Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Power-gating-aware scheduling with effective hardware resources optimization

Nan Wang[a,*], Wei Zhong[b], Song Chen[b], Zhiyuan Ma[c], Xiaofeng Ling[a], Yu Zhu[a]

[a] School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
[b] Department of Electronic Science and Technology, University of Science and Technology of China, Hefei 230027, China
[c] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China

A R T I C L E I N F O

Keywords:
Scheduling
Power-gating
Retention register
Hardware resources optimization

A B S T R A C T

Power-gating technology has been widely used to reduce subthreshold leakage power. However, the efficiency of power-gating degrades significantly with the wide use of retention registers. The scheduling algorithms proposed in this work optimize the number of required retention registers to minimize the leakage power when the supply voltage of data-path is cut off; in addition, the hardware resources, including functional units, registers and interconnections, undergo overall optimization to enhance the quality of the scheduling results. For each possible scheduling result of an operation, the life times of related values are analyzed together to provide a tight lower bound of active values in each control step, the maximum of which equals the number of required registers. A max-cost flow-based algorithm is also implemented on a network derived from the current mobilities, and the total cost over this network evaluates the interconnection benefits of the current schedule. With an overall analysis of hardware resource usages, the operation scheduling order is finally determined by iteratively executing a maximum weight independent set-based method. The experimental results show the proposed algorithms optimize both the leakage power and the area of the circuit by effectively reducing the hardware resource usages.

1. Introduction

Power consumption has been an important consideration in chip designs since the 1990s, and dynamic power was optimized from the beginning. With the development of nanoscale technology, leakage power has come to dominate the total power consumption, especially in 65 nm technology or below, and chip designers have shifted their focus to leakage power optimization.

Many technologies have been developed to achieve low leakage power circuit designs, among which power-gating is especially popular and has been widely used in the semiconductor industry [1–6]. It cuts off the current to the blocks of circuit that are not in use to reduce the leakage power, and the states of the circuit block must be stored before cutting off the supply voltage. Retention registers are special low-leakage flip-flops used to hold the data on the normal registers of the power-gated block, and the state of the block can be retained and then reloaded when this block is woken up.

There are various studies [7–10] on the implementation of retention registers, which invariably incur a substantial amount of overhead in terms of interconnections and leakage power. The study in [11] reports that a retention flip-flop requires 68% additional area than a conventional flip-flop, and the overhead area in power-gated circuits can range from 13% to 28%. Furthermore, the total wirelength of power-gated circuits typically increases by 29–60% due to the additional wires for retention flip-flops and extra control signals [11]. A retention flip-flop is usually fully biased during standby mode to preserve the state, resulting in a continuous gate leakage current, which contributes significantly to the total leakage current, and this situation has deteriorated rapidly with the scaling of CMOS technology [12].

Consequently, reducing the use of retention storage is an important issue in optimizing power-gated circuits, and researchers try to optimize the leakage power of a power-gated circuit by minimizing the number of retention registers during scheduling because the most power-saving opportunities exist at the highest level of design abstraction [13]. A previous study [14] minimizes the number of required retention registers during scheduling by transferring the optimization problem into a set of integrated linear programming (ILP) formulations. However, this method cannot be applied to most large benchmarks because solving ILP formulations is quite time consuming.

* Corresponding author.
E-mail address: wangnan@ecust.edu.cn (N. Wang).

https://doi.org/10.1016/j.vlsi.2017.12.005
Received 7 November 2016; Received in revised form 19 October 2017; Accepted 11 December 2017

Scheduling also decides the number of functional units (FUs), and there are a number of studies [16,17] that optimize the number of FUs during scheduling. In addition, the scheduling result also determines the registers and interconnections of circuits, which are substantial in recent complex designs and must be considered at the very beginning of high-level synthesis. A recent study in [18] indicates that the registers contribute 36% to the total switching power, and multiplexers contribute 17% in their benchmarks. Furthermore, registers and multiplexers contribute 30% and 26% to the total leakage power, respectively. The study in [19] successfully optimizes the number of registers during scheduling by formulating a set of ILP formulations aiming to minimize the total life times of all values. However, this method does not consider the optimization of retention registers and other data-path components, which limits its effectiveness.

Unlike other studies that decouple the optimizations of FUs, registers and interconnections at different stages of high-level synthesis, this work tries to optimize these data-path components of an application specific integrated circuit (ASIC) during scheduling because scheduling has an overall effect on the final circuit. To analyze the register usages during scheduling, the life time of each value is divided into two types: *certain life time* and *uncertain life time*; the uncertain life times of related values are calculated together, and the sum of certain life times and uncertain life times provides a tight lower bound of active values in each control step, the maximum of which equals the number of total registers. Accurately calculate the number of multiplexer (MUX) ports during scheduling is impossible, and thus, we evaluate the interconnection benefits instead. The interconnection benefit between each pair of operations that can be bound to the same FU is first evaluated, and a max-cost flow-based algorithm, which evaluates the total effects of the interconnections, is then conducted on a network transferred from the compatibility graph. With an overall analysis of hardware resource usages, the operation scheduling order is finally determined by iteratively executing a maximum weight independent set-based method on a scheduling violation graph. The experimental results show the effectiveness of our proposed method in optimizing the retention registers along with other hardware resources during scheduling.

The rest of this paper is organized as follows. Section 2 provides the preliminary information and describes the optimization problem. Section 3 analyzes the register usages and the interconnection cost, and Section 4 presents two maximum weight independent set-based scheduling algorithms. Section 5 shows the experimental results, and Section 6 concludes the paper.

2. Preliminaries and problem definition

The target architecture of this work consists of a data-path that has FUs, registers, interconnections, and a power management unit (PMU) that initiates state changes (from active to standby and vice versa) through a *sleep* signal, as well as resetting the circuit (see Fig. 1). The

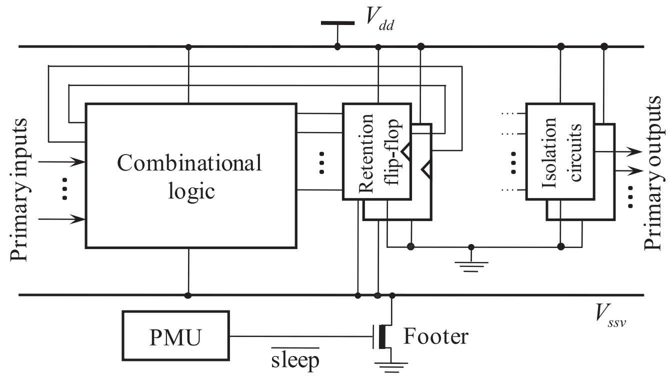

Fig. 1. Example of power-gating circuit.

controller, which is a finite state machine (FSM), is assumed to receive an asynchronous *sleep* signal from PMU, and subsequently generates a *standby* signal to power-gate the data-path, and *ret* signal to enforce the state preservation in the retention register. To clarify the high-level synthesis problem that we are addressing, we assume that:

1. Power-gating is applied to the entire data-path rather than partial of a data-path.
2. A k-bit retention register has k latches to hold k-bit data in the standby mode.
3. Among all control steps within the timing constraint, only one of them involves the actual power gating, and we call this the power-gating control step c_{pg}. If several power-gating control steps may be required, our scheduler can easily be extended to support.

2.1. Preliminaries and motivations

The algorithm description is used as the input of the scheduling algorithm, and it is usually followed by a translation to a graph derived from the data flow (data flow graph, DFG). The DFG is denoted as $G = (V, E)$, where $V = \{v_i | 1 \le i \le n\}$ represents the set of operations, and E represents the set of data dependencies between operations.

Register Requirement: Each operation computes a *value* that must be stored in a register and later used by other operations. The values with non-overlapping lifetimes can share the same register, and maximizing the sharing of registers by values generates the minimum number of registers. Two types of registers are required in a power-gating circuit: normal registers that store the values when the circuit is active and they will be powered-off when the data-path is power-gated; retention registers that retain values in the standby state and can also store values when the circuit is in the active state.

The number of total registers, including normal registers and retention registers, is denoted as R_{tot} and calculated as follows.

$$R_{tot} = max\{|S(c_j)|, \ \forall c_j \in [1, t_c]\} \tag{1}$$

where $|S(c_j)|$ is the number of alive values in control step c_j, and t_c is the given timing constraint. The number of retention registers R_{rr} equals the number of values that are alive in the power-gating control step c_{pg}, which is given as follows.

$$R_{rr} = |S(c_{pg})| \tag{2}$$

However, c_{pg} is a parameter to be determined rather than a fixed one in some scenarios. Consequently, R_{rr} is also described as follows.

$$R_{rr} = min\{|S(c_j)|, \ \forall c_j \in [1, t_c]\} \tag{3}$$

The register requirements vary with different scheduling results, as shown by the example in Fig. 2, where t_c is 5 control steps, and ALU and multiplier need one and two control steps to compute an operation, respectively. Although the numbers of FUs required by these two scheduling results are the same, the scheduling result in Fig. 2(b) needs four registers to save the values, while the scheduling result in Fig. 2(c) needs five registers. Thus, an extra register would be required if we did not consider it during scheduling.

Number of MUX Ports: The number of MUX input ports is used to evaluate the interconnections of circuits in this work, and the exact number of MUX input ports can only be decided after FU binding and register binding. However, the scheduling result also plays an important role in determining the interconnections.

An example in Fig. 3 shows the motivation to consider the MUX ports when scheduling operations, in which operations are bound to the ALUs with the same color. Given the first scheduling result shown in Fig. 3(a), the best binding result is given in Fig. 3(b) with 7 MUX ports. The second scheduling result is presented in Fig. 3(c), and its binding result, shown in Fig. 3(d), only requires 6 MUX ports.

The above examples show the effects on the data-path of different scheduling results, demonstrating the urgent need to optimize these