Appearing in *IEEE International Conference on Field-Programmable Technology* (FPT 2015), December 7–9, 2015

Energy Minimization in the Time-Space Continuum

Hyunseok Park, Shreel Vijayvargiya, André DeHon
Dept. of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA, USA
Email: hnskpark@gmail.com, vshreel@seas.upenn.edu, andre@ieee.org

***Abstract*—Can time-multiplexing save energy? Recent theoretical work suggests that time multiplexed architectures might use less energy than fully spatial FPGAs. Spatial FPGAs conserve energy by avoiding instruction fetch, exploiting locality, and exploiting low activity on wires. However, since they dedicate physical switches and wires to a single signal, they can be larger than designs that time multiplex these physical resources. Can the area savings from time multiplexing reduce wire lengths significantly enough to provide a net win against increased switching activity and the addition of instruction energy? Mapping designs from the VTR 7 no memory benchmarks and spatial FFTs, we show that spatial FPGAs remain the most energy efficient architecture at least up to half a million 4-LUTs. We explain why this is and explore how sensitive our results are to technology and usage assumptions.**

I. Introduction

Trimberger [1] reports that time-multiplexed FPGAs—FPGAs that store multiple configurations on chip and change among them during execution—required higher power than normal, non-time-multiplexed FPGAs. Nonetheless, tradeoffs may change with Moore's Law scaling, and there is a large design space for time-multiplexed architectures to explore. DeHon suggests that time multiplexing can reduce energy under certain circumstances [2]. However, he does not fully articulate a design or map specific benchmarks. As such, he leaves open many issues, including a characterization of *typical* values for switching activity and interconnect requirements that may determine whether or not there is an advantage for real designs.

Most of the energy in FPGAs goes into switching long wires [3], [4]. There are three main competing energy forces in play as we explore spatial and time-multiplexed FPGAs: (1) long wires are highly capacitive, making them consume significant energy each time they switch, (2) dedicated wires switch less often than time-multiplexed wires, exploiting the inherent temporal correlation in the datapath, (3) time multiplexing adds energy to read instructions from memories. Time-multiplexed designs can be smaller than spatial designs [5], so the key question will be whether or not the time-multiplexed design can sufficiently reduce the wire length to cover the cost of increased switching activity and added instruction energy.

We systematically elaborate and explore a design space for time-multiplexed designs. We identify key design parameters including the number of LUTs to sequentialize on a processing element (S), the level of sequentialization for the interconnect, and the physical interconnect richness (p_t). We develop tools to map designs to these different points in the design space (Sec. IV), circuits for the key structures, and a 45 nm energy, delay, and area model for assessing and comparing the costs of the mapped designs. We also explore the different microarchitectures for managing and controlling the time-multiplexed resources that range from a simple, flat design where all resources are controlled with a single wide instruction, to a data-driven design that fully embraces *sparse activation*, only activating any instruction or data memory reads when a resource is actually used (Sec. V). We find that this fully data-driven extreme uses the least energy of the time-multiplexed designs on standard benchmarks. Nonetheless, the best time-multiplexed design uses more energy than the fully spatial design. We detail the key phenomena in play that prevent the time-multiplexed designs from achieving a net win and use these to examine the sensitivity of our conclusions to changes in application, use, and technology parameters (Sec. IX).

Our novel contributions include:

- A detailed microarchitecture and circuit-level design for a time-multiplexed FPGA (Sec. II-B, V)
- Microarchitecture optimized for sparse activation (Sec. V)
- Mapping tools for this design space (Sec. IV)
- Quantified energy costs for mapped designs (VTR7 no memory benchmarks [6], spatial FFTs) (Sec. VI) including fixed, one-size-fits-all architectures (Sec. VIII)
- Sensitivity analysis on technology and model assumptions (Sec. IX)

II. Background

A. Prior Work

Time-multiplexed FPGAs were initially of interest as a way to pack more logic onto limited die area [5], [7], [1], [8]. By storing multiple configurations on the die, it is possible to rapidly switch among different behaviors during a computation. A configuration context can be small compared to the active logic, making computations more compact. The recent multicontext offering from Tabula was able to achieve about $3\times$ the logic density of spatial FPGAs and ran at a fixed context clock of 1.6 GHz [9]. To our knowledge, only [1] reported the energy or power of time-multiplexed designs.

B. Energy Components

In a spatial design, the main energy components are logic, interconnect wires and switches, clocking, and leakage. Time-multiplexed designs also spend energy on memory reads, both from data memory and instruction memory. Tab. I shows the basic technology parameters we assume throughout this paper.

1) Wire Energy: Wire energy is the dominant energy component in spatial FPGA designs [3], [4]. This is driven by charging and discharging the capacitance on wires when they switch. Wire capacitance is linear in the length of the wire, making it important to determine wire lengths to model FPGA wire energy. As a result, our basic wire energy model is

$$E_{wire} = \left(\frac{\alpha}{2}\right) LC_u V^2 \quad (1)$$

where α is the activity rate and L is the wire length.

978-1-4673-9091-0/15/$31.00

TABLE I
45 NM LSTP TECHNOLOGY TABLE

Symbol	Description	Value
F	Minimum feature size	45 nm
FP	Full pitch	90 nm
V_{dd}	Supply voltage	1 V
I_{leak}	Leakage current, per minimum sized transistor	9 pA
C_g	Gate capacitance of a transistor	38 aF
C_u	Wire capacitance per meter	167 pF
R_u	Resistance of the wire per meter	2600 kΩ
R_0	Drain-to-source resistance of a min. sized trans.	39 kΩ
A_b	Area of one SRAM cell	147.5 F^2
M_{layers}	Number of metal layers for interconnect routing	8

TABLE II
ENERGY FOR LUT TOPOLOGIES

	Dynamic Energy	Leakage Energy
Pass Transistor (LP)	11.8 fJ	30 aJ/ns
Transmission Gate (LSTP)	13.6 fJ	0.6 aJ/ns

2) Leakage: In CMOS all active devices—logic, switches, and memories—leak and must spend some energy maintaining their output logic levels. We simulate key components in SPICE using models from the NCSU FreePDK45 [10] to get the leakage energy for each component. From early simulations, we quickly determined that leakage would be a dominant factor if we used a high-performance (HP) or low-power (LP) process, but could mostly be curtailed with a low-standby-power (LSTP) process. The LSTP process operates with a higher V_{th} so that devices can be strongly turned off, reducing subthreshold current. Since our focus is low energy, we use the LSTP process for this work.

3) Logic Energy: Logic energy comes from switching devices. We built key circuits, including LUTs, registers, and switches, and simulated them in SPICE. We revisited the design of the K-input LUT (a 2^K:1 multiplexer) in the context of low supply voltage and high threshold voltage. The conventional NMOS pass transistor design [11] cannot restore the voltage level for logical ones without level restorers; subsequent stages see reduced voltage and increased leakage in all processes and fail to switch in low leakage (high V_{th}) processes (LSTP). The transmission gate always restores the voltage to the full value, reducing leakage (Tab. II), and is more tolerant to low supply voltage and high threshold voltage, both of which are consistent with prior work [12] [13].

4) Data Memory: Each of the LUT inputs in the Processing Element (PE) needs a one-bit wide data memory (Fig. 4). These memories are generally small (Sec. VI), and we use a standard random access memory (RAM) design with a row decoder to select the appropriate row and multiplexers to select the output from the selected row.

5) Instruction Memory: Since time-multiplexed instruction memories (imem) are accessed sequentially, we can save decoding energy by using shift-registers for row and column addressing. Furthermore, when the width of the instruction memory is smaller than the square root of the memory capacity (memory row), we store the entire memory row in a shift register at the output stage. We shift out the bits needed in each cycle, only accessing the memory core once all the stored bits in the shift register have been used. In contrast, a standard RAM would spend energy to fetch the entire row for each word read, wasting the energy to charge the bit lines for the unused portion of the row. This instruction memory is used in the architecture to control the switches in the interconnect and the narrow instruction memories in the PE (Sec. V-A).

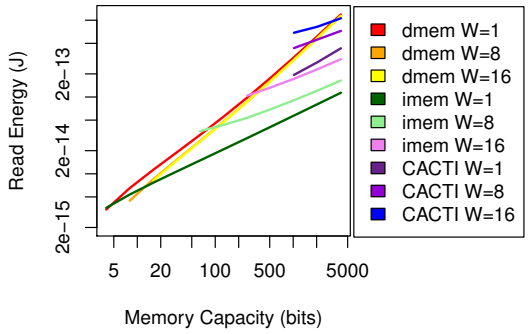

Fig. 1. Memory Energy

Fig. 1 shows how energy grows with capacity and word width in these models. As we expect, the instruction memories (imem) are lower energy than the RAMs used for data memories (dmem). We also include CACTI memory estimates [14]. These are only included for large capacities since CACTI does not provide models for memories below 1024 bits. This shows, roughly, that the data memories are consistent with the CACTI RAM. CACTI achieves lower energy at higher capacities by banking, an optimization omitted for our data memories since we did not require large RAMs.

III. ARCHITECTURE OVERVIEW

Since prior work has shown that 4-input LookUp Tables (4-LUTs) minimize energy [15], our basic architecture uses a compute block containing a 4-LUT with optional output flip-flop. These compute blocks are organized in a hierarchical interconnection network with wiring provisioned based on Rent's Rule [16]. We limit our designs to 4-LUTs and flip-flops, leaving coarse-grained embedded blocks (e.g., multipliers) and embedded memories for future work. The simple architecture allows us to focus on the key issues of logic and interconnect size and the impact of time-multiplexed resource sharing using data and instruction memories.

A. Spatial

The spatial design is organized like the HSRA [17], except that we use a more modern directional-drive architecture [18] (Fig. 2). Rent's Rule suggests, when we partition for locality, the I/O required for a collection of N gates grows as cN^p, where the Rent Exponent, p, characterizes the locality in the design. Consequently, in the HSRA, channel width in the 2-ary tree network increases discretely towards the root ($ch \propto \left(2^h\right)^p$) [16]. At each height, h, in the tree, the parent channel width either conserves the total child channel width using a 2:1 switch, or the parent channel width is the same as the child channel width, effectively reducing total channel width by a factor of two, using a 1:1 switch. The schedule of channel width reduction at each tree height can be selected to just

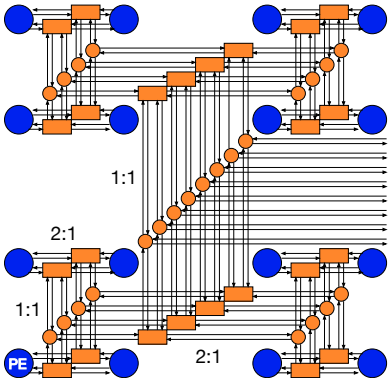

Fig. 2. Spatial Design: HSRA-Style Interconnect with Directional Drive

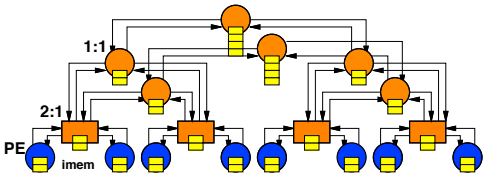

Fig. 3. Time-Multiplexed Design

match the requirements of the application in a *matched* design, or can be *fixed* to accommodate a large range of designs. For simplicity we can talk about this schedule in terms of the particular Rent exponent, p, that it approximates. Switchboxes are linearly populated as shown in Fig. 2. Wires are buffered to minimize total energy. I/O pads are placed at the leaves of the tree as well, assuming they will be supported by area I/O.

B. Multicontext

The multicontext design derives from the spatial design and the heterogeneous multicontext design in [2] (Fig. 3). First, we time-multiplex the physical LUTs. Rather than assigning one logical 4-LUT to each physical 4-LUT at the leaf processing element (PE), we allow a number of logical LUTs, S, to share a single leaf 4-LUT. This reduces the number of leaf PEs in the network by a factor of S compared to the spatial design. These leaf PEs will be larger since they must hold instruction and data memory to control the time-multiplexing.

Once we sequentialize the leaves, it is natural to sequentialize the physical network as well. In particular, if only one of S 4-LUTs can produce a value in a timestep, there is only one S-th as much traffic to route over the network on a timestep. Conceptually, at least, we think of performing the route of one set of data from the outputs produced at the active PE sources to their consumers as a *wave*. Computation proceeds by evaluating a set of 4-LUTs in parallel and performing a routing wave that sends the LUT outputs through the network. In the simplest case, there is one wave per LUT evaluation (sequentialization, S), and the switches in the network need an instruction per wave to control their configuration. However, since we do not cascade 4-LUT evaluations within a wave, precedence constraints in the logical netlist may demand more than S waves, with the 4-LUT depth of the critical path (Tab. III) also serving as a lower bound on the number of waves needed to evaluate the design. Detailed time-multiplexed routing determines the exact number of waves needed (Sec. IV).

We can separately control the size of the physical network, rather than simply designing it to have the same growth schedule as the spatial network except divided by the leaf serialization, S. At one extreme, we might build a binary tree network, which would be composed physically of only 1:1 switches. In general, we can select the Rent exponent for the physical network, p_t, to be any value, and it may be smaller than the logical p required for a routing wave. When $p_t < p$, it becomes necessary to further time multiplex the interconnection network within a single routing wave. For example, if we did use binary tree ($p_t = 0$), and a particular routing wave required 4 signals to cross the root of the tree, it would be necessary to sequentialize communication across the root of the tree over 4 cycles within a routing wave. More generally, if a routing wave requires some $(2^h/S)^p$ signals to cross over a subtree root at height h of a p_t tree that has only $(2^h/S)^{p_t}$ wires in the channels at height h, it will require the communications be sequentialized by $(2^h/S)^{p-p_t}$. When $p_t < p$, the ratio of logical bandwidth to physical bandwidth changes as we go from the root to the leaves, so the level of time multiplexing, and hence the depth of the time-multiplexed memories associated with the switches, varies with the height of the switch. A key source of the asymptotic area, and hence energy, benefit in [2] comes from the ability to keep the wires asymptotically shorter using a $p_t < 0.5$ design when $p > 0.5$.

C. Multiplexing and Wire Activity

Sharing wires, while good for reducing area, can increase the activity factor on a wire and hence increase energy. The simplest case to consider is one where two signals share a single wire driven by a multiplexer. On each logical cycle, the multiplexer first drives one signal, then the other. If both signals are low activity, α, they will typically hold the same value across multiple clock cycles. In the extreme case, if one signal is almost always high and the other almost always low, we take a design with almost no switching and force each signal to switch on each cycle, increasing the effective switching activity to 100%.

IV. Methodology

To compare architectures, we perform complete mapping of benchmark circuits and estimate the energy required to run the benchmarks on the various architectures.

The lack of embedded memories limits us to the VTR 7 no memory benchmarks [6]. We add 16-bit spatial FFTs of size 4, 8, 16, 32, and 64 (Tab. III) to increase the benchmark set size and diversity and to illustrate size scaling for a task. We used VTR 7 (ABC [19]) to map all the designs to 4-LUTs and used the same netlists across all architecture variants within the study. To estimate wire energy for the spatial designs, we performed gate-level simulations with random inputs and captured the signal activity (toggle rates).

For placement on the hierarchical network, we developed a custom, recursive partitioner based on KLFM-style bipartitioning [20] that we use for both spatial and time-multiplexed designs. In the time-multiplexed designs, the recursive partitioning stops when the cluster size reaches S. We place each

TABLE III
BENCHMARK STATISTICS

Benchmark	LUTs	Logic Depth	Weighted Activity	p
stereovision3	378	18	0.164	0.34
sha	3214	54	0.226	0.50
diffeq2	4768	128	0.072	0.43
diffeq1	5144	126	0.075	0.43
fft4	10246	86	0.206	0.43
blob merge	10351	49	0.026	0.50
stereovision0	16675	22	0.103	0.38
fft8	37468	85	0.185	0.45
stereovision1	37580	32	0.066	0.50
bgm	66193	120	0.262	0.67
stereovision2	67830	55	0.218	0.50
fft16	108634	85	0.182	0.50
fft32	285264	95	0.175	0.55
fft64	549331	129	0.204	0.59

design on the tree with the smallest height that accommodates it—i.e., we round up to the closest power-of-two network size. The partitioner allows imbalanced cuts as long as they fit within the allocated network capacity. The partitioner determines the required wire growth schedule to support routing.

For routing, we use a custom, list-scheduling-based, greedy time-multiplexed router. Nets are routed in precedence order in the earliest time-slot available using a Longest-Processing Time (LPT) heuristic [21], [22]. The partitioner determines the logical wiring schedule required, and routing attempts to minimize the number of waves required to perform routing. To support the *data-driven architecture*, the router can also enforce a bound on the maximum number of waves on which a particular switch is used that is lower than the number of waves. The router reports the statistics on the used wires at each tree height so that we can calculate wire energy. In the case of the spatial design, these are weighted by the simulated wire switching statistics. The tools also report usage statistics on LUTs and PE data memories as needed by the energy models. Based on the statistics from the partitioner and router, we compute the design area, memory sizes, wire lengths, delays, and energies.

V. MICROARCHITECTURE

In this section, we start from the most straightforward time-multiplexed microarchitecture, illustrate the energy consequences, and successively refine the microarchitecture to reduce energy consumption.

A. Processing Element

The PE of the Time-Multiplexed (TM) architecture consists of one 4-input LUT, 4 data memories for each of the 4 inputs to the 4-LUT, instruction memories to control the PE, multiplexers to select the write input to the data memories, and flip-flops at the input and the output of the PE (Fig. 4).

The most simplistic PE design is the *Flat PE* design in which a single instruction memory controls all the PE components. The instruction memory for the $S = 8$ design with one input per cycle will be 49 bits wide, including 16b to control the 4-LUT, 1b for data memory read enables, and 8b for each data memory (1b multiplexer select, 1b write enable, 3b read and 3b write address).

Fig. 6 breaks down the energy composition of the flat designs for the `stereovision2` benchmark. The LUT Active energy includes the LUT evaluation energy, data memory read energy, instruction memory read energy, and the output flip-flop energy. The Data Write Active energy is the PE dynamic energy consumption during the data memory writes, including instruction memory read energy, input multiplexing energy, input flip flop energy, and the associated wire energies. The LUT Inactive energy is the PE dynamic energy consumption when no LUT evaluation takes place, including instruction memory read energy and the associated wire energies. Data Write Inactive energy is the PE dynamic energy consumption when no data write activity occurs including reading from the instruction memories and the associated wire energy.

The flat design breakdown shows that the LUT Inactive energy forms the major part. The flat design reads the instruction memory on every cycle, even when the LUT, which contributes half the configuration bits, is not in use. To reduce this energy waste, we explored a series of optimizations that successively separated the LUT (*Sparse LUT* design) and data input control bits (*Sparse Input* design) so that they are only read when needed, reducing inactive energy waste (Figs. 4, 6).

However, we still have inactive energy present since we must read the main memory every cycle to determine when to activate the LUT and data input stores. For example, the $S = 8$ `stereovision2` requires 69 waves, but has fewer than 32 inputs per PE, so most waves will require no instructions in the PE. We should avoid spending any energy reading from memories when there is nothing to do in the PE. We can do this by including information in each configuration on when the **next** activity will occur. This allows a counter to keep track of the idle periods between PE uses, avoiding the need to read PE instruction memory on most cycles. This makes up the *PE Active* design (Fig. 6), which has negligible inactive energy. For the benchmark set, we see that this PE Active design comes out to be the most energy efficient (Fig. 6).

B. Interconnect

Without buffering the interconnect, the clock speed is slow and all transistors leak for the longer duration (See PE Active design in Fig. 6). To minimize the leakage energy, we buffer the longest wire at the top height of the tree. The wires at the lower levels are buffered to satisfy the optimized clock speed set by the longest wire. The *Buffer Wires* case in Fig. 6 shows the benefit of the optimization, where leakage is minimized. While additional buffers increase Wire Inactive and Wire Active energy, we achieve net benefit from buffering.

A significant portion of the remaining energy (Fig. 6) is spent on switching unused wires (wire inactive). When a switch multiplexer at an unused output port is configured to select its input from the transitioning link, the wire connected to the multiplexer output may spuriously switch. To prevent this wasteful wire switching, we introduce an extra latch at each port of the switch that is enabled only when the port is used for routing a signal (Fig. 5). While the switch instruction memories need to be wider, and hence larger, to provide extra control bits for the latches, we achieve net energy benefit from removing spurious wire switching activities, as can be seen from the *No Spurious (Switching)* case in Fig. 6.

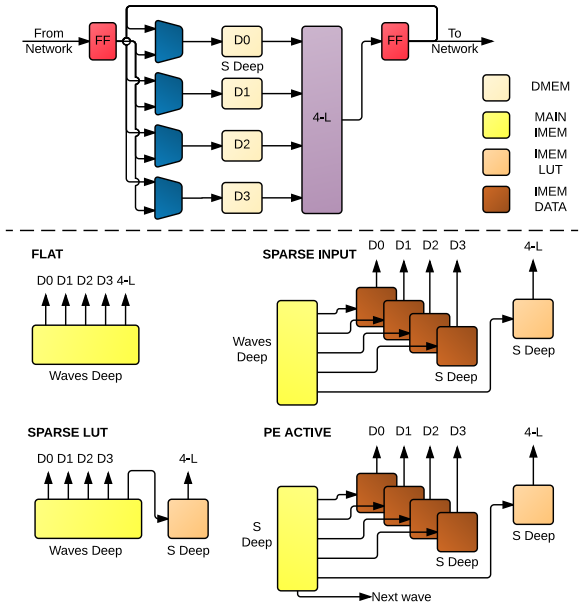

Fig. 4. PE Microarchitecure Models

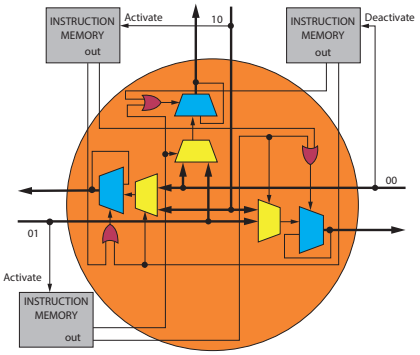

Fig. 5. Data-Driven 1:1 Switch

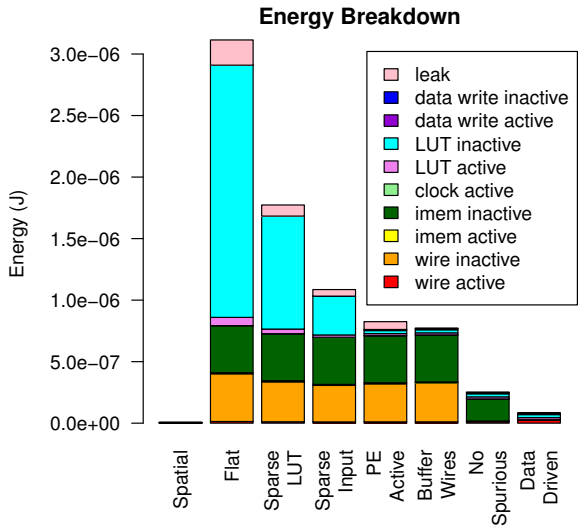

Fig. 6. Impact of Processor and Interconnect Optimizations for `stereovision2` for $S = 8$, $p_t = 0.5$

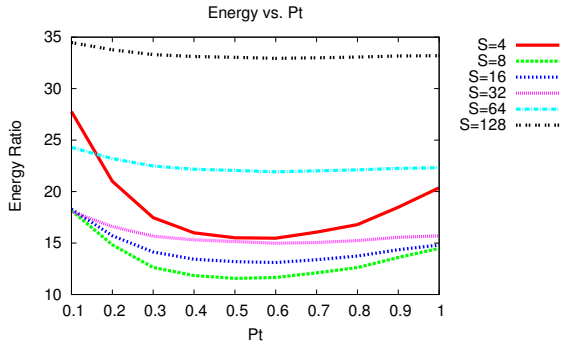

Fig. 7. Energy Ratio to Spatial for `stereovision2` versus S and P_t

While the configuration of the unused switch remains the same as the previous cycle, we still need to spend energy to read identical configuration bits. To minimize the energy consumption in configuring inactive switches, we introduce a data-driven design that has an instruction memory per port of a switch that is only read when the port is used. The instruction memories only store bits necessary for configuring active multiplexers (instruction depths roughly $4 \cdot S \cdot 2^{h(p-p_t)}$). As shown in Fig. 5, a 1:1 switch needs 6 control bits (2 per port) to send incoming data to a desired destination (2:1 switches need 10). To identify when data arrives on an input port, we use a 2-bit packet signal where 00 or 11 signifies that input data is not present and the memory is unused, and 01 or 10 indicates that there is input data, 1 or 0. In this way, no energy is spent on the switches with inactive port(s), as shown in the *Data Driven* case in Fig. 6. Although Wire Active energy is doubled since we have to switch wire twice to support the 2-bit packet signal, no energy is spent on Imem Inactive energy, which results in net energy reduction.

The data-driven design thoroughly utilizes sparse activation opportunities and minimizes the inactive energies spent on the processing elements, wires, and switch instruction memories.

VI. Experimental Results

In the previous section, we have illustrated how to tune the time-multiplexed architecture to minimize energy on a single design and a single architectural point. More generally, we want to understand what is the optimal level of time-multiplexing to minimize energy. Specifically, we consider how much to serialize the computation at the PEs, S, and how much to serialize network communication, p_t. For `stereovision2`, Fig. 7 plots the energy ratio to spatial across this S and p_t parameter space. We can see that $S = 8$ along with p_t in the 0.4–0.6 range minimizes energy, but the energy is an order-of-magnitude higher than the spatial design.

As noted in the introduction, a key opportunity is to reduce the area and hence the length of the wires that must be switched. Fig. 8 compares the side length of the smallest matched chip that will support the `stereovision2` benchmark for the spatial design and various levels of time

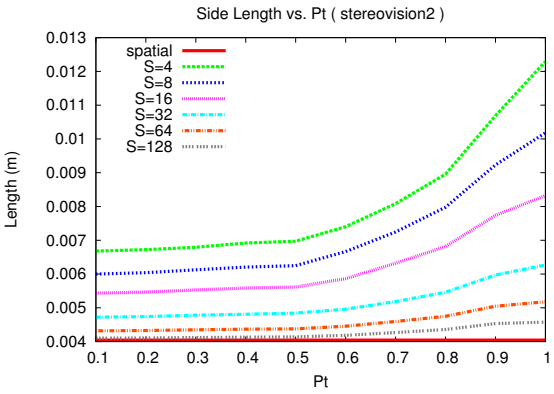

Fig. 8. Side Length for `stereovision2` as a function of p_t

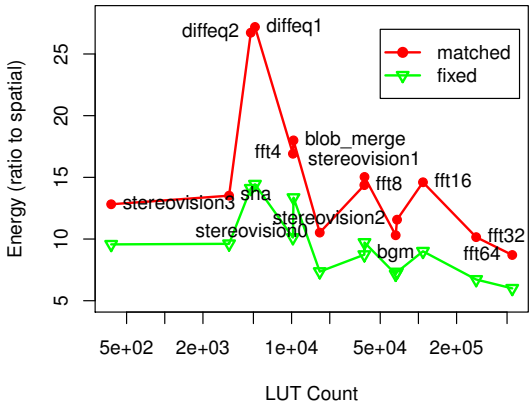

Fig. 11. Net Energy Ratio to Spatial for Data-Driven, Time-Multiplexed $S = 8$, $p_t = 0.5$ Architecture versus Benchmark LUT Count

multiplexing in the (S, p_t) space. As expected, designs with the highest time-multiplexing factors have the shortest wires, which are essential to achieving a net win for the time-multiplexed design. Surprisingly, we see no net reduction in wire length for this design (Fig. 8).

To take both a broader and deeper look at the impact of PE serialization, Fig. 9 shows the energy ratio as a function of S for the entire benchmark set and a detailed breakdown of the energy components for the `stereovision2` benchmark. We see that most designs have a minimum energy around $S = 8$. The detail energy breakdown on the right helps explains how this minimum arises. The leftmost bar shows the spatial design, which has its dominant energy signaling on wires. Next to that we show what would happen if the spatial design had to pay for 100% switching activity on its wires, illustrating the benefit the spatial design is getting from exploiting low activity switching; this is a useful reference since the time-multiplexed designs are spending 200% switching activity on the wires due to the data-driven signaling scheme. We see that the time-multiplexed designs do not make the wires sufficiently shorter (Fig. 8) to compensate for the increased switching energy. As we increase the time-multiplexing of the PE, we reduce the wire energy, but we are also increasing the data and instruction memory in the PE. At S=8, we roughly balance the energy added from the PE instruction memories with the wiring energy, minimizing total energy.

Fig. 10 focuses on the impact of network serialization, p_t, again looking at all designs and showing a detailed breakdown for `stereovision2`. We see that most designs are minimized around $p_t = 0.5$. This happens mostly because the wiring requirement, p, for the designs is close to 0.5 (Tab. III), but some designs have a slightly larger physical p value. The right graph shows how the energy shifts as we increase p_t. As p_t increases the wire lengths increase (Fig. 8), increasing the wiring energy. However, as p_t increases, the lower serialization reduces the switch instruction memory and reduces the leakage energy down to about p_t=0.5. Above p_t=0.5, the instruction memory does not increase for `stereovision2`, but the larger number of unused switches increases the leakage energy. The composite effect of wire length and leakage growth leads to a clear minimum at $p_t = 0.5$.

Using the optimal serialization of S=8, p_t=0.5, Fig. 11 plots the energy ratio between the data-driven, time-multiplexed design and the fully spatial design for all the benchmarks by their LUT count. All designs are using significantly more energy than the spatial design. The ratio has a downward trend for the larger designs, but even the design with half a million 4-LUTs requires 11× the energy of the spatial design. For the designs with p>0.5, asymptotic effects suggest a diminishing spatial energy advantage as the design size increases [2].

VII. Why does the TM design have larger Energy?

The time-mulitplexed design did not reduce its wires sufficiently to get a net energy reduction. In fact, in many cases, it did not reduce the wire lengths and even the energy spent on wires was larger than in the spatial design. DeHon suggests the big win for time-multiplexing comes when the design is wire dominated [2]. In that case, there is no limit to the potential benefit of using a $p_t < p$ wiring network. Even before the asymptotic $\left(2^h\right)^{(p-p_t)}$ takes effect, the reduction of wires by a factor of 32 for the time-multiplexed design can be significant—the $S = 8$, 4-LUT design could need 4×8 inputs, which we sequentialize to a single input wire. However, these designs are **not** wire dominated in this technology (Fig. 10, right) because the p's are generally small and, even when the p's are greater than 0.5, the design sizes are not large enough (Tab. III). In `stereovision2` wires account for less than 10% of the area, with switches accounting for roughly 50%.

When switches rather than wires dominate, switch sharing can still save area, if the switch configuration memory area for one switch use is smaller than the switch area itself. In this design the area for a time-multiplexed switch configuration is larger than a spatial switch, negating this potential advantage. This comes in part due to energy optimization that required additional area, including the latch to prevent spurious switching and the extra configuration bits to prevent instruction reads on unused cycles. Nonetheless, even when there is an area advantage, it is a constant one rather than an asymptotic one. The total number of switches is still linear in N for any $p < 1.0$. Tabula's design was one-third the size of spatial designs and would only have $\sqrt{3}$ shorter wires, suggesting, at most, a 40% reduction in wire energy, which is

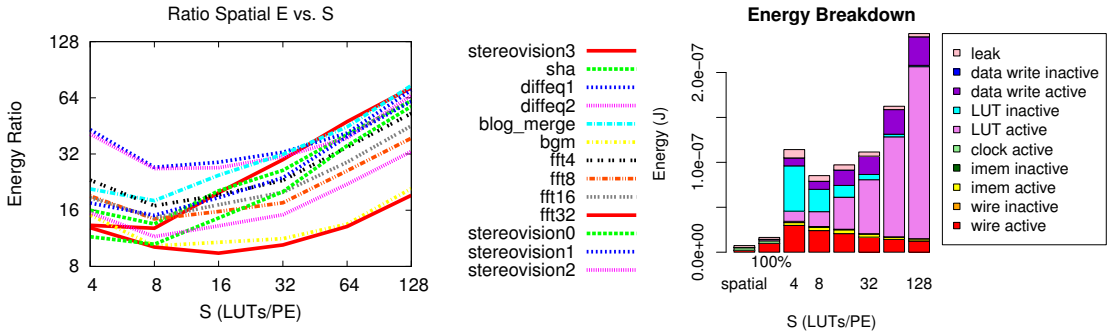

Fig. 9. Impact of PE sequentialization (S) on all benchmarks (left) and energy breakdown for `stereovision2` (right) at p_t=0.5

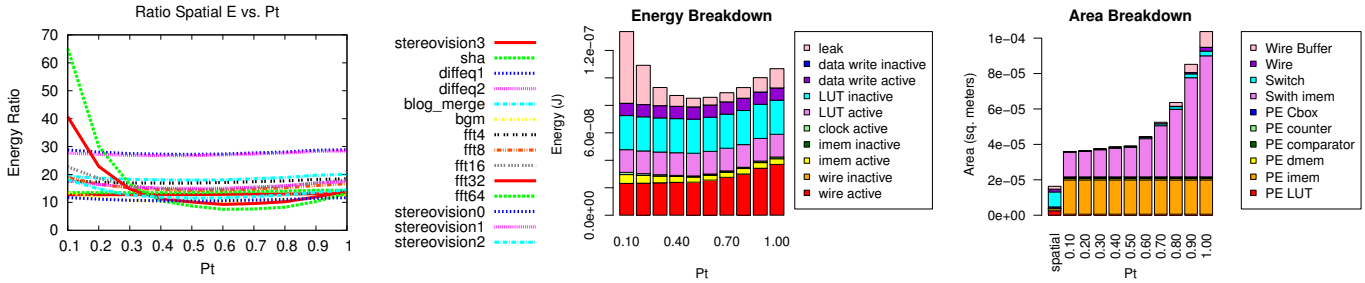

Fig. 10. Impact of p_t on all benchmarks (left) and Energy and Area Decomposition for `stereovision2` (right) at $S = 8$

likely insufficient to compensate for the loss of activity savings from wires that are only active 26% of the time in the spatial case. Traditional island-style mesh designs [11] are even more likely to be switch dominated than wire dominated [23].

VIII. Fixed Resources

In the previous section, we allowed each design to be mapped to the perfectly *matched* architecture—that is, a spatial component with exactly the wiring growth schedule to map the benchmark and a time-multiplexed design where the instruction memory depths were similarly sized to match the benchmark. In practice, an FPGA vendor would only sell one, or a few, variants. Typically, this drives the FPGA vendor to richly provision the interconnect so that it will support most applications customers might want to map. This will make the wires longer for both the spatial and time-multiplexed designs and make the memories deeper for the time-multiplexed design. This could increase wiring requirements since the fixed p will often be larger than the benchmark's own p.

To model this case, we recorded the maximum wiring requirements across our entire benchmark set and came up with a minimum wire growth schedule for the tree that would accommodate all of the designs. We then evaluated the energy for mapped design using this *fixed* wire provision rather than the exact matched provision used earlier. In Fig. 11, we show this ratio of the spatially mapped design to the energy-minimizing time-multiplexed architecture along with the matched designs. The energy penalty for time multiplexing is slightly lower for this fixed case than the matched case, but energy is still worse than the spatial design.

If we take this to the extreme and use a fixed p=1 design, we do begin to see an energy advantage to time

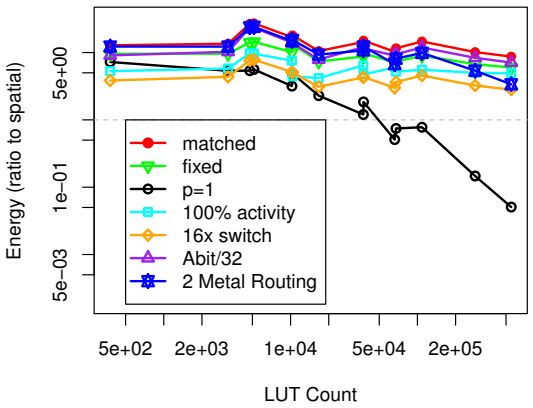

Fig. 12. Net Energy Ratio to Spatial for Data-Driven, Time-Multiplexed $S = 8$, $p_t = 0.5$ Architecture versus Benchmark LUT Count

multiplexing for the larger designs (Fig. 12). This arises for the anticipated reasons. The wire area dominates (97% for spatial `stereovision2`), so that the time-multiplexed design that shares wires has significantly lower area and hence wire lengths (80% lower for `stereovision2`)—sufficiently lower to achieve a net energy reduction. Nonetheless, this is probably not a case for time multiplexing as much as a case against over-engineering the wiring in spatial designs [24]. The spatial design with wiring closer to the benchmark set (*fixed* case) is still lower energy than the time-multiplexed designs.

IX. Sensitivity

A challenge in evaluating these designs is tool quality [25], model accuracy, and changing technology. In this section, we

briefly evaluate how the trends or conclusions might change with different technology, tools, and usage.

The previous sections establish pretty clearly that these designs are not wire dominated. That means the partitioner is at least of reasonable quality. A better partitioner would only push the designs further into the switch- rather than wire-dominated domain, which, as we have seen, is favorable to the spatial design over the time-multiplexed design.

The spatial design benefits from the ability to exploit low activity factors. We estimate these designs have activities under 26%, with some down to 2.6% (Tab. III). As a limit case to understand the impact of higher activity, we include a "100% activity" curve in Fig. 12, showing that higher activity is not enough to make the time-multiplexed designs preferred. This also provides a bracket in case our gate-level activity simulations under estimate spatial glitching activity.

As noted, a key factor that plays against this design is that the switch configuration is larger than the spatial switch. It is possible that our spatial switch estimates are smaller than reasonable (*e.g.*, [26]). To understand this effect, we evaluated variant models where we multiplied the switch size by different factors. Even at a fairly extreme case of 16× switches, which make the switch area 11× the area of the switch configuration in memory, the spatial design still has lower energy than the time-multiplexed designs (Fig. 12).

Memory consumes much of the area in the time-multiplexed design (Fig. 10). Technologies that would reduce the size of memory cells could change the tradeoffs. Non-volatile memories have been demonstrated at $50\,F^2$ [27], and FLASH and DRAM cells can reach $4\,F^2$ [28, ORTC-2c]. To explore this, we evaluated configuration bits that were 32× smaller than our main assumption of roughly 150 F^2. We left the periphery area unchanged. Many of our small data memories are already periphery dominated, so they will see small changes. As a result, the overall impact of even a 32× reduction in memory cell area is small as shown in the "Abit/32" line in Fig. 12.

If we restrict the network routing to 2 metal layers, instead of 8, designs become wire dominated earlier. We see a clearer downward trend for the time-multiplexed energy (Fig. 12), but it remains larger than spatial for this benchmark set.

The results here are for bit-level architectures. Wide-word architectures share configurations, thus reducing the ratio of instruction memory area to switch area and may become wire-dominated and see a benefit earlier than bit-level architectures.

X. Conclusions

Typical designs have low activity and high locality (Rent exponent close to 0.5). As such, they are switch rather than wire dominated and are relatively energy efficient on spatial FPGAs. In these switch-dominated designs, time-multiplexed architectures have limited ability to reduce area and hence wire length, but must pay the cost of higher activity on the wires and the cost for reading instructions from memories. As a result, even our highly optimized time-multiplexed designs require more energy than spatial designs. The time-multiplexed architectures may be able to provide an advantage in wire-dominated regimes, such as large designs with extreme, non-local interconnect requirements.

References

[1] S. Trimberger, D. Carberry, A. Johnson, and J. Wong, "A time-multiplexed FPGA," in *FCCM*, April 1997, pp. 22–28.

[2] A. DeHon, "Fundamental underpinnings of reconfigurable computing architectures," *Proc. IEEE*, vol. 103, no. 3, pp. 355–378, March 2015.

[3] T. Tuan, A. Rahman, S. Das, S. Trimberger, and S. Kao, "A 90-nm Low-Power FPGA for Battery-Powered applications," *IEEE Trans. Computer-Aided Design*, vol. 26, no. 2, pp. 296–300, 2007.

[4] E. Kadric, D. Lakata, and A. DeHon, "Impact of Memory Architecture on FPGA Energy Consumption," in *FPGA*, 2015, pp. 146–155.

[5] A. DeHon, "DPGA Utilization and Application," in *FPGA*, February 1996, pp. 115–121.

[6] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose, and V. Betz, "VTR 7.0: Next generation architecture and CAD system for FPGAs," *ACM Tr. Reconfig. Tech. and Sys.*, vol. 7, no. 2, pp. 6:1–6:30, Jul. 2014.

[7] D. Jones and D. Lewis, "A time-multiplexed FPGA architecture for logic emulation," in *CICC*. IEEE, May 1995, pp. 495–498.

[8] S. Scalera and J. Vázquez, "The design and implementation of a context switching FPGA," in *FCCM*, April 1998, pp. 78–85.

[9] T. R. Halfhill, "Tabula's time machine," *Microprocessor Report*, March 29 2010.

[10] NCSU, "45 nm physical design kit," http://www.eda.ncsu.edu/wiki/FreePDK45:Contents, 2007.

[11] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, Massachusetts, 02061 USA: Kluwer Academic Publishers, 1999.

[12] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic," *IEEE J. Solid-State Circuits*, vol. 32, no. 7, pp. 1079–1090, 1997.

[13] T. Pi and P. Crotty, "FPGA lookup table with transmission gate structure for reliable low-voltage operation," 2003, US Patent 6,667,635.

[14] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to model large caches," HP Labs, Palo Alto, CA, HPL 2009-85, April 2009, latest code release for CACTI 6 is 6.5. [Online]. Available: http://www.hpl.hp.com/techreports/2009/HPL-2009-85.html

[15] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, "Power modeling and characteristics of field programmable gate arrays," *IEEE Trans. Computer-Aided Design*, vol. 24, no. 11, pp. 1712–1724, Nov. 2005.

[16] B. S. Landman and R. L. Russo, "On pin versus block relationship for partitions of logic circuits," *IEEE Trans. Comput.*, vol. 20, pp. 1469–1479, 1971.

[17] 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*, February 1999, pp. 125–134.

[18] D. Lewis, V. Betz, D. Jefferson, A. Lee, C. Lane, P. Leventis, S. Marquardt, C. McClintock, B. Pedersen, G. Powell, S. Reddy, C. Wysocki, R. Cliff, and J. Rose, "The Stratix routing and logic architecture," in *FPGA*, 2003, pp. 12–20.

[19] A. Mishchenko, S. Chatterjee, and R. K. Brayton, "Improvements to technology mapping for LUT-based FPGAs," *IEEE Trans. Computer-Aided Design*, vol. 26, no. 2, pp. 240–253, February 2007.

[20] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proceedings of the 19th Design Automation Conference*, 1982, pp. 175–181.

[21] R. Graham, "Bounds on multiprocessor timing anomalies," *SIAM J. Appl. Math*, vol. 7, pp. 416–429, 1969.

[22] D. S. Hochbaum, Ed., *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.

[23] A. DeHon and R. Rubin, "Design of FPGA Interconnect for Multilevel Metalization," *IEEE Trans. VLSI Syst.*, vol. 12, no. 10, pp. 1038–1050, October 2004.

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

[25] A. Yan, R. Cheng, and S. J. E. Wilton, "On the sensitivity of FPGA architectural conclusions to experimental assumptions, tools, and techniques," in *FPGA*, 2002, pp. 147–156.

[26] F. F. Khan and A. Ye, "Measuring the accuracy of minimum width transistor area in estimating FPGA layout area," in *FCCM*, May 2015, pp. 223–226.

[27] K. Tatsumura, M. Oda, and S. Yasuda, "A pure-CMOS nonvolatile multi-context configuration memory for dynamically reconfigurable FPGAs," in *ICFPT*, 2015, pp. 215–222.

[28] "International technology roadmap for semiconductors," <http://www.itrs.net/Links/2012ITRS/Home2012.htm>, 2012.