Low-power RT-level synthesis techniques: a tutorial

M. Pedram and A. Abdollahi

Abstract: Power consumption and power-related issues have become a first-order concern for most designs and loom as fundamental barriers for many others. While the primary method used to date for reducing power has been supply voltage reduction, this technique begins to lose its effectiveness as voltages drop to below one volt and further reductions in the supply voltage begin to create more problems than are solved. Under these circumstances, the process of design and the automation tools required to support that process become the critical success factors. In the last decade, huge effort has been invested to come up with a wide range of design solutions that help solve the power dissipation problem for different types of electronic devices, components and systems. These techniques range from RTL power management and multiple voltage assignment, to power-aware logic synthesis and physical design, to memory and bus interface design. A number of representative low-power design techniques from this large set are explained. More precisely, basic techniques are described, that are applicable at RT-level and below, and have proved to hold good potential for power optimisation in practical design environments.

1 Introduction

A dichotomy exists in the design of modern microelectronic systems; they must be low power and high performance, simultaneously. This dichotomy largely arises from the use of these systems in battery-operated portable (wearable) platforms. Accordingly, the goal of low-power design for battery-powered electronics is to extend the battery service life while meeting performance requirements. Unless optimisations are applied at different levels, the capabilities of future portable systems will be severely limited by the weight of the batteries required for an acceptable duration of service. In fixed, power-rich platforms, the packaging cost and power density/reliability issues associated with high-power and high-performance systems also force designers to look for ways to reduce power consumption. Thus, reducing power dissipation is a design goal even for nonportable devices, since excessive power dissipation results in increased packaging and cooling costs as well as potential reliability problems. Meanwhile, following Moore's Law, integrated circuit densities and operating speeds have continued to go up in unabated fashion. The result is that chips are becoming larger, faster and more complex, and because of this, consuming increasing amounts of power.

These increases in power pose new and difficult challenges for integrated circuit designers. While the initial response to increasing levels of power consumption was to reduce the supply voltage, it quickly became apparent that this approach was insufficient. Designers subsequently began to focus on advanced design tools and methodologies to address the myriad of power issues. Complicating designers' attempts to deal with these issues are the complexities, logical, physical and electrical, of contemporary IC designs and the design flows required to build them.

This article reviews a number of representative RT-level design automation techniques that focus on low-power design. It should be of interest to designers of power-efficient devices, IC design engineering managers, and EDA managers and engineers. More precisely, it covers techniques for sequential logic synthesis, RT-level power management, multiple-voltage design and low-power bus encoding techniques. Interested readers can find wide-ranging information on various aspects of low-power design in [1–3]. Note that although, in many of today's designs, the leakage component of power consumption has become comparable to the dynamic component, this tutorial does not discuss the leakage issue. Interested readers may refer to any of the excellent references on leakage power, including those in the abovementioned edited books.

2 Multiple-voltage design

Using different voltages in different parts of a chip may reduce the global energy consumption of a design at a rather small cost in terms of algorithmic and/or architectural modifications. The key observation is that the minimum energy consumption in a circuit is achieved if all circuits paths are timing-critical (there is no positive slack in the circuit). A common voltage scaling technique is thus to operate all the gates on non-critical timing paths of the circuit at a reduced supply voltage. Gates/modules that are part of the critical paths are powered at the maximum allowed voltage, thus avoiding any delay increase; the power consumed by the modules that are not on the critical paths, however, is minimised because of the reduced supply voltage. Using different power supply voltages on the same chip of circuitry requires the use of level shifters at the boundaries of the various modules (a level converter is needed between the output of a gate powered by a low V_{DD} and the input of a gate powered by a high V_{DD}, i.e. for a step-up change). Figure 1 depicts a typical level converter design. Notice that if a gate that is supplied with $V_{DD,L}$ drives

IEE Proceedings online no. 20045111

doi: 10.1049/ip-cdt:20045111

Paper first received 17th August and in revised form 3rd November 2004

The authors are with the Department of Electrical Engineering, University of Southern California, 3740 McClintock Ave, Los Angeles, CA 90089, USA

E-mail: pedram@usc.edu

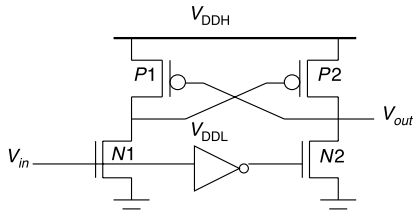

Fig. 1 *Typical level-converter design*

a fanout gate at $V_{DD,H}$, transistors N1 and N2 receive inputs at reduced supply and the cross-coupled PMOS transistors do the level conversion. Level converters are obviously not needed for a step-down change in voltage. Overhead of level converters can be mitigated by doing conversions at register boundaries and embedding the level conversion inside the flip-flops (see [4] for details).

A polynomial time algorithm for multiple-voltage scheduling of performance-constrained non-pipelined designs is presented by Raje and Sarrafzadeh in [5]. The idea is to establish a supply voltage level for each of the operations in a data flow graph, thereby fixing the latency of that operation. The goal is then to minimise the total power dissipation while satisfying the system timing constraints. Power minimisation is in turn accomplished by ensuring that each operation will be executed using the minimum possible supply voltage. The proposed algorithm is composed of a loop where, in each iteration, slacks of nodes in the acyclic data flow graph are calculated. Then, nodes with the maximum slack are assigned to lower voltages in such a way that timing constraints are not violated. The algorithm stops when no positive slack exists in the data flow graph. Notice that this algorithm assumes that the Pareto-optimal voltage against delay curve is identical for all computational elements in the data flow graph. Without this assumption, there is no guarantee that this algorithm will produce an optimal design.

In [6], the problem is addressed for combinational circuits, where only two supply voltages are allowed. A depth-first search is used to determine those computational elements which can be operated at low supply voltage without violating the circuit timing constraints. A computational element is allowed to operate at $V_{DD,L}$ only if all its successors are operating at $V_{DD,L}$. For example, Fig. 2*a* demonstrates a clustered voltage scaling (CVS) solution in which each circuit path starts with $V_{DD,H}$ and switches to $V_{DD,L}$ when delay slack is available. The timing-critical path is shown with thick line segments. Here grey-coloured cells are running at $V_{DD,L}$. Level conversion (if necessary) is done in the flip-flops at the end of the circuit paths. An extension to this approach is proposed in [7], which is based on the observation that by optimising the insertion points of level converters, one can increase the number of gates using $V_{DD,L}$ without increasing the number of level converters. This leads to higher power savings. For example, in the CVS solution depicted in Fig. 2*a*, assume that the path delay from flip-flop FF3 to gate G2 is much longer than that of the path from FF1 to G2. In addition, assume that if we apply $V_{DD,L}$ to G2, then the path from FF3 to FF5 through G2 will miss its target combinational delay, i.e. G2 must be assigned a supply level of $V_{DD,H}$. With the CVS approach, it immediately follows that G3 must be assigned $V_{DD,H}$ although a potentially large positive slack remains in the path from FF1 to G2. The situation is the same for G4 and G5. Consequently, the CVS approach can miss opportunities for applying $V_{DD,L}$ to some gates in the circuit. If the insertion point of the level converter LC1 is

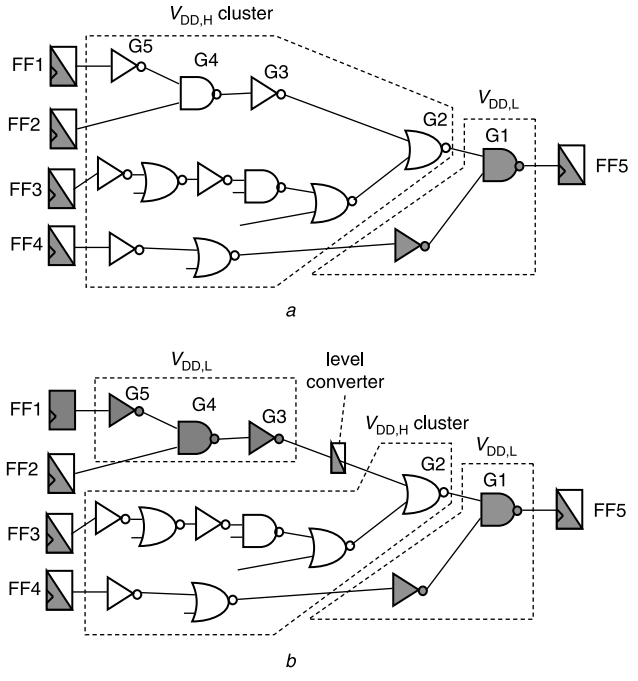

Fig. 2 *Examples*
a CVS solution
b ECVS solution

allowed to move up to the interface between G3 and G2, the gates G3 through G5 can be assigned a supply of $V_{DD,L}$, as depicted in Fig. 2*b*. The structure shown there is one that can be obtained by the extended CVS (ECVS) algorithm. Both CVS and ECVS assign the appropriate power supply to the gates by traversing the circuit from the primary outputs to the primary inputs in a levelised order. ECVS allows a $V_{DD,L}$-driven gate to feed a $V_{DD,H}$-driven gate along with the insertion of a dedicated level converter.

In [8], the authors propose an approach for voltage assignment in combinational logic circuits. First, a lower bound on dynamic power consumption is determined by exploiting the available slacks and the value of the dual-supply voltages that may be used in solving the problem of minimising dynamic power consumption of the circuit. Next, a heuristic algorithm is proposed for solving the voltage-assignment problem, where the values of the low and the high supply voltages are either specified by the user or fixed to the estimated ones.

In [9], the authors present resource- and latency-constrained scheduling algorithms to minimise power/energy consumption when the resources operate at multiple voltages. The proposed algorithms are based on efficient distribution of slack among the nodes in the data-flow graph. The distribution procedure tries to implement the minimum energy relation derived using the Lagrange multiplier method in an iterative fashion.

An important phase in the design flow of multiple-voltage systems is that of assigning the most convenient supply voltage, selected from a fixed number of values, to each operation in the control-date flow graph (CDFG). The problem is to assign the supply voltages and to schedule the tasks so as to minimise the power dissipation under throughput/resource constraints. An effective solution has been proposed by Chang and Pedram in [10]. The technique is based on dynamic programming and requires the availability of accurate timing and power models for the macromodules in a RTL library. A preliminary characterisation procedure must then be run to determine

an energy-delay curve for each module in the library and for all possible supply-voltage assignments. The points on the curve represent various voltage assignment solutions with different tradeoffs between the performance and the energy consumption of the cell. Each set of curves is stored in the RTL library, ready to be invoked by the cost function that guides the multiple supply-voltage scheduling algorithm. We provide a brief description of the method for the simple case of control and data flow graphs (CDFGs) with a tree structure. The algorithm consists of two phases; first, a set of possible power-delay tradeoffs at the root of the tree is calculated, then a specific macromodule is selected for each node in such a way that the scheduled CDFG meets the required timing constraints. To compute the set of possible solutions, a power-delay curve at each node of the tree (proceeding from the inputs to the output of the CDFG) is computed; such a curve represents the power-delay tradeoffs that can be obtained by selecting different instances of the macromodules, and the necessary level shifters within the subtree rooted at each specific node. The computation of the power-delay curves is carried out recursively, until the root of the CDFG is reached. Given the power-delay curve at the root node, that is the set of tradeoffs the user can choose from, a recursive preorder traversal of the tree is performed, starting from the root node, with the purpose of selecting which module alternative should be used at each node of the CDFG. Upon completion, all the operations are fully scheduled; therefore, the CDFG is ready for the resource-allocation step.

More recently, a level-converter-free approach is proposed in [11], where the authors try to eliminate the overhead imposed by level converters by suggesting a voltage scaling technique without utilising level converters. The basic initiative is to impose some constraints on the voltage differences between adjacent gates with different supply voltages based on the observation that there will be no static current if the supply voltage of a driver gate is higher than the subtraction of the threshold voltage of a PMOS from the supply voltage of a driven gate. In [12], the authors propose behavioural-level power optimisation algorithms that use voltage and frequency scaling. In this work, the operators in a data flow graph are scheduled in the modules of the given architecture by applying voltage and frequency scaling techniques to the modules of the architecture, such that the power consumed by the modules is minimised. The global optimal selection of voltages and frequencies for the modules is determined through the use of an auction-theoretic model and a game-theoretic solution. The authors present a resource-constrained scheduling algorithm, which is based on applying the Nash equilibrium function to the game-theoretic formulation.

3 RT-level power management

Digital circuits usually contain portions that are not performing useful computations at each clock cycle. Power reductions can then be achieved by shutting down the circuitry when it is idle.

3.1 Precomputation logic

Precomputation logic, presented in [13], relies on the idea of duplicating part of the logic with the purpose of precomputing the circuit output values one clock cycle before they are required, and then uses these values to reduce the total amount of switching in the circuit during the next clock cycle. In fact, knowing the output values one clock cycle in advance allows the original logic to be turned off during the next time frame, thus eliminating any charging and discharging of the internal capacitances. Obviously, the size of the logic that precalculates the output values must be kept under control since its contribution to the total power balance may offset the savings achieved by blocking the switching inside the original circuit. Several variants to the basic architecture can then be devised to address this issue. In particular, sometimes it may be convenient to resort to partial, rather than global, shutdown, i.e. to select for power management only a (possibly small) subset of the circuit inputs.

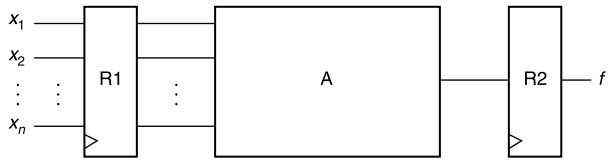

Fig. 3 *Pipeline stage of a data path*

Figure 3 shows a combinational block A that implements an n-input, single-output Boolean function f, with registers R1 and R2 connected to its inputs and output pins, respectively. A precomputation architecture realisation of this same logic block placed between register sets R1 and R2 is depicted in Fig. 4. The key elements of the precomputation architecture are two n-input, single-output predictor functions $g1$ and $g2$, which satisfy the following constraints:

$$g1 = 1 \Rightarrow f = 1$$

$$g2 = 1 \Rightarrow f = 0$$

If, at the present clock cycle, $g1$ or $g2$ evaluate to one, then the load enable signal *LE* goes to zero, and the inputs to block A at the next clock cycle are forced to retain the current values. Hence, no gate output transitions inside block A occur, while the correct output value for the next time frame is provided by the two registers located on the outputs of $g1$ and $g2$. Note that the precomputation logic is a function of a subset of the input variables, hence, it is called a 'subset input-disabling architecture'.

The synthesis algorithm presented in [13] suffers from the limitation that if a logic function is dependent on the values of several inputs for a large fraction of the applied input combinations, then no reduction in switching activity can be obtained. In [14], the authors focus on a particular sequential precomputation architecture, where the precomputation logic is a function of all of the input variables. The authors call this architecture the 'complete input-disabling architecture'. It is shown that the complete input-disabling architecture can reduce power dissipation

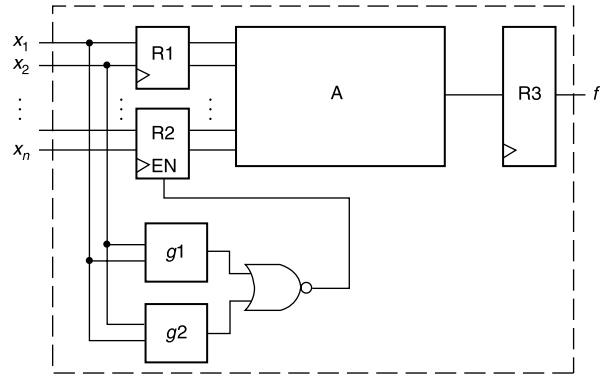

Fig. 4 *Precomputation logic realisation of the pipeline stage (subset-input disabling architecture)*

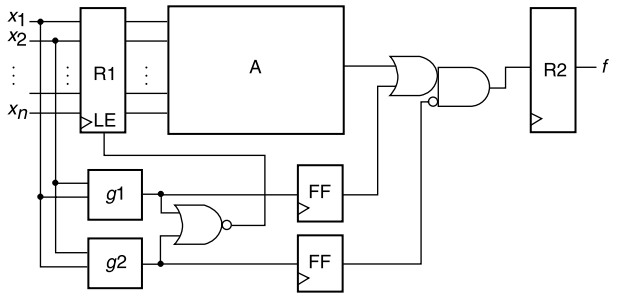

Fig. 5 *Example of a complete input-disabling precomputation architecture*

for a larger class of sequential circuits compared to the subset input-disabling architecture. The authors present an algorithm to synthesise precomputation logic for the complete input-disabling architecture.

In Fig. 5, a complete input-disabling precomputation architecture for a comparator circuit is shown. Functions $g1$ and $g2$ satisfy the conditions of (1) and (2) as before. During clock cycle t; if either $g1$ or $g2$ evaluates to a 1, the load enable signal of register R1 is set to be 0. This means that in clock cycle $t + 1$; none of the inputs to the combinational logic block A change. If $g1$ evaluates to 1 in clock cycle t, the input to register R2 is a 1 in clock cycle $t + 1$, and if $g2$ evaluates to a 1, then the input to register R2 is a 0. Note that $g1$ and $g2$ cannot both be 1 during the same clock cycle, due to the conditions imposed by (1) and (2). The important difference between this architecture and the subset input-disabling architecture shown in Fig. 4 is that the precomputation logic can be a function of all input variables, allowing us to precompute any input combination.

3.2 Clock gating

Another approach to RT-level and gate-level dynamic power management, known as gated clocks [15–17], provides a way to selectively stop the clock, and thus force the original circuit to make no transition, whenever the computation that is to be carried out at the next clock cycle is redundant. In other words, the clock signal is disabled according to the idle conditions of the logic network. For reactive circuits, the number of clock cycles in which the design is idle in some wait states is usually large. Therefore, avoiding the power waste corresponding to such states may be significant.

The logic for the clock management is automatically synthesised from the Boolean function that represents the idle conditions of the circuit (cf. Fig. 6.) It may well be the case that considering all such conditions results in additional circuitry that is too large and too power consuming. It may then be necessary to synthesise a simplified function, which dissipates the minimum possible power and stops the clock with maximum efficiency. The use of gated clocks has the

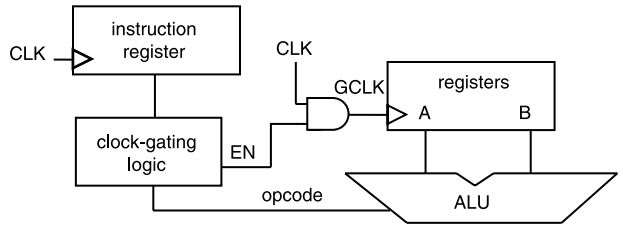

Fig. 6 *Clock gating logic for ALU in a typical processor microarchitecture with negative-edge triggered flip-flops*

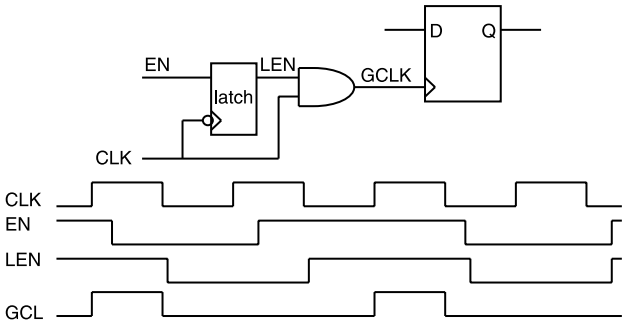

Fig. 7 *Clock is disabled when EN = 0; furthermore, a hazard on EN will be stopped from reaching GCLK*

drawback that the logic implementing the clock-gating mechanism is functionally redundant, and this may create major difficulties in testing and verification. The design of highly testable gated-clock circuits is discussed in [18].

Another difficulty with clock-gating is that one must stop hazards/glitches on the EN signal from corrupting the clock signal to the register sets. This can be accomplished by introducing a transparent negative latch between EN and the AND gate as shown in Fig. 7.

3.3 Computational kernels

Sequential circuits may have an extremely large number of reachable states, but during normal operation these circuits tend to visit only a relatively small subset of the reachable states. A similar situation occurs at the primary outputs; while the circuit walks through the most probable states, only a few distinct patterns are generated at the combinational outputs of the circuit. Many researchers have proposed approaches for synthesising a circuit that is fast and power-efficient under typical input stimuli, but continues to operate correctly even when uncommon input stimuli are applied to the circuit.

Reference [19] presents a power optimisation technique by exploiting the concept of computational kernel of a sequential circuit, which is a highly simplified logic block that imitates the steady-state behaviour of the original specification. This block is smaller, faster and less power-consuming than the circuit from which it is extracted, and can replace the original network for a large fraction of the operation time.

The p-order computational kernel of an FSM is defined with respect to a given probability threshold p and includes the subset of the states S_p of the original FSM whose steady-state occupation probabilities are larger than p. The combinational kernel also includes the subset of states R_p, where for each state in R_p there is an edge from a state in S_p to that state. As an example, consider the simple FSM shown in Fig. 8*a* in which the input and output values are omitted for the sake of simplicity and the states are annotated with the steady-state occupation probabilities calculated through Markovian analysis of the corresponding state transition graph (STG.) If we specify a probability threshold of $p = 0.25$, then the computational kernel of the FSM is depicted in Fig. 8*b*. States in black represent set S_p, while states in grey represent R_p. The kernel probability is $\text{Prob}(S_p) = 0.29 + 0.25 + 0.32 = 0.86$.

Given a sequential circuit with the standard topology depicted in Fig. 9*a*, the paradigm for improving its quality with respect to a given cost function (e.g. power dissipation, latency) is based on the architecture shown in Fig. 9*b*.

The basic elements of the architecture are the combinational portion of the original circuit (block CL), the

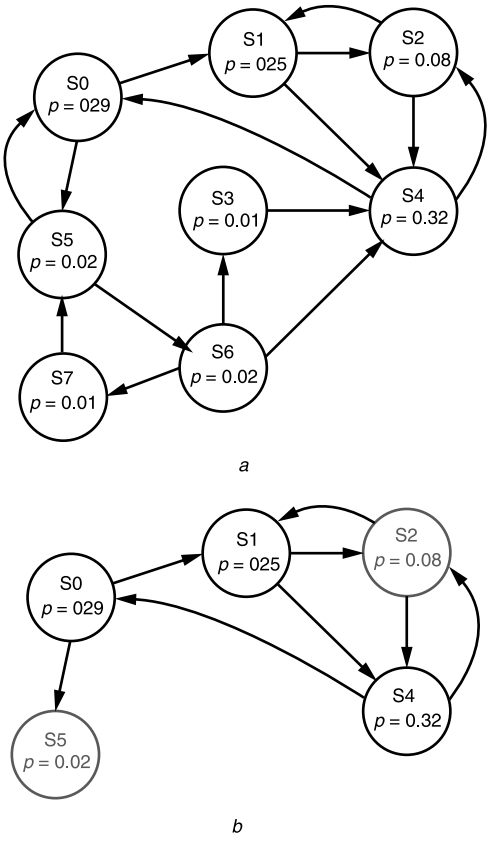

Fig. 8 *Moore-type FSM and its computational bernel*

a Moore-type FSM
b Its 0.25-order computational kernel

computational kernel (block K), the selector function (block S), the double-state flip-flops (DSFF) and the output multiplexers (MUX).

The computational kernel can be seen as a 'dense' implementation of the circuit from which it has been extracted. In other terms, K implements the core functions of the original circuit and, because of its reduced complexity, it usually implements such functions in a faster and more efficient way. The purpose of selector function S is that of deciding what logic block, between CL and K, will provide the output value and the next-state in the following clock cycle. To take a decision, S examines the values of the next-state outputs at clock cycle n. If the output and next-state values in cycle $n + 1$ can be computed by the kernel K, then S takes on the value 1. Otherwise, it takes on the value 0.

The value of S is fed to a flip-flop, whose output is connected to the MUXes that select which block produces the output and the next-state. The optimised implementation is functionally equivalent to the original one. Computational kernels are a generalisation of the precomputation architecture from combinational and pipelined sequential circuits to finite-state machines. The authors in [19] proposed an algorithm for generating the computational kernel of a FSM by iterative simplification of the original network by redundancy removal.

In [20], the authors raise the level of abstraction at which the kernel-based optimisation strategy can be exploited and show how RTL components, for which only a functional specification is available, can be optimised using the computational kernels. They present a technique for computational kernel extraction directly from the functional specification of a RTL module. Given the state transition graph (STG) specification, the proposed algorithm

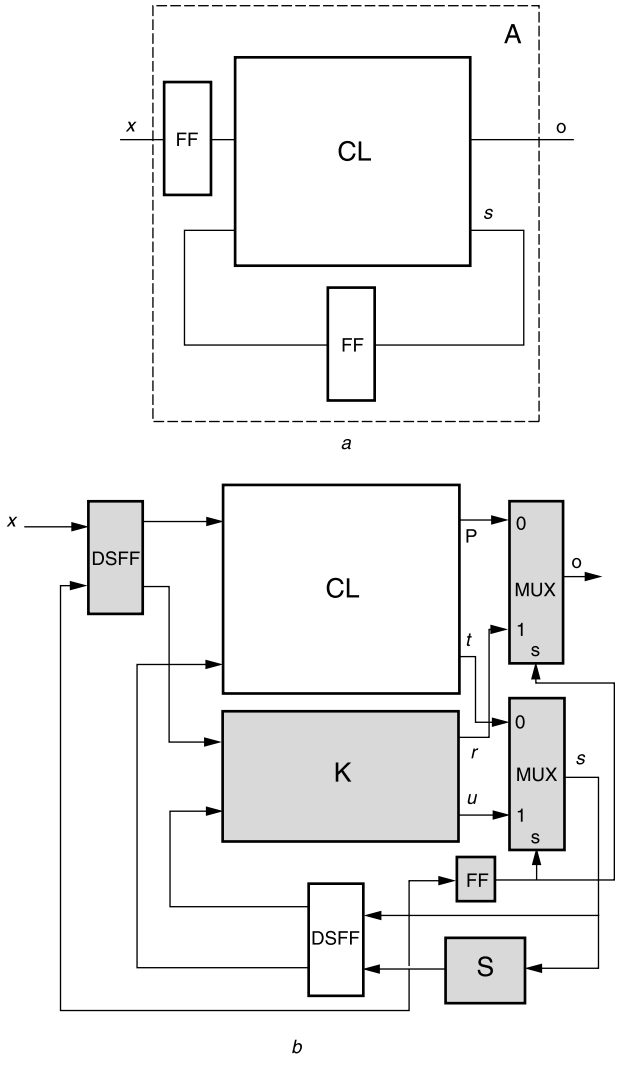

Fig. 9 *Kernel-based optimised architecture*

calculates the kernel exactly through symbolic procedures similar to those employed for FSM reachability analysis. The authors also provide approximate methods to deal with large STGs. More precisely, they propose two modifications to the basic procedure. The first one replaces the exact probabilistic analysis of the STG with an approximate analysis. In the second solution, symbolic state probability computation is bypassed and the set of states belonging to the kernel is determined directly from RTL simulation traces of a given (random or user-provided) stream.

3.4 State machine decomposition

Decomposition of finite-state machines for low power has been proposed in [21]. The basic idea is to decompose the STG of a finite-state machine (FSM) into two STGs that jointly produce the equivalent input–output behaviour as the original machine. Power is saved because, except for transitions between the two sub-FSMs, only one of the sub-FSMs needs to be clocked. The technique follows a standard decomposition structure. The states are partitioned by searching for a small subset of states with high probability of transitions among these states and a low probability of transitions to and from other states. This subset of states will then constitute a small sub-FSM that is active most of the time. When the small sub-FSM is active, the other larger sub-FSM can be disabled. Consequently, power is saved

because most of the time only the smaller, more power-efficient sub-FSM is clocked.

In [22], the combinational logic block is partitioned (for example to CL1 and CL2) and the active part is decided based on the encoding of the present state. The states selected for one of the sub-FSMs (i.e. M1) are all encoded in such a way that the enable signal is always on for CL1, while it is off for CL2. Conversely, for all states in the other sub-FSM (i.e. M2), the enable signal is always off for CL1, while it is on for CL2. Consequently, for all transitions within M1, only CL1 will be active and vice versa.

Consider as an example *dk*27 FSM from the MCNC benchmark set, depicted in Fig. 10. Assume that the input signal values, 0 and 1, occur with equal probabilities. The steady-state probabilities, which are shown next to the states in this Figure, have been computed accordingly. Suppose we partition the FSM into two sub-machines M1 and M2 along the dotted line. Then, around 40% of the transitions occur in submachine M1, 40% of the transitions occur in submachine M2 and 20% of the transitions occur between sub-machines M1 and M2. Now suppose that the FSM is synthesised as two individual combinational circuits for sub-machines M1 and M2. Then, we can turn off the combinational circuit for submachine M2 when transitions occur within submachine M1. Similarly, we can turn off the combinational circuit for submachine M1 when transitions occur within submachine M2. The states are partitioned such that the probability of transitions within any sub-FSM is maximised and the estimated overhead is minimised.

These methods for FSM decomposition can be considered as extensions of the gated-clock for FSM self-loops approach proposed in [23]. In FSM decomposition the cluster of states that are selected for one of the sub-FSMs can be considered as a 'super state' and then transitions between states in this cluster can be seen as self-loops on this 'super state'.

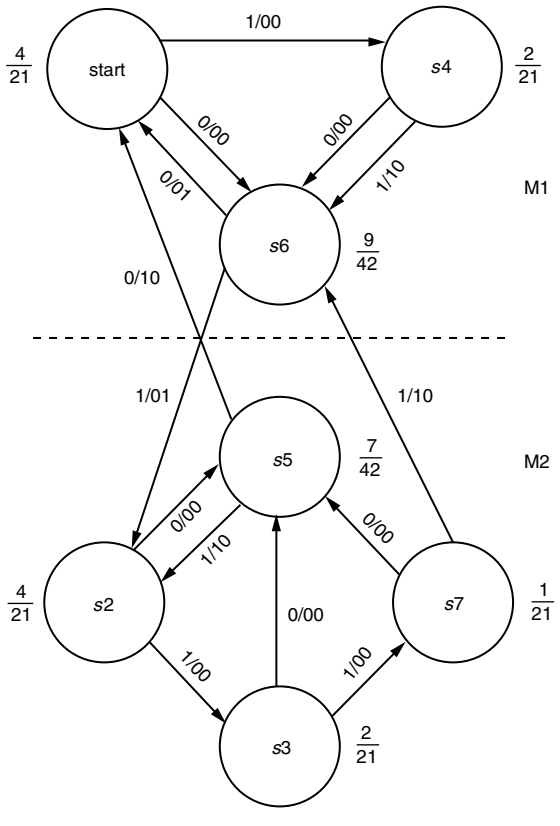

Fig. 10 *Example of an FSM (dk27) that may be decomposed into two sub-FSMs such that one sub-FSM can be shut off when the other is active and vice versa*

3.5 Guarded evaluation

Guarded evaluation [24] is the last RT-level and gate-level shutdown technique we review in this Section. The distinctive feature of this solution is that, unlike precomputation and gated clocks, it does not require one to synthesise additional logic to implement the shutdown mechanism. Instead, it exploits existing signals in the original circuit. The approach is based on placing some guard logic, consisting of transparent latches with an enable signal, at the inputs of each block of the circuit that needs to be power-managed. When the block must execute some useful computation in a clock cycle, the enable signal makes the latches transparent. Otherwise, the latches retain their previous states, thus blocking any transition within the logic block.

Guarded evaluation provides a systematic approach to identify where transparent latches must be placed within the circuit and by which signals they must be controlled. For example, let C be a combinational logic block (cf. Fig. 11*a*), X be the set of primary inputs to C, and z be a signal in C. Furthermore, let F be the portion of logic that drives z and let Y be the set of inputs to F. Finally, let $D_Z(X)$ be the observability don't-care set for z (that is, the set of primary input assignments for which the value of z does not influence the outputs of C). Now consider a signal s in C which logically implies $D_Z(X)$, that is, $s \Rightarrow D_Z(X)$. Then, if $s = 1$, then the value of z is not required to compute the outputs of C. If we call $t_e(Y)$ the earliest time at which any input to F can switch when $s = 1$, and $t_l(s)$ as the latest time at which s settles to one, then signal s can be used as the guard signal for F (cf. Fig. 11*b*) if $t_l(s) < t_e(Y)$. This is because z is not required to compute the outputs of C when $s = 1$, and therefore block F can be shut down. Notice that the condition $t_l(s) < t_e(Y)$ guarantees that the transparent latches in the guard logic are shut down before any of the inputs to F makes a transition.

This technique, referred to as pure guarded evaluation, has the desirable property that when applied, no changes in the original combinational circuitry are needed. However, if some resynthesis and restructuring of the original logic is

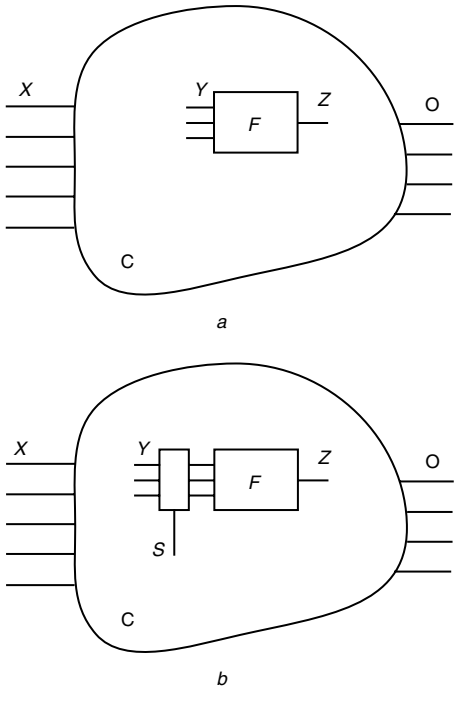

Fig. 11 *Example of guard logic insertion*

allowed, a larger number of logic shutdown opportunities may become available.

4 Sequential logic synthesis for low power

Power can be minimised by appropriate synthesis of logic. The goal in this case is to minimise the so-called switched capacitance of the circuit by low-power-driven logic minimisation techniques.

4.1 State assignment

State encoding/assignment, as a crucial step in the synthesis of the controller circuitry, has been extensively studied. Roy and Prasad were was the first to address the problem of reducing switching activity of input state lines of the next state logic, during the state assignment, formulating it as a minimum weighted Hamming distance problem [25]. Olson and Kang used a linear combination of switching activity of the next-state lines and the number of literals as the cost function [26]. Tsui *et al.* [27] used simulated annealing as a search strategy to find a low-power state encoding that accounts for both the switching activity of the next-state lines and switched capacitance of the next-state and output logic.

For example, consider the state transition graph for a BCD to Excess-3 converter depicted in Fig. 12. Assume that the transition probabilities of the thicker edges in this Figure are more than those of the thin edges. The key idea behind all of the low-power state assignment techniques is to assign minimum Hamming distance codes to the states pairs that have large interstate transition probabilities. For example, the coding $S_0 = 000$, $S_1 = 001$, $S_2 = 011$, $S_3 = 010$, $S_4 = 100$, $S_5 = 101$, $S_6 = 111$, and $S_7 = 110$ fulfills this requirement.

In [28], Wu *et al.* proposed the idea of realising a low-power FSM by using T flip-flops. The authors showed that the use of T flip-flops results in a natural clock gating and may result in reduced next-state logic complexity. However, that work was mostly focused on BCD counters, which have cyclic behaviour. The cyclic behaviour of counters results in a significant reduction of combinational logic complexity and, hence, lowers power consumption. Reference [29] introduces a mathematical framework for cycle representation of Markov processes and, based on that, proposes

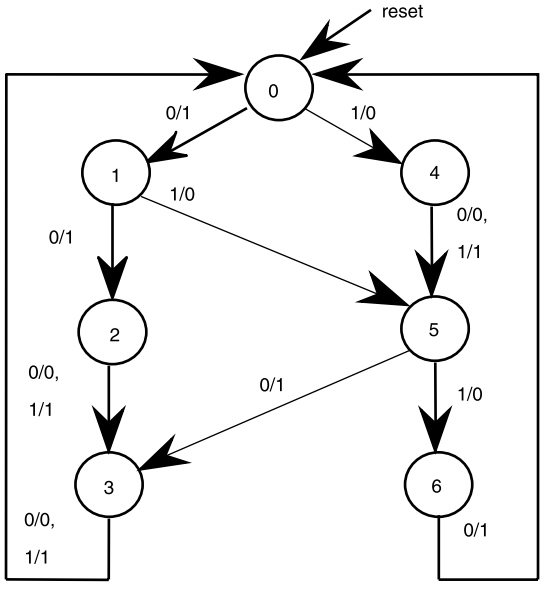

Fig. 12 *Excess-3 converter state transition graph*

solutions to the low-power state assignment problem. The authors first identify the most probable cycles in the FSM and encode the states on these cycles with Gray codes. The objective function is to minimise the weighted Hamming distance. This reference also teaches how a combination of T and D flip-flops as state registers can be used to achieve a low-power realisation of a FSM.

4.2 Retiming

Retiming is to reposition the registers in a design to improve the area and performance of the circuit without modifying its input–output behaviour. The technique was initially proposed by Leiserson and Saxe [30]. This technique changes the location of registers in the design in order to achieve one of the following goals: (1) minimise the clock period (2) minimise the number of registers (3) minimise the number of registers for a target clock period.

Minimising dynamic power for synchronous sequential digital designs is addressed in the literature. In [31], Monteiro *et al.* presented heuristics to minimise the switching activity in a pipelined sequential circuit. Their approach is based on the fact that registers have to be positioned on the output edges of the computational elements that have high switching activity. The reason for power savings is that in this case the output of a register switches only at the arrival of the clock signal as opposed to potentially switching many times in the clock period. Consider the simple example of a logic gate belonging to a synchronous circuit and a capacitive load driven by the output gate. In CMOS technology, the power dissipated by the gate is proportional to the product of the switching activity of the output node of the gate and the output load. At the output of the gate some spurious transitions (i.e. glitches) may occur, which can result in a significant power waste. Suppose a register is inserted between the output of the gate and the capacitive load. In the new circuit, the output of the register can make, at most, one transition per clock cycle. In fact, the gate output may have many redundant transitions, but they are all filtered out by the register. Hence, these logic hazards do not propagate to the output load.

The heuristic retiming technique of [31] applies to a synchronous network with pipeline structure. The basic idea is to select a set of candidate gates in the circuit such that if registers are placed at their outputs, the total switching activity of the network gets minimised. The selection of the gates is driven by two factors; the amount of glitching that occurs at the output of each gate and the probability that such glitching propagates to the gates located in the transitive fanout. Registers are initially placed at the primary inputs of the circuit, and backward retiming (which consists of moving one register from all gate inputs to the output) is applied until all the candidate gates have received a register on their outputs. Then, registers that belong to paths not containing any of the candidate gates are repositioned, with the objective of minimising both the delay and the total number of registers in the circuit. This last retiming phase does not affect the registers that have been already placed at the outputs of the previously selected gates. In [32], fixed-phase retiming is proposed to reduce dynamic power consumption. The edge-triggered circuit is first transformed to a two-phase level-clocked circuit, by replacing each edge-triggered flip-flop by two latches. Using the resulting level-clocked circuit, the latches of one phase are kept fixed, while the latches belonging to the other phase are moved onto wires with high switching activity and loading capacitance.

Fixed-phase retiming is best illustrated by the example shown below. Figure 13*a* shows a section of a pipelined circuit with edge-triggered flip-flops. The numbers on the edges represent the potential reduction in power dissipation when an edge-triggered flip-flop is present on that edge, assuming that the rest of the circuit remains unchanged. Negative values of power reduction indicate an increase in power dissipation when a flip-flop is placed on an edge. This reduction in power dissipation can be achieved if the edge has a high glitching-capacitance product [3]. After replacing each edge-triggered flip-flop by two back-to-back level-clocked latches, the resulting circuit is fixed-phase retimed to obtain the circuit in Fig. 13*b*.

Assuming a non-overlapping two-phase clocking scheme $\pi = \langle \phi_0 = 4,\ \gamma_0 = 1,\ \phi_1 = 4,\ \gamma_1 = 1 \rangle$ such as the one shown in Fig. 13*c*, power dissipation can be reduced by 11.8 units. Specifically, the glitching on edges B $\rightarrow$ D, E $\rightarrow$ F and E $\rightarrow$ H is 'masked' for 60% of the clock cycle which decreases power dissipation by $0.6 \times (12 + 13 - 2) = 13.8$ units of power. At the same time, the glitching on edges G $\rightarrow$ J and H $\rightarrow$ K is 'exposed' for 40% of the clock cycle which increases power dissipation by $0.4 \times (10 - 5) = 2$ power units. In order to simplify the computation of changes in power dissipation for this example, it is assumed that glitching is uniformly distributed over the entire clock period and that the relocation of latches does not change glitching significantly.

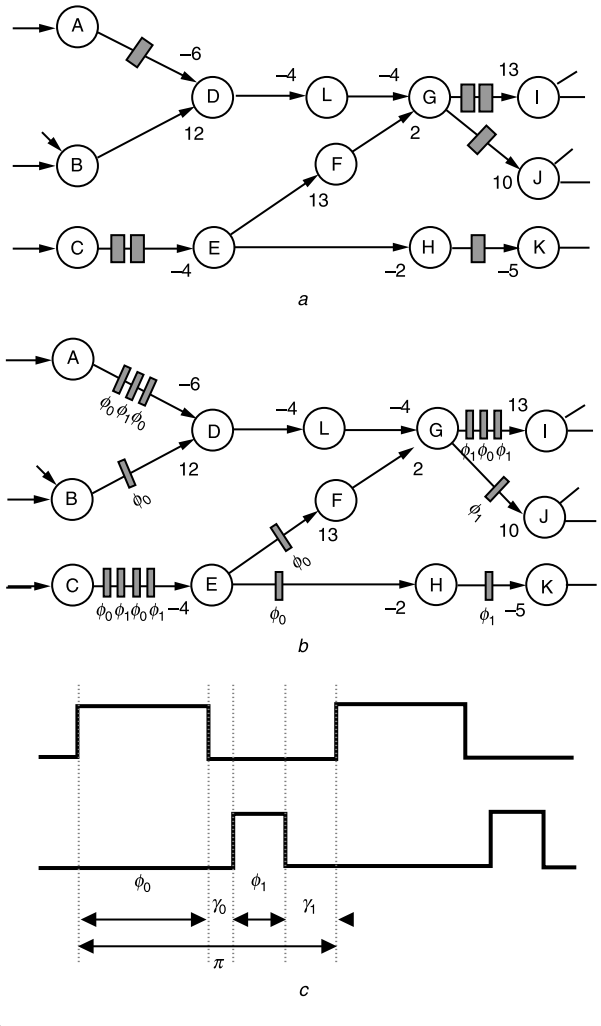

Fig. 13 *Illustration of fixed-phase retiming*

a Initial edge-triggered circuit
b Fixed-phase retimed circuit
c Two-phase clocking scheme $\pi = \langle \phi_0 = 4,\ \gamma_0 = 1,\ \phi_1 = 4,\ \gamma_1 = 1 \rangle$

In [33], the authors propose a hybrid retiming and supply voltage scaling. They observe that critical paths are related to the position of registers in a design so they try not only to scale down the supply voltage of computational elements that are off the critical paths, but also to move registers to maximise the number of computational elements that are off the critical paths, thereby further minimising the circuit power consumption. Registers have to be moved from their positions by the standard retiming technique. Instead of unifying basic retiming and supply voltages scaling, the authors propose to apply 'guided retiming' followed by the application of voltage scaling on the retimed design. Polynomial time algorithms based on dynamic programming to realise the guided retiming, as well as the supply voltage scaling on the retimed design, are proposed.

5 Bus encoding for low power

A lot of power is consumed in the on-chip and off-chip buses in a VLSI circuit. These buses, which connect various internal blocks of the circuit or connect the circuit to the external environment, have large capacitive loads and high transition counts. Power on these buses can be reduced by properly coding the data and/or address bus values so as to minimise the number of transitions that occur on the bus.

Musoll, *et al.* proposed the working zone method in [34]. Their method takes advantage of the fact that data accesses tend to remain in a small set of working zones. For the addresses that lie in each of these zones, a relatively high degree of locality is observed. Each working zone requires a dedicated register called zone register that is used to keep track of the accesses in that zone. When a new address arrives, the offset of the address is calculated with respect to all zone registers. The address is, thus, mapped to the working zone with the smallest offset. If the offset is sufficiently small, one-shot encoding is performed and the result is sent on the bus using transition signalled (by transition signalled we mean that instead of sending the code itself we XOR it with the previous value of the bus). Otherwise, the address itself is sent over the bus. The working zone method uses one extra line to show whether encoding has been done or the original value has been sent. It also uses additional lines to identify the working zone that was used to compute the offset. Based on this information, the decoder on the other side of the bus can uniquely decode the address.

The working zone method also has the ability to detect a stride in any of the working zones. A stride is a constant offset that occurs between multiple consecutive addresses repeatedly and, if detected, can be used to completely eliminate the switching activity for such addresses. For instruction addresses, stride corresponds to the offset of sequential instructions. Stride is very important when instruction address encoding is tackled. In fact, the large number of sequential instructions with constant stride is the foundation of considerable transition savings that is usually seen in instruction address encoding techniques. For data addresses, stride can occur when, for example, a program is accessing elements of an array in the memory. Except for special cases, detecting and utilising strides has a very small impact on decreasing the switching activity of data addresses.

Another encoding method that can be used for data addresses is the bus-invert method [35]. The bus-invert selects between the original and the inverted pattern in a way that minimises the switching activity on the bus. The resulting patterns together with an extra bit (to notify whether the address or its complement has been sent) are

Table 1: Example showing the T0, BI and T0–BI codes

Address (Hex)	Source word	T0 Code word	BI Code word	T0–BI Code word
31	0011 0001	0 0011 0001	0 0011 0001	00 0011 0001
32	0011 0010	1 0011 0001	0 0011 0010	10 0011 0001
33	0011 0011	1 0011 0001	0 0011 0011	10 0011 0001
C2	1100 0010	0 1100 0010	1 0011 1101	01 0011 1101
C3	1100 0011	1 1100 0010	1 0011 1100	11 0011 1101
C4	1100 0100	1 1100 0010	1 0011 1011	11 0011 1101
C2	1100 0010	0 1100 0010	1 0011 1101	01 0011 1101
C3	1100 0011	1 1100 0010	1 0011 1100	11 0011 1101
C4	1100 0100	1 1100 0010	1 0011 1011	11 0011 1101
	Tr. Cnt = 19	Tr. Cnt = 11	Tr. Cnt = 16	Tr. Cnt = 9

transition signalled over the bus (cf. Table 1, column 4.) This technique is quite effective for reducing the number of 1's in addresses with random behaviour, but it is ineffective when addresses exhibit some degree of locality. To make the bus-invert method more effective, the bus can be partitioned into a handful of bit-level groups and a bus-invert can be separately applied to each of these groups. However, this scheme will increase the number of surplus bits required for the encoding, which is absolutely undesirable.

In [36], Benini *et al.* proposed the T0 code, which exploits data sequentiality to reduce the switching activity on the address bus. The observation is that addresses are sequential, except when control flow instructions are encountered or exceptions occur. T0 adds a redundant bus line, called INC. If the addresses are sequential, the sender freezes the value on the bus and sets the INC line. Otherwise, INC is de-asserted and the original address is sent (cf. Table 1, column 3.) Several methods that are combinations of the bus-invert and T0 encodings were proposed in [37]. For instance, one of the introduced methods called T0–BI, adds two redundant bits, named INV and INC to the bus. If the addresses are sequential, T0 encoding is applied and the bus is frozen; otherwise, the new address, which is not sequential, is encoded based on the bus-invert coding. INC and INV bits are used to correctly decode the bus value on the receiver side (cf. Table 1, column 5).

The major drawback of the encoding methods introduced in this work is that they introduce redundant bits. In T0 code, one extra bit was used to identify between these two cases in the receiver. Aghaghiri *et al.* [38] improved on this technique by eliminating the redundant bit in T0-concise. The idea is to send previous source plus stride if the bus value is equal to the current nonsequential source word. This is the only thing that the receiver does not expect, therefore, it can correctly decode it as a jump back to the address that was frozen on the bus at the beginning of the current sequential access.

Reference [39] proposes a low-power coding framework for address and data buses. They describe the general architecture of a low power encoder (cf. Fig. 14.) In this Figure, choices for function F1 include identity or increment transformations, choices for F2 include XOR operation, subtraction or difference-based mapping, and choices for F3 are inversion-or probability-based mapping. For example, the INC–XOR encoder, also known as T0–XOR, generates the new bus value as the XOR of the previous bus value and the new code word (known as the transition signalled over the bus). The new code word is in turn obtained as the XOR of the new source word and the summation of the previous source word and the stride value. Obviously, when consecutive addresses grow by the stride, no transitions will occur on the bus. The offset–XOR encoder also relied on transition signalled. However, the new code word is obtained as the new source word minus the summation of the previous source word and the stride value. In [40], Aghaghiri *et al.* presented offset–XOR–SM encoding, whereby the new code word is again transition signalled over the bus. The new code word itself is generated as LSB–invert of the offset–XOR code word followed by a codebook-based mapping. The LSB-invert function is a simple mapping function that reduces the number of 1's in the binary representation of small negative numbers $(\mathrm{LSB\text{–}INV}(X) = \mathrm{if}(\mathrm{X} > 0) \quad \mathrm{X};$ otherwise $\mathrm{X} \oplus (2^{N-1} - 1)$. The codebook maps small offsets (say up to 10 bits) to K-limited codes in order to reduce the number of 1's in the new code word (recall that a 1 in the code word translates to a bit-level activity after transition signalled the code word over the bus).

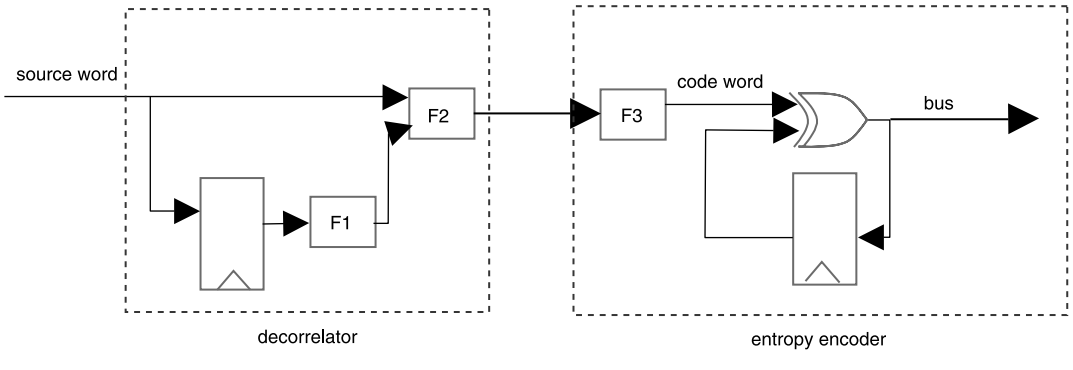

Fig. 14 *Block diagram of a generic low-power encoder*

In [41], Mamidipaka *et al.* proposed an encoding technique based on the notion of self-organising lists. They use list to create one-to-one mapping between addresses and codes. The list is reorganised in every clock cycle to map the most frequently used addresses to codes with fewer 1's. For multiplexed address buses, the authors used a combination of their method and INC–XOR. The size of the list in this method has a significant impact on the performance. To achieve satisfactory results, it is necessary to use a long list. However, the large hardware overhead associated with maintaining long lists makes this technique quite expensive. Furthermore, the encoder and the decoder hardware are practically complex and their power consumption appears to be quite large.

In [42], the authors introduced a class of irredundant low-power techniques for encoding instruction or data source words before they are transmitted over buses. The key idea is to partition the source word space into a number of sectors with unique identifiers called *sector heads* (SH). These sectors can, for example, correspond to address spaces for the code, heap and stack segments of one or more application programs. Each source word is then mapped to the appropriate sector and is encoded with respect to the sector head. Suppose X is an N-bit source word to be encoded. There are 2^k fixed sectors with 2^k sector heads, $\mathrm{SH}[0] \ldots \mathrm{SH}[2^k - 1]$. The code word is comprised of k most significant Sec–ID bits used for identifying the sector, and $N - k$ least significant difference bits representing the XOR difference between the source word and the corresponding sector head. The encoder takes a source word $X: (X_N \ldots X_1)$, and assigns it to the corresponding sector by examining its Sec–ID bits. Next, it sets the $N - k$ LSBs of the code word to the XOR difference between the $N - k$ LSBs of the source word and the corresponding bits of the SH for the identified sector. The SH of the identified sector is set to X. Finally, the code word is transition signalled over the bus. As an example, consider a five-bit space with four sector heads initialised at equal distances from each other, i.e. {00000,01000,10000,11000}. Table 2 shows the results of the fixed four-sector encoder.

Note that the sectors are fixed, but the sector heads are dynamically updated. In a generalisation of this approach the sectors can dynamically be defined based on program behaviour. This feature is very useful because the source word space is very large while the total working zone of a program is usually small. Therefore, it pays off to have dynamically defined sectors which can 'zoom into' the working zone of a program. The sector-based encoding techniques are quite effective in reducing the number of inter-pattern transitions on the bus, while incurring rather small power and delay overheads.

In [43], the authors provide a modified bus-invert (MBI) technique which, besides reducing delay and power, also minimises the crosstalk noise that results from inductive coupling between the bus lines. Their proposed approach is based on the observation that opposite skews can reduce the crosstalk noise. Therefore, the authors propose to minimise the number of transitions that are in the same direction by selectively inverting the data patterns. The method requires an extra line which carries the 'invert signal' and is used by the decoder in order to restore the original data. In the encoder the bus lines are partitioned into pairs and each pair of adjacent lines, as well as their values from the previous clock cycle, drive the inputs of a logic cell, which encodes the types of events that occur on the pair of bus lines. This cell generates 11 if the transitions occur in the same direction, 00 if both lines are idle, 01 if either only one line switches or both lines switch in opposite directions. A majority voter takes the outputs of the logic cells and the previous invert signal and sets the invert signal to 0 when the count of 1's is less than n and to 1 otherwise.

Table 2: Example showing a fixed four-sector encoder

X	SH[0]	SH[1]	SH[2]	SH[3]	Code(X)
01111	00 000	01 000	10 000	11 000	01111
00010	00 000	01 111	10 000	11 000	00010
00011	00 010	01 111	10 000	11 000	00001
01110	00 011	01 111	10 000	11 000	01001
10001	00 011	01 110	10 000	11 000	10001
00011	00 011	01 110	10 001	11 000	00000
01100	00 011	01 110	10 001	11 000	01010
Tr. Cnt = 12	00 011	01 100	10 001	11 000	Tr. Cnt = 8

6 Conclusions

Several key elements emerge as enablers for an effective low-power design methodology. The first is the availability of accurate, comprehensive power models. The second is the existence of fast, easy to use high level estimation and design exploration tools for analysis and optimisation during the design creation process, while the third is the existence of highly accurate, high-capacity verification tools for tape-out power verification. As befitting a first-order concern, successfully managing the various power-related design issues will require that power be addressed at all phases and in all aspects of design, especially during the earliest design and planning activities. Advanced power tools will play central roles in these efforts.

This paper reviewed a number of techniques for low-power design of VLSI circuits, including RT-level synthesis, bus encoding and voltage scaling. Emphasis was placed on runtime power management techniques and sequential circuit synthesis. A review of techniques for low power design of combinational logic circuits can be found in many references, including [44, 45].

7 References

1 Pedram, M., and Rabaey, J. (Eds.): 'Power aware design methodologies' (Kluwer Academic Publishers, Boston, 2002)
2 Macii, E. (Ed.): 'Ultra low-power electronics and design' (Kluwer Academic Publishers, Boston, 2004)
3 Piguet, C. (Ed.): 'Low power electronics design' (CRC Press, 2004)
4 Hamada, M., Takahashi, M., Arakida, H., Chiba, A. Terazawa, T., Ishikawa, T., Kanazawa, M., Igarashi, M., Usami, K., and Kuroda, T.: 'A top-down low power design technique using clustered voltage scaling with variable supply-voltage scheme'. Proc. IEEE Custom Integrated Circuits Conf. (CICC'98), May 1998, pp. 495–498
5 Raje, S., and Sarrafzadeh, M.: 'Variable voltage scheduling'. Proc. Int. Workshop on Low Power Design, Aug. 1995, pp. 9–14
6 Usami, K., and Horowitz, M.: 'Clustered voltage scaling technique for low-power design'. Proc. Int. Workshop on Low Power Design, 1995, pp. 3–8
7 Usami, K., Igarashi, M., Minami, F., Ishikawa, T., Kanzawa, M., Ichida, M., and Nogami, K.: 'Automated low-power technique exploiting multiple supply voltages applied to a media processor', *IEEE J. Solid-State Circuits*, 1998, **33**, (3), pp. 463–472
8 Chen, C., Srivastava, A., and Sarrafzadeh, M.: 'On gate level power optimization using dual supply voltages', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2001, **9**, pp. 616–629
9 Manzak, A., and Chakrabarti, C.: 'A low power scheduling scheme with resources operating at multiple voltages', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2002, **10**, (1), pp. 6–14
10 Chang, J.M., and Pedram, M.: 'Energy minimization using multiple supply voltages', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1997, **5**, (4), pp. 436–443
11 Yeh, Y.-J., Kuo, S.-Y., and Jou, J.-Y.: 'Converter-free multiple-voltage scaling techniques for low-power CMOS digital design', *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 2001, **20**, pp. 172–176

12 Murugavel, A.K., and Ranganathan, N.: 'Game theoretic modeling of voltage and frequency scaling during behavioral synthesis'. Proc. VLSI Design, 2004, pp. 670–673
13 Alidina, M., Monteiro, J., Devadas, S., Ghosh, A., and Papaefthymiou, M.: 'Precomputation-based sequential logic optimization for low power', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1994, **2**, (4), pp. 426–436
14 Monteiro, J., Devadas, S., and Ghosh, A.: 'Sequential logic optimization for low power using input-disabling', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1998, **17**, (3), pp. 279–284
15 Benini, L., Siegel, P., and De Micheli, G.: 'Automatic synthesis of gated clocks for power reduction in sequential circuits', *IEEE Des. Test Comp.*, 1994, **11**, (4), pp. 32–40
16 Benini, L., and De Micheli, G.: 'Transformation and synthesis of FSM's for low power gated clock implementation', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1996, **15**, (6), pp. 630–643
17 Benini, L., De Micheli, G., Macii, E., Poncino, M., and Scarsi, R.: 'Symbolic synthesis of clock-gating logic for power optimization of control-oriented synchronous networks'. Proc. European Design and Test Conf., Paris, France, Mar. 1997, pp. 514–520
18 Benini, L., Favalli, M., and De Micheli, G.: 'Design for testability of gated-clock FSM's'. Proc. European Design and Test Conf., Paris, France, Mar. 1996, pp. 589–596
19 Benini, L., De Micheli, G., Lioy, A., Macii, E., Odasso, G., and Poncino, M.: 'Synthesis of power-managed sequential components based on computational kernel extraction', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2001, **20**, (9), pp. 1118–1131
20 Benini, L., De Micheli, G., Macii, E., Odasso, G., and Poncino, M.: 'Kernel-based power optimization of RTL components: exact and approximate extraction algorithms'. Proc. Design Automation Conf., 1999, pp. 247–252
21 Monteiro, J., and Oliveira, A.: 'Finite state machine decomposition for low power'. Proc. Design Automation Conf., June 1998, pp. 758–763
22 Chow, S-H., Ho, Y-C., and Hwang, T.: 'Low power realization of finite state machines a decomposition approach', *ACM Trans. Des. Autom. Electron. Syst.*, 1996, **1**, (3), pp. 315–340
23 Benini, L., Siegel, P., and De Micheli, G.: 'Automatic synthesis of low-power gated-clock finite-state machines', *IEEE Trans. Comput -Aided Des. Integr. Circuits Syst.*, 1996, **15**, (6), pp. 630–643
24 Tiwari, V., Malik, S., and Ashar, P.: 'Guarded evaluation: pushing power management to logic synthesis/design'. Proc. ACM/IEEE Int. Symp. on Low Power Design, Dana Point, CA, Apr. 1995, pp. 221–226
25 Roy, K., and Prasad, S.: 'Syclop: synthesis of CMOS logic for low-power application'. Proc. Int. Conf. on Computer Design, Oct. 1992, pp. 464–467
26 Olson, E., and Kang, S.M.: 'Low-power state assignment for finite state machines'. Proc. Int. Workshop on Low Power Design, April 1994, pp. 63–68
27 Tsui, C.Y., Pedram, M., and Despain, A.M.: 'Low-power state assignment targeting two- and multilevel logic implementation', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 1998, **17**, (12), pp. 1281–1291
28 Wu, X., Wei, J., Wu, Q., and Pedram, M.: 'Low-power design of sequential circuits using a quasi-synchronous derived clock', *Int. J. Electron.*, 2001, **88**, (6), pp. 635–643
29 Iranli, A., Rezvani, P., and Pedram, M.: 'Low power synthesis of finite state machines with mixed D and T flip-flops'. Proc. Asia and South Pacific Design Automation Conf., Jan. 2003, pp. 803–808
30 Leiserson, C.E., and Saxe, J.B.: 'Optimizing synchronous systems', *J. VLSI Comput. Syst.*, 1983, **1**, (1), pp. 41–67
31 Monteiro, J., Devadas, S., and Ghosh, A.: 'Retiming sequential circuits for low power'. Proc. Int. Conf. on Computer-Aided Design, Santa Clara, CA, Nov. 1993, pp. 398–402
32 Lalgudi, K.N., and Papaefthymiou, M.: 'Fixed-phase retiming for low power'. Proc. Int. Symp on Low-Power Electronics and Design, 1996, pp. 259–264
33 Chabini, N., and Wolf, W.: 'Reducing dynamic power consumption in synchronous sequential digital designs using retiming and supply voltage scaling', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2004, **12**, (6), pp. 573–589
34 Musoll, E., Lang, T., and Cortadella, J.: 'Exploiting the locality of memory references to reduce the address bus energy'. Proc. Int. Symp. on Low Power Electronics and Design, 1997, pp. 202–207
35 Stan, M.R., and Burleson, W.P.: 'Bus-invert coding for low power I/O', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1995, **3**, (1), pp. 49–58
36 Benini, L., De Micheli, G., Macii, E., Sciuto, D., and Silvano, C.: 'Asymptotic zero-transition activity encoding for address buses in low-power microprocessor-based systems'. Proc. 7th Great Lakes Symp. on VLSI, Mar. 1997, pp. 77–82
37 Benini, L., De Micheli, G., Macii, E., Sciuto, D., and Silvano, C.: 'Address bus encoding techniques for system-level power optimization'. Proc. Design Automation and Test in Europe, 1998, pp. 861–866
38 Aghaghiri, Y., Fallah, F., and Pedram, M.: 'A class of irredundant encoding techniques for reducing bus power', *Circuits Syst. Comput.*, 2002, **11**, (5), pp. 445–457
39 Ramprasad, S., Shanbhag, N., and Hajj, I.: 'A coding framework for low power address and data buses', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 1999, **7**, (2), pp. 212–221
40 Aghaghiri, Y., Fallah, F., and Pedram, M.: 'Irredundant address bus encoding for low power'. Proc. Symp. on Low Power Electronics and Design, Aug. 2001, pp. 182–187
41 Mamidipaka, M., Hirschberg, D., and Dutt, N.: 'Low power address encoding using self-organizing lists'. Proc. Int. Symp. on Low Power Electronics and Design, 2001, pp. 188–193
42 Aghaghiri, Y., Fallah, F., and Pedram, M.: 'Transition reduction in memory buses using sector-based encoding techniques', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2004, **23**, (8), pp. 1164–1174
43 Lampropoulos, M., Al-Hashimi, B.M., and Rosinger, P.M.: 'Minimization of crosstalk noise, delay and power using a modified bus invert technique'. Proc. Design Automation and Test in Europe, 2004, pp. 1372–1373
44 Rabaey, J., and Pedram, M. (Eds.): 'Low power design methodologies' (Kluwer Academic Publishers, Boston, Oct. 1995)
45 Pedram, M.: 'Power minimization in IC design: principles and applications', *ACM Trans. Des. Autom. Electron. Syst.*, 1996, **1**, (1), pp. 3–56