoring the processor speed to a variable computation load. Because of this possible tradeoff, the power requirements for portable applications are typically specified in mA/MHz,

---

*This work was supported in part by NSF CAREER award MIP-9703440 and by an IBM research grant.

[†]Now with Lucent Bell Labs.

[‡]Corresponding author. e-mail: mircea@virginia.edu

not in mW, with current specifications at $< 0.5 \text{ mA/MHz}$. In order to achieve such aggressive power targets attention needs to be paid to all aspects of chip design at all levels of abstraction, from process and library technology, to physical design, circuit and logic, to register transfer (RT) level and behavior.

I.2. The Power Wheel

The *power wheel* [1] seen in Figure 1 was used in this project for focusing the low power design activities. The wheel depicts major components needed for low power design and is inspired by Gajski's well known Y-chart [2]. At the center of the wheel are the low power technology and

supporting cell models, with the outer rings representing higher levels of abstraction from transistor to gate to RT and behavior. The components of the wheel are closely integrated through the low power design methodology used for the project.

I.3. Low Power Design Methodology

Low-power products have traditionally been custom designed, with the help of accurate power estimators and power reporting tools, in order to deliver first-pass silicon that meets the target power specification. Typically, a semicustom, cell-based, design methodology was considered less attractive for low power products because

FIGURE 1 The power wheel represents the components of a low power design methodology (from [1]).

less control is available for the designer to achieve aggressive design specifications. In this paper, we present the process and techniques by which we achieved the low power design of a DSP core for mobile (battery powered) applications using a semicustom methodology. These techniques can be used for other semicustom designed cores and have been incorporated into the low power design methodology shown in Figure 2 [1].

To achieve the desired power reduction target (10 times less than a standard first pass design),

several major efforts were undertaken:

- an ASIC design system was chosen that supports dual rail power supplies, with the core running at a lower voltage than the I/O (described in Section II),
- new low power core library elements were developed, including SRAM, ROM, PLL, DC-DC voltage regulator, and one new logic cell type (described in Section III),
- a method was created to enable fast and accurate power consumption reporting at the RT, logic and transistor levels (described in Section IV),

FIGURE 2 Diagram of the low power design methodology (from [1]).

- the architecture and structure of the design were both modified to reduce dynamic power (described in Section V),
- novel methods for active power reduction were developed, including pseudo-microcode, transition-once logic, very wide word (VWW) control (described in Section VI),
- novel clock gating methods were derived by taking into account both the logic and the physical design levels (described in Section VII),
- bit-stacking (bit-slicing) and latch/clock splitter “power groups” were extensively used in physical design (described in Section VIII).

In the following sections more details are given on the various low power techniques used in the design of the processor core.

II. TECHNOLOGY

The process used for designing this DSP core is a 0.25 μm drawn (0.18 μm effective) CMOS process with 5 layers of Aluminum interconnect and a nominal voltage of 2.5 V. The core is run at a reduced 1.8 V supply which results in a power reduction of over 20% compared to nominal and, thanks to the semicustom design methodology, can be easily migrated to other CMOS processes for extra power savings through simple scaling. Power shutdown circuitry is incorporated into the DC–DC voltage regulators in order to reduce static power during standby and a diode is used during shutdown mode to hold core voltage to non-critical circuits.

III. CORE LIBRARY

The macros used for the project were optimized to meet low power objectives. Quasi-differential drive latches were used to reduce the clock input loading by 80%. Low power clock splitters are used exclusively to drive low power latches and enable clock gating. The ASIC library was expanded to

include drive strengths with smaller device widths so power could be minimized on non-critical paths.

An analog PLL was designed with low power features. Starting from an existing design that was intended for high performance ICs, the maximum operating frequency was reduced from 400 MHz to 100 MHz. Clock shutdown modes specified at the architecture level allow the PLL to be suspended during chip standby and sleep modes. Node toggling was eliminated on logic branches that are unused during a given mode of operation and DC paths were reduced or eliminated.

A ROM was also newly designed such that input addresses are latched only if a ROM access is requested. Partitioning and proper sequencing of the ROM timing events was orchestrated to minimize overlap of events that could cause DC paths or high power conditions. The word line decoder divides the 256-word lines into 32 subdivisions of 8-word lines each, with only one subdivision being active at a time. For each data output bit, only one of the 32-bit lines that could be accessed is pre-charged. Finally, ROM output data is held until the next ROM READ operation is executed, reducing glitching downstream.

The 3-way multi-port SRAM designed for the core uses a number of techniques to minimize power consumption. Clock-gating is performed in the interface logic based on port request signals, minimizing clock tree and latch power. The multi-way accessing is implemented using three sequential operations to a standard six-transistor cell to reduce area. Support circuits are fully interlocked, maximizing the power/performance efficiency. To minimize array power, a NAND decode word-line system is used, and word-line length is kept at 64 bits, reducing bit-line pre-charge power and all associated sense amp and control signal power. Standby power is reduced primarily by the use of longer than minimum channel length devices, thus reducing the subthreshold leakage of the OFF word line driver PFET transistors.

IV. POWER REPORTING AT THE RT, LOGIC AND TRANSISTOR LEVELS

The developed low power methodology supports power reporting at the RT, logic, and transistor levels. The equivalent capacitances used for computing power are not only the physical ones (gate, diffusion, wires), but also equivalent capacitances which model second order effects as a function of slew rate, input and output transition edges, previous state of the gate, process, temperature, *etc.*

The power calculation expected accuracy is intrinsically reduced at higher levels of abstraction but basic correlation must exist to ensure that power reduction at the RT level results in corresponding reductions at the logic and transistor levels. Statistical, probabilistic or random switching models are used to *estimate* power at different levels of abstraction. The power consumption can be also accurately *calculated* if switching factors are available from logic simulation, but finding a set of representative applications that will yield power consumption close to the average power of the hardware is generally difficult. The advantage for battery-powered embedded applications is that the micro-processor is typically limited to the code in the ROM, thus the range of applications is bounded, embedded processors spending most of their powered-up cycles in certain segments of code. This bounded code can confidently be used to characterize average dynamic operating power and identify instructions with high usage frequency, this being the method used in this project.

IV.1. Power Consumption Reporting

The power consumption report is a key component of the power reduction process. The power report developed for this project is formatted at different levels of abstraction such that a designer can promptly implement power reduction changes. Too low a level of detail, such as at the cell level,

may be useless to a designer who is thinking in terms of high-level blocks. The customizable power reporting program developed for this project abstracts the detailed power information from the internal EDA tool (PowerCalc) report and presents it at a level determined by the designers. Power can be reported not only at the net or gate level, but also at the bus, data-path, unit, or sub-unit abstraction levels, all under user control.

IV.2. Power Estimation and Optimization at the Logic Level

At the logic level, an accurate power calculation is possible, limited only by the accuracy of the cell power models and the extracted parasitics. Early in the design process, estimated parasitics coupled with a zero-delay simulator provide enough accuracy, while post-wiring parasitics and nominal delay simulation are required for more accurate calculations later on. Logic synthesis selects gate sizes based on net capacitance and/or slew driven power minimization.

Once estimated parasitics are obtained from a place and route of the gate level netlist, a nominal delay simulation can be performed to extract the network switching factors using the power verification suite. The slew-rate driven power optimization tool is run to find the optimal slew rate and gate sizes for power consumption that will result in higher or lower powered cells being selected. At this point minimum shortcircuit current during CMOS switching is obtained while meeting the network timing constraints.

As mentioned before, it is more important to have a good correlation between power estimates at different levels of abstraction, even if the absolute numbers are expected to become more accurate only when the design gets refined. The correlation between RT level power estimates *via* node toggles and gate level power calculations varied for this project from 5% to over 30% difference from unit to unit.

V. ARCHITECTURAL AND STRUCTURAL CHANGES FOR LOW POWER

The two main approaches to RT level power reduction are first, to minimize the power required to implement a function, and second, to minimize how often the function needs to be executed. Two methods that were explored in this project for reducing power by structural changes are latch encoding and pipeline restructuring. Other two methods which also entail structural changes will be presented in the next section.

V.1. Latch Encoding

Latches are one of the major sources of power consumption in any synchronous system due to their loading of both the clock tree and data signals. Many times the latches are used in a mutually exclusive manner, in which case a low power alternative is the encoding/decoding of latched data with a reduction in the required number of latches as in Figure 3. Since the savings in clock tree power and latch power are typically much larger than the encoding/decoding overhead, such an approach can save significant power.

V.2. Pipeline Restructuring

A typical two-stage decoding pipeline consists of a first decoding stage followed by a second access stage as in Figure 4. A way to reduce power through structural changes is to move as much of

FIGURE 3 Latch encoding is useful if the extra power for encode/decode is $<$ than the savings in the clock tree and latch power: A. unencoded case, B. latches with encoding/decoding.

FIGURE 4 Pipeline restructuring reduces power by moving logic to prior pipeline cycle: A. Before restructuring, B. After restructuring.

the logic to the first (decode) stage, which will result in a simpler second (access) stage. The extra logic in the first stage only affects latches (for which glitches are removed at the clock boundary), but simplifying the second stage results in a more significant reduction of switching in the dataflow that is controlled by the access stage.

VI. NOVEL METHODS FOR ACTIVE POWER REDUCTION

Two methods developed for this project but with a significant applicability to other applications are the use of pseudo microcode and the transition-once logic cells (mux and buffer).

VI.1. Pseudo Microcode

The decoding techniques commonly used in the industry are distributed decode, where each logic control signal is derived as an independent cone of logic, and microcode, where each opcode is translated to an entry point address to a Read Only Memory (ROM) or Programmable Logic Array (PLA) that provides the complete set of control signals (see Fig. 5).

A new method of reducing active power was devised for this project to minimize cycle-to-cycle unnecessary toggles. With this method, next cycle control signals are derived as a function of both

FIGURE 5 Standard decode solutions: A. Distributed decode, B. microcode.

next cycle desired function, and the current cycle state. Thus, by minimizing cycle-to-cycle functional toggles, power can be saved independent of function implementation. The idea is simple, if a functional path is not required in the next cycle, ensure that all the control signals and data paths remain in the current state, eliminating all node toggles. For achieving this functionality we combined the simplicity of microcode with the flexibility of distributed decode into a style we call *pseudo-microcode*.

With pseudo-microcode all the control signals that can be set by a given opcode are bundled together as a total identity. These signals include the pipeline control, the address generation control, and the execute control bits. At each cycle boundary, a first stage very wide word (VWW) dispatch (see Fig. 6) is sent to n microcode units (22 for this processor). The VW words are decoded in each microcode unit for the given instruction type and the value for every control signal within that microcode unit is set. The important characteristic is that for every instruction decode, the control signals can be set not only to 0 or 1, but also to the previous value. By selectively changing only the signals that are required for each function, unnecessary toggles are eliminated.

FIGURE 6 Pseudo-microcode structure.

FIGURE 7 State machine pseudo-microcode decode.

Each opcode state becomes also a function of the previous opcode (see Fig. 7) and only conflicting control signals need to be resolved. In the example in the figure, during the first NOP instruction, the controls to the Arithmetic Logic Unit (ALU) are left in the MUL state even though the ALU is not needed to perform the NOP operation, *etc.*

With this method, the control signals for each unit remain in a stable state and only change when a new state is required. The processor becomes a complex state machine with the added ability to embed logic within the microcode, such as conditional checks within an opcode type. This can reduce the number of microcode addresses which also reduce the overall size of the decode unit, further saving power.

VI.2. Data-path Toggle Reduction with Transition-once Logic Cells

A common element in data-path processors is the bus multiplexer. Multiplexers select one of several data buses to be driven onto a single output bus as in Figure 8. If care is not taken, unwanted glitches on multiplexer inputs can propagate to the output buses and to downstream logic, even when the multiplexer is not used during the current cycle. The standard approaches for multiplexer toggle reduction are to either hold the previous select value, or to force the multiplexer to a default select state when not in use. While these approaches can save power when the multiplexer is idle for many cycles, if the multiplexer is required to change values frequently, no power savings can be obtained this way. An alternative method was created for data-path multiplexing which guarantees that the multiplexer outputs will switch once and only once during the cycle when they are needed and never switch on cycles where the multiplexer is not functionally needed. In a data-path design, due to individual path delay differences, different bits of a bus will arrive at different times within the cycle. This may result in repeated useless arithmetic or logical calculations downstream as an input bus settles on the final value for

FIGURE 8 Glitching power for datapaths can be reduced at the RT level by controlling the muxes that feed the datapath.

the cycle. To minimize data-path power, inputs to arithmetic units, such as multipliers, shifters and adders, should be applied only once during the cycle on which they are used.

To achieve these goals, a new transition-once multiplexer was designed (see Fig. 9A). The transition-once multiplexer ensures a single transition on the output bus per cycle by guaranteeing that data is propagated through the multiplexer only when the input data buses are valid and stable. The transition-once multiplexer holds the previous value during transitions to a new value. If the multiplexer is not selected in the current cycle, the output is also held at the previous cycle's state. The following four rules define the transition-once multiplexer:

- (1) No new data on the inputs until select off,
- (2) No new select until new data on the inputs,
- (3) If no select, hold previous value,
- (4) If same select, deselect (and hold previous value) during data transition time.

To accomplish these four rules, two selects are defined for every data port, one being called the *fast* select, the other the *slow* select. The fast select is timed to always arrive earlier than the fastest data bit for that input port. The slow select is timed to arrive later than the slowest bit of the same data port. The two selects are fed into an exclusive NOR (XNOR). Both selects must have the same value (either both 0 or both 1) for the multiplexer input value to be propagated to the output stage. Between the input stage and the output stage is a soft latch which holds the output state when needed as seen in Figure 9B. If the same input port is to be selected from one cycle to another, then both selects need to be toggled (if both 0, then both become 1 sequentially in the next cycle). By toggling both selects, the input port is deselected during the input data transition and no glitches are propagated. The fast select changes state before the slow select, temporarily disconnecting the output port from the input during the transition of the input port. Once the slow select toggles to the same value as the fast select, the data

FIGURE 9 A. Transition-once mux symbol, B. Transition-once mux schematic, C. Example behavior.

transition is complete, and the new value of the selected input port is allowed to propagate to the output stage. By cutting down the intermediate transitions of data multiplexers and preventing false calculations of downstream logic from occurring, the transition-once multiplexer plays a major role in the dynamic power reduction effort. Note that, even if the fast and slow selects are not timed to arrive at their optimal times, there is no danger of malfunction and the only penalty is less reduction in glitching.

The truth table for generating the fast and slow select and a circuit that correctly generates the two selects are shown in Figure 10. This circuit

assumes that the select value is known one cycle before being actually needed which is typically true in a pipelined processor.

A variation of the transition-once multiplexer is a transition-once buffer which can be used within random logic. When a pinch-point exists in the design, a transition-once buffer can isolate the downstream logic cone from the upstream cone, reducing glitches. Using a similar control logic to the transition-once multiplexer (only one pair of selects in this case), the buffer output is allowed to transition only when the input signals are stabilized. Insertion of timed buffers is thus an easy way of reducing the AC toggle in random logic. The

FIGURE 10 A. Truth table for generating the fast and slow select, active selects (both 0 or both 1) are marked, B. Circuit for generating the fast and slow select.

overhead of the additional timed selects and the buffer itself are typically more than compensated by the elimination of downstream glitches that were due to the intermediate transitions in the previous stage.

VII. CLOCK GATING METHODOLOGY

Since the clock distribution network typically consumes a large percentage of the processor power, clock gating, where the clocks are turned off to portions of the network that do not require it, can save a lot of active power. Our ASIC clock distribution methodology uses a single clock phase which is distributed through a re-driven network to clock splitters, which create two non-overlapping, out-of-phase clocks. These clocks are then used to drive master/slave latches.

VII.1. Opcode Typing and Clock Gating at RT Level

Clock gating works well for pipelined data-flow logic, where clocking requirements can be pre-determined at least one cycle ahead, but is difficult within the current cycle for control registers due to the random nature of the control logic. In

addition, the clock gating signals must be valid halfway into the cycle to gate off the capture clock of a two-phase clocking scheme as used in a level-sensitive scan design (LSSD) as this project.

We developed an automated method for generating clock gating signals, called *opcode typing*, that maximizes the number of latches that are turned off every cycle and at the same time reduces the total number of clock gating groups for minimum overhead. This method searches the microcode functions and groups control signals into clock gating groups with the goal of maximizing the number of opcodes for which the signals in the clock gating group stay constant. A program with a flow as in Figure 11 iteratively picks different groupings of signals to be typed together, analyzes opcode coverage, and picks the type groupings that maximize opcode coverage with the minimum number of groupings. Figure 12 shows the clock distribution network for an LSSD-based design.

Once optimum clock gating groups are defined and opcodes have been assigned to types, typed clock gating can begin. Early in the cycle, opcodes are pre-decoded to generate a type field corresponding to the groups of registers. The current opcode's type field is then compared to the previous cycle type field. If the previous cycle type

FIGURE 11 Typing search flow.

is ON, and the current type is ON, then clock gating can be enabled since the control signals will have the same value from the previous opcode to the current opcode for that type. The number of clock gating signals for the registers in our microprocessor decode was 10, but in general this is entirely a function of the number of registers and the desired amount of type coverage. While this method of typing the control signals by opcode/microcode/function cannot gate the clock on a per-bit basis, a significant ratio of cycle-to-cycle clock gating can be obtained for control logic registers.

VII.2. Clock Gating at the Logic and Physical Design Levels

After determining the desired clock gating groups for datapath and control at the logic level, it is necessary to determine the savings in power due to gating before and after physical design. We developed a specialized clock distribution power reporting program that post-processes the PowerCalc power report and gathers all the clock

FIGURE 12 Clock distribution network with hierarchical single phase drivers and leaf two-phase splitters.

network power by unit and breaks it down by driver and splitter power. This data is very helpful when prioritizing and structuring the clock gating logic and analyzing the clock distribution network.

Depending on where gating is applied in the clock tree we considered the following three cases:

- *top clock gating* at the top level of the tree (this disables the entire circuit),
- *flat clock gating* only at the leaf (splitter) level of the tree (this disables the inactive loads but keeps the higher levels in the clock tree active at all times),
- *hierarchical clock gating* where gating is done both at the leaf level (to disable loads and ensure correct functionality) and also at higher levels in the tree based on typing information (to further disable portions of the clock tree distribution itself).

An important finding during the course of this project was that not only the logical level but also

FIGURE 13 Hierarchical gating routing problem: A. Flat clock gating (only at the leaf level), B. Hierarchical clock gating with increased wiring overhead due to nonclustered placement of the clock groups.

the physical design aspects (placement and routing) are very critical for evaluating the power reduction potential for clock gating [3]. Clock gating is always effective if the clock gating groups at the logic level are also clustered after placement but this is rarely the case. Figure 13 shows a possible problem with hierarchical clock gating if the gating groups at the logic level are not clustered during placement. Unfortunately, for most ASICs the placement is done automatically with the clock as a global signal which is not driving the placement algorithms, hence the resulting clock groups are typically not clustered. Decisions whether to gate or not a specific set of registers need to be taken based on switched capacitance. Figure 14 shows how gated clocks with a high degree of correlation can be combined together in order to reduce the overhead due to gating.

We developed an algorithm called *ungate* that outperforms the standard *hard groups* clock gating

FIGURE 14 Clock gating domain analysis: Clock gates A and B are similar, hence they can be combined into a single gating signal (A + B) for reduced overhead.

(i.e., all clock gating groups at the logical level are strictly maintained independent of physical design outcomes), and also outperforms a simplistic *soft groups* method which traverses the clock tree bottom-up and groups sinks based on switched

FIGURE 15 Ungate algorithm for hierarchical clock gating (from [3]).

capacitance. Figure 15 shows a simple example of the ungate algorithm while Figure 16 shows the results of using ungate on two real-life example core designs. The clock gating optimization flow is shown in Figure 17, more details on the clock gating aspects can be found in [3, 4].

Another important finding is that hierarchical gating can be quite effective for binary trees with fine-grain distributed drivers and balanced wiring (*e.g.* 20% savings), but is less effective

Algorithm	ASIC Core #1	ASIC Core #2
NO GROUPS	12.15 Units	7.21 units
HARD GROUPS	+31%	+17%
SOFT GROUPS	-14%	-9%
UNGATE	-26%	-16%

FIGURE 16 Ungate algorithm results (from [3]).

FIGURE 17 Clock gating optimization flow.

FIGURE 18 Latch clustering for reducing clock wiring overhead at the splitter level.

for non-binary trees with coarse-grain distributed drivers and Steiner routing. Unfortunately balanced-wire trees are used typically only in high-performance ICs (where the wiring overhead is justified by the required low skew and performance), but not for the ASICs. Our project uses a coarse-grain clock network with Steiner routing for which flat gating only at the lowest level proved to be more effective and was the final chosen solution.

VIII. BIT-STACKING AND POWER GROUPING

Several methods were used to force the physical design placement of regular structures into tightly clustered groups. Datapath elements were placed as bit-stacks (bit-slices), and this results in reduced wiring overhead for control signals which can be routed straight through the bit stack structure.

The last stage of the clock tree is typically the most critical for power dissipation. A binary tree with uniform distribution of the sinks has roughly half of the entire clock tree capacitance in the last stage, while for a non-binary tree the ratio is even

worse [4]. In our design methodology the impact of the last stage is compounded by the fact that two phases are needed (and generated by splitters just in the last stage) for driving LSSD latches. In order to minimize the impact of the last stage wiring load, latches were grouped around their corresponding splitters as in Figure 18. This grouping significantly reduces the power for the last clock stage at the expense of a slight increase of the wiring overhead for the latch input and output ports.

IX. CONCLUSION

We have described design methods for creating a low power processor design in a semi-custom ASIC methodology. Designing for lowest possible power consumption requires attention to the entire design process, including: technology, library selection, power reduction methodology development, architectural and gate-level logic design techniques, and physical design methodology. A complete RT-to-gate level methodology has been used to successfully identify, track, and implement a power reduction strategy. Emphasis was placed on power reporting and analysis, and automatic

low power synthesis capabilities. Several innovative design techniques were described for dynamic power reduction, including register typing for control logic clock gating, transition-once logic to reduce data-path node toggles, and pseudo-microcode to minimize overall instruction stream dependent power. The results of the RT level power optimizations (not including circuit or technology related savings) are summarized in Figure 19.

Method	% Power Saved
Low Power Synthesis	15
Clock Gating	8
Logic/Architectural Changes	45
Voltage Reduction	32

FIGURE 19 Savings for the methods not related to circuit design.

Acknowledgements

The authors thank the following individuals for their contributions to this work: John Adams, Dave Blum, J. Patrick Brennan, Rob Busch, Ken Goodnow, Roger Gregor, Stephen Kenyon, Gary Koch, Manabu Ohkubo, Chris Ro.

References

- [1] Brennan, J. P. *et al.*, "Low power methodology and design techniques for processor design", In: *International Symposium of Low Power Electronics and Design* (Monterey, CA), pp. 268–273, Aug., 1998.
- [2] Gajski, D., Dutt, N., Wu, A. and Lin, S. (1992). *High-level Synthesis, Introduction to Chip and System Design*, Boston, MA: Kluwer Academic Publishers.
- [3] Garret, D., Stan, M. R. and Dean, A., "Challenges in clock gating for low power for ASIC cores", In: *International Symposium on Low Power Electronics and Design* (San Diego, CA), July, 1999.
- [4] Garrett, D., Stan, M. R. and Dean, A. (2000). "Clock gating techniques for ASICs", *Submitted to IEEE Transactions on VLSI*.