Performance modeling of virtualized custom logic computations

Michael J. Hall
Roger D. Chamberlain

Michael J. Hall and Roger D. Chamberlain. “Performance Modeling of Virtualized Custom Logic Computations,” in *Proc. of IEEE 25th International Conference on Application-specific Systems, Architectures and Processors* (ASAP), June 2014, pp 72-73.

Dept. of Computer Science and Engineering
Washington University in St. Louis

Performance Modeling of Virtualized Custom Logic Computations

Michael J. Hall and Roger D. Chamberlain
Department of Computer Science & Engineering
Washington University in St. Louis
Email: {mhall24, roger}@wustl.edu

***Abstract*—Virtualization of custom logic computations (i.e., by sharing a fixed function across distinct data streams) provides a means of computing multiple streams using shared hardware resources. The hardware can be context-switched to support virtualization using C-slow techniques (fine-grained context-switching) or by adding a secondary memory (coarse-grained context-switching). The performance of these computations depends on the circuit, technology, number of pipeline stages, number of streams, cost of a context switch, scheduling period, and arrival rate. In this paper, we analyze a virtualized hardware design and develop a set of analytic modeling equations for predicting the performance of these circuits. We then validate the model equations using a discrete-event simulation.**

I. Introduction

Virtualization is frequently used to share computing resources among multiple users. Examples of sharing hardware [1] include DSP blocks, memory controllers, memory bandwith, and IP cores.

We have developed a model of the performance of a virtualized fixed logic computation that supports fine-grained (context changes each clock cycle) and coarse-grained context-switching (the state of the computation after awhile is swapped out to a secondary memory) in a fixed hierarchical round-robin schedule. We virtualize the logic computation by sharing hardware resources to support context-switching of distinct data streams into the hardware. With this model, we can optimize a schedule for performance.

In our previous work [2], we presented high-level modeling results calibrated using empirical data from an AES encryption application and then demonstrated use of the model through an example. Here we present the model development and validation of the model using a discrete-event simulation.

II. Model Development

The general hardware configuration we consider is shown in Figure 1. An arbitrary sequential circuit has been C-slowed (i.e., has C pipeline stages) and augmented with a secondary memory that can load and unload copies of the "active" state. The circuit consumes one data element, x, (from one of N inputs specified by a schedule) each clock tick.

We represent the performance of context-switched hardware via an open queueing network model with N queueing stations, one for each input data stream. Each individual queueing station consists of a FIFO queue and associated server, and

This research was supported by NSF grant CNS-0931693 and Exegy, Inc.

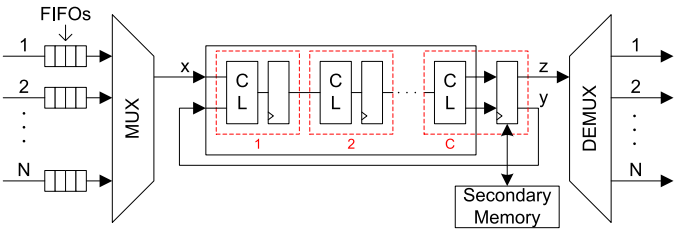

Fig. 1. General hardware configuration of a C-slowed sequential logic circuit with secondary memory supporting N data input/output streams.

represents one virtual copy of the hardware computation with arrival rate λ_i $^{\text{element}}/_{\text{s}}$ for the i^{th} data stream. The effective service rate, μ_s, of the servers is determined by the clock frequency achievable by the C-slowed circuit shared amongst the servers and the schedule. We assume the arrival process is Poisson and that both the secondary memory and the input buffers operate at the clock rate of the pipeline. State transfers to/from secondary memory take a given S clock cycles (enabling the model to support a range of context switch overheads).

First, we will start our model development for the case when $N = C$. If we employ a fixed, round-robin schedule (with no context switches to secondary memory), data elements will be dequeued from each input at a fixed rate of $\mu_s = f_{CLK}/C$. This corresponds to a deterministic service process for each server, with mean service rate μ_s $^{\text{elements}}/_{\text{s}}$, modeled as an independent $M/D/1$ queueing station (*M*arkovian, or memoryless, arrival process; *D*eterministic service process; 1 server) [3]. For this system, the maximum achievable throughput (per stream) is

$$\mu_s = \frac{1}{C \cdot t_{CLK}} \tag{1}$$

and the total achievable throughput is $T_{TOT} = C \cdot \mu_s$.

The average (mean) waiting time of each data element in the queue is [3]

$$W_q = \frac{1}{\mu_s} \cdot \frac{\rho}{2\,(1-\rho)}, \tag{2}$$

where $\rho = \lambda/\mu_s$. The (deterministic) time in the pipelined circuit is

$$W_s = \frac{1}{\mu_s} = C \cdot t_{CLK}, \tag{3}$$

and therefore the average latency (elapsed time from arrival to the completion of processing) for each data element is

$W = W_q + W_s$ with an average (mean) occupancy of each queue of $N_q = \lambda W_q$ [4].

Next, we will consider the case when $N > C$ and we are exploiting both fine-grained and coarse-grained context switching. We will make the simplifying assumption that N is an integer multiple of C. The schedule is a fixed, hierarchical, round-robin schedule with period R_S that acts as follows: (1) a set of C input streams is chosen to share the hardware resource and within that set a round-robin schedule is used; (2) after R_S rounds, the current set of input streams' state is swapped out to secondary memory and the next set of C input streams' state is swapped in (the time required to complete this operation is given as S clock cycles), this set is then scheduled to use the hardware in round-robin fashion; (3) the entire collection of N/C input stream sets is also chosen in round-robin fashion (hence the label hierarchical round-robin schedule), such that once every individual input stream has had R_S input elements processed the high-level schedule returns to the first set of C input streams.

For this case, the number of clock cycles to complete a full round of the hierarchical schedule (during which each server services R_S elements) is $R_S \cdot N + S \cdot N/C$. This implies the effective service rate is

$$\mu_s = \frac{R_S}{(R_S N + SN/C) \cdot t_{CLK}} \text{ elements/s}, \tag{4}$$

which simplifies to (1) when $S = 0$ (i.e., context switches are free) and $N = C$. The total achievable throughput, T_{TOT}, is now $N \cdot \mu_s$.

To develop an expression for the average (mean) time that each data element waits in one of the input buffers, we start by using (2), the $M/D/1$ expression for mean queue waiting time, and add a term, W_h, to reflect the additional time waiting in the queue due to the hierarchical round-robin schedule. The additional waiting time is experienced by the fraction of data elements that arrive when their input stream is swapped out (i.e., not receiving service). This fraction is $\frac{R_S(N-C)+SN/C}{R_S N+SN/C}$, or the number of clocks a stream is swapped out divided by the number of clocks in a full schedule round. The average additional time experienced by this fraction of input elements is one half of the time the stream is swapped out, $(R_S(N-C) + SN/C) \cdot t_{CLK}/2$. Multiplying the additional time by the fraction that experience the additional time yields

$$W_h = \frac{(R_S(N-C) + SN/C)^2 \cdot t_{CLK}}{2(R_S N + SN/C)}. \tag{5}$$

The time in the pipelined circuit does not change from $W_s = C \cdot t_{CLK}$, and therefore the average latency from arrival to completion of processing is $W = W_q + W_h + W_s$. The expression for the average (mean) queue occupancy, N_q, does not change from $N_q = \lambda W_q$. The mean number of elements waiting in the buffer due to the hierarchical round-robin scheduling is $N_h = \lambda W_h$.

The above discussion provides analytic performance expressions for total achievable throughput, T_{TOT}, and the average latency experienced by each data element, W, both in the buffer, $W_q + W_h$, and in computation, W_s. Also available is the average occupancy of the buffer, $N_q + N_h$.

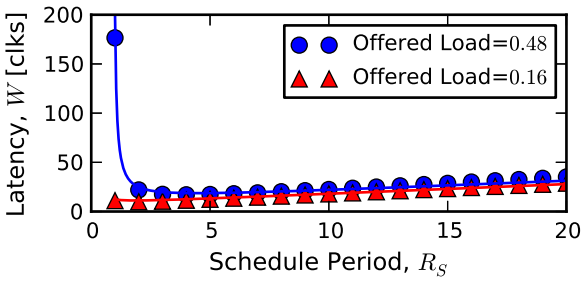

Fig. 2. Latency vs. schedule period at two offered loads (0.48 and 0.16). The curves are from the analytical expression for W, and the points are empirically measured via a discrete-event simulation.

III. Validation

To validate these modeling expressions, we developed a cycle-accurate discrete-event simulation of the system and measured the average latency of data elements from when they enter the input queue to when they exit the system. Consider a candidate design with 4 fine-grained contexts ($C = 4$), 8 total contexts ($N = 8$), a 100 MHz clock rate, and a 4 clock overhead to perform a coarse-grained context switch ($S = 4$). Figure 2 plots the total latency, W, vs. the schedule period, R_S, for two different offered loads. The points on the graph correspond to empirical results from a discrete-event simulation run with the same parameters. The offered load is the ratio of the aggregate arrival rate (of all streams) to the peak service rate of the system (i.e., when $S = 0$). Offered load then evaluates to $N \cdot \lambda \cdot t_{CLK}$. We draw two conclusions from this figure. First, there is good correspondence between the analytical model and the empirical simulation results. Second, as is readily apparent in the graph, the schedule period that optimally minimizes latency is different for different offered loads.

IV. Conclusions

An analytic model of virtualized custom logic computations that employ fixed hierarchical round robin schedules is developed for achievable throughput, latency (including traditional queueing, increased queueing due to hierarchical scheduling, and compute time), and occupancy of the input buffers. The novel portion of this model, the wait time expression W_h for increased queueing delay due to the hierarchical schedule, has been empirically validated.

References

[1] A. Canis, J. H. Anderson, and S. D. Brown, "Multi-pumping for resource reduction in FPGA high-level synthesis," in *Proc. of Conf. on Design, Automation and Test in Europe*, 2013, pp. 194–197. [Online]. Available: http://dl.acm.org/citation.cfm?id=2485288.2485338

[2] M. J. Hall and R. D. Chamberlain, "Performance modeling of virtualized custom logic computations," in *Proc. of 24th ACM Int'l Great Lakes Symposium on VLSI*, 2014.

[3] L. Kleinrock, *Queueing Theory, Volume 1*. Wiley-Interscience, 1975.

[4] J. D. C. Little, "A proof for the queuing formula: $L = \lambda W$," *Operations Research*, vol. 9, no. 3, pp. 383–387, 1961.