Power-efficient RAM Mapping Algorithms for FPGA Embedded Memory Blocks

Russell Tessier, *Member, IEEE,* Vaughn Betz, *Member, IEEE,* David Neto, Aaron Egier, *Member, IEEE,* and Thiagaraja Gopalsamy

Abstract— **Contemporary FPGA design requires a spectrum of available physical resources. As FPGA logic capacity has grown, locally-accessed FPGA embedded memory blocks have increased in importance. When targeting FPGAs, application designers often specify high-level memory functions which exhibit a range of sizes and control structures. These logical memories must be mapped to FPGA embedded memory resources such that physical design objectives are met. In this work a set of power-efficient logical-to-physical RAM mapping algorithms are described which convert user-defined memory specifications to on-chip FPGA memory block resources. These algorithms minimize RAM dynamic power by evaluating a range of possible embedded memory block mappings and selecting the most power-efficient choice. Our automated approach has been validated with both simulation of power dissipation and measurements of power dissipation on FPGA hardware. A comparison of measured power reductions to values determined via simulation confirms the accuracy of our simulation approach. Our power-aware RAM mapping algorithms have been integrated into a commercial FPGA compiler and tested with 34 large FPGA benchmarks. Through experimentation, we show that, on average, embedded memory dynamic power can be reduced by 26% and overall core dynamic power can be reduced by 6% with a minimal loss (1%) in design performance. Additionally, it is shown that the availability of multiple embedded memory block sizes in an FPGA reduces embedded memory dynamic power by an additional 9.6% by giving more choices to the CAD algorithms.**

***Index Terms*— power demand, field programmable gate arrays, memory architecture, design automation**

I. Introduction

As field-programmable gate arrays (FPGAs) have grown in logic capacity, the need for on-chip data storage has increased since almost all modern designs contain memory. In contemporary FPGAs, most on-chip storage is implemented in large RAM blocks integrated into the FPGA architecture. These storage blocks allow for the implementation of a variety of memory structures, including FIFOs, scratch pad memories, and shift registers, within close physical proximity of logic resources. Due to their extensive use, embedded memory blocks have been found to consume between 10-20% of core dynamic power in typical FPGA designs [1]. For example, Fig. 1 illustrates the core dynamic power breakdown for 124 FPGA designs of varying sizes and functionalities mapped to Altera Stratix II [2] devices. On average, embedded memory consumes as much power as lookup tables (LUTs) in these designs. As the amount of FPGA logic and on-chip memory increases over the next few years and the application domain of FPGAs expands to include mobile and power-sensitive environments, the power-efficient use of embedded memory blocks will become increasingly important.

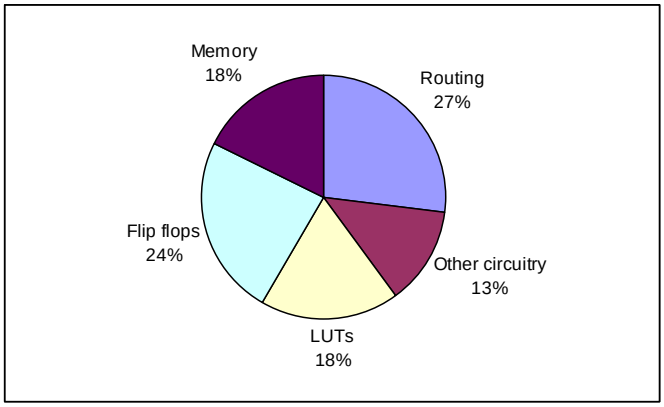

Fig. 1: Core dynamic power distribution for 124 benchmarks mapped to Stratix II devices. Test vectors are not available for these designs, so logic is assumed to toggle during 12.5% of clock cycles.

Embedded memory blocks in contemporary FPGAs are typically implemented with synchronous SRAM [2][19] to improve design performance. Like other synchronous SRAM architectures, FPGA embedded memory accesses are performed in concert with a design clock and a series of interface signals including read/write (R/W) enables, clock enables, address, and data signals. During application development, designers usually do not specify RAM blocks that precisely match the size and fully specify the control signals of the physical RAM blocks on an FPGA. More typically, a higher-level, *logical memory* representation is specified and the CAD flow automatically implements this specification using *physical memories* and any required control

Manuscript received March 15, 2006. Revised July 28, 2006

Russell Tessier is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003 USA (e-mail: tessier@ ecs.umass.edu). Vaughn Betz, David Neto, and Aaron Egier are with the Altera Toronto Technology Centre, Toronto ON CANADA. Thiagaraja Gopalsamy is with Altera Corporation, San Jose, CA USA.

circuitry.

Synchronous FPGA embedded memories primarily consume dynamic power as a result of internal RAM clocking. To save power, RAM control signals can be configured to suppress internal clocking when RAM access is unnecessary on a specific clock cycle. Although user-defined or generated control signals provide for valid functional embedded memory behaviour, their configuration may not efficiently suppress unnecessary clocked memory accesses, leading to wasted RAM dynamic power. These limitations motivate the development of RAM mapping algorithms that take power objectives into account while maintaining valid functional behaviour.

In this paper we describe a series of algorithms to automatically map user-specified logical memories to available physical embedded memory block resources with the goal of reducing overall FPGA dynamic power consumption. In considering feasible RAM mappings, our approach estimates the relative dynamic power consumption of each potential implementation and selects the most power-efficient implementation subject to on-chip RAM availability constraints. When necessary, user-specified RAM control signals (R/W enable, clock enables) are remapped to achieve a logically-equivalent RAM implementation with reduced dynamic power consumption. If an FPGA contains embedded memory blocks of different sizes, a mapping using each block type is considered.

Our mapping techniques have been integrated into the Altera Quartus II synthesis system [1] and targeted to several Altera FPGA families which contain embedded memories. To determine the benefit of our approach we evaluate the power reduction for 34 designs which contain RAM using a power estimation methodology based on digital simulation and circuit-level power models. To evaluate the accuracy of the simulation-based approach, we first map a sample set of six large FPGA designs and a group of primitive RAM instantiations to an FPGA-based board which allows for accurate dynamic power measurements. Through experimentation with the sample designs, it is shown that the measured and predicted power savings due to our mapping optimizations differ by a little more than one percentage point. Subsequent simulation-based experimentation with the 34 RAM-based designs demonstrates an average embedded memory dynamic power reduction of 26% and overall core dynamic power reduction of 6% on Stratix II devices. Additionally, the availability of multiple types of embedded memory blocks in an FPGA device reduces memory dynamic power and overall core dynamic power by an additional 9.6% and 2%, respectively.

In the next section we discuss related power-aware memory mapping techniques. In Section III the basic operation of FPGA embedded memories is described along with details of the basic mapping flow used to translate user-specified logical memory to physical embedded memory blocks. Section IV provides the details of our power-aware RAM mapping techniques and supporting algorithms. Experimental results are presented in Section V. Section VI summarizes the paper and provides directions for future work.

II. Related Work

RAM dynamic power-reduction techniques for ASICs and microprocessor systems have been considered at the application-mapping, compiler, and circuit levels. Although these approaches provide insight into reducing FPGA embedded memory power, none are directly applicable. Several synthesis techniques for application-specific embedded systems create power-optimized memory structures based on application address traces. In Benini et al. [5], the memory trace of an embedded application is analyzed by an algorithm to determine the portion of program and data memory that is most frequently accessed. These addresses are then grouped into memory banks which are implemented with scratch pad memories. Infrequently accessed addresses are grouped into larger physical memory blocks. Later work by Cao et al. [6] extends this optimization to consider data width scaling. Wuytack et al. [18] have developed techniques to optimize the entire memory hierarchy of an application for power consumption based on application information. These previous approaches rely on application trace information to perform memory partitioning.

A number of compiler techniques have been developed for processor-based systems which optimize power while mapping data to fixed system memory resources. For example, in Unsal et al. [16], a series of memory locations for multimedia applications are remapped to a small, local scratch pad memory to save dynamic power. In Petrov and Orailoglu [13], the organization and power consumption of a translation look-aside buffer are adjusted on a per-application basis. In Gebotys [8], memory energy is managed through memory and register allocation using a network flow algorithm. In Ferrahi et al. [7], a compiler technique to optimize sleep mode operation for memories is described. Memory reactivations are minimized via scheduling to save dynamic power.

Numerous circuit-level techniques for power reduction have been explored [11] including reduced swing pre-decode lines, multi-stage address decoding, and divided word and bit lines, among others. These techniques may be used in the future by FPGA designers to reduce FPGA embedded memory block power and are additive to the approaches described in this paper.

Although FPGA logic and routing dynamic power reduction has been studied [10], these techniques were not applied to embedded memory blocks. Except for [15], previous research efforts that map design logic to embedded memory blocks in ASICs [4][14] and FPGAs [9] do not consider power optimization as a mapping goal.

This paper extends our earlier work in [15] by providing more detailed algorithm descriptions and many new experimental results. We validate our simulation-based power estimation approach by comparing the power reductions estimated via simulation with those measured on physical

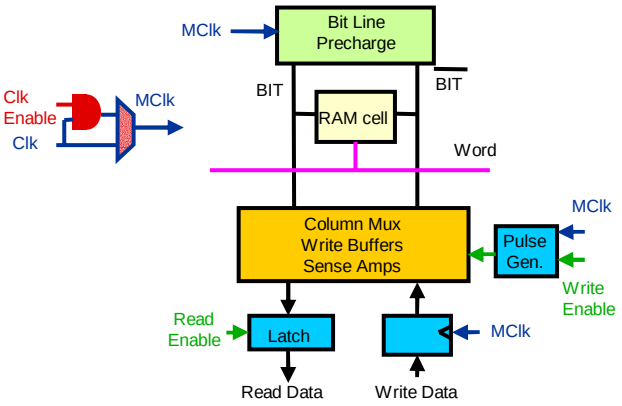

Fig. 2: Internal view of embedded memory read/write port

hardware for a subset of our design suite, and show that the results are very consistent. We also measure all power reductions due to our RAM mapping algorithm using a more recent version of the Quartus CAD tool than the one used in [15]. This more recent version of the Quartus software includes synthesis, placement and routing algorithms which incorporate power optimizations, so the results in this paper show that power-aware RAM mapping saves significant power even when a CAD suite incorporates other power-reduction techniques. Finally, this work extends [15] by evaluating the effect of RAM power-reduction techniques across multiple FPGA device families.

III. Background

The development of a power-efficient embedded RAM mapping strategy requires insight into the internal behaviour of synchronous SRAM. Typically, each port of an embedded memory block is controlled by one or more read/write (R/W) enable signals, clock (Clk) enable signals, and clock signals. As shown in Fig. 2, these signals directly or indirectly control data movement in different parts of the embedded memory port.

During a typical memory **read** operation the following events occur in sequence, in response to a rising clock edge:

- The memory port clock (MClk) is strobed causing the BIT lines to be precharged to Vcc.
- The read address is decoded and one word line is activated.
- The BIT line difference is identified by sense amps causing the read data to be strobed into a column multiplexer.
- Read data passes through the column multiplexer and a latch conditioned by Read Enable to the RAM external Read Data lines.

Memory **write** operations require a similar sequence of operations which occur in the following order:

- MClk is strobed causing the BIT lines to be precharged to Vcc.
- The Write Enable signal, conditioned by MClk, creates a write pulse which transfers write data to the write buffers and a word line is activated following write address decode.
- The write buffer data is stored in the RAM cell

For both synchronous read and write RAM operations, most dynamic power is consumed via BIT line precharging [12]. To control clocking, embedded memory ports often have a clock enable signal which can eliminate internal precharging, word-line decoding, and RAM cell access. The disabling of the clock enable signal when memory port access is not required provides the best technique to eliminate embedded memory dynamic power consumption for a memory port. If a RAM port is inactive on a given clock cycle and its clock can be suppressed via an inactive clock enable, the RAM port will not consume significant dynamic power.

A number of contemporary FPGAs support embedded memory blocks with enable and clock enable signals. Altera Stratix II and Cyclone II [3] devices support both R/W enables and clock enables on each of the two ports on every memory block. Each Xilinx Virtex-II [20] and Virtex-4 [19] embedded SelectRAM block contains write enable and clock enable control signals on each port, but no separate read enable. While Stratix II devices support three different embedded memory block sizes, Cyclone II, Virtex-II, and Virtex-4 contain embedded memory blocks of a single size.

The goal of power-aware RAM mapping is to implement the functionality of a user-defined RAM module (logical memory) in one or more FPGA embedded memory blocks so that memory precharges are limited. This optimization goal attempts to minimize RAM dynamic activity through the use of RAM port clock enables whenever possible. The effective use of clock enable signals ensures that the bulk of embedded memory block dynamic power is consumed when a *required* access to data within a RAM is performed. In some cases this goal may require the synthesis of one of more clock enable signals during the mapping process. This mapping must achieve the same functional behaviour for the RAM as specified by the designer while allowing for possible tradeoffs between design power consumption, area and performance.

A. Typical RAM Mapping Flow

FPGA embedded memory blocks are used to implement a variety of RAM components including FIFOs, shift registers, and single and dual-port memories. Logical RAMs are specified by the designer in RTL or schematic form, created by the FPGA compiler and mapped [1], as shown in Fig. 3:

1. **Logical memory creation** – User-defined RAM descriptions are processed by the FPGA compilation software to create logical memories with the desired characteristics.
2. **Logical-to-physical RAM processing** - Logical RAMs are converted into one or more RAM blocks which

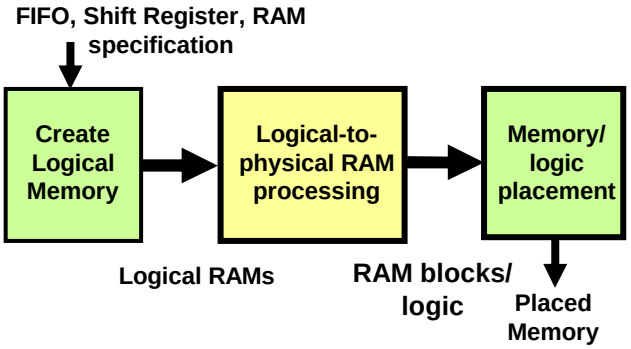

Fig. 3: Typical logical RAM to embedded memory block mapping flow

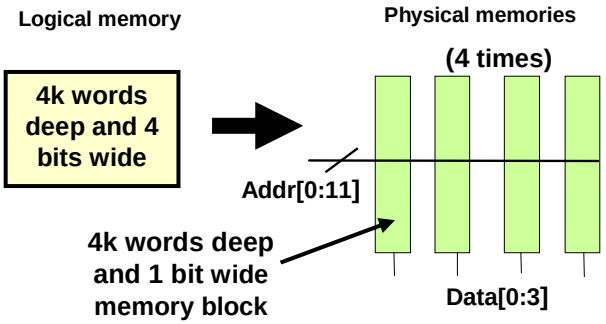

Fig. 4: Area-efficient mapping of a 4Kx4 logical RAM to 4 Kbit memory blocks

match the external interface and size constraints of available embedded memory blocks.

3. **Embedded memory block placement** – RAM blocks and associated control logic are assigned to available on-chip embedded memory block and logic resources.

The power-aware algorithms developed in this work are applied in the logical-to-physical RAM processing step. Traditionally, RAM mapping has targeted logical RAM performance and FPGA area minimization [9] rather than power consumption. As shown in Section IV.B, however, an area-optimal embedded memory implementation does not always minimize dynamic power. To conserve dynamic power it is desirable to map the required memory functions to the available physical memories so that power consumption is optimized, while meeting area and delay constraints.

The size of both logical and physical (embedded) memory blocks can be defined in terms of the number of addressable locations (*depth*) and output bits per memory (*width*). The number of address bits required for both logical and physical memories is directly related to memory block depth. The number of data in and data out bits is related to memory block width. To promote flexibility, an FPGA embedded memory block may typically be programmed to support a range of depth versus width configurations [2][19].

Until the relatively recent adoption of synchronous SRAMs, most user-defined RAM targeted asynchronous memories which use read and write enable for data access control. Although embedded memory blocks now allow for the use of either operation-specific enable or clock enable signals to provide access control, many designers continue to use the operation-specific enable approach, ignoring the clock enable. Contemporary RAM mapping flows (e.g. Fig. 3) automatically map these user-defined enable signals to the R/W enable signals located on the embedded memory block ports. Unspecified clock enables are set to be continuously active. The use of read and write enable signals for data access control instead of clock enable signals leads to sub-optimal power consumption in many cases.

A second impediment to reduced RAM power dissipation is related to logical RAM size. In most cases the size of a user-specified logical memory will not exactly match the width and depth dimensions of an embedded memory block. Since RAM mapping flows typically focus on optimizing delay and resource usage, rather than power, logical memories are usually mapped using a minimum of external logic. As an example, Fig. 4 illustrates the mapping of a 4Kx4 logical memory to four 4Kx1 embedded memory blocks. In this case, each memory block is configured as 4Kx1 so that a single bit of each addressable location is located in each block. This configuration requires no external logic. However, all four memory blocks must be active during each logical memory access, so this is a high-power implementation.

IV. Power-Aware RAM Mapping

Our RAM mapping approach consists of two algorithms that obtain a power-efficient mapping of logical memories to FPGA embedded memory blocks. Two specific cases are targeted:

1. Since most embedded memory block dynamic power is a result of clock-induced precharging, we identify cases where user-specified logical RAM read and write enable signals can be automatically **converted** or **combined** with corresponding read and write clock enable signals while maintaining correct functional behaviour.

2. For cases where more than one embedded memory block is required to implement a logical RAM, we implement a multi-banked RAM mapping. As a result of this banked mapping, only one embedded memory block is clocked per access. In some cases the banked structure may require the inclusion of supporting logic.

A. Conversion of read and write enable to read and write clock enable

In general, synchronous embedded memory blocks exhibit the same RAM behaviour if either an enable or a clock enable is used to control a read (or write) access and the alternate signal is set to an active state. If present, both read enable and read clock enable signals must be active to successfully perform an

```
If  Clken = 1 and Enable = User Enable
    Set Clken = User Enable and Enable = 1
If Clken = User Clken and Enable = User Enable
    Set Clken = User Enable & User Clken and Enable = 1
If Clken = User Clken and Enable = 1
    Perform no change
```

Fig. 5: Steps required for enable signal conversion and combining. This analysis is performed on each design logical RAM

embedded memory block read transaction [15]. Consider a scenario where a read enable input is attached to a control signal and read clock enable input is always tied to active logic 1. Since the inputs both must be active for reads, the read clock enable input can be driven by the signal previously tied to read enable input, and read enable input can be tied to logic 1.

Similarly, write enable and write clock enable signals must be active simultaneously to successfully perform an embedded memory block write transaction [15]. Consider a scenario where a write enable input is attached to a control signal and the write clock enable input is always tied to active logic 1. Since the inputs both must be active for writes, the write clock enable input can be driven by the signal previously tied to the write enable input, and the write enable input can be tied to logic 1.

The **conversion** of user-defined read and write enable signals to respective clock enables primarily reduces power by eliminating BIT line precharging when embedded memory block data access is not required. The same functional RAM behaviour is maintained. For some logical memories, a designer may specify both an enable and a clock enable signal for an embedded memory port. In these cases, simple conversion cannot be performed. Additional logic (an AND gate) must be added to the user design to allow the user-defined enable signal to condition the associated memory port clock. The **combining** of the enable and clock enable signal forms a new combined clock enable signal which can be attached to the memory port clock enable input. Depending on designer timing constraints, the addition of logic delay to the clock enable path may negatively impact mapped design performance. As a result, this approach may only be appropriate if design power reduction is considered more important than design performance or preliminary timing information is available to determine that performance is not likely to be affected.

The mapping steps in Fig. 5 are performed on each logical RAM. These steps perform enable-to-clock enable conversion and combining for embedded memory block inputs Clken and Enable and designer signals User Clken and User Enable.

B. Power-Aware RAM Partitioning

As shown in Fig. 4, a logical memory which exceeds the size of an embedded memory block must be mapped to multiple blocks. Although the mapping shown in Fig. 4 does not require

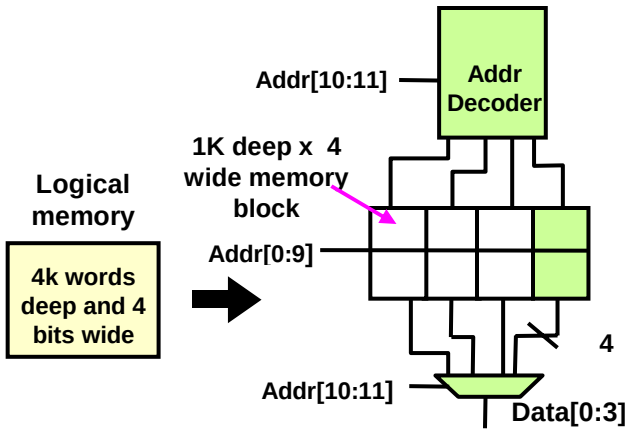

Fig. 6: Alternate mapping of a 4Kx4 logical RAM to 4 Kbit memory blocks

any supporting logic, each memory block is active during each memory access, requiring substantial power consumption. In this case, the *depth* of each physical memory block matches the depth of the logical memory and the width of each physical memory block is smaller than its logical memory counterpart. This mapping is an example of *vertical* memory slicing.

In general, an FPGA embedded memory block can be structured to have a variety of depth and width configurations, each with the same bit storage capacity. For example, an Altera M4K (4608 bit) embedded memory block can be organized into configurations ranging from 4096x1 to 256x18 [2]. This allows a range of choices in mapping a logical RAM to physical memory blocks. For example, Figs 4 and 6 provide two example mapping alternatives. In the mapping in Fig. 6, the *width* of each physical memory block matches the width of the logical memory while the depth of each physical memory block is reduced compared to its logical memory counterpart. This mapping can be considered an example of *horizontal* memory slicing. This second mapping requires the inclusion of address decoding circuitry to determine which memory block contains the requested data. Additionally, a multiplexer is required on the read port to select the requested word during read requests. Although dynamic power is consumed by the added address decoder and multiplexer, all but one of the embedded memory blocks is disabled during RAM accesses, saving considerable dynamic power. Unused memory blocks are disabled by connecting the outputs of the address decoder to memory block clock enable signals.

The vertical and horizontal RAM slicing implementations shown in Figs 4 and 6 represent the end points of a spectrum of feasible logical-to-physical RAM mappings (e.g. 2Kx2 RAM block configurations are also possible). If, as a result of a mapping change, an embedded memory block is converted from a given depth to one that is half as deep, the following additional mapping changes are required:

1. Each write port data line must be tied to twice the number of source/destination embedded memory blocks.

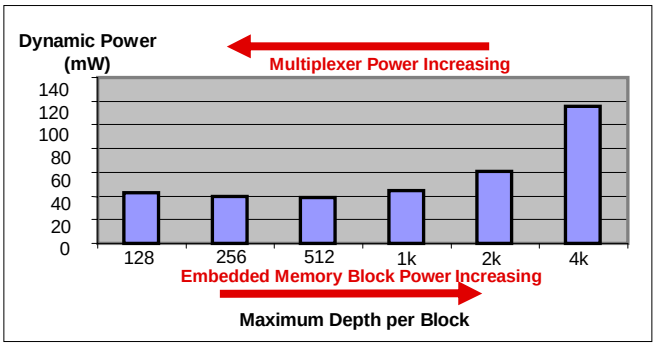

Fig. 7: Dynamic power consumption of a 4Kx32 logical RAM at 100 MHz in different slicing configurations

2. The size of the address decoder increases by a factor of 2.
3. The bit input size of each multiplexer on the embedded memory read port increases by a factor of 2.
4. One address line is removed from each of the embedded memory blocks.

The *relative* power consumed by each logical-to-physical mapping can be evaluated by assessing the power consumed by the memory blocks during a data access, the address decoder, the output multiplexer, and associated routing. As mappings approach the vertical slicing implementation (maximum physical block depth), memory block power is increased and multiplexer and address decoder power is decreased. As mappings approach the horizontal slicing implementation, multiplexer and address decoder power is increased and memory block power is decreased. Fig. 7 shows the dynamic power consumed by various mappings of a 4Kx32 logical RAM in a Stratix II device for a selection of embedded memory block depths as reported by the Quartus II PowerPlay power analyzer. The plot shows that the power optimal mapping for this logical RAM falls between the horizontal slicing on the left and vertical slicing on the right. All mappings achieve the same functional behaviour.

C. *Logical RAM Partitioning Algorithm*

A power-aware RAM partitioning algorithm has been developed to evaluate the relative power consumption of a series of logical-to-physical RAM mappings. Each mapping is evaluated based on the number of active embedded memory blocks per port, the required address decoder and multiplexer circuitry, and the estimated routing required. Since contemporary FPGAs contain a set of different embedded memory block sizes, mapping evaluation is performed for each block type to determine the most power-efficient choice.

The relative cost for each mapping is determined based on the estimated dynamic power consumption of the mapping. This cost can be expressed for each port of each logical RAM:

$$\mathbf{Cost} = \mathbf{W} * \mathbf{P_{mux}} + \mathbf{N} * \mathbf{P_{ram}} + \mathbf{P_{addr\_decode}} \qquad (1)$$

where **Cost** is the relative power cost for the mapping, **W** is the width of the logical RAM, $\mathbf{P_{mux}}$ is the per-bit dynamic power of a read port multiplexer, **N** is the number of required embedded memory blocks, $\mathbf{P_{ram}}$ is the per-block dynamic power, and $\mathbf{P_{addr\_decode}}$ is the dynamic power consumption of the address decoder. Specific algorithm steps are shown in Fig. 8. An initial, resource-feasible mapping is first determined for each logical memory based on vertical memory slicing. The goal of this step is to find a RAM mapping such that all the

1. For each logical memory
   a. Identify smallest memory block type with depth ≥ logical memory depth
      i. Determine number of embedded memory blocks, **N**
   b. For each logical memory port
      ii. Look-up memory block power, $\mathbf{P_{ram}}$
      iii. Set $\mathbf{Cost_{port}}$, for logical memory port to $\mathbf{N} * \mathbf{P_{ram}}$
   c. Sum $\mathbf{Cost_{port}}$ values across ports to form $\mathbf{Cost_{init}}$
   d. Store mapping and $\mathbf{Cost_{init}}$ for logical memory
2. Verify mappings of all logical memories are feasible within device memory constraints.
   a. If mapping is infeasible, Retry 1 with alternate embedded memory block mappings
3. For each logical memory
   a. For each memory block type
      i. For each possible memory block depth and width configuration
         1. Determine **N** and the size of the address decoder.
         2. For each logical memory port
            a. Look-up memory block power, $\mathbf{P_{ram}}$
            b. Scale $\mathbf{P_{ram}}$ by number of memories, **N**
            c. Look up per-bit dynamic power of bit of output read port multiplexer, $\mathbf{P_{mux}}$
            d. Scale $\mathbf{P_{mux}}$ by read port width, **W**
            e. Look up dynamic power of address decoder, $\mathbf{P_{addr\_decode}}$
            f. Sum power components to determine $\mathbf{Cost_{port}}$ via Eq. (1)
         3. Sum $\mathbf{Cost_{port}}$ values across logical memory ports to form $\mathbf{Cost_{type}}$
      ii. Store depth and width of configuration with minimum $\mathbf{Cost_{type}}$ if $\mathbf{Cost_{type}} < \mathbf{Cost_{init}}$
   b. Rank memory block types by $\mathbf{Cost_{type}}$
   c. Set $\mathbf{Cost_{mem}}$ to the minimum $\mathbf{Cost_{type}}$ value
4. Rank logical memories by $\mathbf{Cost_{init}} - \mathbf{Cost_{mem}}$
5. For each logical memory in ranked order
   a. Select memory block type which has minimum $\mathbf{Cost_{type}}$ and is feasible.

Fig. 8: Power-aware memory partitioning algorithm

logical memories are mapped to physical memories without exceeding the supply of any RAM block or logic resource in the FPGA. Next, all possible logical-to-physical mappings are enumerated and evaluated based on their estimated dynamic power consumption. The minimum power depth and width configuration for each memory block type is then stored for each logical memory. Following enumeration, the initial physical mapping for each logical memory is replaced by a lower-power alternative mapping, subject to embedded memory block availability. This final step is ordered so that re-mappings that will save the most dynamic power are considered first.

Our approach is effective for both single- and dual-port logical RAMs. The key power savings aspect of the approach is the connection of address decoder outputs to embedded memory block clock enables. Only the *addressed* memory block is precharged on a given clock cycle, saving considerable RAM dynamic power.

The inclusion of a memory block read port multiplexer can negatively impact design performance for designs which include the RAM block output on the design critical path. Design performance is not explicitly considered by the partitioning algorithm. However, to minimize performance impact, only configurations which require a 4-to-1 or smaller multiplexer on each read port output bit are considered. Larger multiplexers require additional levels of logic (at least 2) and routing, significantly reducing design performance.

In addition to possibly affecting performance, the inclusion of multiplexers consumes device logic. This added logic may result in an overflow of required design logic elements for a target device, so care must be taken not to add excessive logic.

D. Parameter Evaluation

The algorithms described in Sections IV.A and IV.C have been integrated into Quartus II and are included in 5.1 and later versions. Before experimental results on a range of benchmarks were evaluated, the technology parameters noted in Eq. (1) were determined via experiments with a representative set of logical RAMs. The RAMs used for parameter evaluation include ROMs and single and dual port RAMs of sizes ranging from 512x2 to 8Kx132. Parameter evaluation was performed for the Altera Stratix II architecture, which contains three types of embedded memory blocks, each of a different size: 576 bit (M512), 4,608 bit (M4K), and 589,824 bit (M-RAM) [2]. Each memory block allows for implementation of both single and dual-port synchronous RAMs.

Each logical RAM used for parameter evaluation was mapped to each of the three Stratix II memory block types using multi-block partitioning ranging from horizontal slicing to vertical slicing. Following synthesis with Quartus II, the memory designs were placed and routed using Quartus II. All synthesis, place, and route steps used an unattainable 1 GHz timing constraint to ensure maximum optimization effort by the CAD software. A digital simulation of each design at 100 MHz with random input vectors was performed to find the toggle rate of each signal. This simulation includes *glitch filtering*, where changes in logic state that are too rapid to propagate through the device routing or functional blocks are removed from the simulation waveform – this improves power estimation accuracy [1]. Dynamic power was then estimated by using the Quartus II PowerPlay power analyzer to combine the signal toggle rates with detailed models of the power dissipated by FPGA circuitry for each toggle. All the designs were able to satisfy a minimum clock frequency of 100 MHz.

Statistical averaging was then used to determine the following values based on the reported power estimates for all RAM implementations:

- Power consumed by a single bit of an **n**-to-1 multiplexer, $\mathbf{P_{mux}}$, Values for only 2-to-1 and 4-to-1 multiplexers were determined since shallower embedded memory blocks depth slicings are not performed by our system due to performance concerns.
- Per-port design power consumed by an active physical memory block, $\mathbf{P_{ram}}$, for an M512, M4K, and M-RAM embedded memory block.
- Power consumed by a **k**-to-**n** address decoder, $\mathbf{P_{addr\_decode}}$, for a 2-to-4 and 1-to-2 decoder.

The variance of the multiplexer, memory block, and address decoder dynamic power parameters listed above across the various designs was found to be less than 1%. Variations of the parameter values across devices in the same device family were found to be negligible since logic block and routing constructs are consistent across devices. Because the power analyzer takes detailed placement and routing into account when producing a power estimate, the averaged values for $\mathbf{P_{mux}}$ and $\mathbf{P_{addr\_decode}}$ take the effects of control signal, address, and data net fanout into account.

Although the calculated parameters measure dynamic power values averaged across the RAM parameter evaluation design set, the access patterns of user logical RAMs may differ. Since our algorithm considers relative rather than absolute dynamic power values in making tradeoffs, we consider the subsequent use of these parameters across a range of user benchmarks to be acceptable and representative of most RAM access patterns.

V. Results

A. Validation of Simulation-Based Power Estimation

The power-saving benefits of our approach have been tested experimentally using both physical measurements and power estimates obtained via the digital simulation and Quartus II power analyzer flow described in Section IV.D.

In an initial experiment, six designs which contain memory were mapped to a board which allows for accurate FPGA power measurements. These designs include:

Table I: Measured versus predicted changes in RAM dynamic power

	Circuit Characteristics			Unopt. power	Enable convert			Enable convert/combine + mem partition		
	LUTs	Memory bits	Flip flops	Meas. (mW)	**% change** Predict	**% change** Meas.	Absol Diff.	**% change** Predict	**% change** Meas.	Absol Diff.
des3_6	5850	2688	1014	2240.6	-0.4%	0.0%	0.4%	-0.3%	0.0%	0.3%
fft	4585	286956	4039	136.5	0.0%	-0.1%	0.1%	-8.3%	-11.3%	3.0%
512x9_x72	840	331776	762	108.5	-3.2%	-4.8%	1.6%	-3.2%	-4.8%	1.6%
fir_63tap_18bit	3331	1134	3263	189.9	-0.5%	0.0%	0.5%	-0.5%	0.0%	0.5%
1024x18_x36	1414	663552	781	220.0	-2.1%	-5.2%	3.1%	-32.4%	-35.1%	2.7%
jpeg	12068	1164986	9390	747.9	-15.5%	-15.9%	0.4%	-18.7%	-18.2%	0.5%
average					***-3.6%***	***-4.3%***	***1.0%***	***-10.6%***	***-11.6%***	***1.4%***

Fig. 9: Test platform for Stratix II EP2S60 power measurements

- **des3_6** – an implementation of the triple data encryption standard algorithm using six pipeline stages. This design is derived from the circuit described in [17].
- **fft** – an implementation of the Fast Fourier Transform algorithm including buffer storage.
- **512x9_d72** - 72 instances of a 512x9 RAM block. Each block contains one read port and one write port.
- **fir_63tap_18bit** – a 63 tap FIR filter with 18 bit data and coefficients.
- **1024x18_x36** - 36 instances of a 1024x18 RAM block. Each block contains one read and one write port.
- **jpeg** – an implementation of a JPEG image encoder.

Table I lists LUT, memory bit, and flip flop counts for each design. The memory instance designs (**512x9_d72** and **1024x18_x36**) contain some memory block selection logic which accounts for reported LUTs and flip flops. Although the number of designs in this set is insufficient to fully assess the power saving potential of our mapping approaches, they provide a sufficient sample set to validate the relative accuracy of our simulation-based power estimation approach versus physical measurement. The estimation approach is subsequently applied to a substantially larger design set.

As shown in Fig. 9, the board used for power measurement contains an Altera Stratix II EP2S60 device. The power rails for the FPGA are isolated and individually regulated to allow precise measurement of supply current per power rail. The six designs described above were coded in architecture independent HDL. Each design was wrapped inside the FPGA with a random vector generation circuit which generates a sequential series of test vectors. The structure of all six designs is such that random vectors provide a reasonable input stimulus. This circuit reduces the external signal requirements of the FPGA to a clock signal and an enable signal. The designs were compiled using Altera Quartus II version 6.0 at an unattainable 1 GHz target clock frequency. The Quartus II software includes several power optimization algorithms, and all algorithms except the power-aware RAM mapping algorithm were set to their highest effort level.

The dynamic power consumption of each design was measured in a series of tests. Following device configuration, FPGA power consumption was first measured with an input clock frequency of 0 MHz, providing the design static power consumption. Then, overall FPGA power is measured at a series of clock frequencies, up to the maximum frequency of the design. The slope of the plotted line connecting these points provides a dynamic power mW/MHz ratio for the design. The ratio is then multiplied by 50 to determine measured design dynamic power at 50 MHz. For comparison, the same designs were simulated at 50 MHz and dynamic power analysis was performed using the Quartus II PowerPlay power analyzer to determine predicted power.

Each design was compiled three times using Quartus II, each time with a different RAM power optimization setting, as listed below.

Power optimization cases:

1. No RAM power optimization
2. Read/write enable conversion to read/write clock enable.
3. Memory partitioning in addition to read/write enable conversion and combining

The optimizations for Cases 2 and 3 were described in Section IV. The dynamic power consumption of each design at 50 MHz was determined using the physical measurement and simulation methods described above for each case. The *percentage change* in dynamic power for a design compiled using RAM power optimization (e.g. Case 2 or Case 3) versus no optimization (e.g. Case 1) is:

$$\textbf{\% change} = 100 * (\mathbf{P_{opt}} - \mathbf{P_{unopt}}) / \mathbf{P_{unopt}} \quad (2)$$

where $\mathbf{P_{unopt}}$ is design dynamic power if RAM power optimization is not used and $\mathbf{P_{opt}}$ is the dynamic power of the design compiled with optimization. Percent changes can be individually determined for values predicted via simulation ($\mathbf{\% change_{predict}}$) or physically measured ($\mathbf{\% change_{meas}}$). The absolute difference in power saving percentage between the simulated (predicted) and measured dynamic power values for each design optimized under Case 2 or Case 3 is:

$$\textbf{Abs diff} = |\ \mathbf{\% change_{predict}} - \mathbf{\% change_{meas}}| \quad (3)$$

Table I shows the measured and predicted percentage change in dynamic power for the six designs based on Case 2 and 3 power optimization cases. The measured power consumption for unoptimized compilation (Case 1) is provided for reference. The maximum per-design absolute difference in power saving percentage for measured and predicted values is 3.1% for enable conversion (*1024x18_x36*) and 3.0% for partitioning, enable conversion and combining (*fft*). On average, the absolute difference in power saving percentage for the six designs was 1.0% for enable conversion and 1.4% for all techniques.

Table II: Benchmark design statistics for 34 designs

Design	LUTs	Memory bits	Flip flops	Target Device
1	4617	246488	6283	EP2S15F672C3
2	12005	66336	12366	EP2S90F1508C3
3	12005	66336	12366	EP2S90F1508C3
4	7130	43008	4013	EP2S30F672C3
5	6348	231970	6813	EP2S15F672C3
6	11145	548	12951	EP2S60F1020C3
7	8199	292608	5388	EP2S15F672C3
8	3395	63744	7362	EP2S60F1020C3
9	3697	6432	5944	EP2S60F1020C3
10	18890	327680	3241	EP2S60F1020C3
11	225	331776	43	EP2S60F672C3
12	256	331776	77	EP2S60F672C3
13	20833	327680	3312	EP2S90F1508C3
14	2424	512	4919	EP2S60F1020C3
15	533	8192	273	EP2S60F672C3
16	35404	89600	19465	EP2S90F1508C3
17	8938	36096	5461	EP2S30F672C3
18	5481	47264	6993	EP2S15F672C3
19	8356	65536	8461	EP2S30F672C3
20	2583	35280	3260	EP2S60F1020C3
21	6542	128452	22139	EP2S60F1020C3
22	3283	94588	3886	EP2S15F672C3
23	26310	270336	29394	EP2S60F1020C3
24	3584	111872	4695	EP2S15F672C3
25	9005	426512	10416	EP2S60F1020C3
26	7527	98304	4497	EP2S60F1020C3
27	2136	126118	2693	EP2S15F672C3
28	3706	168416	6972	EP2S15F672C3
29	12320	184320	15075	EP2S60F1020C3
30	13993	88048	25516	EP2S60F1020C3
31	15907	337501	18874	EP2S30F672C3
32	8911	293856	9164	EP2S30F672C3
33	4248	153864	2842	EP2S60F1020C3
34	18553	996096	12816	EP2S60F1020C3

As a final test of the accuracy of our simulation-based power evaluation approach, a set of 150 ROMs and single and dual-port RAMs ranging in size from 256x1 to 128Kx9 were mapped to a Stratix II device and evaluated using the procedure outlined above. One instantiation of each design RAM was evaluated in each test. On average, read/write enable conversion resulted in a measured 0.28% reduction in core dynamic power (0.27% predicted). The low power reduction for this method was primarily due to the presence of a designer-specified clock enable on many designs. Memory partitioning in addition to read/write enable conversion and combining resulted in a measured 16.1% core dynamic power reduction (18.1% predicted). Maximum per-design and average absolute differences in power reduction percentage were similar to the six designs noted above. Although these RAMs do not reflect typical FPGA designs, they provide an additional platform for the validation of the simulator.

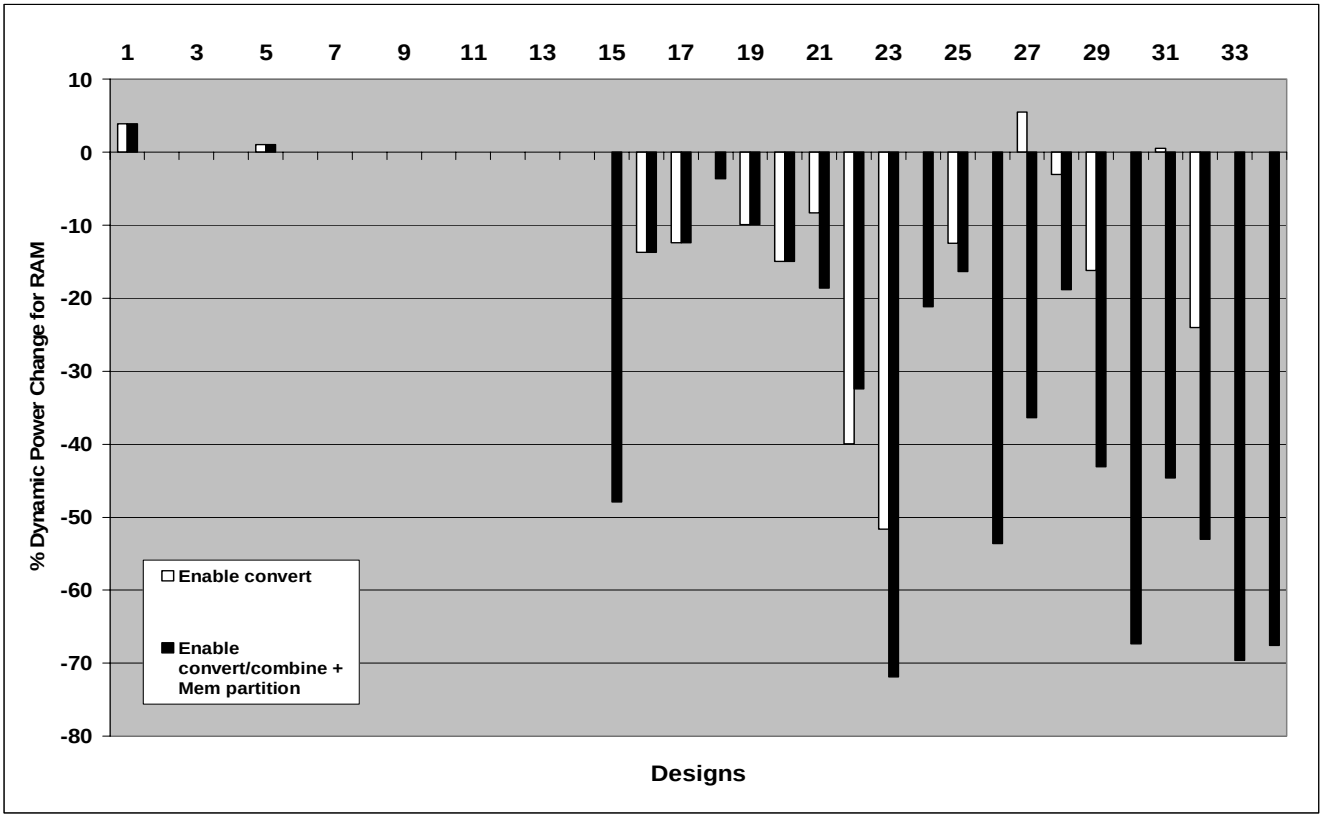

Figure 10: Percentage change in data RAM power for benchmark designs due to RAM power optimizations

Since many of the designs in our benchmark suite cannot be adequately exercised by random input vectors, we cannot physically measure power reductions on the entire suite. However, the power reduction accuracy of both of these calibration experiments validates the use of our simulation-based power evaluation approach on the larger set of benchmark circuits.

B. Power-Aware Mapping Algorithm Results for Stratix II

Following the determination of the tuning parameters, the integration of our algorithms with Quartus II, and the validation of our simulation-based power evaluation approach, experimentation was performed on 34 designs with test vectors. LUT, memory bit and flip flop counts for each design are shown in Table II. This benchmark set includes designs from the encryption, signal processing, and communications processing domains. Most of the benchmark set consists of proprietary designs collected by Altera and used for evaluation of current and future FPGA architectures. Several designs are Altera-created IP cores (e.g. Design 6 is an FIR filter, Design 7 is a network interface core, and Design 16 is a digital correlator). Designs 11 and 12 are single- and dual-port versions, respectively, of the **512x9_x72** design described in Section V.A. As discussed later in this section, the designs in Table II are ordered by achieved overall power savings (smallest to largest). The ratio of memory bits to LUTs in the designs is consistent with the ratio found in Stratix II devices (about 70 memory bits per LUT).

Table III: Benchmark power statistics for 34 test designs

Average % dynamic power - embedded block memory	22.3%
Average % dynamic power – combinational logic	15.6%
Average % dynamic power – registers	25.9%
Average % dynamic power – routing	24.1%

As seen in Fig. 3, optimization occurs after complex memory functions (e.g. FIFOs, shift registers) are converted to logical RAMs, but before structures are assigned to specific embedded memories. The specific device used for each design is listed in Table II. The 34 designs were targeted to the smallest Stratix II device which would hold them.

Designs were implemented using the compilation settings described in Section V.A, and dynamic power consumption was evaluated using the power analysis flow detailed in Section IV.D. A series of test vectors were used to simulate each design at 100 MHz to obtain the toggle rate of each signal. Dynamic power analysis was then performed with the Quartus II PowerPlay power analyzer using these toggle rates. In contrast to the experimentation described in [15], all Quartus II power optimizations, except for our RAM power optimization algorithms where noted, were included during experimentation.

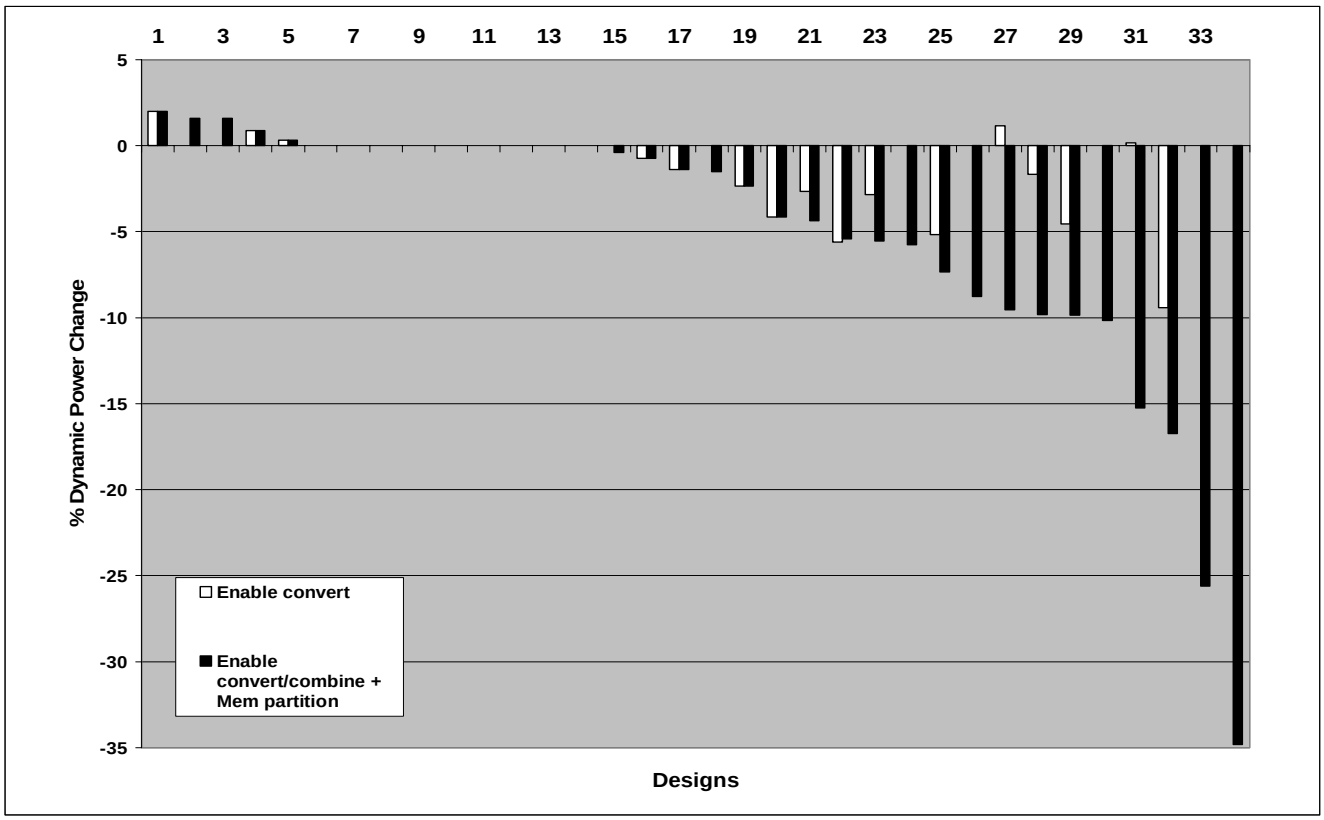

Figure 11: Percent change in overall core dynamic power for benchmark designs due to RAM power savings

To validate our approach, a series of experiments were performed using combinations of the algorithms with the 34 benchmark circuits. Dynamic power statistics related to the benchmarks appear in Table III for initial compilation with default parameters and no RAM power optimizations. Dynamic power percentages were determined versus overall design core dynamic power. As shown in Table III, RAM dynamic power forms a significant part of average design core dynamic power. In addition to compilation without RAM power optimizations, each design was compiled using the following combinations of automatic RAM power optimizations described in Section IV.

Power optimization cases:

1. Read/write enable conversion to read/write clock enable.
2. Read/write enable combining with an existing clock enable in addition to read/write enable conversion.
3. Memory partitioning in addition to read/write enable conversion and combining.

A bar graph illustrating the per-design percent change in memory dynamic power due to these optimizations for Cases 1 (enable conversion) and 3 versus compilation with no RAM power optimization appears in Fig. 10. The dynamic power percentage change is computed via Eq. 2. Fig. 11 shows the percent change in overall core dynamic power. The designs appear in the same order numerically in each plot and in Table II. Case 3 data for each graph includes any increase in combinational logic and register dynamic power due to logic added for multiplexing, address decoding, and clock enable combining.

Table IV: Summary of RAM optimization results for 34 benchmark designs (all averages geometric). Each percentage represents the percent change of a value obtained via compilation with no RAM power optimization due to the specified optimization

	R/W Enable convert	R/W Enable convert/ combine	R/W Enable convert/ combine + Mem partition
Core dynamic power	-1.1%	-1.5%	-5.5%
Memory dynamic power	-6.8%	-10.1%	-26.0%
Max clk freq	0.0%	-0.4%	-1.1%
LUT count	0.0%	0.0%	0.4%

These plots show that although some designs achieve no benefit from the new approaches, others benefit significantly (up to 72% RAM dynamic power and 35% overall core dynamic power).

Table IV shows the average percentage change in core and RAM dynamic power for all three cases. The use of memory partitioning more than doubles the average core dynamic power savings (5.5% vs. 1.5%) and RAM dynamic power savings (26.0% vs. 10.1%).

Table V: Summary of RAM optimization results for logical RAMs targeted to specific embedded memory blocks versus unconstrained RAM placement using 34 benchmark designs.

	M512	M4K	M-RAM
Designs completed	21	34	5
Core dynamic power	19.6%	1.9%	41.6%
Memory power	149.1%	9.6%	433.0%
Max clk freq.	-3.6%	1.4%	-13.4%
LUT count	2.4%	0.0%	-0.4%

Table IV also shows that the RAM dynamic power optimizations have little effect on area or performance. The percentage reduction in the achievable average design clock is shown in the table for all three cases. As expected, Case 3, which includes memory partitioning, exhibits the largest performance loss due to the inclusion of multiplexers at the logical RAM output (1.1%). As discussed in Section IV.C, this performance loss was mitigated by our restriction of a maximum 4-to-1 read port output bit multiplexer size.

Case 3 also shows the largest increase in required LUTs (0.4%), primarily used to implement multiplexing logic. Case 1 (enable conversion) requires no additional logic and shows minimal performance decrease. Although not optimal, our memory partitioning algorithm is effective. For each design except Design 5, sufficient embedded memory block resources were available in the target FPGA to select the power optimal logical-to-physical memory mapping for each logical memory (Case 3). On average, memory partitioning required 0.7 sec. (worst case 10 sec. for Design 16). This represents 0.05% of Quartus II synthesis time on average (1.3% worst case for Design 16).

C. The Use of Multiple Embedded Memory Block Sizes to Reduce Dynamic Power

As stated in Section IV.C, the memory partitioning algorithm considers mapping each logical memory to each type of embedded memory block on a target device and selects the most power-efficient implementation relative to available resources. To illustrate the dynamic power benefits of the availability of multiple embedded memory block sizes on a target FPGA we re-mapped each of the 34 benchmark designs to a Stratix II EP2S180 using the constraints described in Section IV.D. Four separate compiles were performed for each design, each using one of the following constraints:

a. Memory partitioner selects the target physical embedded memory for each logical memory
b. All logical memories mapped to M512s
c. All logical memories mapped to M4Ks
d. All logical memories mapped to M-RAMs

For each compile, all Quartus II power optimizations, including RAM power optimizations, were used, including memory partitioning. Due to RAM resource limitations it was not possible to successfully map all designs for Cases b, c, and d. Table V shows the number of designs that were successfully mapped for each case and the percentage changes for Cases b, c, and d mapping versus Case a for several parameters. Although it was possible to map all designs using solely M4Ks for embedded memory, a 9.6% RAM power and 1.9% core dynamic power penalty was observed. More drastic results versus the base case were observed by restricting memory mapping to solely M512s and M-RAMs. For example, RAM dynamic power for M512-only mapping more than doubled (e.g. a 149% increase indicates a final value 2.49x larger than the original). Table VI shows similar percentage change results for the 21 designs that were successfully mapped for Cases a, b and c. These results indicate that if only one type of embedded memory block could be included in an FPGA, a block of intermediate size would be best for power efficiency.

Table VI: Summary of RAM optimization results for logical RAMs targeted to M512s and M4Ks versus unconstrained RAM placement using 21 benchmark designs

	M512	M4K
Core dynamic power	19.6%	0.8%
Memory dynamic power	149.1%	6.4%
Max clk freq	-3.6%	0.6%
LUT count	2.4%	0.0%

D. The Effect of Read and Write Enable on Memory Block Dynamic Power

As mentioned in Section III, Xilinx Virtex-II and Virtex-4 and Altera Cyclone II devices contain embedded memory blocks of a single size (18 Kbit, 18 Kbit, and 4.5 Kbit, respectively). Additionally, Virtex-II and Virtex-4 memory blocks do not have read enable signals. To assess the benefits of our approach for write port only optimization on devices which contain a single embedded memory block type, the 34 benchmark designs were re-mapped to Cyclone II devices using the procedure described in Section V.B for four specific sets of optimization:

Power optimization cases:

A. Write enable conversion to write clock enable
B. Read/write enable conversion to read/write clock enable.
C. Memory partitioning in addition to write enable conversion and combining.
D. Memory partitioning in addition to read/write enable conversion and combining.

For Cases A and C, read enable signals were left in their original locations. As shown in Table VII, the availability of both read and write enable conversion (Case B) versus write enable-only conversion (Case A) nearly doubles memory power reduction from 3.5% to 6.7%. Core dynamic power is modestly reduced by 0.4%. Figure 12 illustrates that this power decrease is primarily the result of four designs (17, 22, 23, and 32). The designs in Figure 12 are shown in the same order as in Table II. If enable combining and memory partitioning are

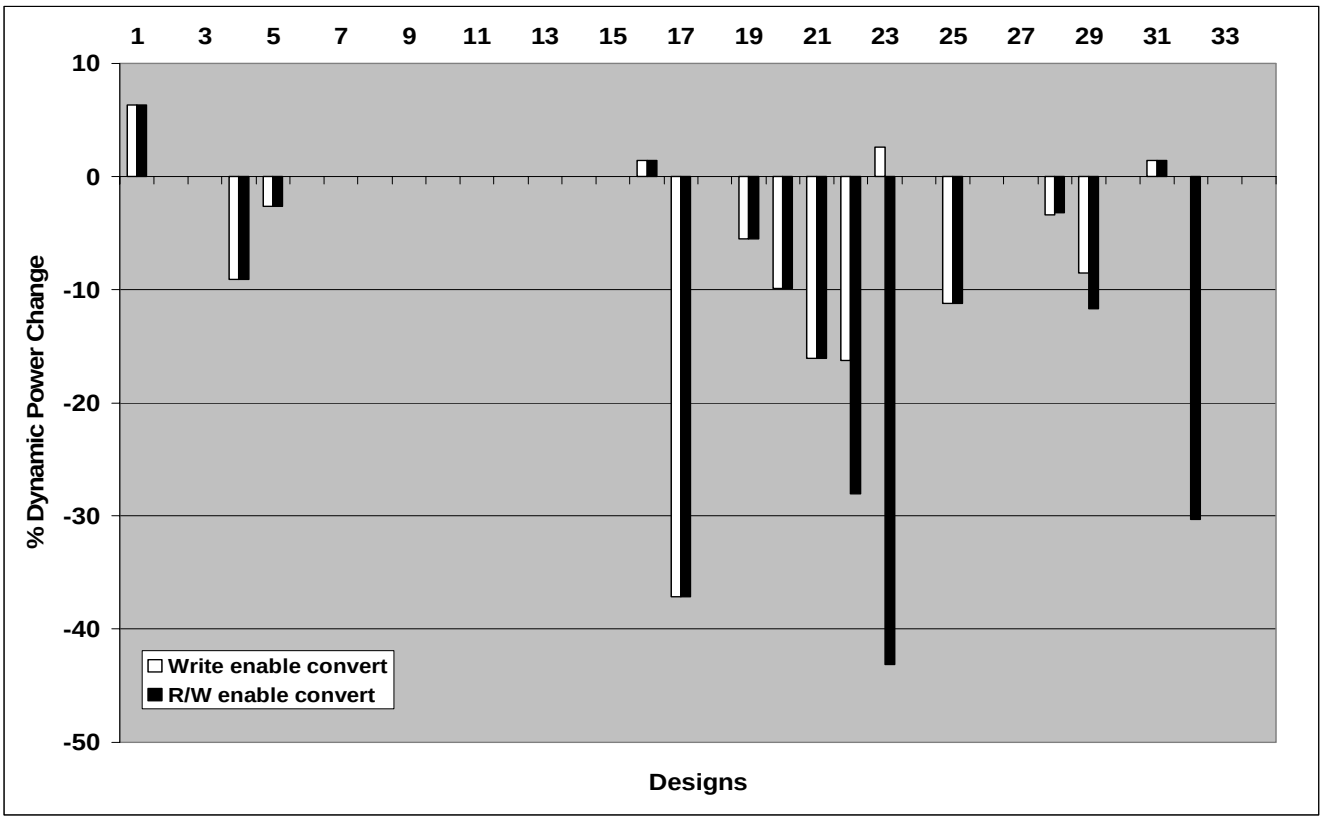

Figure 12: Percent change in data RAM power for benchmark designs due to RAM optimizations mapped to Cyclone II devices

Table VII: Summary of RAM optimization results for 34 designs mapped to Cyclone II devices. Each percentage represents the percent change of a value obtained via compilation with no RAM power optimization due to the specified optimization

	Write enable convert	R/W enable convert	Write Enable convert/ combine + Mem partition	R/W Enable convert/ combine + Mem partition
Core dynamic power	-0.7%	-1.1%	-3.3%	-6.1%
Memory dynamic power	-3.5%	-6.7%	-16.1%	-26.1%
Max clk freq	-0.4%	-0.4%	-1.0%	-1.6%
LUT count	0.0%	0.0%	0.7%	0.7%

included with write enable conversion (Case C) and R/W enable conversion (Case D), overall core dynamic power is more substantially reduced (by 3.3% and 6.1%, respectively). For each design, sufficient embedded memory block resources were available in the target FPGA to select power optimal logical-to-physical memory mapping for each logical memory. For Cases C and D, memory partitioning required about 0.02% of design synthesis time on average (max 0.1% for Design 34). It should be noted that these results do not advocate for the inclusion of a RAM read enable input on embedded memory blocks. A designer could select to use an available read clock enable input on an embedded memory block and gain the same power reduction benefit without the need for conversion/combining.

VI. Conclusion and Future Work

In this paper we have presented a set of RAM mapping algorithms that are targeted to FPGA embedded memory blocks. These techniques take advantage of the internal structure of FPGA embedded memory to reduce memory dynamic power dissipation. When possible, embedded memory block clock enables are used to deactivate RAM block precharging. Our mapping algorithms maintain the functional behavior of each designer-specified RAM. These techniques achieve a 26% RAM dynamic power reduction and a 6% core dynamic power reduction for 34 large benchmark designs with a performance and logic cost of about 1%. The availability of three embedded memory block sizes leads to a 10% memory power and 2% dynamic power reduction versus using only 4.5 Kbit embedded memory blocks. Our power reduction estimates have been verified both via board-level power measurement and via simulation-based power estimates.

Several optimizations to our power saving approaches could

be implemented in the future. An analysis of our benchmark designs shows that, on average, 18% of logical memories in a design share address decoding circuitry with other design logical memories. Currently, we rely on the Quartus II logic synthesis tool to identify and eliminate these and other structural logic redundancies and to pack compatible logical memories into the same physical memory. Higher-level logical RAM clustering may provide additional dynamic power savings. Another possible optimization is the RTL analysis of state machines to determine when embedded memory block accesses are not needed. More complex RAM shut-down signals could then be generated. Finally, an investigation to determine the optimal size and availability of different-sized embedded memory blocks is needed. In this paper it has been shown that a diverse selection of memory block sizes is beneficial and medium sized blocks (e.g. 4-16 Kbit) are desirable for power reduction. The exact mix of block sizes for optimal power reduction remains an open problem.

ACKNOWLEDGMENT

The authors wish to thank Elden Chau, Marcel LeBlanc, David Lewis, David Lin, Ricky Tai, and Meghal Varia for their insights regarding this work.

REFERENCES

[1] Altera Corp. Quartus II Handbook, Chapter 7, vol. 1, July 2005.
[2] Altera Corp. Stratix II Device Handbook, vol. 2, July 2005.
[3] Altera Corp. Cyclone II Device Handbook, vol. 1, June 2006.
[4] S. Bakshi and D. Gajski, A memory selection algorithm for high-performance pipelines, In *Proceedings of the European Design Automation Conference*, Brighton, England, Sept. 1995, pp. 124-129.
[5] L. Benini, A. Macii, and M. Poncino. A recursive algorithm for low-power memory partitioning. In *Proceedings of the International Symposium on Low Power Electronics and Design*, Rapallo, Italy, July, 2000, pp. 78-83.
[6] Y. Cao, H. Tomiyama, T. Okuma and H. Yasuura. Data memory design considering effective bitwidth for low-energy embedded systems, In *Proceedings of the IEEE International Symposium of System Synthesis*, Kyoto, Japan, Oct. 2002, pp. 201-206.
[7] A. Ferrahi, G. Tellez, and M. Sarrafzadeh. Memory segmentation to exploit sleep mode operation, In *Proceedings of the ACM/IEEE Design Automation Conference*, San Francisco CA, Jun. 1995, pp. 36-41.
[8] C. Gebotys. Low energy memory and register allocation using network flow, In *Proceedings of the ACM/IEEE Design Automation Conference*, Anaheim, CA, Jun. 1997, pp. 435-440.
[9] W. Ho and S. Wilton. Logical-to-physical memory mapping for FPGAs with dual-port embedded memories, In *Proceedings of the International Workshop of Field Programmable Logic and Applications*, Glasgow, UK, Aug. 1999, pp. 111-123.
[10] J. Lamoureux and S. Wilton. On the interaction between FPGA CAD algorithms, In *Proceedings of the IEEE International Conference on Computer-Aided Design*, San Jose, CA, Nov. 2003, pp. 701-708.
[11] M. Margala. Low-power SRAM circuit design, In *Proceedings of the IEEE International Workshop on Memory Technology, Design, and Testing*, San Jose, CA, Aug. 1999, pp. 115-122.
[12] M. Mamidipaka and N. Dutt. An Enhanced Power Estimation Model for On-Chip Caches. CECS Technical Report #04-28, University of California, Irvine, 2004.
[13] P. Petrov and A. Orailoglu. Virtual page tag reduction for low-power TLBs, In *Proceedings of the IEEE International Conference on Computer Design*, San Jose, CA, Oct. 2003, pp. 371-374.
[14] H. Schmit and D. Thomas. Address generation for memories containing multiple arrays, IEEE Transactions on VLSI Systems, vol. 17, pp. 377-385, May 1998.
[15] R. Tessier, V. Betz, D. Neto, and T. Gopalsamy. Power-aware RAM mapping for FPGA embedded memory blocks, In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, Feb 2006, pp. 189-198.
[16] O. Unsal, R. Ashok, I. Koren, C. Krishna, and C. Moritz. Cool-cache for hot multimedia, In *Proceedings of the ACM/IEEE International Symposium on Microarchitecture*, Austin, TX, Dec. 2001, pp. 274-283.
[17] S. Wilton, S. Ang, and W. Luk. The impact of pipelining on energy per operation in field-programmable gate arrays, In *Proceedings of the International Workshop of Field Programmable Logic and Applications*, Antwerp, Belgium, Aug. 2004, pp. 719-728.
[18] S. Wuytack, F. Catthoor, L. Nachtergaele and H. De Man. Power exploration for data dominated video applications, In *Proceedings of the IEEE International Symposium on Low Power Design*, Monterey, CA, Aug. 1996, pp. 359-364.
[19] Xilinx Corp. Virtex-4 User's Guide, July 2005.
[20] Xilinx Corp. Virtex II Platform FPGAs: Complete Data Sheet, March 2005

Russell Tessier (M'00) is an associate professor of electrical and computer engineering at the University of Massachusetts, Amherst, MA. He received the B.S. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY in 1989 and S.M. and Ph.D. degrees in electrical engineering from MIT, Cambridge, MA in 1992 and 1999, respectively. Dr. Tessier was a founder of Virtual Machine Works, a logic emulation company, and has also worked at BBN, Ikos Systems, and Altera. Prof. Tessier currently leads the Reconfigurable Computing Group at UMass. His research interests include computer architecture, field-programmable gate arrays, and system verification.

Vaughn Betz (S '88, M '92**)** received the B.Sc. degree from the University of Manitoba in 1991, the M.S. degree from the University of Illinois at Urbana-Champaign in 1993, and the PhD degree from the University of Toronto in 1998, all in Electrical and Computer Engineering. Currently he is a Director of Software Engineering at Altera Corporation's Toronto Technology Centre. Dr. Betz was a co-founder and Vice President of Engineering at Right Track CAD until its acquisition by Altera in May of 2000. His research interests include placement and routing algorithms, field programmable gate array architecture, and power and timing modeling of deep-submicron circuitry.

David Neto earned a BSc in Computer Science and Mathematics in 1991, an MSc in Computer Science in 1993, and a PhD in Computer Science in 1999, all at the University of Toronto. His technical interests include power analysis, CAD optimization algorithms for power and other metrics, and software performance and quality.

Aaron Egier (S'00-M'06) received the B.A.Sc. and M.A.Sc. degrees in computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2002 and from the University of Toronto, Toronto, ON, Canada, in 2005, respectively. For the M.A.Sc. degree, he created software tools to automate the design and layout of FPGAs. Currently, he is

an Advanced Software Engineer in the power modeling group at the Altera Toronto Technology Centre.

Thiagaraja Gopalsamy has a Masters degree in computer science and engineering from the Ohio State University. He is currently working at Altera Corporation as a Senior Engineer. His research interests include FPGAs and mobile ad-hoc wireless networks.