Tile Size Selection for Low-Power Tile-Based Architectures

John Oliver
University of California
Davis, CA
jyoliver@ucdavis.edu

Ravishankar Rao
University of California
Davis, CA
ravishankar@ucdavis.edu

Michael Brown
Cal Poly State University
San Luis Obispo, CA
whiteknite143@aol.com

Jennifer Mankin
Cal Poly State University
San Luis Obispo, CA
jmankin@calpoly.edu

Diana Franklin
Cal Poly State University
San Luis Obispo, CA
franklin@csc.calpoly.edu

Frederic T. Chong
University of California
Santa Barbara, CA
chong@cs.ucsb.edu

Venkatesh Akella
University of California
Davis, CA
akella@ece.ucdavis.edu

ABSTRACT

In this paper, we investigate the power implications of tile size selection for tile-based processors. We refer to this investigation as a tile *granularity* study. This is accomplished by distilling the architectural cost of tiles with different computational widths into a system metric we call the *Granularity Indicator* (GI). The GI is then compared against the communications exposed when algorithms are partitioned across multiple tiles. Through this comparison, the tile granularity that best fits a given set of algorithms can be determined, reducing the system power for that set of algorithms. When the GI analysis is applied to the Synchroscalar tile architecture [1], we find that Synchroscalar's already low power consumption can be further reduced by 14% when customized for execution of the 802.11a reciever. In addition, the GI can also be a used to evaluate tile size when considering multiple applications simultaneously, providing a convenient platform for hardware-software co-design.

Keywords

Media Processors, Multi-Core Processors

1. INTRODUCTION

As power and complexity have become increasingly problematic in modern microprocessors, tile-based architectures have become increasingly attractive (ie. [2] [3] [4]). In essence, these systems trade architectural complexity for communications, spreading work across a number of sparsely-connected small tiles rather than among richly-connected functional units of a monolithic, wide core.

However, the choice of tile size for tile-based architectures has been largely an ad-hoc, qualitative process. While this may be because of practical reasons (such as availability of cores), this may not yield an efficient design.

In this paper, we find that in systems where low power operation is critical, proper tile size selection is important. How does an architect find the best tile size for low-power operation? To investigate this we first generate cores with different amounts of computational power. We note that the larger, more richly interconnected tiles have higher average switching capacitance per operation, but also have a larger locality of data available to them. Then, we tile these cores until a fixed amount of total computational parallelism is reached, providing us with a set of tile architectures with different computational *granularity* but with the same amount of total computational power. By mapping the power efficiency per operation, we then generate a power efficiency curve that we call the Granularity Indicator (GI).

Once we find the GI for a tile architecture with different granularities, we then partition and map different algorithms to the different granularities of the tile architecture and execute them. This process yields the computation cost and the communications cost required for the algorithms to execute across multiple tiles for a tile architecture of differing granularities.

Finally, we can then compare the cost of partitioning an algorithm against the energy efficiency of the tile architecture which is embodied within the GI. If large amounts communications are exposed by partitioning the algorithm, larger tiles that invest more heavily in connectivity are favored, as they are more apt to hide communications. This

is despite the fact that the extra connectivity within larger tiles contribute to higher average switching capacitance. On the otherhand, if little communications is exposed when partitioning an algorithmn, then smaller, more power efficient tiles are favorable. The result of this comparison is to find the the granularity of tile that has the best power consumption for a given algorithm.

To drive this exploration, we use Synchroscalar [1] architecture as a basis, but other tile-based architectures could be used with a similar methodology. We use the GI framework on the Synchroscalar architecture to investigate how the computation power of the Synchroscalar tiles can be tailored to execute a the 802.11a PHY layer application at low power. We then weigh the cost of this customization against other applications that Synchroscalar may execute. We find that by tailoring the tile granularity to a given application may signifcantly negatively impact the power consumption of other algorithms, making tile granularity an important decision for low-power tile architectures.

The rest of this paper is structured as follows. First, we develop a set of cores with different amounts of computational parallelism. With these cores, we populate a tile architecture similar to Synchroscalar and generate multiple variants of Synchroscalar with different tile granularities. Using these Synchroscalar-like processor-variants, we can find the GI.

Next, we describe the methodology used to partition, map and execute different algorithms on our tile architectures with differing tile granularities. This process yields the computation time required to execute and algorithm and the number of cycles required for inter-tile communication that are required to maintain data coherency across multiple tiles.

These cycle results then allows us to compare the communications requirements against the GI. This, we will demonstrate, can tell us which granularity of tile executes a given algorithm at the lowest power.

Finally, we use the GI as a guide and we re-design the Synchroscalar architecture for low power 802.11a PHY layer execution. We will show how much power can be saved through the choice of proper tile granularity and also investigate the implications of this optimization on other applications executed on Synchroscalar.

We finish this paper with related works and then conclude.

2. TILE SCALING MODELS

In this section, we develop tiles with different amounts of computational parallelism with which to create a tile-based architecture. While the models presented here could be developed in a number of different ways, it is important to remember that the central message of the GI arises from non-linear scaling as issue-width grows wider. We argue that this is a valid assumption for any set of tile sizes, since linear scaling would, in essence, reduce a large tile to a collection of small tiles.

To connect the tiles of our tile architecture, we develop models for a bus, statically scheduled mesh, and dynamically scheduled mesh interconnects. These interconnect topologies are intended to be general and cover a wide range of interconnect topologies. Other interconnect networks, such as Raw's Scalar Operand Network [5], could be employed in a similar study.

The rest of this section describes the details of how the models in this study were created. We first begin describing how the tile area and power models were created. Then, we look at how we derived the interconnection models used in this study.

2.1 Tile Area

The goal of our tile model is to capture the first-order scaling effects of computational width on area and power. We define the computational width of a tile as the maximum number of arithmetic operations that can be completed per clock cycle, where the operands are in the local register file. The smallest tile we consider in this study can compute a single operation in every cycle, while the largest tile we consider can compute 32 operations in parallel every cycle. We assume a VLIW-based architecture, which can be efficiently scheduled for data-parallel applications like media applications. The register file of our model is assumed to provide one write and two read ports for each operation.

We first developed a tile based on the Blackfin Digital Signal Processor (DSP) [6], which can be viewed as having a computational width of two. In order to get a power and area estimate for this processor, we modeled the control logic of this processor in VHDL and synthesized it using the Synopsys Design compiler. The data-path units, i.e. multipliers, register file and memory, were estimated using published numbers [7, 8, 9].

Using the width-2 Blackfin DSP as a basis, we extrapolate the tile area for tiles with a computational width of one, four, eight, sixteen and thirty-two. We assume that the area of control logic scales linearly with computational width as well as the area contributions of the ALU, shifter, accumulator and multiplier. Memory capacity is assumed to grow linearly with computational width at 32 KB of instruction and 32 KB of data memory per computational width.

For the register file, we assume that the number of ports in the register file, as well as the capacity, grows linearly with the computational width. This produces a quadratic increase in both power and area in the register file. Finally, the on-chip wiring/data-forwarding paths are also assumed to grow quadratically, in a similar manner as the register file.

Figure 1 shows the area results of our tile model. We hold the total computation width constant at 32 computational widths, so when we halve the computational width of a tile, we double the number of tiles we are using. The left most bar in Figure 1 shows the area breakdown for a single tile with a computational width of 32. The next bar to the right shows a tile model with two tiles with width 16. The column furthest on the right shows the area of 32 tile each with a single computational width. The single large tile with a width of 32 has a 93% area growth over the array of 32 tiles with computational widths of one. Note that if processor area is a design constraint, this will need to be weighed in conjunction with any power saving we present in this paper. However, for this study, we concentrate only on saving power.

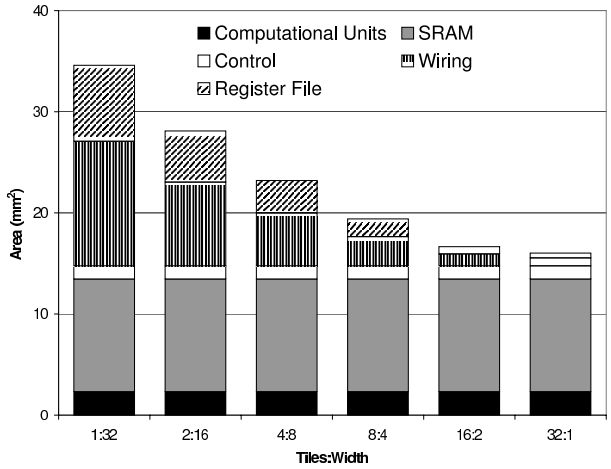


Figure 1: Tile Area scaling for 1 32-wide tile (1:32) to 32 1-wide tiles (32:1).

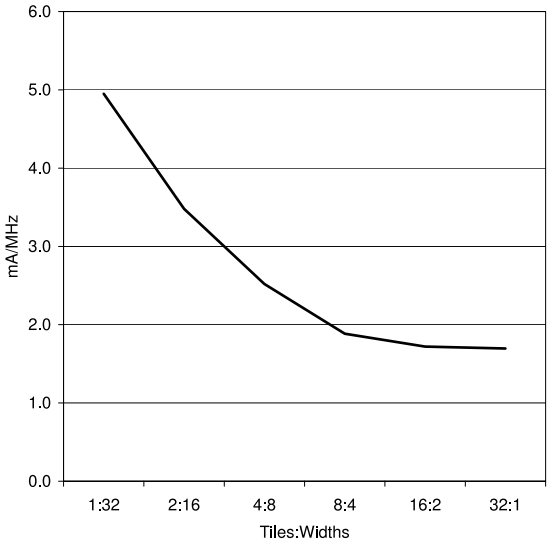


Figure 2: Current required for our tile architecture model with 32 computational widths of differing granularities.

2.2 Tile Power

The tile power of our tile model is composed of two portions, the active power and the leakage power. To find the active power, we use power numbers based upon synthesis of the Blackfin core, as described in the previous section. This yielded a power estimate of 0.1 mA/MHz at 1 V, on average.

For the other granularity of tiles the average current for each of tiles is assumed to be proportional to the area relative to our Blackfin core. This is a decent approximation if two conditions are met. First, the activity factor of the tiles must be constant. Since the partitioning of data parallel multimedia applications used in this study are done in a load-balanced manner, this should hold approximately true. Second, for those micro-architectural structures that have non-linear area growth, their power consumptions must track the growth in area. This is true for the register file to a first approximation, as register files have been shown to have active power consumption that is linearly proportional to area [9].

For our leakage model, we assume that leakage power is proportional to the number of transistors. Using an average of 830 pA of leakage per transistor [10], we approximate that the Blackfin DSP leaks 1.5mA. This provides a range where the smallest, single-width tile leaks 0.74 mA of current, and the largest, width-32 tile leaks 23.68 mA of current.

Having established the assumptions for our tile power scaling, we find that this provides a range of currents consumed for different sized tiles. A single computational width tile uses 0.05 mA/MHz on average, while the largest 32-width tile consumes 4.87 mA/MHz on average. For a total of 32 computation ways, this yields a tile architecture that has current requirements as shown in Figure 2.

2.2.1 Tile Power Model Correlation

Although the GI metric and analysis methodology can be applied to a wide array of tile power scalings, in order to demonstrate the usefulness of the GI, our tile model needs to reflect the scaling trends that real processors will observe. In order to see if our tile power model scales as industrial processors do, we have plotted published power results from similar processors from industry. We expect our power models to lie below the curve of the realistic processors for two reasons. First, we model only the core components, not the I/O devices and special purpose circuits. In addition, often commercial projects scale not only the width, but also the functionality, adding specialized units and other functions while we are only looking at computational cores. Figure 2.2.1 shows commercial processors, normalized for process technology. Next to Figure 2.2.1, we show Table 2.2.1 which contains the references for the processors used in Figure 2.2.1. We can see that, as expected, our scaling model shows a similar trend but has lower absolute power than the published results.

2.3 Inter-Tile Interconnect

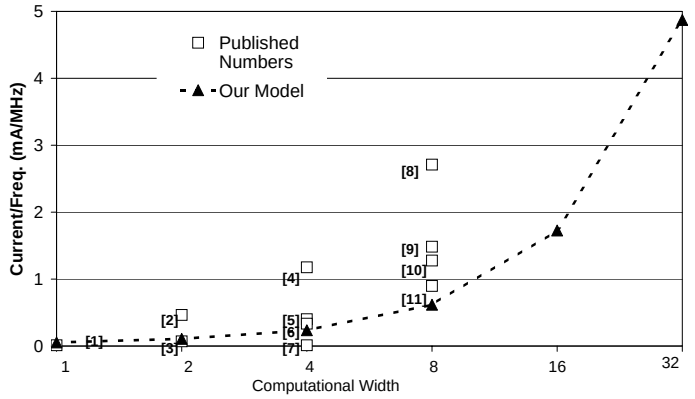
To properly account for the power due to inter-tile communication, we need two values - the delay caused by the interconnect (and thus the idle cycles of the tiles) and the power required by the interconnect to perform communication. In this study, we evaluate both a single bus and a generalized mesh interconnect topologies. We also assume that data is communicated between tiles using explicit message passing. A similar study could be done with a shared memory system with a hardware-enforced coherence protocol.

2.3.1 Interconnect Delay

In order to calculate communications delay, we must know how much communication occurs and how long each transmission takes. Details on how we find the amount of communication required by an algorithm are described later in Section 4.

For a shared bus, the delay for each communication is a single cycle, regardless of the source and destination. Since the distance of the bus is small and operational frequencies are limited in the Synchroscalar architecture, single cycle communications is possible using a bus.

The delay of a mesh is a function of the contention on



Plot #	Processor	Citation
1	Analog Devices ADSP-2191	[11]
2	TI TMS320C2810	[12]
3	NEC SPXK5	[13]
4	Hitachi SH-Mobile3	[14]
5	Infineon Tricore 2	[15]
6	Analog Devices TS-101	[16]
7	Transmeta Crusoe TM5700	[17]
8	Transmeta Efficeon TM8820	[18]
9	TI TMS320C62x	[19]
10	TI TMS320C64x	[20]
11	TI TMS320DM643	[21]

Figure 3: Our power model correlated to published power consumptions numbers for similar VLIW processors. The numbers in the chart on the left correspond to the plot numbers listed on the table to the right.

the mesh. This requires a traffic simulator to accurately find contention on the mesh. Mesh simulations were completed using the FlexSim mesh simulator from USC [22]. FlexSim was configured in a 2-D space for up to 32 switches, where each switch is attached to an end-node with one injection channel to the switch. FlexSim was modified in two ways. First, the default latencies were reduced to allow low-overhead flit-level routing as expected for an on-chip network. Also, an optional mode was introduced in which only the link overhead was counted, and the routing overhead was discounted, in order to simulate the delay for a statically-scheduled mesh. This allows us to more closely emulate the statically scheduled nature of the Synchroscalar inter-tile interconnect.

2.3.2 *Interconnect Power*

For our inter-tile interconnect power model, we employ power costs as abstracted from the Orion interconnect power model [23] from Princeton University. We find that our wires are using in the neighborhood or 10 pJ/bit for a 10mm trace, similar to Stanford's Smart Memories [24].

As observed in previous studies by the RAW project [25] and the Synchroscalar project [1], interconnect switching power can be a small portion of the overall power consumption. The reason for this is two-fold. First, the number and size of tiles used in this study are relatively small, thus not requiring an abundance of interconnect resources. Second, the frequencies of operation of the tiles in this study is relatively low compared to high-speed processors, resulting is low frequency communications as well.

3. THE GRANULARITY INDICATOR (GI)

Now that the tile area and power model used in this study has been introduced, we will encapsulate the power of different granularities of tiles into a metric we call the Granularity Indicator (GI).

The GI expresses the *architectural power characteristics* of a tile architecture that is comprised of tiles with different computational granularities. At it's simplest form, the GI is a measure of the relative energy efficiency per operation of different granularities of tile architectures, similar to Figure 2 except with a simple added transformation. The

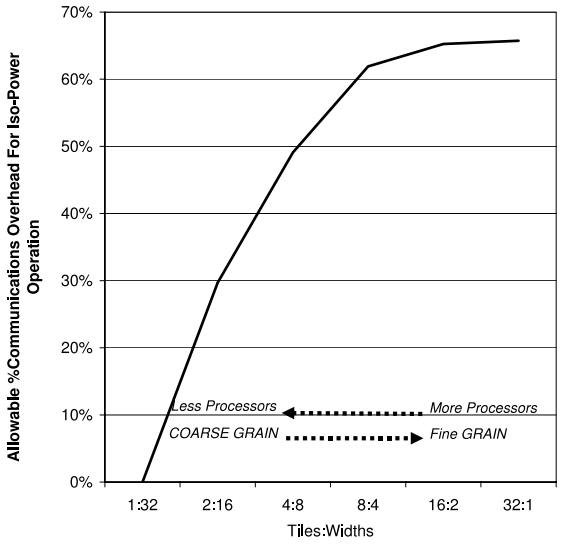


Figure 4: The GI for tile model. A single 32-width tile is shown on the left and thirty-two 1-width tiles is shown on the right. Smaller tiles have less average switching capacitance, which can be re-invested into communications.

additional energy saved on every operation by a finer granularity tile architecture is reinvested into a communications budget. So, for every pJ of energy a smaller tile saves in energy consumption versus a larger tile, it can re-invest that energy into communications. This has the effect of changing the vertical axis of Figure 2 from power consumption to allowable communications overhead while maintaining iso-power consumption.

For our tile model, with an power consumption curve as shown in Figure 2, this transformation creates the GI which is shown in Figure 4. On the left of Figure 4, we see a single large tile with 32 computational widths. As we move to the right on Figure 4, we double the number of tiles but halve the widths. By moving to finer-grain tiles, we know that the average energy consumption per operation is reduced, as shown in Figure 2. However, this is shown as allowable communications given iso-power consumption in Figure 4.

This change in axis is convenient for comparing the energy efficiency of a tile architecture with a given granularity for a given communications requirements for a given algorithm because the communications overhead allowed can be easily matched against the communications requirements of a partitioned algorithm.

Alternatively, to find the system power of an architecture executing a given algorithm, extensive simulation is typically involved. This transformation to the GI allows us to decouple the architectural contributions to power consumption from the algorithm's demands for computation and communications cycles. This decoupling allows an architect to make architectural-based decisions to minimize power while quickly evaluating the effectiveness of those architectural decisions for a given set of algorithms. In section 3.1 we look at the impact of voltage scaling on the GI. In section 3.2 we describe how a mode that puts tiles into a low-power mode when completing inter-tile communications can affect the GI. In section 5, it will be clear how these shifts can lead to understanding of the effectiveness of certain architectural features for a given algorithm or application.

3.1 Tile Voltage-Frequency Scaling

As we can see from the GI in Figure 4, a tile architecture comprised of smaller tiles has a higher allowable communications overhead than a tile architecture made of larger tiles and still have the same power consumption. However, to support this communications overhead, additional cycles are required which will result in a higher operational frequency for a given throughput. Additionally, this higher frequency requires a higher operational voltage, which also will increase the power consumption of a tile architecture comprised of smaller tiles. Therefore, in the presence of voltage scaling, a tile architecture with many small, higher frequency tiles will consume relatively more power than a tile architecture with fewer, lower frequency tiles.

Figure 5 shows four GI curves, corresponding to four different base frequencies. The base frequency is defined as the frequency of operation of a single tile with a computational width of 32. We see that the GI is shifted down-wards at higher frequencies. So the impact of voltage scaling is to reduce the amounts of allowable communications for a finer grain tile architecture for the same amount of power consumption as a coarser grain tile architecture.

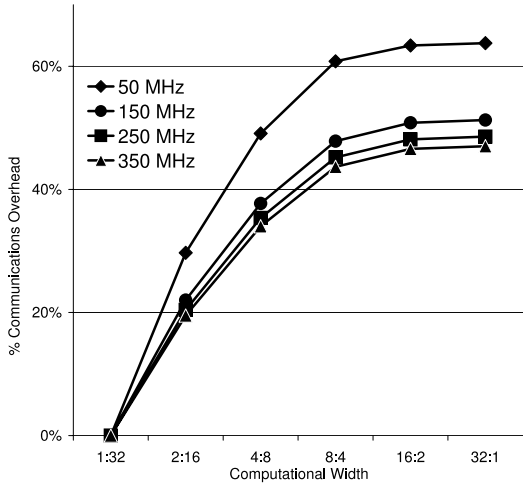


Figure 5: Finer grain tiles require more cycles to execute a given algorithm, because of added communications costs. These added cycles require higher frequency of operation, thus higher supply voltages as well. As a result, due to voltage scaling, finer grain tiles have the amount of allowable communications for iso-power consumption reduced.

3.2 Low Power Idle Tiles for Low Power Communication

The GI can also show how using an idle communication mode in the tiles impacts the power consumption. The Blackfin DSP requires a total of two cycles to enter two cycles to exit from idle mode. While in idle mode, the core consumes approximately a fifth of the active power consumption. For communications that cannot be overlapped with computation, this mode can be used to reduce the overall system power.

Figure 6 shows the impact on the GI when using tiles with this idle mode for non-overlappable communications. Intuitively, communication with the addition of this mode now costs relatively less. The result is that fine-grain tiles should become more attractive since fine-grain tiles require more communication than coarse grain tiles for a fixed amount of aggregate parallelism. Likewise, we would expect the power consumption of an application mapped onto fine grain tiles to decrease. This effect is shown on the GI in Figure 6. We see that this can make a dramatic difference in the communication supported by smaller tiles. Not surprisingly, as more tiles (requiring more communication) are used, more power is saved by the implementation of the idle mode.

Now that we have introduced the GI and presented a pair of features that may impact the GI, we will now talk about how we find the communications overhead required by partitioned algorithms.

4. ALGORITHM AND APPLICATIONS PARTITIONING METHODOLOGY

The goal of our algorithm partitioning and mapping analysis is to find out, for each granularity of tile architecture, the amount of communication and computation that needs

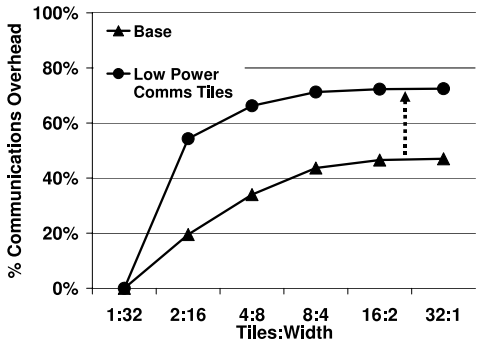


Figure 6: A low-power idle mode used to limit the current consumption of tiles when idle reduces the cost of communications. Since finer grain tiles expose more communications, features that reduce the power cost of communications favor finer grain tiles. The result is that the GI shifts upwards.

to occur for completion of that algorithm. Calculating this was a multi-step process.

The algorithms we chose to evaluate are those that can be executed on the Synchroscalar tile architecture, namely static media-based applications. Due to the static nature of these algorithms, we adopted a graph-oriented approach using the best-known graph partitioning algorithms to obtain the best parallelization possible for each granularity. For dynamic workloads, other partitioning methods may be used in conjunction with the GI to find the optimal power consumption of those workloads and architectures.

The first step is to express the algorithms as data flow graphs (DFGs). Next, an algorithm is then partitioned onto multiple tiles. To partition and map the DFGs, we iteratively employ Chaco [26], which is a graph partitioning tool that is used in the scientific computing community for high-performance multiprocessors. In particular, Chaco uses recursive spectral bisection (which performs minimum cuts through eigenvalues of an adjacency matrix) with a Kernighan-Lin heuristic to improve the partition resolution. The result is load balanced partitions with minimal N-section bandwidth.

To find the execution time of the algorithms, we can first use the Blackfin simulator to find the execution time of the computational nodes in the graph. Then we use the FlexSim cycle accurate network simulator [22] to simulate the communications cycles required. This process is repeated for each tile granularity and each algorithm, yielding the number of computational and communications cycles required for each algorithm on each granularity of tile architecture.

Now that the algorithms partitioning methodology has been detailed, we can proceed to compare the efficiency of different tile granularities against the communications requirements of the partitioned algorithms.

5. PARTITIONING RESULTS AND GRANULARITY ANALYSIS

We now show the partitioning results of several algorithms and compare these results against the GI. This will allow us to see which algorithms execute most efficiently on a given granularity of tile architecture.

Figure 7 shows the amount of inter-tile communication overhead of the different algorithms on three different interconnects - bus, statically routed mesh, and dynamically routed mesh. We assume that the interconnect wire-widths is 32b for each of the topologies.

We can see in Figure 7 that LDPC requires the most communication, and both trellis-based algorithms, the FFT and Viterbi ACS, have high degrees of communication. For an FIR block filter, our partitioning was able to hide most of the required communications. Likewise, MPEG4 encoding and Software Radio do not expose large amounts of communications when partitions cross multiple tiles.

Now, lets compare the communications requirements of these algorithms against the GI. The dotted line in Figure 7 is the GI for our tile scaling model for the Synchroscalar architecture. Remember, the GI curve shows the maximum amount communications while maintaining iso-power consumption for tile architectures with different tile granularities. Therefore, algorithms that have inter-tile communications requirements above the GI for a given tile granularity will execute more efficiently a coarser-grain tile architecture. If the exposed communications curve is above the GI at all granularities, then a single large tile the most efficient choice. Conversely, algorithms with communications requirement below the GI will execute most efficiently at the granularity that has the largest distance between the GI curve and the exposed communications curve. It is in this way that the GI exposes relative power consumptions for different granularities of tiles for a given algorithm.

For instance, in Figure 7, we can see that on a static mesh interconnect, the 64 point FFT (as marked by white triangles) is above the dotted GI line at all tile granularities except for a single large tile. This indicates that the amount of communication exposed by partitioning the FFT is greater than what is allowed by the GI to maintain iso-power execution. Therefore, a single large tile is the most efficient granularity for a 64 point FFT that requires 32 widths of total performance. Likewise, we can see that the Viterbi ACS trellis (as marked by black squares), dips below the GI when executed on a static mesh for two tiles with a computational width of 16. Since the Viterbi ACS exposed communications curve is above the GI at all other granularities, the Viterbi ACS trellis will execute at lowest power on two tiles with a computational width of 16.

To show that the relative location of the communications curve compared with the GI is a good indicator of minimal power consumption, we also show in Figure 8 the power consumption of these algorithms on Synchroscalar-based tile architecture. In Figure 8, for a static mesh, we can see that FFT (again marked by white triangles) has a higher power consumption for all partitionings, so a single large tile is best. For Viterbi ACS, the power consumption is lowest for the 2:16 case for the statically scheduled mesh, just as indicated by the GI analysis.

5.1 Using the GI

Thus far, we have developed the GI to describe the best-fit granularity of tiles for a given application. However, the GI can also be used to direct architectural design.

Suppose an architect is interested in building a media processor to run FFTs. Furthermore, the architect has flexibil-

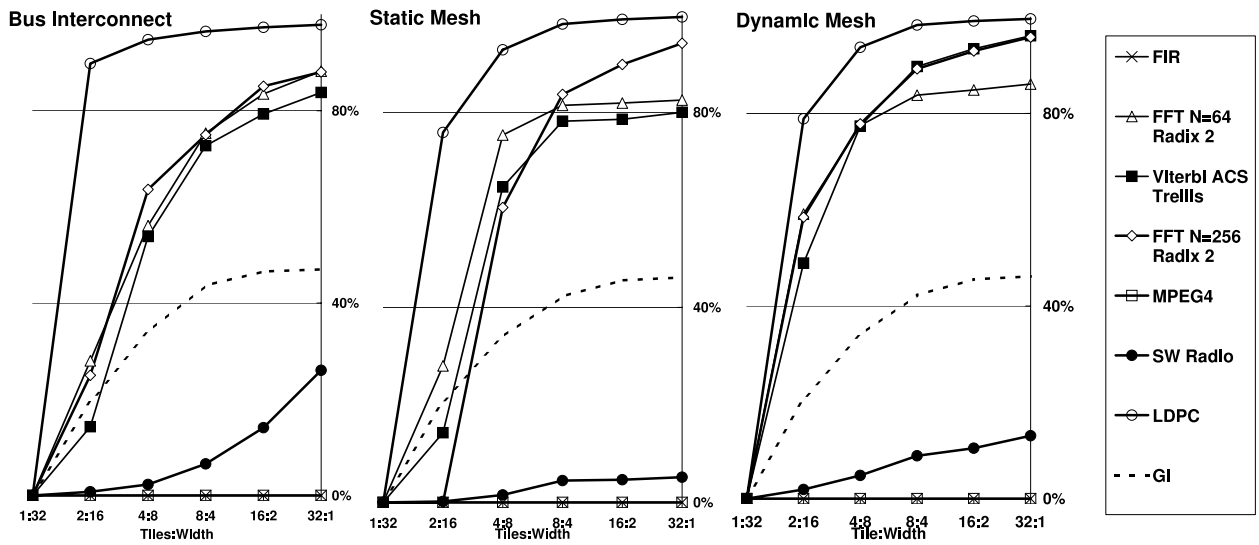


Figure 7: The communication overheads of our algorithms on three interconnection networks. The left chart shows results for a bus interconnect, the middle for a statically scheduled mesh, and the right for a dynamically scheduled mesh.

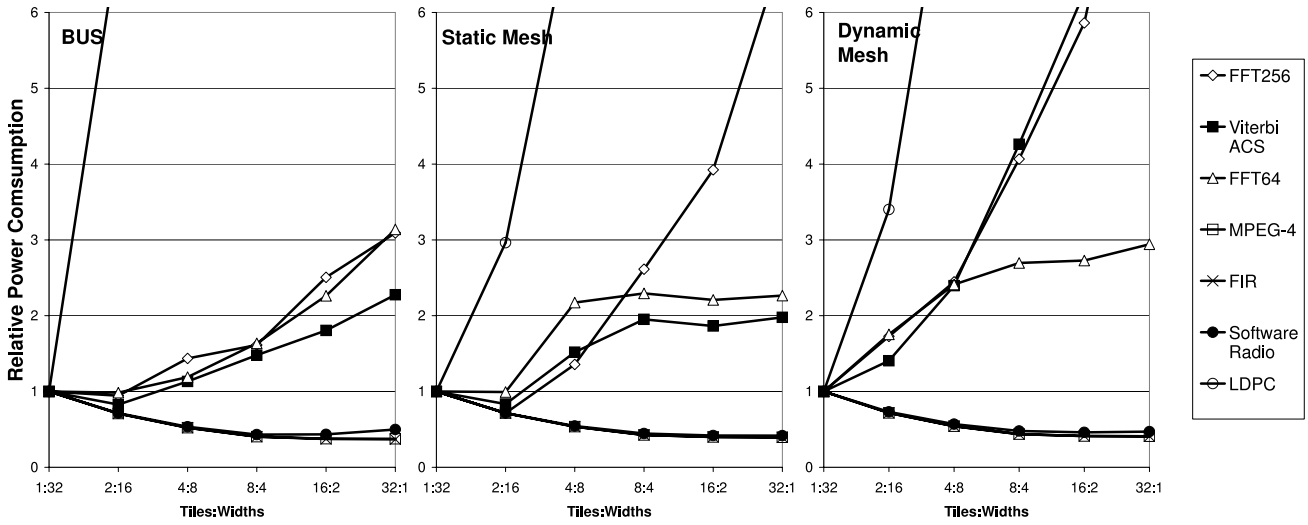


Figure 8: The relative power consumptions of our algorithms on three interconnection networks. Note that these are relative to the difference between the communications requirements of an algorithm and the GI.

ity to choose what size of tile to use as well as the width of the mesh-based interconnect, but is constrained to a total of 32 computational widths. The GI can be used to guide the architect to these decisions.

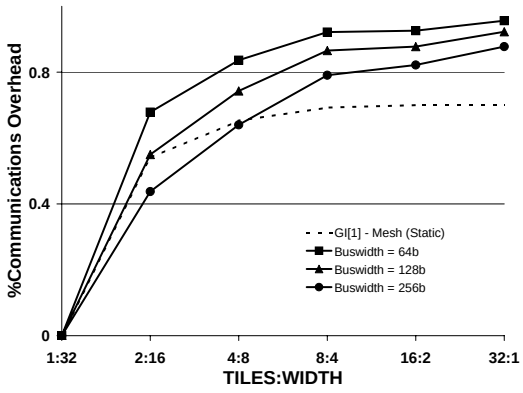


Figure 9: The GI and the communication requirements for a 64 point FFT for different mesh bandwidths.

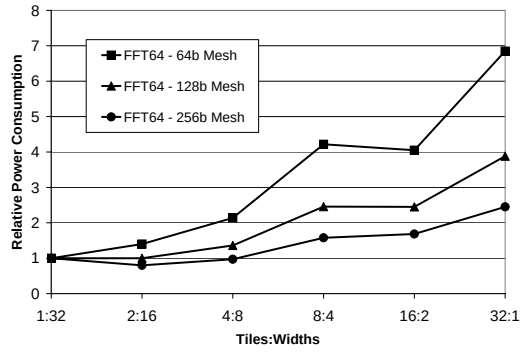


Figure 10: Relative power for FFT plotted for different mesh bandwidths. We can see that the bandwidth of the inter-tile interconnect impacts the best granularity of tile for low-power execution. For a 256b Mesh two 16-width tiles is most efficient, while for a 64b mesh, a single 32-width tile is most efficient.

Figure 9 shows the communication overheads for a 64 point FFT, mapped with the base-line GI(1). From this figure, we can see that the FFT requires more communication than a 64b mesh can support at any granularity, except for a single large tile. Therefore, for a tile architecture with a 64b mesh, running on a single tile is the best option. However, as we increase the bandwidth of the mesh, the inter-tile communication overhead is reduced. This has the effect of making large tiles that have high amounts of local on-tile interconnect relatively less powerful than smaller tiles. Indeed, for a 256b mesh, we see that the FFTs communication overhead curve has dipped below the GI for 2:16 and 4:8 points. This indicates the large 32-wide tile is no long the most power efficient.

Furthermore, by utilizing the GI Numbers for the FFT with our tile model, we can make trade-offs between tile granularity and inter-tile interconnect. These trade-offs are shown in Figure **??**. For instance, we can see that 32 width-1 tiles using a 128b mesh has a lower GI number than eight width-4 tiles using a 64b mesh. The GI Number gives the architect the ability to weigh the added (predominantly) area cost of the larger interconnect with the power saved by using a larger mesh.

Now that we have seen how the GI, in conjunction with algorithm communications overheads, can be used to find the granularity of tile with the lowest power consumption, we will show how the GI can be used to to improve the Synchroscalar architecture.

6. APPLICATION OF THE GI TO IMPROVE SYNCHROSCALAR

Using the GI, we can revisit the design of the Synchroscalar architecture. Synchroscalar was designed as a system based on 2-wide Blackfin tiles. We will attempt to use the GI and customize the tile granularity for efficient execution of 802.11a PHY layer baseband processing.

We will assume the same tile scaling model as presented in Section 2. First we need to compute the GI for our tile model. We will also assume that this version of Synchroscalar uses a generalized, statically scheduled mesh as an inter-tile interconnect. The GI for this model is shown in Figure 11 as a dotted line.

Next, we need to find the communications overhead of 802.11a when it is partitioned across 2, 4, 8, 16 and 32 tiles. This is shown in Figure 11 as a solid black line.

Now, to find the most efficient granularity, all we need to do is find the place where the communications exposed by partitioning 802.11a is lowest relative to the GI. From Figure 11, we can clearly see that this occurs with two tiles with a computation width of 16. Again, for validation, the power consumption of 802.11a on differing granularities of Synchroscalar has also been plotted in Figure 11, as marked by triangles. Indeed, we can see that two 16-width tiles is the lowest power consuming granularity, saving Synchroscalar 14% power over Synchroscalar's already very low power consumption.

Before moving on, in sections 3.1 and 3.2 we discussed how architectural features can shift the GI up or down. We can now see how this is useful information. A downward shift in the GI would likely mean that for the Viterbi decoder on a static mesh, two 16-width tiles would no longer be the most efficient operating point. This is because if the GI shifted down-wards (perhaps by implementing dynamic voltage-frequency scaling on the tiles), the Viterbi decoder's communications requirements would likely be above the GI curve. So, if the GI shifted down-wards, this indicates that a single large tile would be the most efficient for executing the Viterbi decoder. A similar result can be seen here in the case of 802.11a. Conversely, if tile idle modes were implemented, the GI would shift up-wards. This would perhaps allow some more algorithms to execute more power efficiently on finer grain tiles.

Now that we have found the tile granularity that most efficiently executes for 802.11a, lets investigate the impact of this on the other applications that Synchroscalar supports. In Figure 12, the power consumptions of four different applications are shown, both for the original width-2 tile Synchroscalar array and the width-16 tile Synchroscalar array.

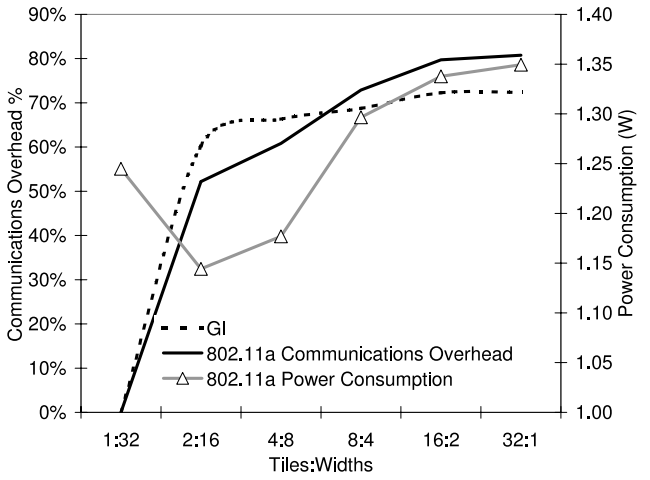


Figure 11: The dotted Line is the GI and the solid Line is the communications overhead exposed when partitioning the 802.11a signal chain on up to 32 tiles. The Power Consumption of 802.11a tracks the relative distance between the GI and communications overhead.

The changes in power consumption are shown numerically on top of each pair of bars. We see that 802.11a saves about 14% power over the orginal Synchroscalar array, but this comes at a cost of a 65% increase in power consumption for MPEG4. The architect can then easily find the best trade-off of tile granularity and power consumption for all the applications of interest by using the GI as a guide. This makes the GI a useful hardware/software co-design tool.

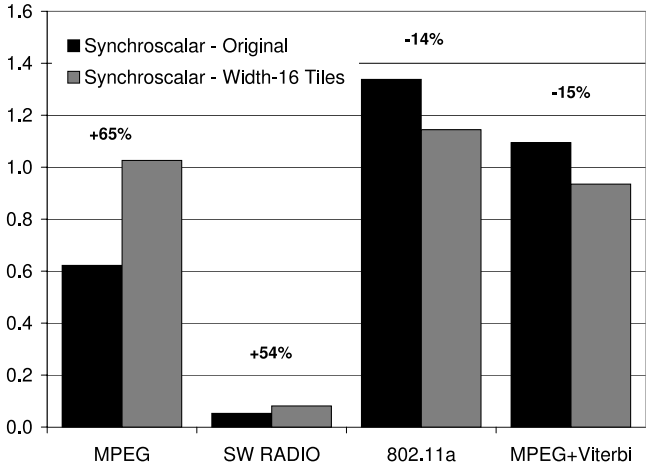


Figure 12: Four different applications are plotted for the original width-2 Synchroscalar array and the width-16 Synchroscalar array. Added power consumption percentages are shown above each set of bars.

7. RELATED WORK

Our work attempts to build intuitive understanding of a design space occupied many diverse projects. The MIT SCALE project [27] is developing a tile-based power efficient architecture based on their Vector-Thread paradigm. In their prototype SCALE processor, they are able to develop a simple micro-architecture that attains high performance and low power execution by avoiding complex control structures and utilizing spatial locality. The EnyAC group at Carnegie Mellon [28] is investigating globally asynchronous, locally synchronous designs to allow for dynamic voltage and frequency scaling for low power consumption.

On the processor-power efficiency front, Zyuban [29] has developed an architectural based power-performance efficiency metric for a single microprocessor which allows efficiency to be evaluated during the development of the ISA. In a complementary study, Hartstein and Puzak [30] develop an power efficiency metric and investigate the power efficiency of deep pipelines on a processor. While these studies are concerned with the power efficiency of a single tile, our study extends the study of power efficient processors to multiple-processors on a single chip. In the paper, Custom Fit Processor [31], a VLIW tile model is developed and performance is weighed against area cost, but not against power. A study similar to ours for energy efficient interconnects [32] has been published by Heo and Asanovic.

Finally, it is because of the many different tile based architectures that are being researched that this study was developed. The RAW project [2] [33] uses MIPS-based cores as tiles and shows performance scalability through their robust, three-level inter tile communication structure. Also, in a similar effort is the Smart Memories project citesmart-memories. Smart Memories uses finer-grain tiles than the RAW processor. The TRIPS architecture [3] also attacks wire-scalability by utilizing multiple cores. Additionally, TRIPS is a malleable architecture that can adapt to different types of workloads to gain performance, yet maintain performance for general purpose workloads. One study that looks at a heterogeneous tile structure was done at Technion [34] and allows the core with the best power efficiency to execute.

7.1 Future Work

In this work, we assume a flat topology, where the inter-tile interconnect bandwidth is evenly distributed across multiple ALUs. One interesting addition would be to incorporate hierarchical interconnect topologies. Additionally, extending the GI framework to include shared memory multiprocessors with hardware enforced coherence protocols would be a valuable extension.

8. CONCLUSIONS

The Granularity Indicator (GI) provides a novel way to encapsulate power scaling factors when trying to meet performance targets with parallelism. The GI can be used to discover which algorithms can be executed in a power efficient manner on small or large tiles. Additionally, through the use of the GI and knowledge of communication overheads from algorithms, tile architectures can be optimized for granularity of targeted application mixes.

We have presented the GI and used it to show many different forms of analysis. We have explored how base frequency and idle-modes affect the power-performance scal-

ing and how applications behave with different tile widths. Finally, we used the GI to revisit tile granularity in Synchroscalar. We found that the use of the Blackfin DSP as our tile, was non-optimal in terms of power for our centerpiece application, the 802.11a receiver.

9. ACKNOWLEDGMENTS

We would like to thank the SMART Interconnects group at USC for support with the FlexSim interconnect simulator, the Orion group at Princeton University for support with the Orion power-performance interconnect simulator and the MIT Raw group for the software radio benchmark.

This work is supported by NSF ITR grants 0312837 and 0113418, and NSF CAREER and UC Davis Chancellor's fellowship awards to Fred Chong.

Diana Franklin's faculty position is funded by a Forbes Endowment.

10. REFERENCES

[1] J. Oliver, R. Rao, P. Sultatna, J. Crandall, E. Czernikowski, L. W. Jones, D. Franklin, V. Akella, and F. T. Chong, "Synchroscalar: A multiple clock domain, power-aware, tile-based embedded processor," in *31th Annual International Symposium on Computer Architecture (31th ISCA-2004) Computer Architecture News*, ACM SIGARCH / IEEE, June 2004.

[2] M. B. Taylor *et al.*, "The Raw microprocessor: A computational fabric for software circuits and general-purpose programs," *IEEE Micro*, vol. 22, pp. 25–35, Mar./Apr. 2002.

[3] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, N. Ranganathan, D. Burger, S. W. Keckler, R. G. McDonald, and C. R. Moore, "Trips: A polymorphous architecture for exploiting ilp, tlp, and dlp," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 1, pp. 62–93, 2004.

[4] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz, "Smart memories: A modular reconfigurable architecture," in *27th Annual International Symposium on Computer Architecture (27th ISCA-2000) Computer Architecture News*, (Vancouver, British Columbia, Canada), ACM SIGARCH / IEEE, June 2000. Published as 27th Annual International Symposium on Computer Architecture (27th ISCA-2000) Computer Architecture News, volume 28.

[5] M. B. Taylor, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, J. Kim, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. P. Amarasinghe, and A. Agarwal, "Evaluation of the raw microprocessor: An exposed-wire-delay architecture for ilp and streams.," in *ISCA*, pp. 2–13, 2004.

[6] R. Kolagotla, J. Fridman, B. Aldrich, M. Hoffman, W. Anderson, M. Allen, D. Witt, R. Dunton, and L. Booth, "High Performance Dual-MAC DSP Architecture," *IEEE Signal Processing Magazine*, July 2002.

[7] S. Gupta, S. Keckler, and D. Burger, "Technology independent area and delay estimates for microprocessor building blocks," in *Technical Report TR2000-05, Department of Computer Science, University of Texas*, 2000.

[8] A. Wolfe, J. Fritts, S. Dutta, and E. S. T. Fernandes, "Datapath design for a vliw video signal processor.," in *HPCA*, pp. 24–, 1997.

[9] S. Rixner, W. Dally, B. Khailany, P. Mattson, U. Kapasi, and J. Owens, "Register Organization for Media Processing," in *International Symposium on High Performance Computer Architecture (HPCA)*, (Toulouse, France), January 2000.

[10] S. Thompson, M. Alavi, M. Hussein, P. Jacob, C. Kenyon, P. Moon, M. Prince, S. Sivakumar, S. Tyagi, and M. Bohr, "130nm logic technology featuring 60nm transistors, low-k dielectrics, and cu interconnects," *Intel Technology Journal*, vol. 6, pp. 5–13, May 2002.

[11] "ADSP-2191 Processor Data Sheet." 2002.

[12] "TMS320C28x Processor Manual." July 2001.

[13] M. Y. T. Kumura, M. Ikekawa and I. Kuroda, "VLIW DSP for Mobile Applications," *IEEE Signal Processing Magazine*, July 2002.

[14] H. Mizuno, N. Irie, K. Uchiyama, Y. Yanagisawa, S. Yoshioka, I. Kawasaki, and T. Hattori, "SH-Mobile3: Application Processor for 3G Cellular Phones on a Low-Power SoC Design Platform," *Hot Chips 16*, August 2004.

[15] E. Norden, P. Leteinturier, J. Barrenscheen, K. Scheibert, and F. Hellwig, "A Fast Powertrain Microcontroller," August 2004.

[16] "TS-101 Data Sheet." August 2002.

[17] "Transmeta Crusoe TM5700/5900 Processors." 2003.

[18] "Transmeta Crusoe TM8300/8600 Processors." 2004.

[19] "TMS320C62x Processor Manual." July 2001.

[20] S. A. et. al., "A 600MHz VLIW DSP," February 2002.

[21] "TMS320DM642 Data Sheet." 2005.

[22] U. SMART Interconnect Group, "Flexsim 1.2 flit level simulator." http://ceng.usc.edu/smart/tools.html.

[23] X. Chen and L.-S. Peh, "Leakage power modeling and optimization in interconnection networks," in *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design*, pp. 90–95, ACM Press, 2003.

[24] R. Ho, K. Mai, and M. Horowitz, "Efficient on-chip global interconnects," in *IEEE Symposium on VLSI Circuits*, June 2003. Stanford Univeristy.

[25] J. S. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff, "Energy characterization of a tiled architecture processor with on-chip networks," in *ISLPED '03: Proceedings of the 2003 international symposium on Low power electronics and design*, pp. 424–427, ACM Press, 2003.

[26] B. Hendrickson and R. Leland, "The chaco user's guide, version 2.0, technical report sand94-2692," 1994. http://www.ti.com/corp/docs/press/backgrounder/omap.shtml.

[27] R. Krashinsky, C. Batten, M. Hampton, S. Gerding, B. Pharris, J. Casper, and K. Asanovic, "The vector-thread architecture," *SIGARCH Comput. Archit. News*, vol. 32, no. 2, p. 52, 2004.

[28] D. Marculescu, "Application adaptive energy efficient clustered architectures," in *ISLPED '04: Proceedings*

of the 2004 international symposium on Low power electronics and design, pp. 344–349, ACM Press, 2004.

[29] V. Zyuban, "Unified architecture level energy-efficiency metric," in *GLSVLSI '02: Proceedings of the 12th ACM Great Lakes symposium on VLSI*, pp. 24–29, ACM Press, 2002.

[30] A. Hartstein and T. R. Puzak, "The optimum pipeline depth considering both power and performance," *ACM Trans. Archit. Code Optim.*, vol. 1, no. 4, pp. 369–388, 2004.

[31] J. A. Fisher, P. Faraboschi, and G. Desoli, "Custom-fit processors: letting applications define architectures," in *MICRO 29: Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture*, pp. 324–335, IEEE Computer Society, 1996.

[32] S. Heo and K. Asanovic;, "Replacing global wires with an on-chip network: a power analysis," in *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, (New York, NY, USA), pp. 369–374, ACM Press, 2005.

[33] M. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Ho, m Lee, P. Johnson, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Frank, S. Amarasinghe, and A. Agarwal, "The raw microprocessor: A computational fabric for software circuits and general purpose programs," 2002.

[34] T. Y. Morad, U. C. Weiser, A. Kolodny, M. Valero, and E. Ayguade, "Performance, power efficiency and scalability fo asymmetric cluster chip multiprocessors," in *CCIT Technical Report 514*, Technion, 2005.