IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 August 2023, accepted 29 August 2023, date of publication 31 August 2023, date of current version 7 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3310940

RESEARCH ARTICLE

Efficient Implementation of Many-Ported Memories by Using Standard-Cell Memory Approach

HANAN MARINBERG[1], (Member, IEEE), ESTEBAN GARZÓN[2], (Member, IEEE), TZACHI NOY[1], MARCO LANUZZA[2], (Senior Member, IEEE), AND ADAM TEMAN[1], (Member, IEEE)

[1]Emerging Nanoscaled Integrated Circuits and Systems (EnICS) Laboratories, Faculty of Engineering, Bar-Ilan University, Ramat Gan 5290002, Israel
[2]Department of Computer Engineering, Modeling, Electronics and Systems, University of Calabria, 87036 Rende, Italy

Corresponding author: Esteban Garzón (esteban.garzon@unical.it)

This work was supported by the Magneton Program of Israel Innovation Authority. The work of Esteban Garzón was supported by the Italian MUR through the Call—Horizon Europe 2021–2027 Program under Grant H25F21001420001.

ABSTRACT Multi-ported memories are widely used in many applications, such as for high-speed and high-performance parallel computations. While conventional SRAM-based memory macros are limited in both flexibility (e.g., to accommodate a large number of write inputs and read outputs) and performance, standard cell-based memories are much more flexible and can be parameterized. The physical implementation approaches of the existing tools have great difficulty in dealing with these memories due to the multiplicity of wiring and misunderstanding of the regular structure of these arrays, to the point of inability to converge under certain conditions. This paper presents novel methodologies for the logical and physical implementation of many-ported standard cell memories (MPSCMs). Two methodologies are proposed to replace the standard design flow by controlling and guiding the design tools to improve power consumption, area and performance of these arrays. A commercial 65 nm CMOS technology was used to evaluate and benchmark the two design methodologies on MPSCM macros of different sizes as compared to other equivalent macros designed with standard methodologies and state-of-the-art designs. Physical implementation results show that as compared to a standard RTL approach, the implementation of a 3-write, 5-read port (3W5R) register file with the tightly-controlled methodology leads a 2 × increase in placement density along with significant reductions in write power (–66%), read power (–37%) and leakage power (–80%), while also improving the access time (–6%). When considering an extreme case of a many-ported memory with 20-write and 20-read ports (20W/20R), the guided methodology leads to improvements in delay (–13%), write power (–62%), and leakage power (–51%). Both methodologies were implemented within an automation utility based on the ''Salamandra'' open source netlisting tool, enabling fast and easy migration to additional process nodes and standard cell libraries for generating MPSCMs with various sizes and features.

INDEX TERMS Standard cell memories (SCMs), multi-ported memories, controlled placement, low-power, vector register file, register file, many-ported memory.

I. INTRODUCTION

On-chip memory is a major bottleneck for performance, energy-efficiency, and area footprint of state-of-the art systems-on-chip (SoCs) and modern microprocessors [1]. On-chip memories are commonly implemented with six-transistor static random access memory (6T-SRAM) based macro-cells, which are provided as hard macros by memory vendors or directly by the foundry. These memories are often provided with various design targets, such as

The associate editor coordinating the review of this manuscript and approving it for publication was Harikrishnan Ramiah.

high-density, high-speed and low-leakage, but at their basis, they are optimized for size, and are therefore based on foundry-supplied ''pushed rule'' bitcells, which are very constrained in their usage envelope. For robust operation in advanced process nodes and to meet the high memory throughput requirements, the 6T-SRAM macros require dedicated, custom-designed peripherals [2], [3], [4], [5], [6]. Since these macros are designed for general use, they often do not support special design requirements, such as low-voltage operation, non-standard array or periphery sizes, multi-ported functionality, etc. [7], [8]. Therefore, when an embedded memory with special functionality is required, the choice is usually between the long, expensive and high-risk approach of custom-designing the memory, the use of architectural techniques to achieve the required features with standard SRAM macrocells, or behaviorally defining the required memory as part of the register transfer level (RTL) description of the system [8], [9].

For non-standard (e.g., multi-ported) memories up to a certain size (usually several kB), the third solution is usually chosen. Defining the memory with a standard hardware description language (HDL) and then synthesizing and implementing it through the standard design flow can lead to some advantages. First, the portability of the memory design to different technologies is greatly simplified. Next, it is relatively easy to modify design parameters (such as the number of words, the number of bits per word or the number of access ports) at design time and even at quite advanced stages of the design cycle. Finally, since top-level designs comprising standard-cell-based memory blocks can be placed automatically using a standard placement tool, the data locality is improved, while the routing can be also reduced [9]. However, the straightforward behavioral description of such a block misses out on several functionally equivalent characteristics of a memory that cannot be exploited by the standard flow. This leads to synthesis of the RTL into gate-level structures that cannot be further optimized in the way that custom-designed memory macros are. One example of this is the clocking of storage elements – whereas a custom-designed memory macro only has a single clock port that has a relatively low toggling load (e.g., only the input latches), an RTL defined memory requires distributing a clock to each storage element, leading to a high-degree of clock tree buffering, routing complexity, and clock power consumption. Another problem with the RTL-defined approach is that the addition of such an array of clocked elements creates a large number of timing paths, which can overload the timing optimization tools, leading to extremely long tool runtimes.

An alternative solution consists of carefully defining the memory block in what is referred to as the semi-custom, or standard cell memory (SCM) approach [9], [10], [11], [12]. The SCM approach, which takes into consideration the structural compilation of the digital block and not just its behavioral functionality, leads to improved implementation as compared to the purely-behavioral RTL-defined alternative. While the block will remain functionally-equivalent, it may not be logically-equivalent, and therefore, a synthesis tool will not reach such a solution based on the straightforward behavioral description. For example, since memory writes are applied to a single decoded row, the clock signal can be replaced by each row's write word line, which only toggles when that row is selected, significantly reducing the power consumption. Taking this approach a step further, controlled-placement can be applied to the structurally-defined macrocell, further exploiting the characteristics of the array for gains in power, performance and area [7], [8], [13]. Another benefit of the controlled placement approach is the significant reduction in physical implementation runtime, since the blocks can be either implemented independently of the full design or removed from the timing optimization process, such that the tools are not loaded by the additional timing paths [7].

Previous work on SCM implementation has exclusively focused on two-ported memories; i.e., macros with one read and one write port (1W1R), which are natively supported by the SCM approach. However, many applications require memories with additional ports; from a small number of ports for processor register files to a very high number of ports for highly-parallel computational units, such as digital signal processors (DSPs). We refer to memories with more than two ports as ''many-ported'', since these memories are non-standard in the sense that it is very uncommon to find SRAM-based memory compilers supporting these requirements. This work presents a novel design methodology for the physical implementation of many-ported memories that are based on standard cells. We have evaluated the inherent structure and properties of many-ported memories, and subsequently replaced the implementation through the conventional digital implementation flow (DIF) with a novel methodology based on specially developed algorithms and utility tools for efficient implementation in terms of performance, power, and area (PPA). In particular, the proposed solution aims to manipulate the netlist according to the desired architecture with the goal of improving the place and route steps by properly guiding the electronic design automation (EDA) tools according to the physical structures of the common components. Our study covers all the stages of digital design from memory specification through full layout, as demonstrated with a commercial 65 nm CMOS technology and standard cell library.

Two separate methodologies are proposed. The first, referred to as the ''tightly-coupled'' technique is intended for highly-optimized macros with clear specifications, such as a 3W5R register file required to meet the requirements of a dual-issue RISC-V processor pipeline [14]. The second, referred to as the ''guided'' technique is a flexible approach intended for macros with a large number of ports and supporting design flows that support a parameterized definition, such as the vector register file (VRF) of a

high-performance DSP [15]. Post-layout simulation results have been benchmarked against other equivalent macros designed with standard methodologies. As compared to a standard RTL approach, the implementation of the aforementioned 3W5R processor register file with the tightly-controlled methodology leads a 2 × increase in placement density along with significant reductions in write power (–66%), read power (–37%) and leakage power (–80%), while also improving the access time (–6%). At the other extreme, a many-ported VRF with 20-write 20-read ports (20W/20R) provides very significant gains in leakage power (2 ×), write power (–62%) and access time (–13%), while ensuring high memory density, and considerably reducing the implementation run time (–47%), as compared to the behavioral RTL design approach.

The rest of the paper is organized as follows. Section II presents the background of our work. Section III introduces the proposed methodology for implementation of many-ported memories. Section IV presents and discusses obtained implementation results. Finally, Section V summarizes our work.

II. BACKGROUND

A. WHY MULTI-PORTED EMBEDDED MEMORIES?

Embedded memories with multiple read/write (R/W) ports are very appealing for modern SoCs [1], [2], [16], [17]. Various blocks that are usually integrated in these systems, such as media and graphics processing units (GPUs) as well as computational cores, require multi-port memories to support instruction-level parallelism with the goal of increasing processing speed while avoiding data serialization and reducing wait states during pipelining [2], [3], [18], [19]. As an example, dual-port memories are widely used for video processing units, since two read and write access operations can be performed within the same clock cycle [20], [21], [22], [23], [24]. An even larger number of read ports is also utilized in graphics processors, digital signal processors, and for shared system level memories that provide simultaneous access to multiple processing units [3], [16], [25], [26], [27]. In addition, a large number of write ports can be useful in multi-threaded applications and vector processing. Furthermore, many read/write ports (>10) are needed in high-end microprocessor register files to support high degrees of multi-threading, wide issues and other features [28], [29]. These, and many other applications, raise the need for many-ported memory design.

B. STANDARD CELL MEMORIES

On-chip memories are commonly implemented with custom-designed macrocells based on 6T-SRAM, which is optimized by the foundry to achieve very dense layouts that enable integration of as much memory as possible within the allocated silicon area. However, for small storage arrays, a common practice is to define the registers in RTL and synthesize them along with the rest of the digital logic. While structural design of such arrays was probably commonly practiced beforehand, the groundbreaking papers by Meinerzhagen et al. [9], [11] coined the term "SCM" for this semi-custom design approach for register arrays. In these works, the case for SCMs was presented, describing various options for implementation of the SCMs and comparing them with compiled 6T-SRAM macrocells in 180nm to 65nm technologies. A primary conclusion from [9] was that the tradeoff point, at which SCMs overtake 6T-SRAM macros in area and start presenting a growing overhead, is at approximately 1 kbit (32 × 32) bits. That said, the work presented in [11] shows the value of SCMs as a low power alternative to 6T-SRAM, which fails under voltage scaling and is not appropriate for subthreshold operation.

The study of SCMs was taken a step farther in [13] and [7], where a controlled-placement approach was proposed and an architecture for implementing SCMs with high placement utilization and low wirelength was developed. These works presented comparisons of controlled SCMs in 28nm FD-SOI technology, comparing various memory sizes and configurations with several types of compiled SRAM macros and register files. This and following work in additional technologies showed significant energy savings through controlled placement, reaching as high as 50% and 70% as compared to non-controlled SCMs and 6T-SRAM, respectively, while retaining similar performance. Furthermore, the controlled placement technique was shown to enable scaling of SCMs to large sizes, with a 16 kbyte (512 × 256 bit) macro demonstrated in a 28nm bulk CMOS process [30]. These approaches have been applied to SCMs fabricated in many research chips, including from the PULP team at ETH-Zurich [8]. Similarly, IBM has reported extensive use of synthesized soft arrays (SCMs with some custom-designed parts) in their POWER9 processor, using controlled placement to implement hundreds of scannable register files [29].

C. MANY-PORTED STANDARD CELL MEMORIES

Fig. 1 shows the top-level of a conventional many-ported standard cell memory (MPSCM), which is composed of four main blocks:

1) The storage array with $R \times C$ memory elements, where R is the number of rows and C is the number of columns.
2) The write logic, including the write address decoders, that is fed by a set of input data (DIN), write enable (WE) and write address (WADDR) signals for each write port.
3) The read logic, which for each read port, comprises read out multiplexers that drive the data out (DOUT) signals and read address decoders that receive read address (RADDR) and read enable (RE) signals.
4) The clock controller, which provides the gated clock signals to the registers during write operations and the sampling clock for optional input/output registers or latches.

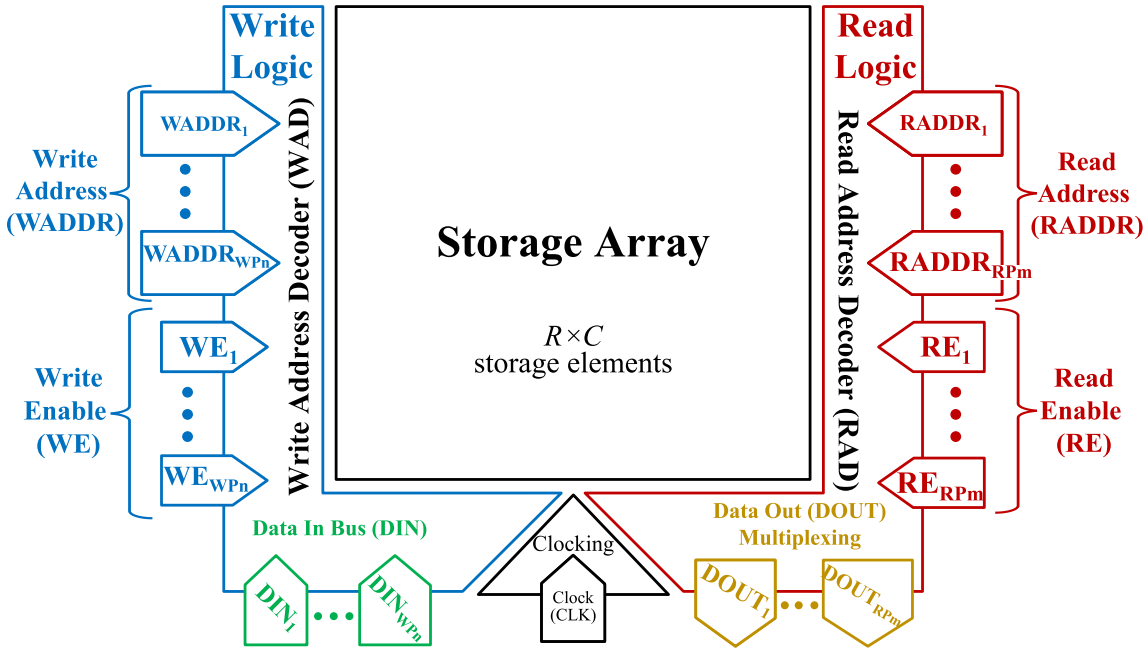

FIGURE 1. Simple block diagram of a many-ported standard cell memory (MPSCM) with $R \times C$ storage elements, WPn-write and RPm-read ports, and the corresponding peripheral with decoupled read and write paths.

As shown in Fig. 1, for n-write (WPn) and m-read (RPm) ports, each input to the decoupled write and read logic corresponds to a single write and read port, respectively.

The storage elements could be implemented using flip-flops or latches. Flip-flop based implementation is more straightforward and can mitigate potential race conditions, while a latch-based approach offers a reduced area footprint [7]. Additionally, the latch-based implementation can improve access time by leveraging the time-borrowing property of latches. Due to the static CMOS nature of standard cells, both storage elements can easily support multi-port writing and reading, given the logic added for correct operation. This logic can be trivially described in RTL and implemented through synthesis. The SCM write and read logic from Fig. 1 have been extensively studied in [7] for a two-port (1R1W) memory. For many-ported memories, the majority of the logic remains the same. In particular, for the readout block, slight modifications should be taken into account to efficiently implement multiple read ports. The readout block has to choose the bit that will be transferred to the output in each column, according to the read address. When implementing the readout path from the storage elements to the output, previous explorations, reported in [7] and [11], indicate that multiplexer-based readout is more efficient than the tri-state buffer-based logic. Accordingly, to propagate the data stored in the targeted row to the output port (DOUT), an $R:1$ multiplexer is needed. To provide a mapping of such a multiplexer, this can be synthesized, potentially using high fan-in cells, which are undesired for low voltage operation, and could also result in complicated routing with long wires and high pin density, especially when multiple read ports are required.

Differently from an SRAM-based implementation, whose clock signal is fed only to the peripheral circuitry, every SCM storage cell requires a clock signal for writing new data. This results in several implementation challenges:

- The addition of a large number of clock sinks leads to an extreme rise in complexity of clock tree synthesis, resulting in multiple-level buffering, unconventional placement, inherent routing congestion, large skews and long run times.
- The clock pin in each storage cell toggles regardless of read/write activity, resulting in a large increase in dynamic power consumption.
- Each bitcell introduces additional sequential timing paths. This large number of additional paths must be accounted for and optimized by the timing and optimization engines, leading to complexity and long run times.

In this work, we propose solutions to the above-mentioned issues to be applied in the physical design phase.

III. MULTI-PORT STANDARD CELL MEMORY (MPSCM) USING CONTROLLED PLACEMENT

A. MULTI-PORT STANDARD CELL MEMORY MICRO-ARCHITECTURE

Previous work dealing with controlled-placement of SCMs [7] has shown that, when implemented with standard logic, memories suffer from complex wiring that leads to high congestion of interconnections and sub-optimal implementation. Many-ported memories require multiplexing address

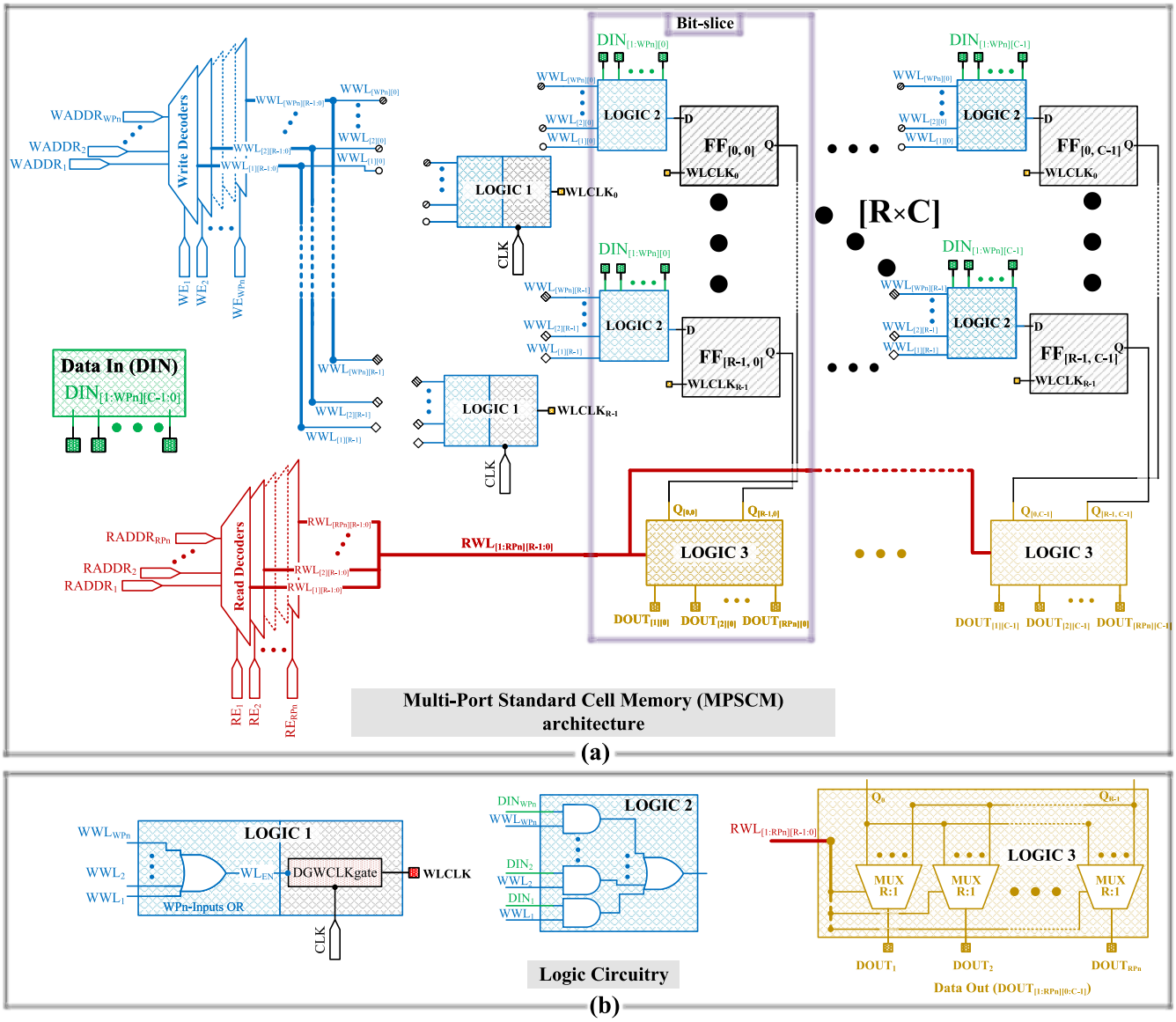

FIGURE 2. (a) General multi-port standard cell memory micro-architecture with $R \times C$ storage elements. (b) Read/Write logic circuitry.

and data busses from and to each port, resulting in additional wiring and routing to areas with high pin densities. This issue is further exacerbated as the number of read and write ports grows, leading to severe congestion, convergence challenges and diminished PPA.

Fig. 2(a) shows a general MPSCM micro-architecture for an array of $R \times C$ storage elements with WPn-write and RPm-read ports. First, to carry the data in/out (DIN/DOUT) from or to the memory by selecting the appropriate row, an additional decoder is needed for each read and write port (not illustrated in Fig. 2) to access the corresponding read or write address, respectively.

For each independent write port, there is a write decoder that receives WADDR WIDTH bits, and outputs $2^{\text{WADDR WIDTH}}$ write word line (WWL) signals – one to each row of the array. For example, considering WPn ports, the output of the last write decoder is $\text{WWL}_{[WPn][R-1:0]}$, which refers to the WWL of the write port WPn with a bus of $[R\text{-}1:0]$ signals. Given the assumption that no writing operation is allowed from multiple ports to the same row during a given clock cycle, only a single WWL signal from each write decoder will be active at a given time. The WWL signal for a particular row feeds all the registers of that row and has two roles. The first is to enable the clock to the selected row (please, refer to the "clock control" circuit in Fig. 2(a)-(b)). The second role is to serve as a switch for the appropriate data entry of the port from which the address came (please, refer to the "mux logic" circuit in Fig. 2(a)-(b)). This mux logic is local to each individual register, as illustrated in Fig. 2(a).

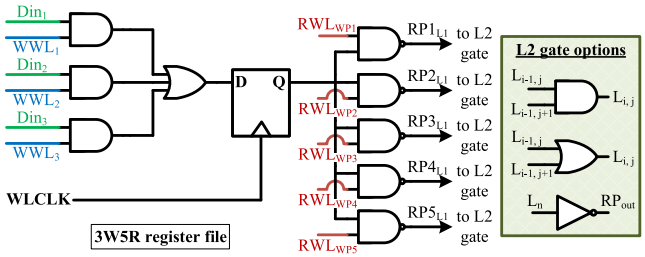

FIGURE 3. Implementation of an expanded bitcell for a 3W5R register file.

Similar to the write ports, each port requires a separate decoder, which drives a read word line (RWL) bus signal that is propagated to all registers across the entire word. For a memory with RPm read ports, the output of the last read decoder will be $RWL_{[RPm][R-1:0]}$. The RWL signal is also used here as a switch for the reading mux (refer to the ''reading mux'' circuit in Fig. 2(a)-(b)); however, in contrast to the write requirements, in the case of read, there is no limit on reading from two different ports on the same row. The read mux component for each port also needs to be placed next to each register. More specifically, the ''reading mux'' circuit is shared between columns.

This microarchitecture provides the foundation for dividing the register file into $R \times C$ expanded bitcells, composed of a flip-flop, n write mux elements, and m read mux elements. An example of such an expanded bitcell is illustrated in Fig. 3 for 3-write and 5-read (3W5R) ports. The write mux is simply constructed of an AND-OR tree, where the DIN of each write port is ANDed with the WWL of the corresponding port and the outputs are all ORed to create the data signal to be (potentially) written into the storage element. The clock pin of the flip-flop is fed by the WLCLK signal that is shared by the entire row, as generated by the control clock element of Fig. 2(b). The output of the flip-flop is connected to m minimum sized NAND gates, which receive the corresponding read port's RWL signal as their second input.

The output of these NAND gates ($RP_{i,L1}$) is '0' only if a '1' is stored and the row was selected for readout for this row. Therefore, all other $RP_{i,L1}$ of the same column will be '1' and we need to propagate the '0' to the column output. This is done by chaining interleaved NAND and NOR gates, as detailed in [7], however only one additional gate is required for each bitcell to create this logic. Therefore, a NAND, NOR, or output buffer will be included in the expanded bitcell, based on its row number and the total number of rows in the array.

By constructing these expanded bitcells, no additional logic is required to realize the write and read out muxes. To complete the array, the only additional peripherals are write and read decoders and control clock units, as illustrated in Fig. 2(b). Furthermore, scan can be supported by simply expanding the write mux to support an additional port. This structure enables building an efficient physical implementation based on controlled placement, as described in the following section.

B. PHYSICAL IMPLEMENTATION BASED ON CONTROLLED PLACEMENT

In order to determine the locations of all components in the design, we developed two algorithmic techniques for determining the physical placement of standard cells, hereby referred to as *Tightly-Controlled* and *Guided*.

1) TIGHTLY-CONTROLLED TECHNIQUE

The Tightly-Controlled technique, is similar to that carried out for the prior-art two-port SCM [7]. Each cell is placed at a specific coordinate to achieve maximum utilization and minimum wire length. The location of each standard cell is accurately calculated, taking into account horizontal and vertical wiring. As a consequence, a particularly high utilization of area is achieved along with minimal and straight wiring. This results in improved performance and power with respect to conventional automatic placement approaches. Note that such an approach can only be taken with a detailed structural description of the structural netlist, where each gate can be identified and its location can be calculated. The structural description of Fig. 2 is compatible with this approach; however, defining this structure with an HDL (e.g., using ''generate''-type statements) and running it through a synthesis tool is both cumbersome to write and parameterize, and is non-trivial (and tool-dependent) to manipulate specific gates in the place and route tool. For this reason, a utility for generation of the netlist with the particular structure and instance names was developed and used, as described in Section III-C.

2) GUIDED TECHNIQUE

To overcome the design-time limitations of the Tightly-Controlled technique, we propose an alternative that allows flexible automation in terms of number of ports and features, while maintaining high utilization and low wire-length. In this approach, rather than providing the exact location of each gate, we treat a single memory bit cell with all the components of its read and write ports as a single grouped entity. Thereafter, a group is provided with an area (''fence'') large enough to accommodate all of its components, and this fence is placed at a precise location. The place and route tool is then given the option to perform optimization, buffering, and legalization within and in between the groups.

3) COMPARISON OF PLACEMENT TECHNIQUES

Table 1 summarizes the key differences between the two proposed techniques. The Tightly-Controlled and Guided placement techniques offer complementary benefits. The Tightly-Controlled approach is best suited for small, fixed port counts and achieves maximum utilization and minimum wiring through precise coordinate-based placement, providing full control over each cell, but requires significant design time for each new architecture. In contrast, the Guided

TABLE 1. Key differences between the Tightly-Controlled and Guided placement techniques.

	Tightly-Controlled	Guided
Description	Precise coordinate-based placement of each cell	Groups of related cells placed in fenced regions
Suitability	Small, fixed port counts	Flexible, parameterizable, large port counts
Design Time	Significant fine-tuning required	Completely Automated, less optimization
Control	Full control over each cell	Control at group level
Utilization	Maximum Possible, near 100%	Very high
Wiring	Minimum length	Low wirelength
Performance/ Power	Optimized	Optimized

technique is highly parameterizable, supporting any number of ports without altering the algorithm. While exhibiting slightly lower utilization and wirelength, it provides control at the group level and significant area, performance and power benefits versus standard unguided approaches. Moreover, by placing groups of cells in defined regions rather than individual components, the Guided technique enables automated application to new architectures without extensive floorplan tuning. This makes the Guided approach suitable as a generic compiler technique for varied SCM designs.

C. PROPOSED CONTROLLED PLACEMENT METHODOLOGY FOR MANY-PORTED STANDARD CELL MEMORIES

Previous works on standard cell memories were carried out by describing the structure in RTL and providing synthesis directives and constraints to ensure that the specified structure would be kept when generating the netlist. However, this approach is both tool specific and was found to be very limited when trying to configure various features, such as the number of ports, reset configurations, scan configurations, etc. Furthermore, all controlled placement had to be handled in the place and route tool, which was again both tool specific and very hard to control/configure. Therefore, to support this work, we started by designing a netlisting framework for defining the specific architecture and generating the netlist and location directives in a very flexible and non tool-specific fashion. We developed this framework as an extensible Python library called ''Salamandra'', which is now available as an Open Source project [31].

1) SALAMANDRA FRAMEWORK

Open-source projects to simplify IC design and enable efficient workflows have become increasingly popular over the recent past [32], [33], [34]. MyHDL [32], for example, is a Python-based hardware-description language that enables generation of Verilog/VHDL RTL and test-benches using Python. Chisel [33] has been proposed as a Scala-based alternative to traditional HDLs that provides much improved productivity. Chipyard [34] is an open-source integrated SoC design, simulation and implementation framework, which provides a unified framework and work flow for agile SoC development. Salamandra is such a framework, intended to simplify structural design definition and providing tool-independent attributes that can be passed on to the implementation flow.

The Salamandra framework provides an intuitive interface for generating components in a hierarchical fashion and defining the connectivity between them. Various properties can be added to components, nets and pins to provide additional information about the design, such as placement coordinates for a given logic gate or hierarchical component. Salamandra further supports reading-in standard cell libraries and writing-out various EDA-supported formats, such as Verilog netlists and Tcl commands for place and route tools. All of this functionality can be wrapped in a Python script, allowing Salamandra to be used as a Python library/API and exploiting all the features and existing libraries of this popular programming language.

For this work, the customized SCM architecture was designed in Salamandra by instantiating components like registers, logic gates and memories, and connecting them hierarchically, while attaching location coordinates for controlled placement. This enables the implementation of the algorithms and logic for paramaterizing the construction of the macrocells with various features. The design was then mapped to the target technology through a set of defines, without needing to go through logic synthesis. The Salamandra-based tool emitted the Verilog netlist and Tcl placement commands for the CAD tool, where the Tcl controls placement according to the coordinates set during SCM design. This allowed flexible design and generation of the netlist and placement crucial for implementing the customized SCM architecture, without being constrained to a particular EDA tool flow.

2) PROPOSED MPSCM COMPILER

Fig. 4(a)-(b) shows the flow charts of conventional SCM and the proposed MPSCM compiler operation, corresponding to the typical auto-placement and controlled placement techniques, respectively. By exploiting the Salamandra framework, we developed an MPSCM compiler, as shown in Fig. 4(b), for generating netlist and placement commands for a given SCM specification. The compiler uses a slightly modified standard digital design flow; instead of writing semi-structural RTL with generate statements and standard cell instantiations, which are then synthesized with complex directives and constraints, the SCM structure is entirely designed in Salamadra. Mapping to a technology is achieved with a simple set of defines without the need to go through a synthesis tool, as shown in Fig. 4(b). During the design process, location data is calculated and attributed to each component, such that along with the output Verilog netlist, a physical-aware format, like DEF or Tcl placement commands, can be exported for use in the place and route tool.

The rest of the digital implementation flow is the same, as detailed in Fig. 4, including logic simulation, completion

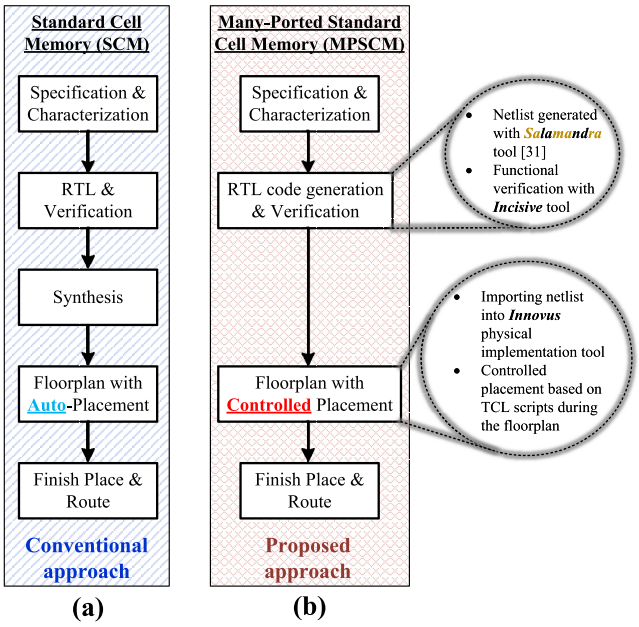

FIGURE 4. Block diagram of (a) conventional standard cell memory (SCM) and (b) the proposed many-ported standard cell memory (MPSCM) compilers.

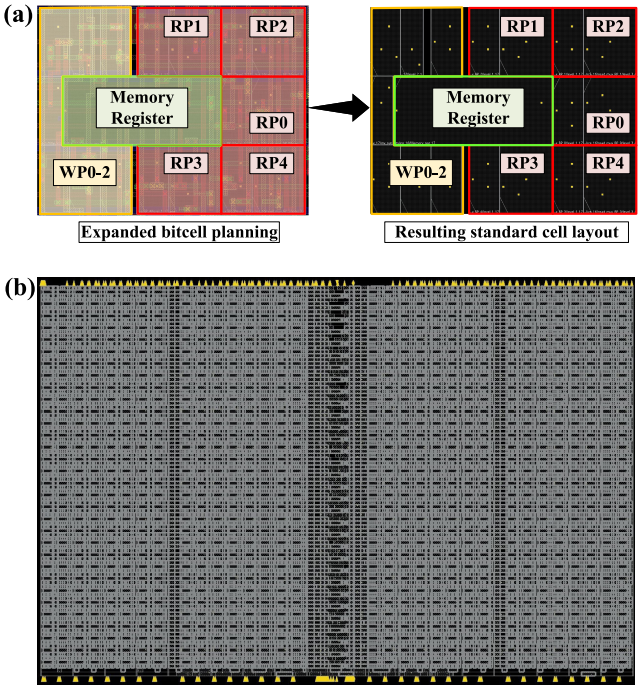

FIGURE 5. Tightly-Controlled 3W5R memory: (a) single bit, (b) floorplan of a 32 ×32 memory array.

of the place and route stages, and post-layout validation and power estimation. This flow can accommodate two hierarchical approaches to SCM implementation. The first is a stand-alone (“black box”) approach, where the entire flow is applied to the memory macro to provide a full layout, which can then be passed as a semi-hard macro to a top-level floorplan for integration with the other parts of the design. The other approach is to export the SCM netlist for integration as a soft macro in the top-level design and provide relative placement coordinates to the top-level floorplan. In this approach, internal routing and timing closure are carried out as part of the top-level place and route flow. While the first approach simplifies the integration, reduces run-time, and enables straightforward reuse of multiple identical SCM instances, the second approach merges the SCM implementation with the top-level floorplan, enabling better utilization of routing resources and more timing flexibility.

IV. RESULTS AND COMPARISON

To evaluate the effectiveness of our controlled placement methodology, we chose two many-ported memory examples for implementation and comparison with the conventional RTL approach, as well as with other multi-ported memories reported in the literature [3], [28], [35], [36]. The first benchmark circuit – on the low-end of the multi-ported spectrum – is a 32 × 32-bit 3-write 5-read (3W5R) memory block, used as the register file (RF) of a dual-issue RISC-V processor[1] [14]. The second benchmark takes the proposed methodology to the other extreme by implementing a 20-write 20-read (20W/20R) RF, intended to be used as VRF for a high-performance DSP core [15]. The presented results are extracted from post-layout gate-level simulations considering a commercial 65 nm CMOS technology and standard cell library featuring a nominal supply voltage of V_{DD}=1.2 V. For power measurements, Cadence Voltus IC Power solution was applied to an extensive test-case that includes 1000 random-data write cycles to random write ports and addresses, which were subsequently read out during the 1000 cycles in a random order.

A. FLOORPLAN AND LAYOUT

The expanded bitcells of the 5W3R register file were hand-designed based on specific standard cells to achieve the highest density possible. The smallest register in the library was chosen and set at the center of the bitcell area with the write port logic gates placed on the left and the read port logic gates tightly wrapping the right side of the register, as shown in Fig. 5(a). This expanded bitcell was then replicated into a full 32 ×32 memory array and the additional peripherals were added to arrive at the layout shown in Fig. 5(b). As can be seen, this Tightly-Controlled approach provides a placement utilization approaching 100% for the smallest area possible. Despite the high utilization and the high pin-density of the multiplexer-heavy logic, the structured approach enables easy routing convergence, thanks to the shares nets and reduced wirelength. This also results in low power consumption

[1]The RISC-V base architecture requires two source (read) and one destination (write) operand, resulting in a 1W2R RF. The expansion to dual-issue doubles the required number of ports. The chosen processor also supports post-increment instructions, which require an additional read+write operation per instruction. The final result is a 3W5R RF.

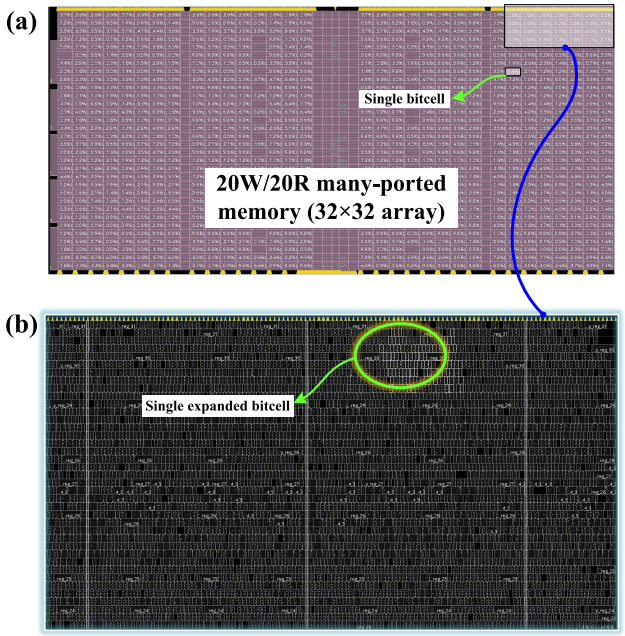

FIGURE 6. Guided approach: (a) floorplan of a 32 ×32 memory array, highlighting a 20W/20R bitcell and a group of bitcells; (b) floorplan of a group of bitcells, highlighting the 20W20R bitcell (includes register files and write/read circuitry).

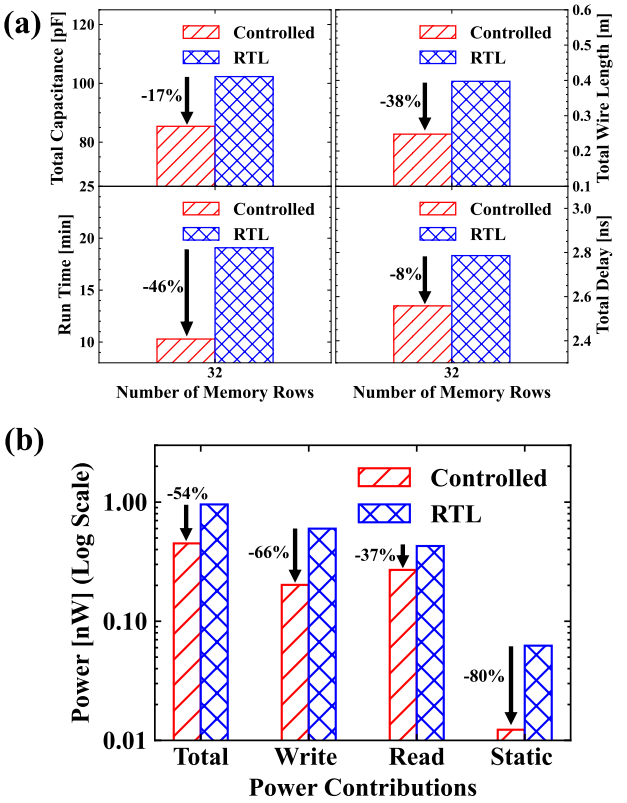

FIGURE 7. Post-layout gate-level simulation results in terms of (a) capacitance, write length, run time, and delay, and (b) power consumption for a multi-port (3W5R) 32×32 register file at a 65 nm node and $V_{DD} = 1.2$ V. Results consider the tightly-controlled placement technique.

and limited usage of routing resources on higher metal layers.

The many-ported 20W/20R memory array was implemented using the Guided-placement approach. Salamndra was used to define the expanded bitcells and the connectivity between them, and to define placement regions (fences) where the gates belonging to a given bitcell would be placed. The floorplan of the many-ported memory is shown in Fig. 6(a), with each small rectangle marking the boundary of a single expanded bitcell. The regions were calculated automatically based on the features of the standard cells, providing some additional area for upsizing during optimization. Once the initial placement was finished, the fences were removed to enable the EDA tool to apply standard optimization techniques to meet timing and design rules. The resulting layout is shown for a zoomed-in portion of the floorplan in Fig. 6(b), with the logic gates belonging to a single expanded bitcell highlighted. Here, too, the resulting utilization is very high and routing easily converged, despite the challenge successfully routing a block with extremely wide muxes and high pin count.

B. POST-LAYOUT RESULTS

Fig. 7(a)-(b) shows the post-layout gate-level simulation results in terms of capacitance, wire-length, run time, delay, and power consumption, for the 3W5R 32×32-bit RF test case. In contrast to the RTL-defined implementation, the tightly-controlled technique results in a well-structured memory layout, which leads to reduced wire capacitance (–17%), reduced wire-length (–38%), and a slight improvement in the access delay (–8%). The run-time required for standalone implementation of the tightly-controlled block is almost 2 × faster than a traditional implementation. By using a hard-macro approach, this run-time savings substantially increases, as the routing and timing of the memory are decoupled from the top-level design, reducing the complexity of the top-level place and route.

The dual-level clock gating and the reduced wire-length lead to significant savings during write (–66%) and read (–37%) operations, as shown in Fig. 7(b). Furthermore, the SCM structure, which includes single signal gating of the output multiplexer [9], as well as the limited buffering and upsizing requirements due to the tight layout, lead to an 80% reduction in leakage power. Comprehensively, the total power consumption is reduced by about 54% for the considered many-ported structure.

The results of the 20W/20R VRF benchmark are summarized in Fig. 8 and Fig. 9, considering a varying number of words from 16 to 128 rows. Three vector widths (16, 32, and 64-bits) are considered for both Controlled and RTL approaches, presenting a total of six data points (i.e., six bars) per memory row. Specifically, RTL technique is represented with dash-line bars shown on the top of the

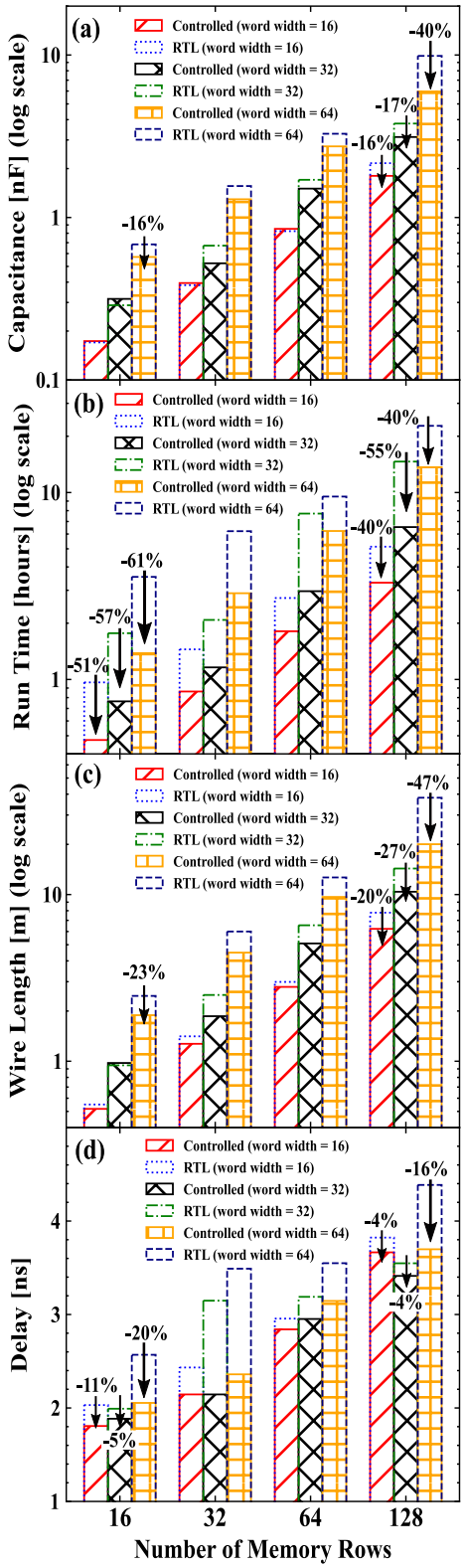

FIGURE 8. Post-layout gate-level simulation results in terms of capacitance, wire length, run time, and delay, for a many-ported (20W/20R) register file as function of the memory capacity at a 65 nm node and V_{DD} = 1.2 V. Results consider the proposed Guided controlled placement technique.

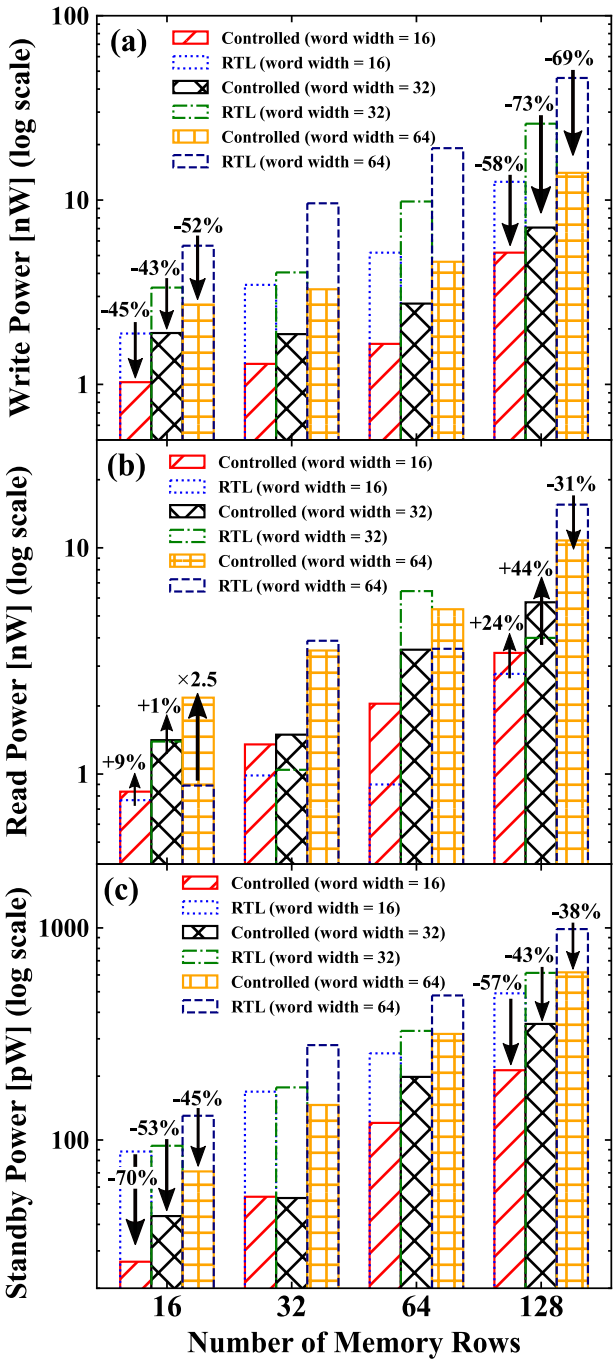

FIGURE 9. Post-layout gate-level simulation results in terms of power consumption for a many-ported (20W/20R) register file as function of the memory capacity at a 65 nm node and V_{DD} = 1.2 V. Results consider the proposed Guided controlled placement technique.

Controlled approach represented with solid-line with filled pattern. Note that all the designs were implemented with the guided-technique, which enables changing the memory specs with an automated utility without the need for manual intervention or optimization.

TABLE 2. Summary comparison of many-ported memories.

Metrics	This work (proposed)	This work (proposed)	This work (typical)	This work (typical)	[35]	[36]	[3]	[28]
Design approach	SCM (Guided)	SCM (Tightly-controlled)	SCM (RTL)	SCM (RTL)	Full-Custom	Full-Custom	Full-Custom	Full-Custom
Process Node (nm)	65	65	65	65	65	65	90 (65)**	90 (65)**
Supply Voltage (V_{DD})	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.1
Write Ports	20	3	20	3	2	2	1	10
Read Ports	20	5	20	5	2	4	5	12
Area (μm^2/bit)	174.8	45.9	174.8	45.9	11.7	44.9	13.9 (7.32)**	–
Word width (bits)	32	32	32	32	64	32	32	64
Num. of words	32	32	32	32	32	32	128	12
f @ V_{DD} (MHz)	466	391	318	359	2048	424‡	1587 (2128)**	1800 (2405)**
Read power (μW/bit)	1.46	0.26	1.02	0.42	7.76	1.25‡	1.41* (0.92)**	521* (338)**
Write power (μW/bit)	1.83	0.20	3.96	0.56	–	1.25‡		
Leakage power (μW/bit)	0.05	0.01	0.17	0.06	0.09	–	0.07 (0.05)**	178† (116)**

* works from [3], [28] reported only the overall power
† register file within Intel Montecito Microprocessor
‡ original values reported at $f = 150$ MHz and V_{DD}= 0.6 V were scaled to 1.2 V [37]
** estimated values exploiting the scaling trends and equations reported in [37]

Fig. 8 shows that, as compared to the RTL implementation, the guided-technique provides savings of as much as 40%, 47%, and 16% in total capacitance, wire-length, and access delay, respectively. Furthermore, the implementation requires between 40%–61% less run-time using the proposed approach. However, the most substantial savings are in write and standby power, as shown in Fig. 9, despite read power increasing for some of the VRF sizes. Specifically, for write power, the Controlled technique results in reductions of 43–59% compared to the RTL approach across the various word widths and RF sizes (16, 32, 64, 128). Similarly, for standby power, the Controlled method leads to substantial decreases of 38–70% versus RTL counterpart.

C. COMPARISON

Table 2 summarizes the main results of the post-layout gate-level simulations for different register file implementations. In particular, the primary metrics of 32×32-bit 3W5R and 20W/20R many-ported designs, implemented with the proposed tightly-controlled and guided techniques, respectively, are provided alongside equivalent macros implemented with a conventional RTL approach within the same silicon footprint. The metrics of several published full-custom-designed register files in 65 nm and 90 nm processes with 2–10 write ports and 2–12 read ports are also provided in the table, with the 90 nm results scaled to 65 nm for direct comparison. The tightly-controlled and guided macros excel in almost every metric. In particular, when comparing the proposed 3W5R register file against existing designs with fewer than 10 ports [3], [35], [36], we can observe that, owing to the efficient allocation of the SCMs, the proposed design leads to the best alternative in terms of power consumption, while offering competitive operating frequency. Furthermore, in contrast to a published many-ported register file [28], the proposed implementation methodology for the 20W/20R register file provides outstanding power savings (several orders-of-magnitude) at the cost of reduced frequency.

V. CONCLUSION

In this work, we presented novel controlled-placement approaches for low-power and area-efficient many-ported memories. Out study was performed using a slightly modified digital implementation flow and exploiting a commercial 65 nm 1.2 V CMOS technology. To evaluate the performance benefits and power-savings, the proposed techniques were benchmarked against conventional RTL implementations with the same requirements in a similarly sized floorplan. Owing to the careful utilization of the routing resources, the main advantage of the proposed controlled-placement techniques over their non-controlled counterparts is their power savings. Implementing well-defined many-ported memories with a relatively small number of ports using the tightly-controlled technique leads to as much as 2 × total power savings, with a 5 × reduction in leakage power and slightly improved access times, as compared to RTL techniques. In the case of a 20W/20R many-ported memory, the proposed guided technique enables power savings of as much as 69% and 70% for write operation and leakage, respectively, at the expense of increased read power for some macro sizes. Slight improvements in terms of delay

have also been seen between the proposed methodology and typical RTL techniques. This shows that the correct allocation of memories with many write and read ports can be a noteworthy alternative for low power, area efficient, and high-performance microprocessors and accelerators.

ACKNOWLEDGMENT

The authors would like to thank CEVA-DSP for their collaboration in problem definition and use-case specification.

REFERENCES

[1] P. Meinerzhagen, A. Teman, R. Giterman, N. Edri, A. Burg, and A. Fish, *Gain-Cell Embedded DRAMs for Low-Power VLSI Systems-on-Chip*. Berlin, Germany: Springer, 2018.

[2] H. Nichols, M. Grimes, J. Sowash, J. Cirimelli-Low, and M. R. Guthaus, "Automated synthesis of multi-port memories and control," in *Proc. IFIP/IEEE 27th Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, Oct. 2019, pp. 59–64.

[3] S.-F. Hsiao and P.-C. Wu, "Design of low-leakage multi-port SRAM for register file in graphics processing unit," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 2181–2184.

[4] T. Huynh-Bao, S. Sakhare, J. Ryckaert, A. Spessot, D. Verkest, and A. Mocuta, "SRAM designs for 5 nm node and beyond: Opportunities and challenges," in *Proc. IEEE Int. Conf. IC Design Technol. (ICICDT)*, May 2017, pp. 1–4.

[5] S. N. Shahrouzi, A. Alkamil, and D. G. Perera, "Towards composing optimized bi-directional multi-ported memories for next-generation FPGAs," *IEEE Access*, vol. 8, pp. 91531–91545, 2020.

[6] T. Song, W. Rim, H. Kim, K. H. Cho, T. Kim, T. Lee, G. Bae, D.-W. Kim, S. Kwon, S. Baek, J. Jung, J. Kye, H. Jung, H. Kim, S.-M. Jung, and J. Park, "24.3 A 3 nm gate-all-around SRAM featuring an adaptive dual-BL and an adaptive cell-power assist circuit," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 64, Feb. 2021, pp. 338–340.

[7] A. Teman, D. Rossi, P. Meinerzhagen, L. Benini, and A. Burg, "Power, area, and performance optimization of standard cell memory arrays through controlled placement," *ACM Trans. Design Autom. Electron. Syst.*, vol. 21, no. 4, pp. 1–25, May 2016.

[8] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Gürkaynak, A. Teman, J. Constantin, A. Burg, I. Miro-Panades, E. Beignè, F. Clermidy, P. Flatresse, and L. Benini, "Energy-efficient near-threshold parallel computing: The PULPv2 cluster," *IEEE Micro*, vol. 37, no. 5, pp. 20–31, Sep. 2017.

[9] P. Meinerzhagen, C. Roth, and A. Burg, "Towards generic low-power area-efficient standard cell based memory architectures," in *Proc. 53rd IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2010, pp. 129–132.

[10] O. Andersson, B. Mohammadi, P. Meinerzhagen, A. Burg, and J. N. Rodrigues, "Ultra low voltage synthesizable memories: A trade-off discussion in 65 nm CMOS," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 6, pp. 806–817, Jun. 2016.

[11] P. Meinerzhagen, S. M. Y. Sherazi, A. Burg, and J. N. Rodrigues, "Benchmarking of standard-cell based memories in the sub-V_T domain in 65-nm CMOS technology," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 1, no. 2, pp. 173–182, Jun. 2011.

[12] X. Fan, J. Stuijt, B. Liu, and T. Gemmeke, "Synthesizable memory arrays based on logic gates for subthreshold operation in IoT," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 3, pp. 941–954, Mar. 2019.

[13] A. Teman, D. Rossi, P. Meinerzhagen, L. Benini, and A. Burg, "Controlled placement of standard cell memory arrays for high density and low power in 28 nm FD-SOI," in *Proc. 20th Asia South Pacific Design Autom. Conf.*, Jan. 2015, pp. 81–86.

[14] E. Garzon, R. Golman, O. Harel, T. Noy, Y. Kra, A. Pollock, S. Yuzhaninov, Y. Shoshan, Y. Rudin, Y. Weitzman, M. Lanuzza, and A. Teman, "A RISC-V-based research platform for rapid design cycle," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2022, pp. 1–19.

[15] CEVA-DSP. (Aug. 2022). *CEVA-XC16 Product Page*. [Online]. Available: https://www.ceva-dsp.com/product/ceva-xc16/

[16] J. Kadomoto, H. Irie, and S. Sakai, "Multiport register file design for high-performance embedded cores," in *Proc. IEEE 14th Int. Symp. Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, Dec. 2021, pp. 281–286.

[17] S. Ataei, M. Gaalswyk, and J. E. Stine, "A high performance multi-port SRAM for low voltage shared memory systems in 32 nm CMOS," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 1236–1239.

[18] A. M. S. Abdelhadi and G. G. F. Lemieux, "Modular multi-ported SRAM-based memories," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*. New York, NY, USA: Association for Computing Machinery, Feb. 2014, pp. 35–44.

[19] M. Lanuzza, M. Margala, and P. Corsonello, "Cost-effective low-power processor-in-memory-based reconfigurable datapath for multimedia applications," in *Proc. Int. Symp. Low Power Electron. Design*, 2005, pp. 161–166.

[20] K. Sarfraz and M. Chan, "A 1.2 V-to-0.4 V 3.2 GHz-to-14.3 MHz power-efficient 3-port register file in 65-nm CMOS," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 2, pp. 360–372, Feb. 2017.

[21] X. Zhang, H. Sun, S. Chen, and N. Zheng, "VLSI architecture exploration of guided image filtering for 1080P@60 Hz video processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 1, pp. 230–241, Jan. 2018.

[22] V. Nautiyal, G. Singla, L. Gupta, S. Dwivedi, and M. Kinkade, "An ultra high density pseudo dual-port SRAM in 16 nm FINFET process for graphics processors," in *Proc. 30th IEEE Int. System-on-Chip Conf. (SOCC)*, Sep. 2017, pp. 12–17.

[23] Z. Yu, H. Zhou, and L. Jiang, "Optimized allocation of FPGA memory for image processing," *Microprocessors Microsystems*, vol. 80, Feb. 2021, Art. no. 103592.

[24] P. Corsonello, S. Perri, G. Staino, M. Lanuzza, and G. Cocorullo, "Low bit rate image compression core for onboard space applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 1, pp. 114–128, Jan. 2006.

[25] P. Zhao, X. Jia, Z. Zhang, H. Ren, and M. S. Tong, "A novel design of high-speed multi-port memory interface for digital signal processor," in *Proc. Photon. Electromagn. Res. Symp. (PIERS)*, Nov. 2021, pp. 2113–2118.

[26] K. Sethi, "Design space exploration of algorithmic multi-port memories in high-performance application-specific accelerators," 2020, *arXiv:2007.09363*.

[27] G. S. Ditlow, R. K. Montoye, S. N. Storino, S. M. Dance, S. Ehrenreich, B. M. Fleischer, T. W. Fox, K. M. Holmes, J. Mihara, and Y. Nakamura, "A 4R2W register file for a 2.3 GHz wire-speed POWERT processor with double-pumped write operation," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2011, pp. 256–258.

[28] E. S. Fetzer, D. Dahle, C. Little, and K. Safford, "The parity protected, multithreaded register files on the 90-nm itanium microprocessor," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 246–255, Jan. 2006.

[29] P. Salz, A. Frisch, W. Penth, J. Noack, T. Kalla, R. Sautter, M. Kugel, O. Torreiter, G. Sapp, M. Lee, E. Fluhr, A. Rozenfeld, J. Pille, and D. Wendel, "A system of array families and synthesized soft arrays for the POWER9$^{\text{Ü}}$ processor in 14 nm SOI FinFET technology," in *Proc. 43rd IEEE Eur. Solid State Circuits Conf.*, Sep. 2017, pp. 303–307.

[30] O. Maltabashi, "Automatic guided physical implementation of common digital structures," MSc thesis, Fac. Eng., Bar-Ilan Universit, Ramat Gan, Israel, 2018.

[31] T. Noy. (2021). *Project Salamandra*. [Online]. Available: https://github.com/enics-labs/salamandra

[32] J. Decaluwe, "MyHDL: A Python-based hardware description language," *Linux J.*, vol. 2004, no. 127, p. 5, 2004.

[33] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanovic, "Chisel: Constructing hardware in a scala embedded language," in *Proc. DAC Design Autom. Conf.*, Jun. 2012, pp. 1212–1221.

[34] A. Amid, D. Biancolin, A. Gonzalez, D. Grubb, S. Karandikar, H. Liew, A. Magyar, H. Mao, A. Ou, N. Pemberton, P. Rigge, C. Schmidt, J. Wright, J. Zhao, Y. S. Shao, K. Asanovic, and B. Nikolic, "Chipyard: Integrated design, simulation, and implementation framework for custom SoCs," *IEEE Micro*, vol. 40, no. 4, pp. 10–21, Jul. 2020.

[35] K. Sarfraz and M. Chan, "A compact low-power 4-port register file with grounded write bitlines and single-ended read operations," *Integration*, vol. 55, pp. 12–21, Sep. 2016.

[36] P.-Y. Chang, T.-J. Lin, J.-S. Wang, and Y.-H. Yu, "A 4R/2W register file design for UDVS microprocessors in 65-nm CMOS," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 12, pp. 908–912, Dec. 2012.

[37] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration*, vol. 58, pp. 74–81, Jun. 2017.

HANAN MARINBERG (Member, IEEE) is currently a VLSI Physical Design Engineer and the Electrical Engineering M.Sc. Researcher with the Emerging Nanoscaled Integrated Circuits and Systems (EnICS) Laboratories, Bar-Ilan University. He has authored scientific papers in international journals and conferences and has participated in several IC tapeouts. His current research interests include algorithms for implementing digital macros physical design at high-density and low-power applications, including developing management netlists tool, memory compiler algorithms, and physical implementation optimization of many ported standard cells memories.

ESTEBAN GARZÓN (Member, IEEE) received the Ph.D. degree in electronics engineering from the University of Calabria (UNICAL), Italy, in 2022. He is currently a Postdoctoral Researcher with the Department of Computer Engineering, Modeling, Electronics and Systems Engineering, UNICAL. He has authored more than 40 scientific papers in international journals and conferences and has participated in several IC tapeouts. His current research interests include domain-specific hardware accelerators, electronics/spintronics, cryogenic memories, and standard and emerging technologies for logic, memory, and low-power applications.

TZACHI NOY received the B.Sc. and M.Sc. degrees in electrical engineering from Bar-Ilan University, Ramat-Gan, Israel, in 2007 and 2019, respectively. He is currently a Senior RTL Component Design Engineer with Google, Israel. His current research interests include EDA optimization heuristics, hardware implementations for low-power machine learning applications, the design of very large-scale integration circuits, and high-performance computer architectures.

MARCO LANUZZA (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from the Mediterranea University of Reggio Calabria, Reggio Calabria, Italy, in 2005. Since 2006, he has been with the University of Calabria, Rende, Italy, where he is currently an Associate Professor. He has authored more than 120 publications in international journals and conference proceedings. His current research interests include the design of ultralow voltage circuits and systems, the development of efficient models and methodologies for leakage- and variability-aware designs, and the design of digital and analog circuits in emerging technologies. He is an Associate Editor of *Integration, the VLSI Journal*.

ADAM TEMAN (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the Ben-Gurion University of the Negev (BGU), in 2014. He was a Design Engineer with Marvell Semiconductors, from 2006 to 2007. From 2014 to 2015, he was a Postdoctoral Researcher with École Polytechnique Frale de Lausanne (EPFL), Switzerland, through the Swiss Government Excellence Scholarship. Since 2015, he has been with Bar-Ilan University, where he is an Associate Professor and the Co-Director of the Emerging Nanoscaled Integrated Circuits and Systems (EnICS) Laboratories. He has authored more than 100 scientific articles and ten patents. His current research interests include embedded memories, energy-efficient circuit design, hardware for artificial intelligence, open source processor platforms and accelerators, and methodologies for physical implementation. He is a member of the technical and review boards of several conferences and journals. In 2020, he was awarded the Krill Prize for outstanding young researchers. He is an Associate Editor of the *Microelectronics Journal*.

Open Access funding provided by 'Università della Calabria' within the CRUI CARE Agreement