INTEGRATION, the VLSI journal 46 (2013) 119–130

Contents lists available at SciVerse ScienceDirect

INTEGRATION, the VLSI journal

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

A fragmentation aware High-Level Synthesis flow for low power heterogenous datapaths

Alberto A. Del Barrio [a,*], Seda Ogrenci Memik [b], María C. Molina [a], José M. Mendías [a], Román Hermida [a]

[a] Department of Computer Architecture and Automation, Computer Science Faculty, Complutense University of Madrid, José Santesmases s\n, 28040 Madrid, Spain
[b] Department of EECS, Tech. Building, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208, United States

ARTICLE INFO

Article history:
Received 30 May 2011
Received in revised form
3 February 2012
Accepted 20 February 2012
Available online 28 February 2012

Keywords:
Low power
Area
High-Level Synthesis

ABSTRACT

State of the art multi-objective synthesis flows use to degrade some parameters of the circuit while trying to optimize the target one. This paper addresses the power reduction problem in heterogeneous datapaths, while keeping a similar area and execution time with respect to the baseline case. Our specific approach first diminishes the area via fragmentation techniques and afterwards it gives it back with the introduction of Low Power Functional Units (LP-FUs) that occupy more area than their corresponding non-low power counterparts. Furthermore, a fragmentation algorithm more suitable for power reduction is proposed. Results show that it is possible to diminish power by 27% on average (49% in the best case).

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

The power dissipated by a circuit can be optimized at different levels of abstraction. However, the potential impact of strategic decisions made at the higher levels is likely to be most significant [1,2]. High-Level Synthesis (HLS) techniques have therefore tackled power minimization in various ways. Majority of these techniques focuses on the dynamic power consumption. Although dynamic power still dominates the total power envelope, leakage power is becoming an increasingly larger fraction of total power. Leakage optimizations are generally addressed through lower level design optimizations such as dual threshold gates or reverse body biasing [3,4]. HLS also has some indirect impact on power, for example by minimizing the total amount of resources employed in a datapath. Dynamic power, on the other hand, is a function of switching activity, supply voltage level, and the switched capacitive load. Various HLS techniques address these parameters for optimization. However, those focused on reducing voltage [19] or frequency [20], require significant changes in the design process and they impact technology parameters. Alternatively, other techniques only aim to minimize switching activity or the effective amount of resources required without the need to impose any limitations on circuit and technology parameters. For instance, binding operations with correlated switching activity on the same resource in consecutive cycles diminishes switching activity [21]. However, this may not be sufficient for modules composed of a large amount of logic, e.g. a combinational multiplier. Internal signals within such a complex component can behave differently depending on the specific implementation, even though consecutive inputs supplied to the component at the primary ports are correlated.

Similar to some aforementioned HLS techniques, fragmentation [16–18] would have an indirect impact on power; mainly thanks to the fact that the total effective amount of hardware used at a given clock cycle is reduced. Useless switching activity is produced when executing a narrow operation in a bigger FU. This is the case of heterogeneous specifications, where different sizes and data types are taken into account. Traditional HLS allocation techniques select FUs able to execute the widest operations in the specification and hence, there will be some wasted FU parts when computing narrower operations. Thereby, removing these useless parts is a way of reducing both area and power. Fragmentation techniques have been developed in order to tackle this problem. Fragmentation mainly aims to allocate a set of functional units with various width configurations that can execute operations in the specification. The goal is to minimize the total amount of hardware used and in this process some wide operations can be divided into fragments such that they can be executed on several narrower FUs. Scheduling techniques have been proposed to accompany fragmentation, where fragments of the same operation could be rescheduled in non-consecutive cycles [17,18]. As a consequence, there will not be any FU executing smaller-sized operations.

Fragmentation and associated scheduling techniques mentioned above indeed help reduce total area. However, as the

* Corresponding author.
E-mail addresses: albertodbg@fdi.ucm.es (A.A. Del Barrio), seda@eecs.northwestern.edu (S.O. Memik), cmolinap@dacya.ucm.es (M.C. Molina), mendias@dacya.ucm.es (J.M. Mendías), rhermida@dacya.ucm.es (R. Hermida).

0167-9260/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2012.02.005

techniques presented in [17,18] are highly focused on diminishing this parameter, they can produce over-fragmented datapaths. As the wiring complexity affects power consumption [27], over-fragmentation must be reduced. Moreover, in order to minimize the hardware wastage these techniques allow the transformation of products into additions, which can prevent the introduction of efficient low-power multipliers as the ones presented in [5–9].

In this paper, we propose a new flow for scheduling, allocation and module selection for heterogeneous datapaths to explicitly address power optimization instead of relying indirectly on area reduction for power improvement. Besides, to lessen the above-mentioned adverse impacts of existing fragmentation-based flows on power we propose to maintain the same scheduling after fragmenting, and not to allow the transformation of products into additions. In this way, power will be further reduced, in spite of not diminishing area so much as in previous works [17,18]. We start out by utilizing the main principle of fragmentation as a tool for minimizing the hardware utilized per cycle [17]. However, we do not allow to schedule fragments of the same operation in different csteps, as practiced by prior techniques [18]. FUs are fragmented, but operations are executed in a set of linked FUs in the csteps when they were originally scheduled [16]. In this way datapaths will be less fragmented and will keep a similar execution time to the common case, which combined with the power reduction will produce an overall energy decrease. Furthermore, the allocation algorithm is complemented by the prior application of a Fragmented-Aware Scheduling (FAS), which will schedule operations such that the allocation process can take the maximum advantage of. In other words FAS will help the allocation stage to reduce the number and size of the required resources as much as possible.

Next, we observe that the area savings achieved by fragmentation can be traded-off systematically for power reduction. Our specific approach is to introduce multiple choices for FUs with power/area trade-offs for different fragmentation and allocation styles. A module selection algorithm has been developed, which pursues a trade-off between area and power consumption for fragmented datapaths under total area constraints. Our experimental results show that it is possible to reduce power by 27% on average (49% in the best case).

The rest of the paper is organized as follows: Section 2 discusses the related work, Section 3 presents an example in order to motivate our techniques, Section 4 explains in more detail our algorithms, Section 5 describes the area and power models, and the FU library used in the module selection algorithm and finally Sections 6 and 7 present our experimental results and final conclusions.

2. Related work

Usually in datapaths multipliers are the biggest and most power consuming modules. Previous works [5–9] try to minimize power consumption produced by multipliers. Authors propose to reduce switching activity in the partial product matrix with bypassing logic in one dimension [5,7,8], two dimensions [6], or using 2's complement for some operands [9]. All these works reduce switching activity inside the multipliers for diminishing power around 20–30% at the expense of an area increase that ranges between 10 and 25%, except for one case [6] where due to the bidimensional bypassing a significant 75% power reduction is achieved, but with 125% area overhead. These approaches that sacrifice a piece of area for diminishing power are becoming widespread and also single cells [30] or complete adders present a low power version [10], reducing 55% power with 8% area penalty. Note that these LP-FUs present a similar or even lower delay than the corresponding non-LP-FUs. Therefore, achieving the same latency as in the common implementation will be enough for complying with the baseline execution time.

A similar LP approach is performed by Choi et al. in [26], obtaining 24% average power reduction at the expense of 34.5% area increase. They present some LP-FUs that perform partially guarded computation. The FUs described in [26] are divided in two halves: the Most Significant Part (MSP) and the Least Significant Part (LSP). The MSP can operate conventionally, when the non-signed parts of the operands are large enough, or otherwise simply extend the sign, thus reducing the switching activity. The location and width of these parts are determined statically with an algorithm.

Another approach [11] proposes to bind operations that share some operands to the same FU and in consecutive csteps, thus reducing switching activity. Nevertheless, as this technique can only be applied to a specific subset of operations, it is quite restricted. In [12] authors present a methodology for reducing area, power or energy, but there is always one target function at a time. It is not clear how several parameters, e.g. area and power can be combined. Remaining parameters often suffer significant degradation while trying to reach optimality in the target metric. Area overhead when reducing power or energy exceeds 50% for some cases, and 25% on average. In [13] authors tradeoff area and power by different clock selections, but they relax the timing constraint T, producing circuits with 1.5–3.5T. More recently, new techniques considering multi-objective genetic algorithms [28,29] have been presented. Genetic algorithms deal efficiently with the problem of optimizing simultaneously several parameters. However, these works do not take into consideration the possibilities that both fragmentation and LP-FUs offer.

In [14,15] authors propose to customize DSP or FPGA multipliers depending on the constants of the applications, obtaining area, power and latency reductions. This idea is more suitable for structures with abundant resources such as FPGAs and DSPs. This is not usually the case of ASICs, where datapaths must be highly optimized in order to comply with the designer constraints. Besides, this approach creates highly instruction-specific FUs. However, our objective is to use non-specific FUs, because the instruction-specific ones will diminish FU sharing and therefore it could introduce some additional modules, with the corresponding area and power penalties.

On the other hand, fragmentation algorithms have been proposed [16–18] to reduce area. These algorithms perform some transformations over the DFG in order to diminish the hardware waste due to the execution of different sized operations over the same FU. That is, if operations are fragmented it is possible to generate a DFG where the FUs are not wider than the operations that they are executing. Hence, the inclusion of LP modules in the design flow complements the application of fragmentation techniques, because LP modules introduce some area overhead in exchange for some power saving, while maintaining cycle time and latency constraints. However, the area gain produced by fragmentation should be traded-off in a systematic way [31].

Some of the aforementioned fragmentation algorithms [17,18] are especially oriented to diminish area. Nevertheless, the excessive application of fragmentation in order to take advantage of the cycle time slack, and thus reuse FUs as much as possible, can increase the wiring complexity, which is damaging from the power point of view [27]. The appearance of many small FUs is not uncommon in these works. Besides, the techniques presented in [17,18] allow the transformation of products into additions for increasing FUs reuse too, which can prevent the introduction of efficient low-power multipliers as the ones presented in literature [5–9]. Hence, executing all the fragments of every operation in the same csteps as the corresponding non-fragmented operation in

Download English Version:

https://daneshyari.com/en/article/541027

Download Persian Version:

https://daneshyari.com/article/541027

Daneshyari.com