I. INTRODUCTION

Hardware acceleration (defined here as fixed, application specific hardware blocks) has demonstrated orders of magnitude improvements in energy efficiency compared to general-purpose compute platforms [1], [4]. As power and thermal constraints continue to limit computation capabilities there has been a wide adoption of accelerators. Accelerator popularity, practical application, and efficiency potential all suggest that specialization is a viable solution to overcome performance throttling imposed by chip power limitations.

Despite all the fanaticism, related accelerator research, and proposed mechanisms there does not exists a general model which enables fundamental understandings of accelerator design trade-offs. Here we suggest analytical power and performance models to find accelerator design points that maximize energy efficiency for various applications. Through characterizing each application and their optimal design points we can definitively answer how different workloads best execute.

By constructing power and performance models for accelerators we are able to optimize designs and understand the parameters which maximize efficiency. More precisely, we found accelerators have two parameters that dominate power and performance: pipelining and parallelism. Accompanying these parameters are corresponding scaling functions. Scaling functions capture how each application’s power and performance numbers change with respect to increasing parallelism or the number of pipeline stages.

Where the increased dimension of design freedom in hardware accelerators requires more complex models it also provides a unique opportunity to examine how efficient designs are achieved. The accelerator’s lack of definition is where the majority of energy savings come from. Eliminating bloated front end CPU stages, branch predictors, and costly MUX over heads enable hardware accelerators to achieve far greater energy efficiency than general-purpose processors [1]. The accelerator’s lack of all such assumed structure cleanses analysis of any machine or ISA artifacts—enabling parallelism and pipelining to be analyzed with only application implicit characteristics.

Looking forward we seek to better understand the differences in optimums with respect to parallelism and pipelining across a variety of workloads. The abstract models we are continuing to develop enable fundamental understandings as to which program characteristics make a workload more amenable to deeper pipelines or more parallelism. We also believe this analysis will be useful in answering questions regarding heterogeneous systems in general. Leveraging the knowledge of how an application most optimally executes can translate to whether it is better suited running on a: GPU or CPU, a few small cores, or requires a less traditional solution like hardware acceleration.

II. PROBLEM STATEMENT

Our idea is that where analytical models of CPUs were so instrumental in understanding optimal designs and their key limiting factors, if we are able to come up with some similar set of equations we could draw similar conclusions [2], [6]. Here we present the first order design considerations needed to accurately model accelerators. We further examine how these factors differ from those assumed in modeling general-purpose cores and why previous modeling assumptions are not valid when modeling fixed-function hardware.

A. Modeling Accelerators

The existence of a general structure and architecture of the CPU does not exist for accelerators. An accelerator’s datapath is completely dependent on the application it is being built for. Note that all previous models assume some known propagation delay, which serves as the basis for all timing and power models. An analogous timing baseline does not exist for accelerators. Defining this baseline is one of the key contributions of this work.

Our proposed solution assumes that propagation delay is based on a workload specific scaling function. First, a baseline is established for each application by implementing the accelerator without any optimizations, pipelining, or parallelism exploited. Then as more parallelism or pipeline stages are introduced we model the effects of these optimizations with respect to how they scale the baseline delay. This process can be automated leveraging program data dependency graphs similar to the analysis done by Aladdin [5].

B. Contrasting Accelerator and CPU Design

When modeling CPUs there are well defined hazards and their penalties are understood. Accessing hardware performance counters enumerates the number of hazards a
running program encounters. This data can be plugged into established formula to estimate the overall hazard penalties and hence performance degradation due to events such as branch misprediction and cache misses. In an accelerator, such hazards do not necessarily exist as each pipeline is application specific. These design differences require a rebuilding of power and performance models for accelerators.

As such, our solution does not have such a heavy emphasis on pipeline flushes or branch mispredictions. These large units are used to overcome the performance limitations standards like ISAs introduce; whereas accelerators exist in a more pure, unconstrained design space their models cannot be constructed based on the same assumptions. This is why our equations are based on program scaling functions encapsulating the effects of pipelining and parallelism have on power and performance.

III. UNDERSTANDING ACCELERATOR DESIGN

We seek to provide analytical power and performance models for hardware accelerators. This will enable a more fundamental, quantitative understanding of trade-offs among hardware designs. Applying the models to real workloads provides even further insight as to how and why a workload is inherently more amenable to exploit parallelism or pipelining to maximize efficiency.

We have come up with two such equations which model an accelerator’s power and performance. Combining the two we analyze accelerator designs with respect to Energy × Delay to understand what the optimal pipeline depth of an accelerator should be with respect to the amount of parallelism exploited. We also found pipelining and parallelism to be dependent on one another. This led to identifying optimal designs sweeping both parallelism and pipelining independently. The result is a quantitative mechanism to understand if a workload is best suited for deep pipelines running at high frequencies, highly parallel execution, or the precise balance between the two extremes that yields the most energy efficient design. Information on scaling factors, model derivation, and the model equations can be found on the poster presented at BARC 2015.

REFERENCES


B. Discussion

By studying and contrasting optimal design points across these benchmarks some fundamental intuition can be developed. For example, how does a distance one loop carried dependency across every loop iteration manifest itself in the optimal pipeline depth? Moreover, how does this compare to a workload with greater distances between loop carried dependencies? With this academic understanding we will further be able to dissect results by looking at inflection points of plots—these will represent where the pipelining overheads are no longer beneficial and the design should exploit more parallelism to maximize energy efficiency and vice-versa.

IV. LOOKING FORWARDS

We seek to provide analytical power and performance models for hardware accelerators. This will enable a more fundamental, quantitative understanding of trade-offs among hardware designs. Applying the models to real workloads provides even further insight as to how and why a workload is inherently more amenable to exploit parallelism or pipelining to maximize efficiency.

We have come up with two such equations which model an accelerator’s power and performance. Combining the two we analyze accelerator designs with respect to Energy × Delay to understand what the optimal pipeline depth of an accelerator should be with respect to the amount of parallelism exploited. We also found pipelining and parallelism to be dependent on one another. This led to identifying optimal designs sweeping both parallelism and pipelining independently. The result is a quantitative mechanism to understand if a workload is best suited for deep pipelines running at high frequencies, highly parallel execution, or the precise balance between the two extremes that yields the most energy efficient design. Information on scaling factors, model derivation, and the model equations can be found on the poster presented at BARC 2015.

REFERENCES