An Energy Aware Model of Computation

Brad D. Bingham and Mark R. Greenstreet

Department of Computer Science,
University of British Columbia,
{binghamb, mrg}@cs.ubc.ca

1. POSITION STATEMENT

The design and analysis of algorithms requires a model of computation. Such a model should faithfully reflect the physical processes of computation so that a programmer can distinguish efficient computations from inefficient ones. At the same time, the model must be simple enough to be tractable and general enough to continue to apply as the underlying technologies evolve. From a computer architect's perspective, a model of computation describes what the programmer expects, and thereby provides criteria for evaluating architectural alternatives.

The models of computation that are prevalent today are based on operation counting assuming sequential program execution. These models reflect the technology of the first several decades of computing: memory accesses were as fast as ALUs so operation count determined execution time; gates were expensive but wires were cheap; the monetary cost of computing was determined by the hardware rather than the power bill. Over time, each of these assumptions have been overturned, and yet the models of computation have remained remarkably stable. This has largely been made practical through innovations in computer architecture; for example, caches and superscalar execution have hidden the cost of memory accesses. Now, the "power wall" is forcing a transition to explicitly parallel architectures and software, and traditional models of computation no longer reflect the actual costs of computation.

Parallel computing offers a way around the power wall because CMOS technology allows operations to be performed with less energy by using more time. Thus, a parallel algorithm may perform more operations than its sequential counterpart, yet use less time *and* less energy. By combining voltage scaling, circuit design techniques and micro-architectural trade-offs, energy and time can be traded over ranges of several orders of magnitude. When these energy-time trade-offs are considered, the optimal algorithm for a task may be one that neither minimizes operation count nor computation depth. While various models have been proposed for parallel computation such as PRAMs [FW78] and logP [CKP$^+$93], we are aware of no prior model that can address the questions that arise from the energy-time trade-offs that are at the heart of current parallel computing technologies. For example,

- How can a programmer design energy-time optimal algorithms?
- Are there fundamental advantages to heterogeneous architectures (e.g. mixing fast and slow cores)?
- What on-chip interconnect topologies are required to realize energy-time optimal computations?
- What are the trade-offs between latency, throughput and power consumption?

Addressing these questions requires an energy-aware model.

The energy-time trade-offs afforded by CMOS technology have been long understood [HM72, DG$^+$74] and studied intensively by the real-time systems community since the seminal paper by Yao *et al.* [YDS95]. This has led to many papers that examine energy trade-offs for scheduling problems for uniprocessors, multiprocessors, with or without precedence constraints, etc. (e.g. [BKP04, CKL05, Bun06, AMS07]). In all of these papers, the set of tasks to be performed is taken as a given. Likewise, Martin [Mar01] considers the energy-time trade-offs of various concurrent decompositions for algorithms, but does not examine the energy requirements of particular computational tasks such as addition or sorting.

Position Statement: *To make effective use of parallelism, programmers and architects must have a model of computation that reflects the energy-time trade-offs of CMOS technology.* We are aware of no prior research that has presented a model that reflects these trade-offs. In the remainder of this paper, we present such a model; we use the model to derive lower bounds for the complexity of addition, multiplication and sorting; we show existing algorithms that meet these bounds to within constant factors, and we consider future directions for energy aware algorithm design and analysis. We present some "surprising" results for energy-constrained, minimal time computation. For example, if the inputs of a sorting network are required to lie along a line, then "slow" algorithms such as bubble-sort and "fast" algorithms such as odd-even merge-sort have the same asymptotic energy-time complexity. Our construction for an optimal adder shows that broadcasting a bit to all $O(d^2)$ mesh locations within distance d of a source uses the same energy and time (to within a constant factor) as sending the bit to a single location distance d away. The algorithms that we present for sorting and multiplication are energy-time optimal, yet they minimize neither operation count nor computation depth.

2. AN ENERGY AWARE MODEL

We present two variants of our model: a fairly abstract lower-bound model that simplifies analysis, and a more detailed upper-bound that ensures that the proposed algorithms can be implemented in VLSI technology. In both models, computations are performed by an ensemble of *processing elements* (PEs). A PE has $O(1)$ inputs, $O(1)$ outputs and $O(1)$ bits of state. A PE can compute an arbitrary function of its inputs and state in t units of time using $t^{-\alpha}$ units of energy; in

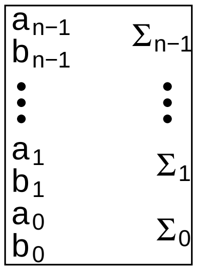

Figure 1: A Perimeter-I/O Implementation of an Adder

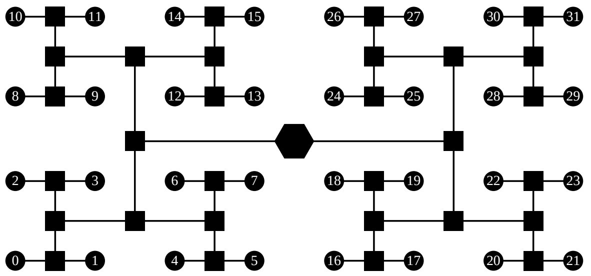

Figure 2: An H-Tree Implementation of an Adder

other words, et^α is constant ($\alpha > 0$). Computations are represented by task graphs with vertices of the task graph mapped to PEs, where vertices represent atomic operations. This is a many-to-one mapping: a PE may perform multiple operations in an algorithm as long as the precedence constraints ensure that these operations are performed during disjoint time intervals.

To model the planar geometry of integrated circuits and printed circuit boards, the lower bound model restricts a PE to having at most d^2 other PEs within distance d of itself, and the triangle inequality applies. To communicate a bit between PEs separated by distance d using time t requires $d^{\alpha+1}t^{-\alpha}$ energy; in other words, $et^\alpha = d^{\alpha+1}$, which means that one bit can be sent d units of distance using d units of time and energy.

In the upper bound model, each PE occupies a unit square, and only nearest neighbor communication is allowed. Thus, wires are realized as chains of PEs. Cross-overs, corners, etc., can be realized by a PE with multiple inputs and multiple outputs.

Further details for these models are presented in [BG08] and omitted here due to space constraints.

3. PRELIMINARY RESULTS

We start with addition. A binary adder has two n-bit input words and one $(n+1)$-bit output word that encodes the sum. At the beginning of the computation, each input bit is initially held by a distinct PE at a predetermined location in the plane. At the end of the computation, each output bit is held by a PE at a predetermined location.

We first consider adders where the centers of the the input PEs lie along a line and likewise for the centers of the output PEs. Figure 1 depicts one such adder. We call such layouts *perimeter-I/O*, and note that such implementations are common in practice. Because the carry generated by the least-significant bit can affect all bits of the result, we can use an adder to broadcast one input bit to all of the output bits. There must be two output PEs separated by distance n because the output PEs lie along a line. By the triangle inequality, there must be an output PE that is at least distance $n/2$ away from the PE holding for the least significant bit (lsb) of one of the operand words. Sending one bit distance $d/2$ requires energy e and time t with $et^\alpha \geq (d/2)^{\alpha+1}$. Letting E denote the total energy required to perform the addition and T denote the time, we conclude that the ET^α complexity for perimeter-I/O implementations of addition is in $\Omega(n^{\alpha+1})$. This bound is tight; it is readily achieved by a carry-ripple adder or a Brent-Kung carry-lookahead adder[1] [BK82]. Observing that a carry-ripple and a carry-lookahead adder have the same asymptotic complexity in our model, we see that when communication time and energy are taken into account, the $n/\log n$ time advantage of a "fast" adder is reduced to a constant factor.

More efficient adders can be achieved by allowing the input and output PEs to occupy arbitrary locations in the plane. We will refer to such implementations as having *planar-I/O*. Following an argument analogous to that for the perimeter-I/O adder, an adder with planar-I/O must have some output PE that is at least distance $\sqrt{n}$ away from the PE for the lsb of one of the operand words. The ET^α cost of sending the input bit to that output PE is in $\Omega((\sqrt{n})^{\alpha+1}) = \Omega(n^{(\alpha+1)/2})$.

To obtain the matching upper bound, we consider a carry-look-ahead adder, organized as an H-tree as shown in Figure 2. Let the root of the tree be at level 0, and the leaves at level m. For $k > 0$, an edge between levels k and $k-1$ is implemented by a chain of length $2^{\lfloor (m-k)/2 \rfloor} - 1$ PEs. We set the time for PEs at level k and the PEs in the chain from level k to level $k+1$ to $2^{k/(\alpha+1)}$. Grinding out the sums yields $E, T \in O(\sqrt{n})$ and thus $ET^\alpha \in O(n^{(\alpha+1)/2})$ matching the lower bound.

The lower and upper bound results together show that it is possible to broadcast a bit to all d^2 PEs within distance d of a source for a constant factor of the cost of sending a bit to a single PE at distance d. It is straightforward to show a complexity for addition of $ET^\alpha \in \Theta(n^{(\alpha+2)/2})$ if all PEs are required to operate with the same computation time. Thus, for addition, a heterogeneous architecture is asymptotically superior to an homogeneous one. We note that the communication pattern for addition is the same as that for the "map-reduce dwarf" [ABC$^+$06]. This suggests that for some computations, heterogeneity may provide a substantial computational advantage.

The multiplication problem can be formulated in a similar manner as addition. We have shown [BG08] that perimeter-I/O multipliers have an ET^α complexity in $O(n^{\alpha+2})$, and we focus here on the planar-I/O case. Following the classical arguments for the AT^2 complexity of multiplication [BK81], shifting can be reduced to multiplication. For each input bit there are at least $3n/4$ output bits that are at least distance $\sqrt{n}/2$ away. A pigeon-hole argument shows that there must be a shift for which at least $3n/4$ input bits must be sent a distance of at least $\sqrt{n}/2$. This yields a lower bound of $ET^\alpha = (3n/4)(\sqrt{n}/2)^{\alpha+1} \in \Omega(n^{(\alpha+3)/2})$. This bound is achieved by Preparata's AT^2 optimal multiplier [Pre83] as viewed in our

[1]Voltage scaling, or heterogeneity of PEs is a requirement for the Brent-Kung design to meet this bound. This is due to the few long wires per stage that must run at a faster speed.

Problem	Algorithm/Implementation	ET^α
addition	carry-ripple (P)	$O(n^{\alpha+1})$
	Brent-Kung (P)	$O(n^{\alpha+1})$
	Kogge-Stone (P)	$O(n^{\alpha+2})$
	Carry-Select	$O(n^{(\alpha+2)/2})$
	H-Tree	$O(n^{(\alpha+1)/2})$
multiplication	carry-save (P)	$O(n^{\alpha+2})$
	Preparata's	$O(n^{(\alpha+3)/2})$
sorting	bubble-sort (P)	$O(n^{\alpha+2})$
	odd-even merge-sort (P)	$O(n^{\alpha+2})$
	Schnorr and Shamir	$O(n^{(\alpha+3)/2})$

Algorithms marked (P) are implemented with perimeter I/O. The other algorithms have input and output PEs placed throughout the implementation.

Table 1: Summary of ET^α complexities.

model. We discuss the relationship between the AT^2 and our model in more detail in section 4.

We now consider the problem of sorting n binary words of w bits each. As for multiplication and addition, we assume that each input bit is stored in a separate PE at a predetermined location at the beginning of the algorithm, and that at the end, each output bit is stored in a predetermined PE. For sorting, we also assume that the bits for any given input or output word are stored in contiguous PEs. If $w \geq \log_2 n$, then we can construct a permutation of the input data that forces the i^{th} input word to go to PEs that are distance $\Omega(\sqrt{(n-i)w})$ away. This leads to a lower bound of $ET^\alpha \in \Omega((nw)^{(\alpha+3)/2})$. A matching upper bound is achieved by a variation on Schnorr and Shamir's mesh sorting algorithm [SS86]. Schnorr and Shamir's algorithm sorts n words on an array of $\sqrt{n} \times \sqrt{n}$ word-wise compare-and-swap modules in $O(\sqrt{n})$ time using only nearest neighbor communication. Because their compare-and-swap module can work on either vertically or horizontally adjacent words in unit time, a w-bit version must have w horizontal and w vertical wires and occupy area $O(w^2)$. This prevents their algorithm from achieving the $\Omega((nw)^{(\alpha+3)/2})$ lower bound we derived above. We replace each compare-and-swap module with a tile of $w \times w$ PEs and only use nearest neighbor communication. Our implementation takes $O(\sqrt{nw})$ time, and each step requires $O(nw)$ energy. This establishes $ET^\alpha \in O((nw)^{(\alpha+3)/2})$ and matches the lower bound.

Table 1 summarizes ET^α complexities that we have derived for the problems of addition, multiplication and sorting. Due to length limitations, we have not presented all of these algorithms here: details are given in [BG08].

4. WHAT'S NEXT?

We have presented a simple model that accounts for energy-time trade-offs in computation and used it to analyze addition, multiplication and sorting. We note that all three of these problems are highly amenable for parallel implementations. For example, addition with an ET^α complexity in $n^{(\alpha+1)/2}$ can have both the time and the energy for the computation grow slower than the input size as both can grow as $\sqrt{n}$. Likewise, for $\alpha > 1$, both multiplication and sorting can have energy and time both grow at a rate that is sublinear in the problem size.

Our model is reminiscent of the AT^2 models for VLSI complexity that were studied in the 1980s [Tho80, BK81, BK82, Pre83, SS86]. In fact, if an algorithm uses only nearest neighbor communication and achieves an $AT^\beta \in O(f(n))$, then it achieves an $ET^{\beta-1} \in O(f(n))$ as well (if easily adapted to our model). The two models, however, are not equivalent. Notice that there is no sense of trading area for time at the PE level, where as our model supports energy-time exchanges to vary both spatially and temporally across otherwise identical PEs. AT^β bounds are typically based on cross-sectional bandwidth requirements, while ET^α bounds are based on the speed (time/distance) that data must move. This has significant implications.

For example, if we consider throughput instead of latency, we can examine AP^β and EP^β where P is the *period* of the computation. We note that AP^β bounds generally match their AT^β counterparts because increasing latency with constant throughput increases both the amount of data to move and the amount of time to move it – the required bandwidth is unchanged. Thus AT^2 bounds cannot model the advantages of pipelining. EP^α bounds, on the other hand, can be lower than their ET^α counterparts because data can move slower if it has more time to reach its destination. For example, a pipelined, add-pass multiplier achieves an $EP^\alpha \in O(n^2)$ even with perimeter-I/O, which is lower than the $O(n^{(\alpha+3)/2})$ bound for the ET^α complexity.

The trade-offs between energy, throughput and latency merit further investigation. As sketched above, a deeply pipelined multiplier can use less energy per operation than a low-latency design for the same throughput. Many, but not all, numerical algorithms are highly tolerant of latency in the floating-point unit. Can we exploit this by building chips with many deeply pipelined floating point units and a few low-latency ones? Likewise, we have shown that the minimum ET^α complexities for addition, multiplication and sorting require planar-I/O, but the observation about multiplication suggests that perimeter-I/O implementations may be adequately (optimally?) efficient in an EP^α model. This would be good news for architects; energy-time optimal implementations may be possible where the functional units for primitive operations are implemented with perimeter-I/O while meshes of cores may provide planar-I/O for larger algorithms. Furthermore, by distinguishing latency and throughput, we can also consider streaming computations. A model like the one we have presented here could help architects and algorithm designers explore trade-offs such as these.

We have presented our model with PEs defined at the bit-level to ensure that our model accounts for all costs in an implementation of an algorithm. However, this approach has the side-effect of forcing all analysis to the bit-level. While we have established bounds for the three problems that we considered, establishing lower bounds for bit-level complexity is known to be a hard problem. Thus, we plan to extend our model to allow PEs that perform simple operations on words. Our analysis of addition and multiplication provides a basis for such an extension. With a word-based model, we plan to examine common numerical tasks such as matrix multiplication and solving linear systems as well as more combinatorial problems such as graph algorithms and search problems.

Power dissipation is the most critical bottleneck for computer system performance. Existing algorithm design and analysis is based on obsolete models that ignore the underlying trade-offs between energy and time. Thus, new models are required before we can even describe the trade-offs involved in designing and analysing algorithms for power-constrained computations. We have presented a simple model for this analysis and demonstrated it by analyzing addition, multiplication and sorting. While we have no doubt that further work will be required to create a model that has the right balance of physical realism and mathematical simplicity, we believe that our approach points in the direction that is required for such research.

References

[ABC+06] K. Asanovic, R. Bodik, B.C. Catanzaro, et al. The landscape of parallel computing research: A view from Berkeley. Technical Report EECS-2006-183, Department of Electrical Engineering and Computer Science, University of California, Berkeley, December 2006.

[AMS07] Susanne Albers, Fabian Müller, and Swen Schmelzer. Speed scaling on parallel processors. In *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, pages 289–298, New York, NY, USA, 2007. ACM.

[BG08] Brad D. Bingham and Mark R. Greenstreet. Computation with energy-time trade-offs: Models, algorithms and lower-bounds. Technical Report TR-2008-03, Department of Computer Science, University of British Columbia, March 2008.

[BK81] R. P. Brent and H. T. Kung. The area-time complexity of binary multiplication. *J. ACM*, 28(3):521–534, 1981.

[BK82] R. Brent and H.T. Kung. A regular layout for parallel adders. *IEEE Transactions on Computers*, C-31(3):260–264, March 1982.

[BKP04] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Dynamic speed scaling to manage energy and temperature. In *Proceedings of the* 45th *Annual IEEE Symposium on the Foundatations of Computer Science (FOCS'04)*, pages 520–529, 2004.

[Bun06] David P. Bunde. Power-aware scheduling for makespan and flow. In *Proceedings of the* 18th *ACM Symposium on Parallel Algorithms and Architectures (SPAA'06)*, pages 190–196, 2006.

[CKL05] Jian-Jia Chen, Tei-Wei Kuo, and Hsueh-I Lu. Power-saving scheduling for weakly dynamic voltage scaling devices. In *WADS*, pages 338–349, 2005.

[CKP+93] David E. Culler, Richard M. Karp, David A. Patterson, Abhijit Sahay, Klaus E. Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. LogP: Towards a realistic model of parallel computation. In *Principles Practice of Parallel Programming*, pages 1–12, 1993.

[DG+74] R.H. Dennard, F.H. Gaensslen, et al. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuts*, 9(5), October 1974.

[FW78] Steve Fortune and James Wyllie. Parallelism in random access machines. In *Proceedings of the* 10th *Annual ACM symposium on the Theory of Computing (STOC'78)*, pages 114–118, 1978.

[HM72] Bruce Hoeneisen and Carver A. Mead. Fundamental limits in microelectronics I: MOS technology. *Solid-State Electronics*, 15:819–829, 1972.

[Mar01] Alain J. Martin. Towards an energy complexity of computation. *Information Processing Letters*, 77:181–187, 2001.

[Pre83] Franco P. Preparata. A mesh-connected area-time optimal VLSI multiplier of large integers. *IEEE Trans. Computers*, 32(2):194–198, 1983.

[SS86] C.P. Schnorr and A. Shamir. An optimal sorting algorithm for mesh connected computers. In *Proceedings of the* 18th *ACM Symposium on the Theory of Computing (STOC18)*, pages 255–263, May 1986.

[Tho80] Clark David Thompson. *A complexity theory for VLSI*. PhD thesis, Carnegie Mellon University, 1980.

[YDS95] Frances Yao, Alan Demers, and Scott Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the* 36th *Annual Symposium on the Foundations of Computer Science*, pages 374–382, October 1995.