Location, Location, Location—The Role of Spatial Locality in Asymptotic Energy Minimization

André DeHon
Department of Electrical and Systems Engineering
University of Pennsylvania
200 S. 33rd St., Philadelphia, PA 19104
andre@acm.org

ABSTRACT

Locality exploitation is essential to asymptotic energy minimization for gate array netlist evaluation. Naive implementations that ignore locality, including flat crossbars and simple processors based on monolithic memories, can require $O(N^2)$ energy for an N node graph. Specifically, it is important to exploit locality (1) to reduce the size of the description of the graph, (2) to reduce data movement, and (3) to reduce instruction movement. FPGAs exploit all three. FPGAs with a Rent Exponent $p < 0.5$ running designs with $p < 0.5$ achieve asymptotically optimal $\Theta(N)$ energy. FPGA designs with $p > 0.5$ and implementations with metal layers that grow as $O(N^{p-0.5})$ require only $O(N^{p+0.5})$ energy; this bound can be achieved with $O(1)$ metal layers with a novel multicontext design that has heterogeneous context depth. In contrast, a $p > 0.5$ FPGA design on an implementation technology with $O(1)$ metal layers requires $O(N^{2p})$ energy.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Type and Design Styles—*VLSI*; C.2.1 [**Computer Communication Newtorks**]: Network Architecture and Design; C.1.3 [**Processor Architectures**]: Other Architecture Styles—*Adapative Architextures*

General Terms

Theory, Design

Keywords

VLSI Theory, Energy, Energy Complexity, Low Power, FPGA, Rent's Rule, Locality, Multicontext

1. INTRODUCTION

Energy is now the dominant concern for many applications and systems. This shows up directly as hours of operation for battery powered devices and indirectly as power density limits for air-cooled laptops and servers. Consequently, considerable research over the last decade has focused on energy reduction at all levels in FPGA design (CAD, microarchitecture, circuit, technology). Microarchitecture studies (*e.g.* [23, 21]) ask detailed questions about parameters in a basic FPGA design such as how long interconnect segments should be, how switchboxes should be populated, how many input a LUT should have, or how many LUTs should be in a cluster. Here we address a broader architecture question: is the basic architectural organization of configured gates and wires energy efficient? or would it be more energy efficient to sequentially evaluate gates as instructions on a processor? is there an inherent energy advantage or disadvantage to multicontext FPGAs?

Since this is a broad question, we stick with an asymptotic energy analysis, ignoring constants. The asymptotic picture is necessarily crude, but it has the advantage of being independent of technologies and a host of implementation assumptions that can obfuscate or, if chosen poorly, invalidate a comparison. The comparison we make is in the spirit of VLSI complexity theory, but focuses on architecturally determined energy complexity rather than application-driven Area-Time complexity (See Secs. 2.3 and 10 for details).

We start by defining a gate-array evaluation model that defines the computation required to evaluate all the gates in a netlist and a VLSI model for the area and energy in the physical substrate (Sec. 3). We consider strawman implementations on spatial crossbars and sequential processors with monolithic memories (Sec. 4) as a baseline.

From this starting point, we derive the need to exploit locality in the problem and use Rent's Rule [19] to characterize the locality in a design. We successively see that there is an asymptotic energy advantage to exploiting locality in:

- *descriptions*—referring to gates with fewer bits when they are "closer" to a consumer rather than using a fixed number of bits to refer to all gates–Sec. 5
- *data movement*—laying out data elements to minimize the distance we need to bring the inputs to a gate together for evaluation; this leads us to give each gate a location and move data to it rather than storing all gate values in a common memory–Sec. 6
- *instruction movement*—placing instructions adjacent to the resources they control, particularly interconnect, rather than centrally or with the gates–Sec. 7

Combining these localities gives us the typical FPGA organization. Despite the fact that FPGA organization may be physically larger than a processor with a dense memory or an ASIC, it is asymptotically optimal ($\Theta(N)$ energy to evaluate N gates) when locality is high (Rent's Rule $p < 0.5$).

Lack of locality is a common challenge for spatial designs and is a key driver in the AT^2 VLSI complexity bounds. For these less local designs ($p > 0.5$), when limited to a constant number of metal layers, traditional FPGA designs require $O(N^{2p})$ energy. We introduce architectures that constructively achieve energy as low as $O(N^{p+0.5})$, including a sequential design and a novel multicontext design. The multicontext design differs from prior proposal for multicontext FPGAs (*e.g.*, [6, 27]) in that it uses a different context depth at different levels in its hierarchical interconnect—a depth driven both by the Rent's Rule demand for interconnect and the availability of wire tracks when limited to a constant number of metal layers (Sec. 7.3). Furthermore, if our technology allows us to add wire layers fast enough, we show that FPGA-like designs can achieve the $O(N^{p+0.5})$ energy bound as well (Sec. 8).

2. BACKGROUND

2.1 Energy Matters

We have now hit the point where power density limits (*e.g.* 100W/cm^2 for force-air cooling, 1–10W/cm^2 for ambient cooling), not transistor integration density, is the key limitation for computations [12]. That is, we cannot afford to turn on all the transistors that we can integrate onto an integrated circuit [15], a phenomenon now termed "Dark Silicon" [29, 10]. This power density limit has driven the end of microprocessor clock scaling [12], and it may drive the end of useful VLSI feature size scaling [11]. *As energy reduction from scaling diminishes, architectural solutions that reduce energy grow in importance.* Even when power does not limit what designers can do, energy consumption is a primary concern for battery life and operating costs.

2.2 Energy Efficiency Hints

A wealth of papers at this conference and the International Symposium on Field-Programmable Custom Computing Machines regularly show FPGA designs that require substantially less energy than processor designs. In one of the early, direct comparisons Budiu maps computational benchmarks to spatial designs and shows order of magnitude lower energy than sequential processor designs [4]. Stitt showed that kernels could be moved from a processor to an attached FPGA co-processor to reduce energy [25]. More recently, Venkatesh shows that custom spatial implementation of basic blocks saves considerable energy compared to sequential implementation on general-purpose processors [29]. Dally shows that the dominant energy in low power embedded processor is in instruction and data memories and this can be reduced by using an array of processors with shallower memories [5]. Trimberger's original paper on multicontext FPGAs [27] noted the high power requirement of instruction memory reads as a potential drawback.

All of this work suggests that spatial designs save energy compared to sequential designs since they avoid the need to read data and instructions from memories. However, we also know that computations on FPGAs can be less compact than processors and, consequently, must pay energy to send bits over longer physical distances than processor. The examples above suggest that the savings in memory energy is more substantial than the cost in interconnect energy. It is therefore desirable to understand if there are fundamental effects that drive the previous observations. While there are many effects at play that are beyond the scope of this paper (See Sec. 11), our results begin to show that there are asymptotic effects that favor spatial implementations as well as showing conditions under which hybrid designs may be better than either spatial or sequential designs.

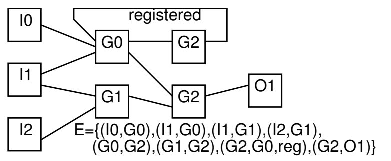

registered
I0
I1
I2
G0
G2
G1
G2
O1
E={(I0,G0),(I1,G0),(I1,G1),(I2,G1),
(G0,G2),(G1,G2),(G2,G0,reg),(G2,O1)}

Figure 2: Example Gate Array Netlist

2.3 VLSI Theory

There is a well-developed theory of VLSI complexity (*e.g.*, [26, 24, 20]). Notably, by modeling the finite width of wires and focusing on graph cut sizes in algorithms, it has been possible to obtain non-trivial bounds on the area of a VLSI design with a particular level of parallelism [2]. For designs with high communication requirements, and hence cut widths, such as sorting and FFT, this led to AT^2 bounds. This says that to run these problems twice as fast, it will require 4 times the area. This area expansion comes not from the need to place more processing elements, but from the need to layout more wiring. Our results in this paper take careful account of wiring and layout to identify constructive organizations and achievable energy bounds.

The energy complexity of VLSI has received less attention than area and time, and most of the VLSI complexity results ignore the area and delay impacts of memory. We believe this is the first work to establish energy complexity results for memories. Section 10 characterizes work on VLSI energy complexity and relates it to the result we develop here.

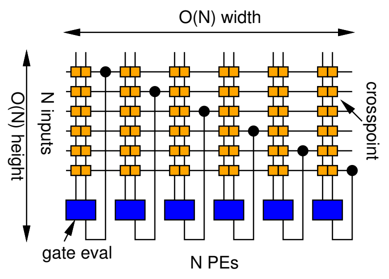
3. MODELS

3.1 Gate Array Evaluation Model

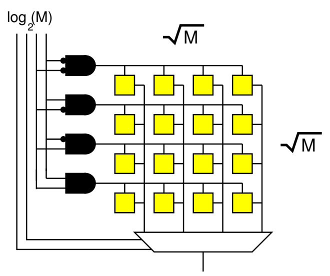
To model a typical FPGA circuit, we use a gate array netlist model. Informally, this is a graph where every node has bounded fanin k and evaluates once per evaluation cycles (*e.g.* Fig. 2). k can be taken as the number of inputs to the LUTs in the FPGA. We take k to be a small constant and make no attempt to differentiate among different choices for k. We characterize a graph by the number of nodes $N = \mid V \mid$. Since k is bounded, the number of edges in the graph $\mid E \mid \leq kN = O(N)$.

Note that one model simplification here is that every edge is assumed to switch once per evaluation. This is essentially making a worst-case assumption about switching. Said another way, we assume homogeneous activity on all edges in the graph. That is, the asymptotic results are unchanged if we say every edge switches 10% of the time. Consequently, we will not specifically reason about variable activity on the netlist graph. Prior work has addressed heterogeneous net activity (See Sec. 10), and treatment of heterogeneous net activity for this model is an important direction to extend our results (Sec. 11).

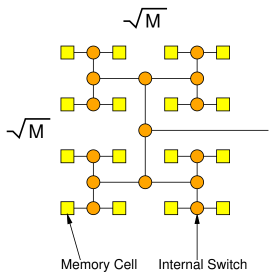
Since we assume k is a constant, every node is of size $O(1)$ and will require $O(1)$ energy per cycle simply to perform the



(a) Crossbar Implementation Architecture



(b) Monolithic Memory



(c) Fully Banked Memory

Figure 1: Crossbar and Memory Organizations

gate evaluation. Since we assume every gate evaluates and potentially switches its output on every cycle, we must, at least, pay energy for the N gates for each evaluation of the netlist. *This means $\Omega(N)$ is a lower bound on the energy that will be required to evaluate the netlist.*

3.2 VLSI Area and Energy Model

We focus on the energy of operations. However, since the energy for a physical wire will depend on the length of the wire, it is also important that we carefully account for area and lengths in the implementation. All gates used in the implementations have $O(1)$ inputs, and hence $O(1)$ area and width. Each wire has $O(1)$ width, and we initially assume $O(1)$ wire layers, so that it takes $O(w)$ width to layout w wires leaving a region of the chip. Energy per fanin, gate switch, or unit length of wire is $O(1)$. Consequently, energy of a physical net is proportional to its total wire length.

3.3 Asynchronous Evaluation

In a combinational circuit implementation, glitches might make the activity on a node greater than 1.0 [18]. That is, the inputs to a gate might change at different times causing the output to switch multiple times per circuit evaluation. As a partial justification of the homogeneous switching model, we note that an asynchronous implementation of the gate that performs a handshake with the inputs can guarantee $O(1)$ switching without changing the asymptotic size of the implementation or number of switching events. If CAD and tuning (*e.g.* [18]) can adequately control glitching, the asynchronous assumption is not necessary to achieve the bounds derived for the fully spatial designs (including traditional FPGAs) (Sec. 7.2 and 8). However, as we will see starting in Sec. 4.2.3, the asynchronous handshaking is necessary to achieving the tightest bounds derived in this paper for the sequential and multicontext designs.

4. SIMPLE STARTING POINTS

We start by analyzing the simplest, most direct architectures for implementing the gate array evaluation model (Sec. 3.1). We see that these are much more expensive than the $\Omega(N)$ lower bound.

4.1 Full Crossbar

For a full crossbar implementation of the gate array evaluation model, we arrange the gates in a line of size N at the outputs of the crossbar (See Fig. 1(a)). Since each gate has at most k inputs, there are at most kN outputs from the crossbar feeding into the gates. Except for the circuit outputs, each gate output is fed back into the crossbar. As a result, the crossbar has N inputs by kN outputs, for a total area that is $O(N^2)$. Significantly, every output drives a wire of length $O(N)$ and every input is of length $O(N)$. A memory cell at each crossbar crosspoint can hold its configuration without changing the asymptotic size of the crossbar. The energy of evaluation for each gate is $O(N)$ to drive its output, plus $O(1)$ for the gate evaluation, plus $O(N)$ for its k inputs to be driven for a total energy of $O(N)$. Evaluating N gates means a total of $O(N^2)$ energy—a factor of N larger than the lower bound of $\Omega(N)$.

4.2 Memory

The crossbar is known to be area expensive because we dedicate a physical connection for every edge in the graph and a physical gate for every vertex. We should be able to store the state in the graph and the description of the edges more compactly in memories. In particular, the vertex outputs require at most $O(N)$ memory locations. If we give each of the N vertices a $\log_2(N)$ bit address, we can describe the connectivity in the graph with $O(N\log(N))$ bits of memory. In order to develop models for architectures that exploit memories, in this section, we characterize the asymptotic energy requirements of memories.

4.2.1 Monolithic

The simplest case is a monolithic memory bank where we store M bits in one large array. We arrange the M bits into a $\sqrt{M}\times\sqrt{M}$ array and use $\log_2(M)$ bits to specify the bit in the array (Fig. 1(b)). This address is broken in half with $\log_2(M)/2$ bits specifying the row address and $\log_2(M)/2$ bits specifying the column address. This organization allows each memory bit to take up $O(1)$ area in the middle of the array. Each memory bit also contributes $O(1)$ capacitance, and hence energy, to the row select and column read lines. Reading out of the core of the memory takes $O(\sqrt{M})$ energy to drive the single activated row select line and $O(\sqrt{M})$ to drive each column read line. Since there are $O(\sqrt{M})$ column read lines, this is a total of $O(M)$ energy. Selecting the row line demands driving $O(\log(M))$ lines of length $O(\sqrt{M})$ for $O(\log(M)\sqrt{M})$ and $O(\log(M)\sqrt{M})$ energy in gates, asymptotically less energy than the $O(M)$ energy in the core. Performing the final bit selection from the $O(\sqrt{M})$ column results takes $O(\sqrt{M})$ energy for a mux tree reduction and less than $O(\log(M)\sqrt{M})$ to connect the column address lines to the muxes; both of these are also asymptotically less than the $O(M)$ energy in the core. Con-

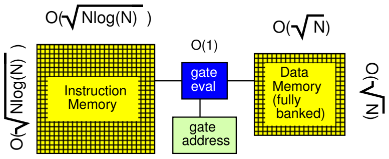


Figure 3: Processor with Memory Organization

sequently, a memory read takes $O(M)$ energy to obtain each randomly accessed bit.

4.2.2 *Sequential Access*

Accessing a single bit is potentially wasteful since we pay energy to bring $\sqrt{M}$ bits out of the array core, but only use one. If we can arrange to read the memory sequentially, we can amortize the energy reading from the array core across all the $\sqrt{M}$ bits read. In particular, instead of using a multiplexer to select the column, we read the column bits into a shift register. We then shift the bits serially out of the shift register. For this case, we perform one read from the core that still takes $O(M)$ energy. For each of the $\sqrt{M}$ bits read, we shift the shift register for a cost of $O(\sqrt{M})$. After $\sqrt{M}$ shifts, we have spent $\sqrt{M} \times O(\sqrt{M}) = O(M)$ energy on the shifts and $O(M)$ energy on the read from the core or a total of $O(M)$ energy. Each bit thus costs us $O(\sqrt{M})$ energy.

Sequential access is an important component of "spatial locality" as the term is used for processor caches. Otherwise, the processor cache use of "spatial locality" is, at best, loosely related to the physical spatial locality developed in the rest of this paper (Secs. 5–7).

4.2.3 *Fully Banked*

Random access into the monolithic memory (Sec. 4.2.1) suffers because we must activate the entire memory. We can reduce the energy by breaking the memory into separate banks and only activating the bank that stores the data being addressed. For example, simply breaking the memory into two banks, means we activate a memory core of half the size and hence half the energy. We do have to pay some energy controlling bank selection, but that is on the order of the address selection $O(\log(M)\sqrt{M})$, which is small compared to compared to the energy of the cores $O(M)$. To get an asymptotic benefit, we recursively subdivide the memory banks, building a tree-based memory access.

Constructively, we layout the fully banked memory as an H-tree (Fig. 1(c)). The leaves of the H-tree are the memory cells of size $O(1)$. Each internal tree node in the H-tree is of size $O(1)$ and serves to route addresses down to the leaves and route results out of the H-tree. The entire H-tree is of size $O(M)$, since the number of internal nodes is also $O(M)$ and the wiring only spreads the leaves and internal nodes by $O(1)$. We cannot afford to clock the memory access tree as that would take $O(M)$ energy per cycle. If we clocked operation at each level of the tree, that would mean $O(M\log(M))$ energy, which is greater than the monolithic memory. Instead, we use asynchronous handshaking at each tree node and are careful to only send address bits down the necessary branch of the tree. Addresses are fed in serially from the top. The first address bit is used by the root node and sets the tree node to send down the left or right branch as appropriate. The remaining address bits follow through the root, each setting the internal node one level down the tree. When the path reaches a leaf node, the value from the leaf is sent back up the tree. As a result, we send $\log(M)$ bits down through the top node, $\log(M)-1$ through the next, $\log(M)-2$ through the next, and eventually one bit to the final internal node above the leaf. Each of these $\log(M)$ active internal nodes then sees one bit coming back out. This means $O(\log^2(M))$ energy spent by the switching gates within the tree. However, we must also account for the energy on the wires. The wires at the top of the tree are of length $\sqrt{M}$. Roughly, the wires at the next level are $\sqrt{M}$ as well. At the following level, the wires have length$\sqrt{M}/2$.

$$
\begin{aligned}
E_{wire} &= \sqrt{M}\sum_{i=0}^{\log_2(M)}\left(i \times \frac{1}{2^{\lceil(\log_2(M)-i)/2\rceil}}\right) \\
&\leq \log_2(M)\sqrt{M}\sum_{i=0}^{\log_2(M)}\left(\frac{1}{2^{\lceil i/2\rceil}}\right) \\
&\leq \log_2(M)\sqrt{M}\sum_{i=0}^{\log_2(M)/2}\left(\frac{2}{2^{i}}\right) \qquad (1)
\end{aligned}
$$

The sum is a geometric series that converges to $O(1)$,[1] so we have:

$$E_{wire} \leq O(\log_2(M)\sqrt{M}) \qquad (2)$$

Wire energy asymptotically dominates gate energy, so the total energy for a bit read from this fully banked memory is $O(\log(M)\sqrt{M})$, which is asymptotically smaller than the bit read energy for a monolithic memory. Writes behave similarly and take the same asymptotic energy.

4.3 Processor with Memory

A simple, sequential case might store the graph description in one memory (instruction memory) and the data value of each gate in another (data memory) as shown in Fig. 3. The sequential processor would process the graph in topological order. For each node, it would:

1. read the gate description from the instruction memory $= 2^k$ bits $= O(1)$ bits since we take k to be a constant
2. read the address of each of the k inputs from the instruction memory; since there are $O(N)$ sources, this is $k\log(N)$ or $O(\log(N))$ bits
3. read the k input bits from the data memory
4. perform the gate evaluation
5. store the result into the current gate address in the data memory
6. increment the gate address

Each vertex requires $O(\log(N))$ bits to specify its inputs, so the total instruction memory holds $O(N\log(N))$ bits. The instruction memory is accessed sequentially, so it can use the more efficient sequential access pattern into memory (Sec. 4.2.2). The data memory only holds $O(N)$ bits, but these must be accessed randomly, so we use a fully banked memory (Sec. 4.2.3). The instruction memory reads (operations 1 and 2) take $O(\log(N)\sqrt{N\log(N)})$ energy. The data memory reads (operation 3) and write (operation 5) take $O(\log(N)\sqrt{N})$. The gate evaluation (operation 4) takes $O(1)$ energy, and the gate address increment (operation 6) takes $O(\log(N))$. Instruction memory dominates, such that

[1]Throughout, we make use of the relationship: $a\left(1+r+r^2+...\right) \leq \frac{a}{1-r}$ when $r<1$.

each memory read takes $O(\log^{1.5}(N)\sqrt{N})$. Processing the entire graph with N nodes takes $O((N\log(N))^{1.5})$.

If we had used a monolithic memory instead of the fully banked memory for the data memory, the data memory would have dominated with energy $O(N)$, for a total energy of $O(N^2)$—comparable to the crossbar.

5. DESCRIPTION LOCALITY

The analysis in the previous section assumes each input to a vertex may come from any other vertex. However, in typical circuits, we expect a certain amount of locality. Most inputs will come from a subset of the vertices that can be made close to the gate. In circuit design, including the design of FPGA architectures [7, 8, 16], it is typical to characterize this locality using Rent's Rule [19] which says the number of wires that exit a region containing N gates is proportional to a fractional power of the gates in the region:

$$IO = cN^p \tag{3}$$

This can be directly mapped to Leighton's α-bifurcator definition [2]. Both characterizations recursively bisect designs minimizing the cut size between regions and use the cut sizes out of each subregion to characterize the locality. In both cases, they suggest a geometric relationship on the cut sizes at successive tree levels in the recursive bisection.

For designs with locality where $p < 1$, not all edges are cut by the top bisection. In particular, only cN^p edges are cut. These edges cut by the top bisection will need all $\log_2(N)$ bits to describe their source. However, the other edges are contained in smaller subtrees and can use fewer bits. This can be used to show that the total number of bits needed to specify routing is $O(N)$ [8]. The number of bits required to specify an edge is proportional to the logarithm of the capacity of the smallest subtree that contains the edge. We can count the number of bits required by charging each edge for each subtree it must exit. That is, when a graph edge needs to cross out of the top of a tree at level i, we need one bit to specify which way the edge connects at that tree level. Thus we need:

$$N_{bits} = \sum_{i=0}^{\log_2(N)} \left(\frac{N}{2^i} \times c\left(2^i\right)^p\right) \tag{4}$$

The first term $\frac{N}{2^i}$ is the number of subtrees at height i from the leaf, while the second term is Rent's Rule (Eq. 3) applied to the size of the subtree (2^i). Pulling out c and N and combining the 2^i terms we get:

$$N_{bits} = cN \sum_{i=0}^{\log_2(N)} \left(\left(2^i\right)^{(p-1)}\right) \tag{5}$$

For $p = 1$, the term being summed is one, so N_{bits} becomes $O(N\log(N))$ as we saw in the previous section when we did not assume any locality. However, when $p < 1$, the exponent $p - 1$ is less than one, making the term being summed a fraction that decreases geometrically with i. As a result, the sum converges to $O(1)$, and we have N_{bits}=$O(N)$.

Impact on Processor Case If we exploit this locality when $p < 1$, the instruction memory in the processor (Sec. 4.3) only needs to be of size $O(N)$. This reduces the area for the processor to $O(N)$. Furthermore, the total bits read from the instruction memory will be $O(N)$ instead of $O(N\log(N))$. The total instruction memory energy reduces

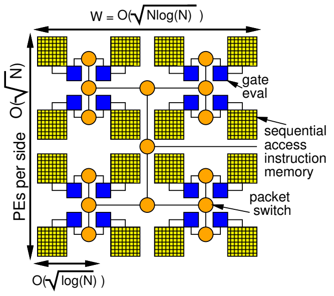


Figure 4: Sequential Communication Exploiting Data Locality

from $O((N\log(N))^{1.5})$ to $O(N^{1.5})$. The data memory energy of $O(\sqrt{N}\log(N))$ per gate means a total energy across all N gates of $O(N^{1.5}\log(N))$ which now dominates instruction energy and determines the asymptotic energy of operation. *Exploiting description locality saves us a factor of* $\sqrt{\log(N)}$.

6. DATA LOCALITY

After exploiting description locality, the dominant energy arises from accessing the data memory. Here, since we bring the data to a single location to evaluate the gate, we must pay $O(\sqrt{N}\log(N))$ energy for every data fetch from the banked memory. That is, we are **not** exploiting any locality in movement of the data.

Alternately, we can perform the gate evaluation at different places and arrange the data for minimal movement. Again, we consider performing the recursive bisection of the graph to minimize cut sizes for Rent's Rule. Each node lives at the leaf of a tree. We layout the tree as an H-Tree (Fig. 4). For this case, we additionally limit the fanout associated with a node to k (See Sec. 6.1). At the leaf associated with a node we include:

1. asynchronous logic to recognize when all k inputs have arrived = $O(1)$
2. storage space for the k inputs to each gate = $O(1)$
3. the description of the behavior of the gate = $O(1)$
4. memory to store the address of the k successors to the gate = $O(\log(N))$

Since each leaf node needs $O(\log(N))$ memory, the entire structure requires area $O(N\log(N))$. If we did not limit the fanout to a constant, we could not guarantee the node memories could be this small.

All links in the H-tree are $O(1)$ wide so that the H-tree has the same asymptotic area as the leaves. The internal nodes in the tree serve as a bit-serial, packet-switched network. This is similar to the fully banked memory, except that routing bits are needed both to route up the tree to the point of cross-over and back down. Since the path up the tree is the same length as the path back down, adding bits to route up the tree does not asymptotically change the number of bits needed to address a destination.

Each leaf node behaves as follows:

1. wait for all inputs to arrive
2. evaluate gate
3. send result bit to all $\leq k$ successor vertices; this also depends on the fanout limit.

Evaluation energy at the leaf nodes remains $O(1)$ per node or $O(N)$ total. We can sequentially access the successor gate address memory. We must read $O(\log(N))$ bits at energy $O(\sqrt{\log(N)})$ per read, for a total of $O(\log^{1.5}(N))$ energy per node or $O(N\log^{1.5}(N))$ energy to perform these reads from all nodes.[2] This leaves the energy required to route the data to the successors over the H-tree network. Here, we must account for the number of edges that must be routed to each height in the tree, the bits that specify the destination, and the lengths of the wires at each level in the tree.

- There are $\frac{N}{2^i}$ subtrees at height i from the leaf.
- By Rent's Rule, we know we have $c\left(2^i\right)^p$ edges that must cross out of each of those subtrees.
- The number of bits in an address will be less than $\log(N)$; for simplicity, we make no further attempt to account for the fact that many stages see fewer bits.
- The length of the top wire in the tree is $O(\sqrt{N\log(N)})$.
- Wire lengths halve every other stage as noted for the fully banked memory.

Putting this together, we get:

$$\begin{aligned} E_{comm} &\leq \sum_{i=0}^{\log_2(N)} \left(\frac{N}{2^i} \times c\left(2^i\right)^p \times \log(N) \right.\\ &\qquad \left. \times O\left(2^{\lceil i/2 \rceil}\sqrt{\log(N)}\right)\right) \\ &\leq O(N\log^{1.5}(N)) \sum_{i=0}^{\lceil \log_2(N)/2 \rceil} \left(\left(2^i\right)^{(2p-1)}\right) \end{aligned}$$

For $p=0.5$, the term in the sum is one, making the total:

$$E_{comm}(p=0.5) \leq O(N\log^{2.5}(N)) \qquad (6)$$

For $p>0.5$, the sum is largest at the maximum value of $i=\lceil \log_2(N)/2 \rceil$ where it evaluates to $O(N^{p-0.5})$. The other terms recede geometrically from the maximum value, so the entire sum comes to $O(N^{p-0.5})$. As a result, we have:

$$E_{comm}(p>0.5) \leq O(N^{p+0.5}\log^{1.5}(N)) \qquad (7)$$

At $p=1$, this is $O(\sqrt{\log(N)})$ larger than the memory case in Sec. 5 due to the area increase to hold $O(N\log(N))$ bits for the description; however, for any $p<1$, the benefit from locality of data movement is greater, and this is scheme has asymptotically lower energy. *Picking spatial locations for computations and moving the data minimally to these locations saves a factor of $O(N^{(1-p)}/\sqrt{\log(N)})$.* For $p<0.5$, the sum is largest at the minimum value of $i=0$ where it evaluates to one. Again, the other terms form a receding geometric sum as i increases, so the summation comes to $O(1)$. For $p<0.5$, we have:

$$E_{comm}(p<0.5) \leq O(N\log^{1.5}(N)) \qquad (8)$$

In all cases this data communication energy equals or dominates gate evaluation and memory read energy.

6.1 Fanout Limit

Imposing a fanout bound does not limit the netlists we can support nor change the asymptotic results. We can transform any netlist with bounded fanin and unbounded fanout into one with bounded fanout without asymptotically changing the number of nodes in the network or its depth [14]. Roughly, an N node graph with a fanin bound of k and unbounded fanout can be transformed into a graph with a bounded fanout of k that has no more than $2N$ nodes. Consequently, the asymptotic results here hold even if we think about starting with a netlist with unbounded fanout.

[2]Strictly speaking, by exploiting the description locality, we read a total of $O(N)$ successor address bits rather than $O(N\log(N))$ when $p<1$, so, the total energy is $O(N\sqrt{\log(N)})$ for $p<1$, but this doesn't change the total asymptotic result that follows since the data movement term dominates the asymptotic energy use.

7. INSTRUCTION LOCALITY

While data communication locality reduces the cost of communication, we spend more energy providing the address for routing the data than we do for actually sending the data. Furthermore, we are forced to give up the full area compactness of description locality. Instead of storing the location address bits at the leaves, we can save additional energy by storing the routing bits in the tree local to the switches that they control.

7.1 Sequential

Starting with the sequential case from the previous section, storing the configuration in the tree means we time-multiplex the switches rather than packet switching with route addresses stored at the leaves. Each internal switch at level i now has a memory of size $O\left(\left(2^i\right)^p\right)$ to tell it the sequence of configurations it must perform in order to route the data (See Fig. 5(a)). The switches remain asynchronous. Each switch reads $O(1)$ bits form the memory to tell it the next route operation to perform, waits for input to arrive on the specified source, handshakes with the source, switches the input data to the specified output, and handshakes with the destination LUT or switch. The Rent's Rule subtree IO model already captures fanout effects. That is, some switches may switch data out two sides, and this effect is already accounted for in the Rent IO of the subtrees to which they connect. Consequently, it is not necessary to assume bounded fanout as in Sec. 6 for the designs in this section and the next (Sec. 8). Each LUT at the leaf waits for all inputs to arrive, computes the result, and sends the result.

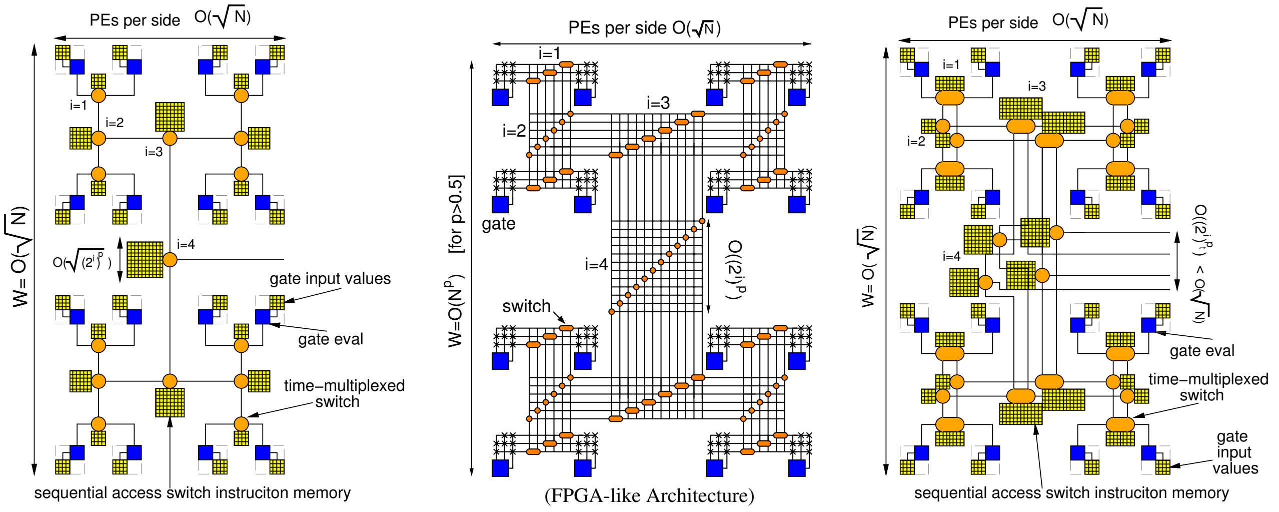
To achieve the $O(\sqrt{M})$ energy for reading from each of the switch memory banks, we must layout the switch memories as a square. This means that the width of the switch at tree level i becomes $O(\sqrt{(2^i)^p})$. For $i=\log_2(N)$, there is one switch width at the root of the tree. Every two tree levels, we double the number of switch widths we must accommodate across the width. Consequently, the width of the tree of capacity N becomes:

$$\begin{aligned} W &= \sum_{i=0}^{\log_2(N)/2} \left(\frac{\sqrt{N}}{2^i} \times O\left(\sqrt{(2^{2i})^p}\right)\right) \\ &= \sqrt{N} \sum_{i=0}^{\log_2(N)/2} \left(O\left(\left(2^i\right)^{p-1}\right)\right) \qquad (9) \end{aligned}$$

For $p<1$, this is a receding geometric sum, so we have W=$O(\sqrt{N})$. This tells us that the entire design requires only $A=W^2=O(N)$ area. It further tells us that the wire lengths for each subtree of capacity N are only $O(\sqrt{N})$, asymptotically the same as if we did not have memories embedded in the tree.

The energy of operation is now composed of:

- Energy per gate evaluation = $O(1)$ per gate.
- Energy for each communication link including the energy reading from the switch instruction memory. At



(a) Sequential Communication (b) Parallel Communication (c) Heterogeneous Multicontext

Figure 5: Instruction Locality Architectures

level i of the tree, we drive a wire of length $O(\sqrt{2^i})$ and spend memory read energy $O(\sqrt{(2^i)^p})$. Since $p < 1$, the energy reading from the memory is less than the energy driving the wire, so the energy per link at level i is $O(\sqrt{2^i})$.

We perform a similar sum to the previous section to account for the number of switches of a given height, the wirelengths driven by each switch height, and the number of edges that must be switched through that switch height.

$$\begin{aligned} E_{comm} &\leq \sum_{i=0}^{\log_2(N)} \left(\frac{N}{2^i} \times c\left(2^i\right)^p \times O\left(2^{\lceil i/2\rceil}\right)\right) \\ &\leq O(N) \sum_{i=0}^{\log_2(N)} \left(O\left(\left(2^i\right)^{p-0.5}\right)\right) \end{aligned} \tag{10}$$

For $p = 0.5$, the term in the sum is $O(1)$, so the communication energy, and hence total energy, is $O(N\log(N))$. For $p < 0.5$, $p - 0.5 < 0$, so the sum converges to $O(1)$ and the communication and total energy is $O(N)$; since this matches the lower bound of $\Omega(N)$, we can conclude the $p < 0.5$ case requires $\Theta(N)$ energy. For $p > 0.5$, the term in the sum is maximum at $i = \log_2(N)$ and takes on the value $O(N^{p-0.5})$. As i decreases, the terms are geometrically smaller, so the sum converges to $O(N^{p-0.5})$, making communication and total energy $O(N^{p+0.5})$. *Keeping the instructions local to the switches they are configuring saves a factor of $O(\log^{1.5}(N))$ and achieves asymptotically optimal energy for $p < 0.5$.*

7.2 Parallel (FPGA)

Alternately, we can dedicate spatial wiring for each edge in the graph. This gives us an architecture analogous to an FPGA. To simplify analysis and build on the designs and calculations we have already performed, we consider a butterfly fat-tree-based spatial interconnect [13, 28]. Instead of having a single wire from each subtree, and hence a single switch linking subtrees, we have a number of switches proportional to the Rent's Rule proscribed IO for each subtree (See Fig. 5(b)). Switches are passively configured with $O(1)$ local configuration bits; as with an FPGA these are continuously applied rather than being read from memory. As such, switches simply pass values combinationally; there is no need for any sequenced behavior. Gates at the leaves perform as in the sequential case, waiting for all inputs to arrive, computing the output, handshaking with the inputs and handshaking with the outputs.

Again, we start by assessing the width of the entire tree. The width of each of the switch groups combining subtrees at height i is now $O\left(\left(2^i\right)^p\right)$ instead of $O(\sqrt{(2^i)^p})$, but the number of switch groups across the width remains the same. We get a width:

$$W = \sum_{i=0}^{\log_2(N)/2} \left(\frac{\sqrt{N}}{2^i} O\left(2^{2pi}\right)\right) = \sqrt{N} \sum_{i=0}^{\log_2(N)/2} \left(\left(2^i\right)^{2p-1}\right)$$

For $p < 0.5$, the sum converges to $O(1)$, for $p = 0.5$, it converges to $O(\log(N))$, and for $1.0 > p > 0.5$, it becomes $O(N^{p-0.5})$. This makes the width $O(\sqrt{N})$, $O(\log(N)\sqrt{N})$, and $O(N^p)$, respectively, and area $O(N)$, $O(N\log(N))$, and $O(N^{2p})$.

With these lengths, we can calculate communication energy. *The $p < 0.5$ case is the same as the previous sum (Eq. 10), so also achieves $O(N)$ communication energy and $\Theta(N)$ total energy.* The $p > 0.5$ case becomes:

$$\begin{aligned} E_{comm}(p > 0.5) &\leq \sum_{i=0}^{\log_2(N)} \left(\frac{N}{2^i} \times c\left(2^i\right)^p \times O\left(\left(2^i\right)^p\right)\right) \\ &\leq O(N) \sum_{i=0}^{\log_2(N)} \left(O\left(\left(2^i\right)^{2p-1}\right)\right) \end{aligned}$$

The sum converges to $O(N^{2p-1})$ bringing communication and total energy to $O(N^{2p})$. For $p = 0.5$, we get:

$$\begin{aligned} E_{comm} &\leq \sum_{i=0}^{\log_2(N)} \left(\frac{N}{2^i} \times c\left(2^i\right)^{0.5} \times O\left(\left(2^{\lceil i/2\rceil}\right)\log(2^i)\right)\right) \\ &\leq O(N) \sum_{i=0}^{\log_2(N)} \left(O\left(i\right)\right) \end{aligned}$$

The sum converges to $O\left(\log^2(N)\right)$, and hence communication and total energy converge to $O(N\log^2(N))$. *For*

$p \geq 0.5$*, the energy requirements are asymptotically larger than the sequential case in the previous section due to the larger area and hence longer wires.*

If CAD mapping can avoid glitching, or limit it to a constant effect, this fully parallel design could be synchronous without damaging the asymptotic result; only the registers at the leaves need to be clocked, and they only need to be clocked once per evaluation cycle. This can be done with no more than $O(N + W \times \sqrt{N}) = O(N + N^{p+0.5})$ energy, which is no larger than the communication energy.

7.3 Multicontext

The fully spatial, fully parallel, FPGA-like case requires asymptotically more energy than the sequential locality case (Sec. 7.1) when the spatial case requires asymptotically more area and hence asymptotically longer wires. Note that the $p < 0.5$ case that did not require asymptotically longer wires did not require asymptotically more energy. This suggests that we might be able to achieve the same asymptotically low energy as the sequential case without giving up all communication parallelism. However, we must be careful not to include too much communication hardware.

Specifically, the previous case suggests that as long as the number of wires at a tree root grows as $O\left(\left(2^i\right)^{p_t}\right)$ with $p_t < 0.5$, we can keep the side length of the tree to $O(\sqrt{2^i})$. Consequently, we build a tree with $p_t < 0.5$ and provide the same asynchronous context memories for the upper level switches as we did in the sequential case (Sec. 7.1) as shown in Fig. 5(c). However, we size these memories based on their required wire sharing. The memory for a switch at the root of a tree at level i must be have $O\left(\max\left(1, \frac{(2^i)^p}{(2^i)^{p_t}}\right)\right)$. For the interesting cases where $p > p_t$, this is: $O((2^i)^{p-p_t})$. *This means that the switches have heterogeneous context depth, increasing toward the root of the tree where edge growth exceeds available 2D wiring.* Switching behavior is the same as the sequential case except that some tree switches have multiple physical parent connections like the spatial case.

The designs fits in $O(N)$ area with width $O(\sqrt{N})$ and the same asymptotic energy as the sequential case. To properly demonstrate this result we need to:

1. validate memory area is asymptotically small enough: At height i, the wire width for the channel is $O\left(\left(2^i\right)^{p_t}\right)$ and the height of the channel is $O(\sqrt{2^i})$. This gives area $O\left(\left(2^i\right)^{p_t+0.5}\right)$. Each of the $O\left(\left(2^i\right)^{p_t}\right)$ memories is of size $O\left(\left(2^i\right)^{p-p_t}\right)$ for a total area of $O\left(\left(2^i\right)^{p}\right)$, which is less than $O\left(\left(2^i\right)^{p_t+0.5}\right)$ as long as $p < p_t + 0.5$. We can guarantee this holds by selecting $p_t \geq p - 0.5$, which we can do and keep $p_t < 0.5$ for any $p < 1$.
2. make sure we can place and route to the memories without changing the asymptotic area or side lengths: The previous calculation assumed that we could pack switches anywhere in the channel. This is possible if we can route the signals from the lower channel to the point of the switch. Since the number of wires from the lower channel are within a constant factor of the number of wires in a channel, if we space out the root wires to allow the lower channel wires to route alongside them to the point where the switch is placed, it only makes the channel a constant amount wider, thus leaving the asymptotic channel width unchanged (See Fig. 6).
3. account for the read energy from memories: The read energy for a switch at level i, $O(\sqrt{(2^i)^{p-p_t}})$, is never greater than the $O(\sqrt{2^i})$ energy of the associated wire.
4. account for the communication energy: This is the same as the sequential case since the wire lengths are asymptotically the same length.

When limited to $O(1)$ *metal layers for planar VLSI, this heterogeneous, multicontext FPGA saves* $O(N^{p-0.5})$ *compared to a fully spatial FPGA that allocates a wire for every edge, achieving the same asymptotic energy as the sequential case.*

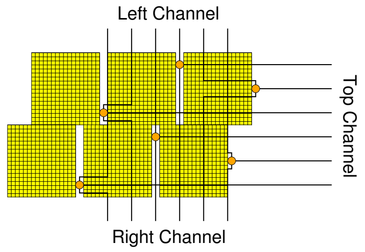


Figure 6: Routing to Multicontext Memories

8. MULTILEVEL METALIZATION

The fully spatial, FPGA-like case with $p \geq 0.5$ consumed more energy than the sequential case because the wires grew asymptotically longer. This is driven by growing channel widths when limited to a constant number of wire layers. *If we can instead use a suitably growing number of wire layers, we can avoid this cost and achieve the same asymptotic energy as the sequential instruction locality case* (Sec. 7.1). In particular, if the wire layers grow as $O((2^i)^{p-p_{2D}})$, where $p_{2D} < 0.5$, we can keep the 2D channel widths down to $O((2^i)^{p_{2D}})$. We must account for both the 2D and z-axis wire lengths in routing, but if we pick $p_{2D} \geq p/2$, the z-axis wire length of $O((2^i)^{p-p_{2D}})$ is never greater than the planar wire length of $O((2^i)^{p_{2D}})$. For $p < 1$, we can pick a $p_{2D} < 0.5$ that satisfies that constraint, so the total wire energy remains asymptotically the same as the sequential and multicontext case. We must guarantee that we can perfectly use the additional wire layers. DeHon and Rubin showed that this was possible using a Mesh-of-Trees interconnect structure [9]. As with the $O(1)$ metal layer FPGA, if we can contain glitching effects, the asynchronous assumption is not necessary to achieve this bound. The total clock energy would only be $O(N)$ per evaluation cycle.

9. LESSONS

Table 1 summarizes the asymptotic bounds developed in this paper. It shows a rich landscape where energy reduces both with the exploitation of more locality in the architecture (roughly top to bottom) and with the gate array netlist locality (Rent p, left to right). The locality exploitation in FPGAs does give them an asymptotic energy advantage compared to processors. The sequence of optimizations shows that exploiting locality to reduce the distances that edge data travel is a larger benefit than being able to share a single physical compute operator. *Locality exploitation is more important than physical operator sharing.*

Table 1: **Asymptotic Energy Required for N-node Gate Array Evaluation**

			Asynch.	Rent Exponent p			
Organization	**Locality**	**Sec.**	**Handshake**	1	1>p>0.5	0.5	<0.5
Crossbar	none	4.1	No	$O(N^2)$			
P Monolithic Mem.	none	4.3	No	$O(N^2)$			
P Fully Banked	description	4.3, 5	Yes	$O\left((N\log(N))^{1.5}\right)$	$O\left(N^{1.5}\log(N)\right)$		
Sequential Comm.	data	6	Yes	$O\left(N^{p+0.5}\times\log^{1.5}(N)\right)$		$O(N\log^{2.5}(N))$	$O(N\log^{1.5}(N))$
Sequential Comm.	instruction	7.1	Yes	$O\left(N^{1.5}\log^{0.5}(N)\right)$	$O(N^{p+0.5})$	$O(N\log(N))$	$\theta(N)$
Parallel (FPGA)	instruction	7.2	No	$O\left(N^{2p}\right)$		$O(N\log^2(N))$	$\theta(N)$
Hetero Multicontext	instruction	7.3	Yes	$O\left(N^{1.5}\log^{0.5}(N)\right)$	$O(N^{p+0.5})$	$O(N\log(N))$	$\theta(N)$
Multilevel (FPGA)	instruction	8	No	$O\left(N^{1.5}\log^{0.5}(N)\right)$	$O(N^{p+0.5})$	$O(N\log(N))$	$\theta(N)$

Multicontext designs can have lower energy than FPGAs when the number of metal layers is limited, but the multicontext designs must be organized differently from previous proposal for multicontext FPGAs. *Sharing of interconnect is important for $p \geq 0.5$ (when limited to $O(1)$ metal layers).* We must be careful not to overbuild wiring to the point that it makes wire lengths grow faster than $O(\sqrt{N})$. For constant numbers of metal layers, this means it is beneficial to share interconnection links to keep the layout size down.

Parallelism is (asymptotically) energy neutral at the architectural level. At fixed voltages, more parallel solutions demand no more asymptotic energy than sequential ones as long as wire lengths do not grow faster than $O(\sqrt{N})$. If we do have fixed performance goals and allow voltage scaling, the parallel solutions may actually be lower energy, since they can use the parallelism to achieve the performance goal with lower operating voltage.

Programmability is (asymptotically) energy neutral. The presence of configuration bits does not impact the asymptotic analysis for the FPGA-like cases (Secs. 7.2 and 8). The constants become smaller, but the asymptotes do not.

10. RELATED WORK

Most of the theoretical work on energy complexity deals with activity factors for particular types of computations (*e.g.*, [1, 17]). This is complementary to the current work that addresses general bounds where circuit classes are only differentiated by their locality.

Another class of work specifically deals with energy-time tradeoffs that arise form VLSI circuits (*e.g.*, [22, 3]). These explore the impact of tuning voltage and sizing (ET^2), which we did not address in this paper. These tradeoffs are important, but not valid over large enough of a range to be fully asymptotic. These works almost entirely address tradeoffs in specific circuits and do not address architectural tradeoffs for general classes of computations. As such, this tuning is also largely complementary to the work described here.

11. OPEN QUESTIONS

The results obtained so far show that there are fundamental reasons that FPGA-like architectures can offer energy advantages. However, there are other phenomena that must be characterized to capture the complete story.

How large are the associated constants and when do the asymptotes matter? We have deliberately focused on asymptotic bounds only. Certainly an asynchronous switch is larger than single SRAM cell and the multicontext design is larger than the sequential instruction locality design. It will be useful to establish how large N must be for various asymptotes to matter (*e.g.* when the fully banked design is lower energy than the monolithic memory, when the sequential instruction locality is lower energy than the processor using fully banked memory). Constants will also be necessary to differentiate the energy in the cases that achieve the same asymptotic energy.

Processors have the option to describe a large computation, but only exercise a small portion of it, or to only exercise portions of the computational graph infrequently. Similarly, gates and wires that do not toggle on a cycle need not consume energy if controlled properly. We deliberately assumed homogeneous *activity* to keep the analysis simple and the results general. As the prior theoretical work show (Sec. 10), specific designs have structurally different activities that can be exploited to identify even lower energy bounds. It will be useful to explore a richer model that accounts for non-homogeneous activities and understand how this impacts the energy picture.

GPGPUs, vector processors, and even scalar microprocessor exploit the fact that instructions can often be productively shared across collections of bits (*e.g.* multi-bit words) and across operations (*e.g.* SIMD, SPMD) or reused (*e.g.*, subroutines, looping), such that the operation description is small compared to the bit operations performed. For this work, we assumed that every gate needed a unique instruction. To complete the picture, it will be useful to characterize the impact of this *instruction sharing and reuse* on the asymptotic energy bounds.

This paper has focused on constructive upper bounds. Except for the $p < 0.5$ case where the constructive upper bound matches the trivial *lower bound*, we have not established non-trivial lower bounds on the energy requirements.

12. CONCLUSIONS

Asymptotically, communication locality exploitation matters. Architectures that are organized to minimize the movement of data and instructions can operate with asymptotically lower energy than alternatives that do not, including sequential processors that move data from a central memory to a single, shared computational block. This gives FPGA-like designs an inherent energy advantage compared to processors. The same energy advantage can be achieved with designs that share interconnect, but operators and data must be spatially distributed similar to FPGAs. Since communication locality is essential, FPGA-like designs must contain processor to processor wiring distances to $O(\sqrt{N})$ (where N is the capacity of the smallest subtree that includes both processors) to achieve the tightest bounds; designs that allow switching or wiring to grow asymptotically faster will not be able to achieve the lowest energy bounds demonstrated. For designs with moderate locality ($0.5 \leq p < 1.0$), wiring requirements grow too fast to achieve the $O(\sqrt{N})$ distance when limited to a constant number of metal layers. It is necessary to either use multilevel metalization with wiring levels growing sufficiently fast or multicontext designs that contain physical interconnect links below $p_t = 0.5$ in order to achieve the lowest energy bounds identified.

13. ACKNOWLEDGMENTS

This work was inspired in part by insights gained from Cory Waxman's time-multiplexed FPGA energy modelling effort. This research was funded in part by National Science Foundation grant CCF-0904577. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

14. REFERENCES

[1] A. A. Ashok, Chandra, and P. Raghavan. Energy consumption in VLSI circuits. In *Proceedings of the ACM symposium on Theory of computing*, pages 205–216, 1988. 10

[2] S. Bhatt and F. T. Leighton. A framework for solving VLSI graph layout problems. *Journal of Computer System Sciences*, 28:300–343, 1984. 2.3, 5

[3] B. D. Bingham and M. R. Greenstreet. Modeling energy-time trade-offs in VLSI computation. *IEEE Trans. Comput.*, 61(4):530–547, April 2012. 10

[4] M. Budiu, G. Venkataramani, T. Chelcea, and S. C. Goldstein. Spatial computation. In *Proc. ASPLOS*, pages 14–26, 2004. 2.2

[5] W. J. Dally, J. Balfour, D. Black-Shaffer, J. Chen, R. C. Harting, V. Parikh, J. Park, and D. Sheffield. Efficient embedded computing. *IEEE Computer*, 41(7):27–32, July 2008. 2.2

[6] A. DeHon. DPGA utilization and application. In *FPGA*, pages 115–121, February 1996. 1

[7] A. DeHon. Balancing Interconnect and Computation in a Reconfigurable Computing Array (or, why you don't really want 100% LUT utilization). In *FPGA*, pages 69–78, February 1999. 5

[8] A. DeHon. Rent's Rule Based Switching Requirements. In *Proc. SLIP*, pages 197–204. ACM, March 2001. 5, 5

[9] A. DeHon and R. Rubin. Design of FPGA Interconnect for Multilevel Metalization. *IEEE Trans. VLSI Syst.*, 12(10):1038–1050, October 2004. 8

[10] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *ISCA*, pages 365–376, 2011. 2.1

[11] D. J. Frank. Power constrained CMOS scaling limits. *IBM J. Res. and Dev.*, 46(2/3):235–244, March 2002. 2.1

[12] S. H. Fuller and L. I. Millett, editors. *The Future of Computing Performance: Game Over or Next Level?* The National Academies Press, 2011. 2.1

[13] R. I. Greenberg and C. E. Leiserson. *Randomness in Computation*, volume 5 of *Advances in Computing Research*, chapter Randomized Routing on Fat-Trees. JAI Press, 1988. Earlier version MIT/LCS/TM-307. 7.2

[14] H. J. Hoover, M. M. Klawe, and N. J. Pippenger. Bounding fan-out in logical networks. *Journal of the ACM*, 31(1):13–18, January 1984. 6.1

[15] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein. Scaling, power, and the future of CMOS. In *IEDM*, pages 7–15, December 2005. 2.1

[16] M. Hutton. Interconnect predition for programmable logic devices. In *Proc. SLIP*, pages 125–131, 2001. 5

[17] G. Kissin. Upper and lower bounds on switching energy in VLSI. *JACM*, 38(1):222–254, January 1991. 10

[18] J. Lamoureux, G. G. F. Lemieux, and S. J. E. Wilton. GlitchLess: Dynamic Power Minimization in FPGAs Through Edge Alignment and Glitch Filtering. *IEEE Trans. VLSI Syst.*, 16(11):1521–1534, November 2008. 3.3

[19] B. S. Landman and R. L. Russo. On pin versus block relationship for partitions of logic circuits. *IEEE Transactions on Computers*, 20:1469–1479, 1971. 1, 5

[20] F. T. Leighton. New lower bound techniques for VLSI. In *Proc. Symp. FOCS*, pages 1–12. IEEE, 1981. 2.3

[21] Y. Lin and J. Cong. Power modeling and characteristics of field programmable gate arrays. *IEEE Trans. Computer-Aided Design*, 24(11):1712–1724, November 2005. 1

[22] R. Melhem and R. Graybill, editors. *Power-Aware Computing*, chapter ET^2: A Metric For Time and Energy Efficiency of Computation. Kluwer Academic Publishers, 2001. 10

[23] K. Poon, S. Wilton, and A. Yan. A detailed power model for field-programmable gate arrays. *ACM Tr. Des. Auto. of Elec. Sys.*, 10:279–302, 2005. 1

[24] J. E. Savage. Planar circuit complexity and the performance of VLSI algorithms. In *VLSI Systems and Computations*, pages 61–68, 1981. 2.3

[25] G. Stitt, B. Grattan, J. Villarreal, and F. Vahid. Using on-chip configurable logic to reduce embedded system software energy. In *FCCM*, pages 143–151, 2002. 2.2

[26] C. Thompson. Area-time complexity for VLSI. In *Proc. ACM STOC*, pages 81–88, May 1979. 2.3

[27] S. Trimberger, D. Carberry, A. Johnson, and J. Wong. A time-multiplexed FPGA. In *FCCM*, pages 22–28, April 1997. 1, 2.2

[28] W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, V. George, J. Wawrzynek, and A. DeHon. HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array. In *FPGA*, pages 125–134, February 1999. 7.2

[29] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor. Conservation cores: reducing the energy of mature computations. In *Proc. ASPLOS*, pages 205–218, 2010. 2.1, 2.2

APPENDIX

A. GATE ARRAY EVALUATION MODEL

A more precise definition of the gate array evaluation model from Sec. 3.1 is as follows:

- The computation is a directed graph $G = (V, E)$.
- Each vertex $v \in V$ is a gate with bounded fanin k.
- A subset of vertices $I \subset V$ are netlist inputs and have no predecessors in G, and a subset of nodes $O \subset V$ are netlist outputs and have no successors in G.
- Each vertex $v_i \in V$ computes its output as a function f_i of its up to k inputs and routes this same output to each of its output edges.
- An edge $e \in E$ is a pair of nodes (v_{src}, v_{sink}) that conveys the output of v_{src} to v_{sink}. An edge, e, may be registered. An unregistered edge conveys the value produced by its source v_{src} to the sink v_{sink} before v_{sink} evaluates its input for a given evaluation. A register edge conveys the value produced by its source v_{src} on the previous cycle of evaluation. Identity vertices can be used in the graph when a value produced by a vertex needs to be used both registered and unregistered and when a value must be retimed for multiple cycles before use.
- In one evaluation of the netlist, each vertex is computed once obeying precedence constraints in the graph. Consequently, every edge (or net) is assumed to switch once per evaluation.

B. VLSI AREA AND ENERGY MODEL

This section provides more background for the VLSI model used in the paper and briefly summarized in Sec. 3.2.

Our primitive building blocks are gates, individual memory cells, and wires. We assume static CMOS gates that dissipate energy only when switched and have dynamic switching energy proportional to $C\left(V_{dd}\right)^2$. We take the supply voltage, V_{dd}, as fixed so energy is proportional to the capacitance switched, C. A primitive gate or memory cell has a constant number of inputs (*e.g.* a 3-input AND gate or a 6T SRAM cell), a constant width, and height, and hence a constant, $O(1)$, area. The inputs present $O(1)$ capacitive load on their drivers, and hence cost $O(1)$ energy, E_{in}, when switched. The gate itself requires $O(1)$ internal energy, E_{gate}, when switched.

Each wire has constant, $O(1)$, width (*e.g.* $2F$ where F is the half-pitch in a particular technology node [1]) and area proportional to its length, l. The capacitance of a wire will be proportional to its area and hence length, making the energy to switch a wire segment proportional to length, $E_{wire}(l) = O(l)$.

We start considering models with a constant number, $O(1)$, of routing metal layers greater than one (*e.g.* two, one for horizontal wiring and one for vertical). Since all wires out of a region or sub-block of a design are of width $O(1)$ and must be routed on $O(1)$ metal layers in this model, the width of a region or sub-block in the design is at least proportional to the number of wires crossing into the region, $Width(region) \geq O(\text{wires cross into region})$. Orthogonal wires can cross on different metal layers. When a wire on one layer crosses n wires on another layer, it must be at least of length $O(n)$.

We account for energy in terms of physical nets with one gate driving a tree of wire segments. Total energy for a physical net will be:

$$E_{net} = E_{gate} + \sum_{i \in \text{ wire segments}} E_{seg_i} + \sum_{j \in \text{input loads}} E_{in_j}$$

Since gate driving and gate input energy is $O(1)$, this gives:

$$E_{net} = O(1) + O(\text{net wire length}) + O(\#\text{ gate input loads})$$

Since there must be finite spacing between gates and gate inputs, the wire length term will asymptotically capture the impact of gate input loads and gate switching. Physical net switching energy then simplifies to:

$$E_{net} = O(\text{net wire length})$$

In Sec. 8 we allow the number of metal layers to increase with N. Logic and memory are contained to a single level on the surface of the chip. We assume a wire segment crossing from the substrate to level l is of length l and hence takes energy $O(l)$ for the crossing. We account for via area in each layer a segment crosses [2].

C. ASYNCHRONOUS EVALUATION

In Sec. 3.3, we noted that the conversion to asynchronous can be done without changing the asymptotic size or number of switching events. In particular, we can represent each original netlist signal using two bits so we can encode the non-presence of data and add a reverse signal to perform handshake acknowledgment of the inputs. A gate waits on the arrival of all of its inputs before producing an output. The 3 signals now used in place of each original wire toggle a constant number of times (4 times) during an evaluation. Similarly, the internal logic is only a constant factor larger than the simple gate. Native asynchronous FPGAs have been developed in previous work [4, 3]. In figures, we show $O(1)$ wire bundles that include directional wires and handshaking as a single line.

D. REFERENCES

[1] International technology roadmap for semiconductors. <`http://www.itrs.net/Links/2011ITRS/Home2011.htm`> , 2011. B

[2] A. DeHon and R. Rubin. Design of FPGA Interconnect for Multilevel Metalization. *IEEE Trans. VLSI Syst.*, 12(10):1038–1050, October 2004. B

[3] J. Teifel and R. Manohar. Highly pipelined asynchronous FPGAs. In *FPGA*, pages 133–142, 2004. C

[4] C. Wong, A. Martin, and P. Thomas. An architecture for asynchronous FPGAs. In *ICFPT*, pages 170–177, 2003. C