Register Files for Embedded Low-Power Applications

Including Microprocessors

by

Vinay Vashishtha

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2014 by the
Graduate Supervisory Committee:

Lawrence Clark, Chair
Jae-sun Seo
Umit Ogras

ARIZONA STATE UNIVERSITY

December 2014

ABSTRACT

Register file (RF) memory is important in low power system on chip (SOC) due to its inherent low voltage stability. Moreover, designs increasingly use compiled instead of custom memory blocks, which frequently employ static, rather than pre-charged dynamic RFs. In this work, the various RFs designed for a microprocessor cache and register files are discussed. Comparison between static and dynamic RF power dissipation and timing characteristics is also presented. The relative timing and power advantages of the designs are shown to be dependent on the memory aspect ratio, i.e. array width and height.

ACKNOWLEDGEMENT

I would like to thank my advisor Dr. Lawrence T. Clark for his guidance throughout the course of my Masters program. He has been a constant source of inspiration and his advice has been invaluable to me. I would also like to thank Dan Patterson, who inspired me with his fastidiousness and helped me understand key timing concepts.

I also thank my defense committee members Dr. Jae-Sun Seo and Dr. Umit Ogras, and my graduate advisor Ms. Esther Korner for their help.

My sincere thanks to Aditya Gujja, Christopher Lieb, Chandarasekaran Ramamurthy, Sandeep Shambhulingaiah, Srivatsan Chellapa, and Sushil Kumar. This work would not have been possible without their support.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

CHAPTER 1. INTRODUCTION

The premise of this thesis lies in the use of static RF architecture for designing caches and register files (RF). This work discusses the static RFs used in various configurations in the design of a microprocessor cache and multi-port register file in the execution unit. The cache and RF are both dual modular redundant (DMR), which implies that two copies of the same logic exist in the design. DMR is a commonly used approach for radiation hardening by the means of introducing hardware redundancy.

Modern designs have become increasingly motivated by minimum energy, rather than the fastest operation—particularly for system on chip (SOC) and internet of things (IOT) applications. RFs are increasingly interesting in that they can use the same power supply as the surrounding logic, even with dynamic voltage scaling (DVS) to low voltages, including subthreshold operation [Chen 2006]. This makes them useful for cache memory as well as their classical use as RF and in other processor structures such as translation lookaside buffers. The use of many RFs to build larger structures, such as cache memory, incentivizes the investigation of their power and delay characteristics with different circuit types.

The designs in this work target the TSMC bulk CMOS 90-nm low power (LP) fabrication process. An important attribute of the static RF cells used in the cache and the multi-port RF is that they are both compatible with the foundry supplied standard cell library. This makes it easier to use the custom designed memory arrays in an automated design flow by allowing for their seamless integration in the synthesis and auto place-and-route (APR) of the caches as well as register files along with the auxiliary logic. This helps to reduce the overall design time.

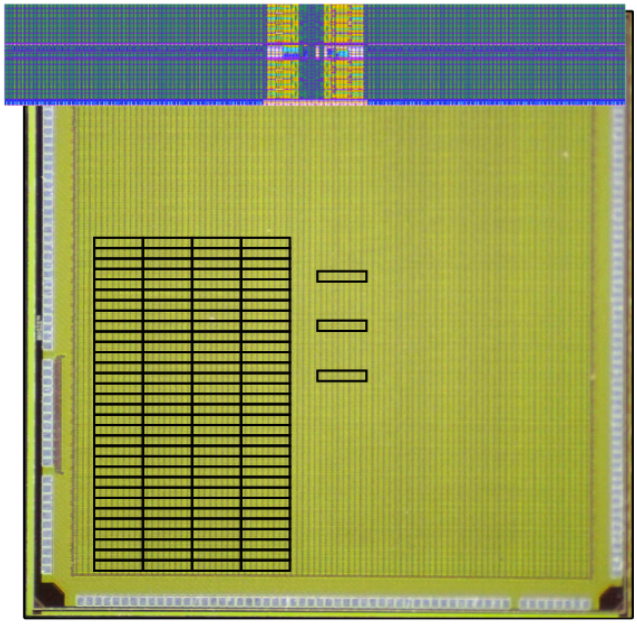


Fig. 1.1 90-nm test die (pre-bonding) showing static RF placement. It is used in two test circuits. The overlay shows the 16 × 128-bit RF layout (horizontal metal 2 and vertical metal 3).

The RF arrays used in the cache are derived from the static RF design used in an advanced encryption system (AES) test logic, and memory controller. The static 16-entry × 128-bit RF has been fabricated in a test chip shown in Fig. 1.1. The design is fully functional from 0.57 V to 1.5 V, the former limited by the foundry library I/O level shifters.

Before proceeding with the description of the RF designs in the upcoming chapters, however, the SRAM and RF cell stability are discussed in this chapter to justify the use of RFs for low power applications. The thesis outline is then delineated in the last section of the chapter.

1.1. SRAM And Register File Stability In the Context of DVS

The proliferation of smart phones and tablets—production volumes of which have exceeded those of microprocessors in the recent years [ITRS 2013]—has established SOC and system in package (SIP) as the mainstay of semiconductor industry growth. This has had the overall effect of changing the primary goal of high performance to low power design, since consumer demand dictates 'all-day-long' operation. As per ITRS estimates, power dissipation due to memories will comprise nearly 65% of the total power consumption in SOCs by 2022. This motivates inquiry into methods to reduce the memory power. A common technique to attain this goal is DVS, whereby the supply voltage (V_{DD}) is scaled down to provide improvements in power dissipation [Govil 1995] that is proportional to the square of supply voltage.

The conventional 6-T SRAM cell stability [Seevinck 1987] is related to the drive strengths of its constituent transistors. Referring to Fig. 1.2, the access transistors PG1 and PG2 must be sized so as to overpower the pull-up transistors PU1 and PU2 when writing the cell. This constraint is referred to as writability. When the SRAM cell is read, a current flows through the node that is at 0 V due to the discharge path formed by the access and pull-down transistors. This current tends to raise the aforementioned node's voltage and can cause a read disturb, whereby the cell flips to an erroneous logic value.

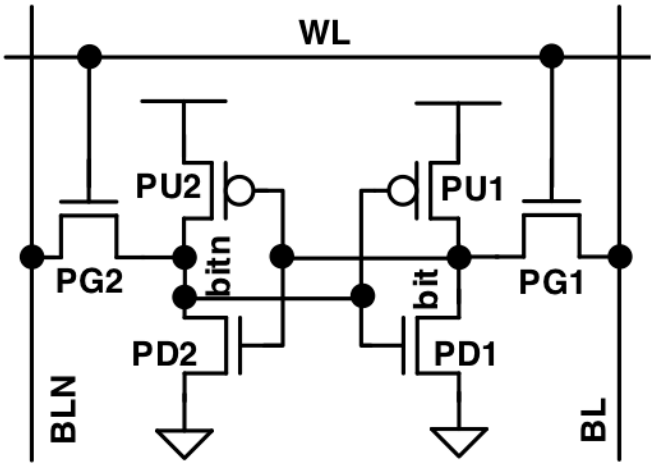


Fig. 1.2 Conventional 6-T SRAM cell.

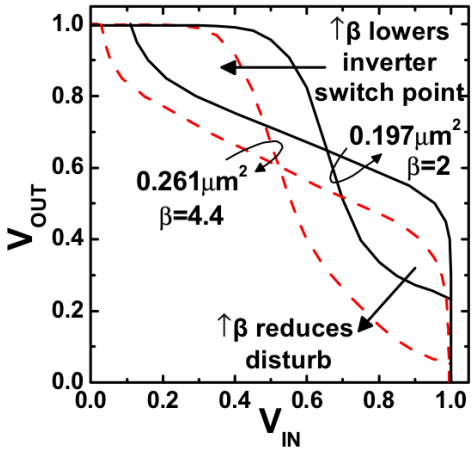


Fig. 1.4 Beta ratio effect on the read SNM of a 6-T SRAM cell (after [Chang 2005]) . Vin and Vout are SRAM input voltages.

To prevent this, the pull-down transistors PD1 and PD2 must be stronger than PG1 and PG2, so as not be overwhelmed by the read current. This gives rise to the read stability constraint. The pull-down to pass gate transistor drive strength ratio β thus affects the SRAM cell's read static noise margin (SNM), which dictates the amount of noise the cell's inputs can tolerate without a disturb to the stored value. The read SNM

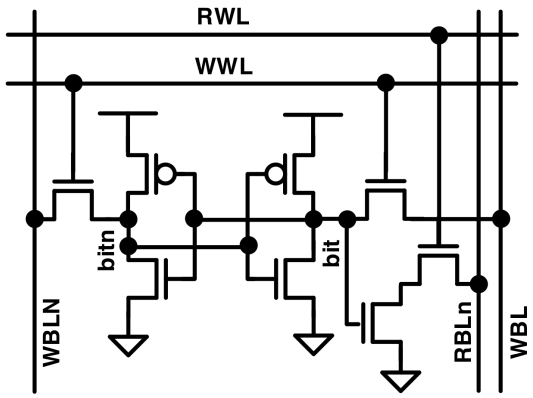


Fig. 1.5 An 8-T RF cell schematic.

is characterized as the smallest square enclosed by the two DC transfer curves associated with the input voltages of the cross-coupled inverters in an SRAM cell. The graph obtained by plotting these two curves together is often referred to as a butterfly diagram due to the shape's resemblance to a butterfly's wings. Fig. 1.4 shows the manner in which increasing the drive strength ratio serves to improve the read SNM by reducing the read disturb.

As mentioned at the beginning of this section, use of DVS for low power operation requires a reduction in V_{th}. However, the subthreshold leakage, which is dominated by V_{th} variations to the low side, increases exponentially as the target V_{th} is reduced. The dominant variation source is due to the random dopant fluctuations [Keyes 1975]. This increases the variability in drive strength ratio due to the lowering of device gate overdrive, thereby adversely affecting the read SNM. Reducing V_{DD} to V_{DDmin}—a quantity that denotes the minimum voltage for reliable SRAM operation—has the effect of further incrementing the drive strength ratio variability, resulting in SRAM yield collapse [Clark 2013].

The drive strength ratio thus creates an impediment to the use of DVS in memories. This issue can be circumvented by using an RF cell, such as the one shown in

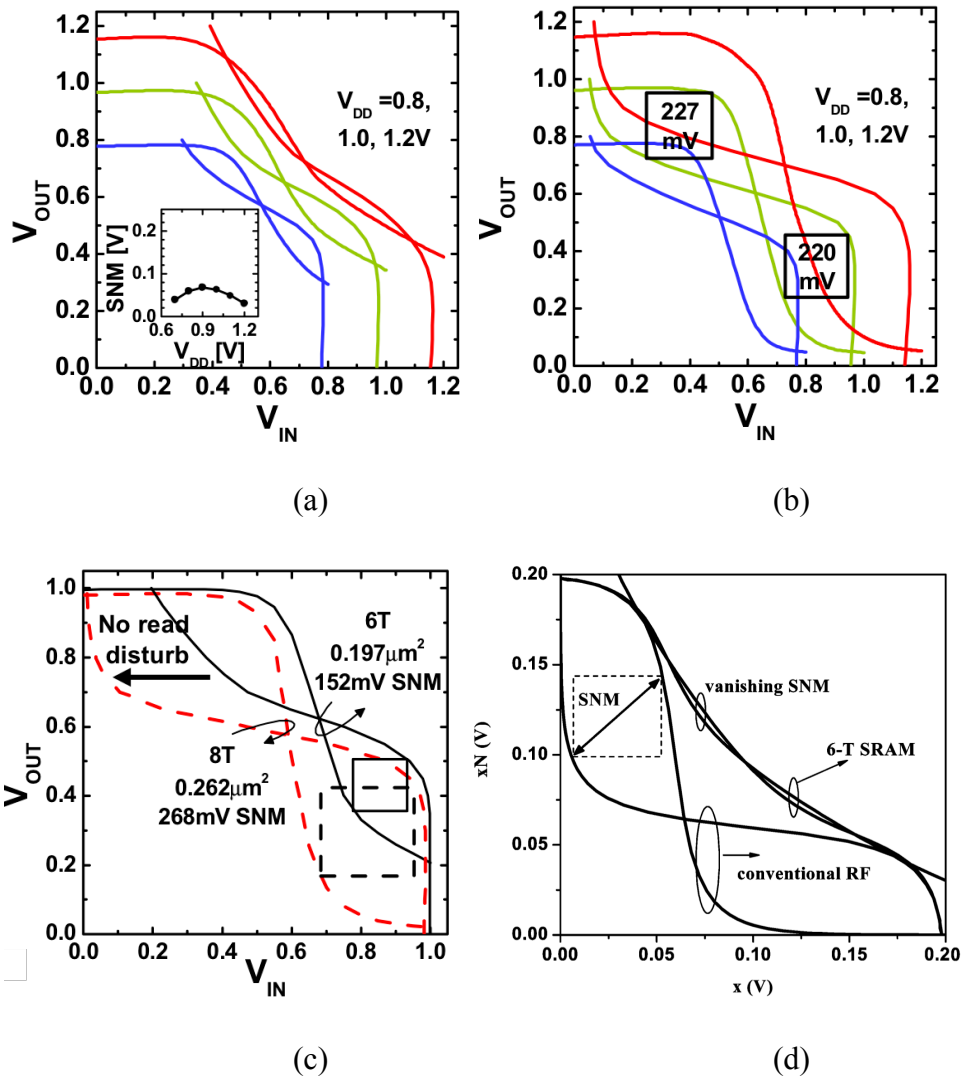


Fig. 1.6 Butterfly curves (after [Chang 2005]) at V_{DD}=0.8, 1, 1.2 V for (a) 6-T SRAM cell SNM, and (b) 8-T RF cell. RF and SRAM cell SNM comparison at (c) 1 V (after [Chang 2005]), and (d) 200 mV (after [Chen 2006]).

Fig. 1.5, where the write and the read ports are decoupled by creating the latter using two additional transistors. Separating these two ports allows for their independent optimization. This improves the read SNM significantly as the read port transistors can be maximized without creating a read disturb since the read current no longer flows through

any of the storage nodes. DVS can therefore be used with memories comprised of RF cells. Fig. 1.6(a) and (b) show the effect of voltage scaling on the SRAM and RF cell, respectively. The difference between the RF cell and SRAM cell—both fabricated on a 32 nm process—SNM due to the elimination of read disturbs is evident in Fig. 1.6(c). The contrast becomes even sharper when the comparison is made in subthreshold region of operation for memory cells fabricated in a 130 nm process.

1.2. Thesis Outline

Chapter 2 discusses the single port static RF used in the microprocessor cache arrays. A 16 × 128-bit data sub-bank and a 16 × 96-bit tag sub-bank were used in the cache design.

Timing characterization was performed for both of these arrays using the Synopsys NanoTime static timing analysis tool and then Cadence abstract generator was used to create the abstracts. The timing model and abstracts thus generated were used in the cache synthesis and APR that are described in chapter 3. Using the arrays as macros for these processes thus allowed to automate the cache design, which is traditionally custom built.

In chapter 4, the multi-port static RF used in the microprocessor's register file unit is presented. The DMR RFs are used as macros at the top-level without having the need to custom design the associated control logic. The process of timing characterization is briefly discussed in this chapter besides the decoder APR.

Finally, chapter 5 describes the design of a traditional dynamic RF and compares it against the single port static RF used in the microprocessor cache.

CHAPTER 2. CACHE ARRAY DESIGN USING STATIC REGISTER FILE

2.1. Overview

This chapter discusses the static RF used in the data and tag sub-banks of a microprocessor cache. The 8KB, 4-way set associative cache is used in the microprocessor's data and instruction caches. The primary reason for selecting the static RF architecture was the ease that it affords in contrast to its dynamic counterpart with respect to timing placement, resulting in improved portability to different processes. The secondary reason for choosing static design was the dynamic RF's higher activity factor, which adversely affects the energy-delay product (EDP), in the context of a read operation due to the clocking required for a read. The third reason is improved noise immunity. Chapter 5 presents comparisons between the two RF design architectures and supports the rationale behind using a static RF.

2.2. Static RF Design

This section discusses the design of the static RF cell used in the cache's arrays, the RF column as well as the decoder design, the RF's read timing, and the write timing race that exists in the RF.

2.2.1. Static RF Cell Design

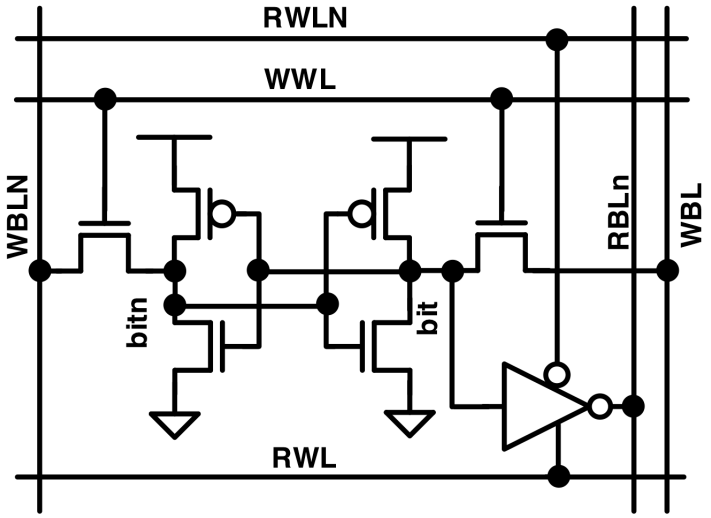
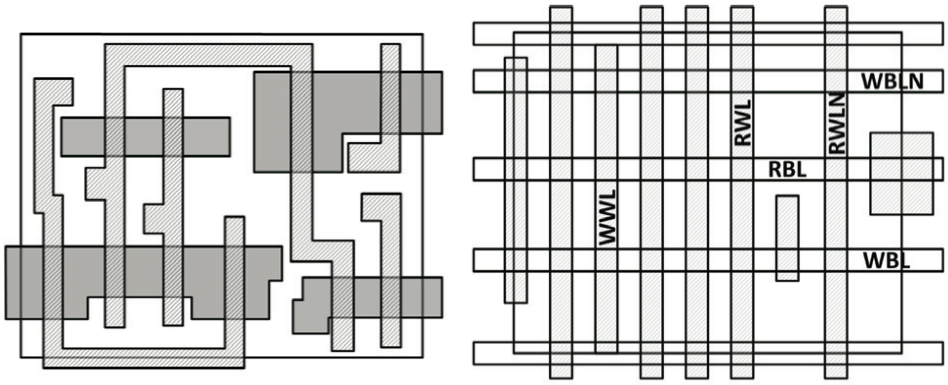


Fig. 2.1 Diagram for the static 10-T single port RF cell used in the microprocessor cache.

The caches use an RF cell with one single-ended read port and one dual-ended write port. A fully static read-out is utilized. This is accomplished by using a tri-state inverter cell output, which, unlike a classic 8-T dynamic RF cell that employs only a read word-line (RWL) for gating the read port, requires a complementary read word line (RWLN) as well to enable the pull-up of the read port. The RF cell is designed to be compatible with the 7-track LP library since the RF cell columns (described in 2.2.2) may then be built using standard cell library gates, thereby reducing the overall custom design effort.



(a) (b)

Fig. 2.2 (a) Static RF cell layout showing poly-silicon and diffusion layers. (b) Diagram showing the metal 2 and metal 3 arrangement in the cell. WLs and BLs are labeled.

The compliance with standard cell library forces metal 3 perpendicular and metal 2 parallel to the poly-silicon gates. This in turn requires the WLs in metal 2 and BLs in metal 3 (see Fig. 2.2 (b)). Although dynamic multi-port RF cells designed using thin-cell layout style have lower area [Hsiao 2014], a static RF cell designed using conventional layout style benefits from the large N-well that can accommodate the static cell's additional PMOS transistors from the tri-state inverter used for read. The RF cell conforms to all aspects of the standard cell, namely well dimensions, height and power-ground rails. The only exceptions to this are the diffusions (see Fig. 2.2 (a))—shared with the adjoining RF cells in a column to lower the diffusion capacitance and attain compact area—that connect to the read bit-line (RBL) as well as one of the write bit-lines (WBL). Additionally, the poly-silicon layer, which connects to the write word-line (WWL) gating the access transistors, protrudes from the place-and-route (PR) boundary due to area constraints, but these are shorted between the RF cells belonging to the same entry without affecting the functionality when the RF bitcell columns are arrayed. This results

in further area savings. As a consequence of the two aforementioned features, column containing special edge cell—either filler or custom decoupling capacitor cells—are required to avoid interaction with standard cells at all the array boundaries. The read port transistor sizes are tuned to minimize the limiting RBL rising edge delay. Larger NMOS devices would fit, but increase RBL capacitance and thus, the rise delay. The RWLN is limiting, so the layout provides extra line-to-line spacing to minimize its coupling capacitance as shown in Fig. 2.2 (b).

2.2.2. RF Column Design

The RF cell column, shown in Fig. 2.3, is comprised of two RF cell groups, each containing eight cells.

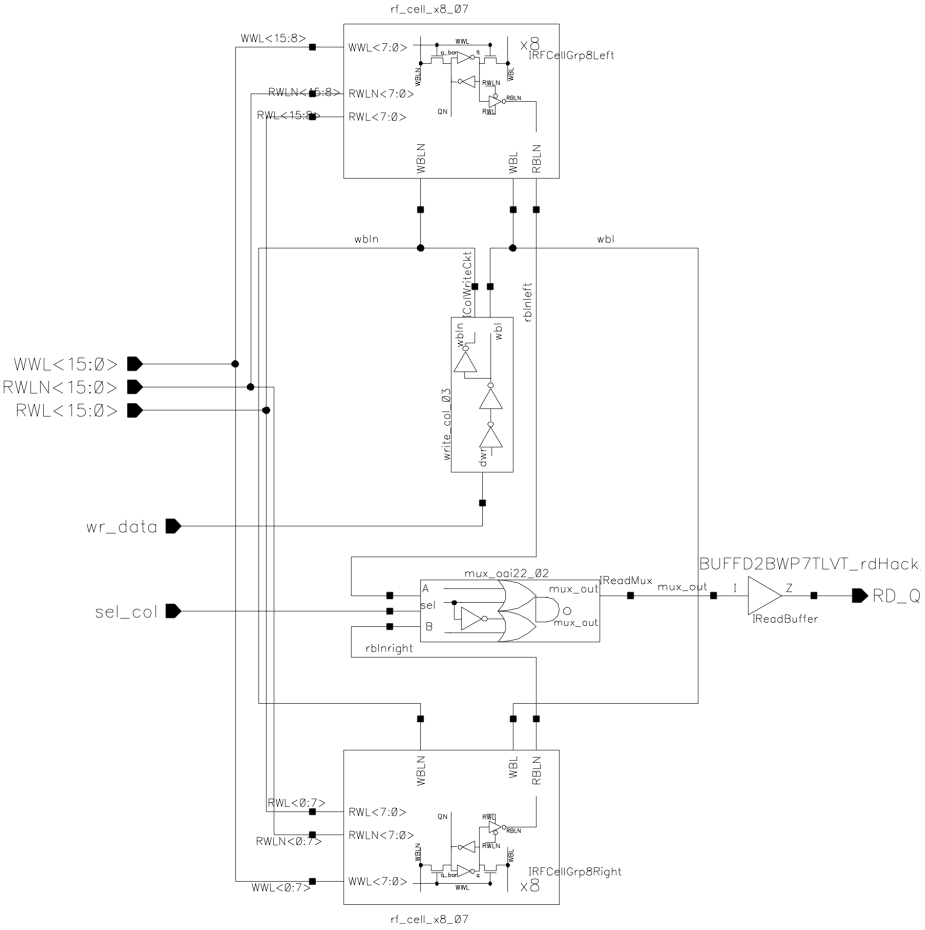


Fig. 2.3 Block-level schematic for the RF cell column used in cache's sub-banks.

The write circuit consists of drivers (inverters and buffers) that drive data of opposite polarity onto the WBL and the complementary WBLN, which are required for a differential write to the cell. Both WBLs are shared by all the cells in an RF column. Unlike the WBLs, however, the single RBL is divided into two, viz. a top and a bottom RBLN, so that each is shared by 8 RF cell. This is done in order to

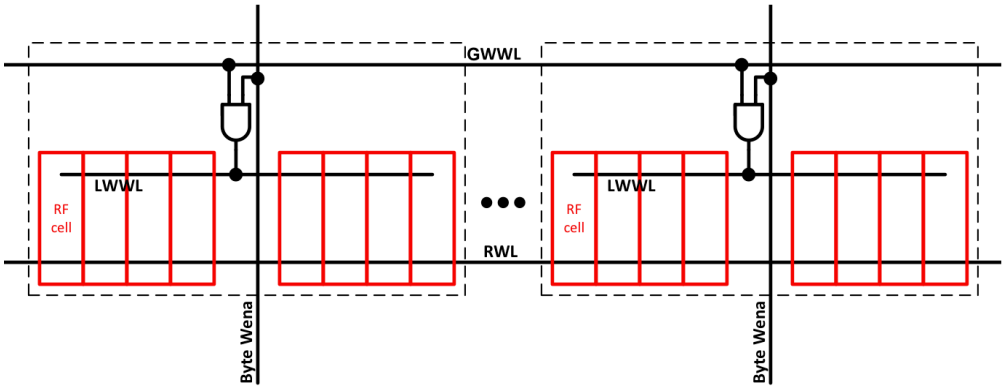


Fig. 2.5 Diagram showing the byte write enable scheme.

minimize the parasitic capacitance and resistance on the RBL and thus, the read power and delay. The read circuit is comprised of a multiplexer—required due to the RBL bifurcation—that selects either the top or the bottom RBL depending upon the multiplexer select signal, which is a buffered version of the read address most significant bit (MSB). The inverting multiplexer is implemented using a static OAI22 gate. No latch is required at the output since the read-out is fully static.

2.2.3. WWL With Local Byte Write Enable Gate

To support byte writes, both data and tag sub-banks consist of a column group (see Fig. 2.5) with local write select gates that qualify a byte write enable signal against the global write word-lines (GWWL) to generate local write word-lines (LWWL) for every eight RF cell columns. Each column group is nine standard cell heights. Instances of local WL generation for saving energy and improving delay can be found in literature ([Itoh 1997, Yoshimoto 1983]). Such a scheme requires that the RF columns belonging to a particular group are adjacent. This, combined with the fact that LWWL cannot be interleaved due to the high metal 2 density, prohibits bit interleaving—a technique used to provide

protection against multiple bit upsets (MBU) due to radiation. The application here allows a design that lacks bit-interleaving. The interleaved disparate bit groups cause the half-select disturb that arises from the read occurring during a write operation for the unselected column groups to which the data is not being written to [Na 2012]. The problem is even more severe in RF cells since they have large access transistors optimized for write that exacerbate the half-select disturb. Future work will address the issue of accomplishing bit interleaving without causing the half-select disturb whilst using the same RF design so that hardening may be achieved without DMR.

2.2.4. Decoder Design

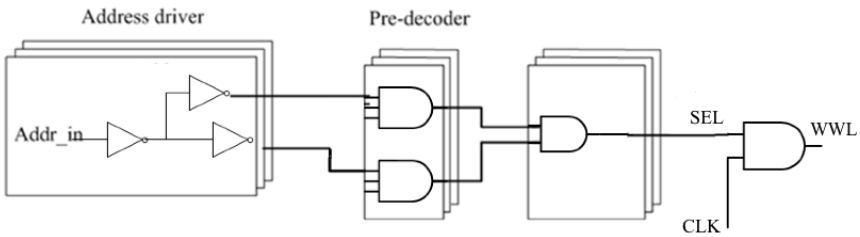


Fig. 2.6 A 4-to-16 WWL decoder with pre-decoders (after [Xavier 2012]).

A static 4-to-16 decoder, such as the one shown in Fig. 2.6, is used for decoding the write addresses. Predecoders are used to reduce the area overhead. The decoded write addresses (SEL) are clock gated to enable write operations. The read addresses are decoded using two 3-to-16 decoders to prevent RBLs from floating, which can otherwise produce large leakage currents at the read multiplexers. RWLN generation for the static read port operation requires additional circuits that result in additional area and power overhead.

Caution must be exercised when choosing the transistor gate widths obtained from 'back-of-the-envelope' logical effort calculations for decoders and it is important to consider the coupling capacitance—a parameter that can often exceed the gate capacitance due to closely spaced metal tracks in a decoder.

In our application, the flip-flops (FF) driving the addresses of individual banks are clock gated, so addresses change only on new read operations. This eliminates all but leakage power for unselected banks.

2.2.5. Timing

This section discusses read and write timing of the single port static RF. The access times for the static RF are presented in chapter 5, alongside those for the dynamic RF.

2.2.5.1. Read Timing

Unlike the dynamic RF design, which requires a clock for read operation that consists of a pre-charge and an evaluate phase, the read operation in a static RF does not require a clock or constitute multiple phases.

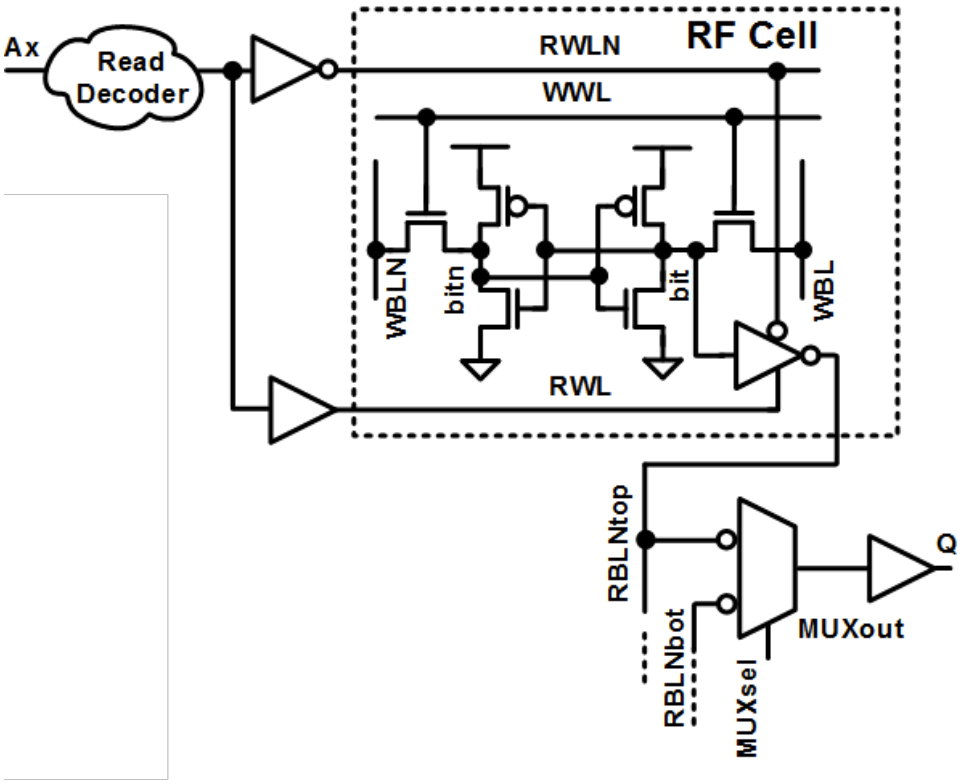


Fig. 2.7 Single port RF storage cell and read path diagram.

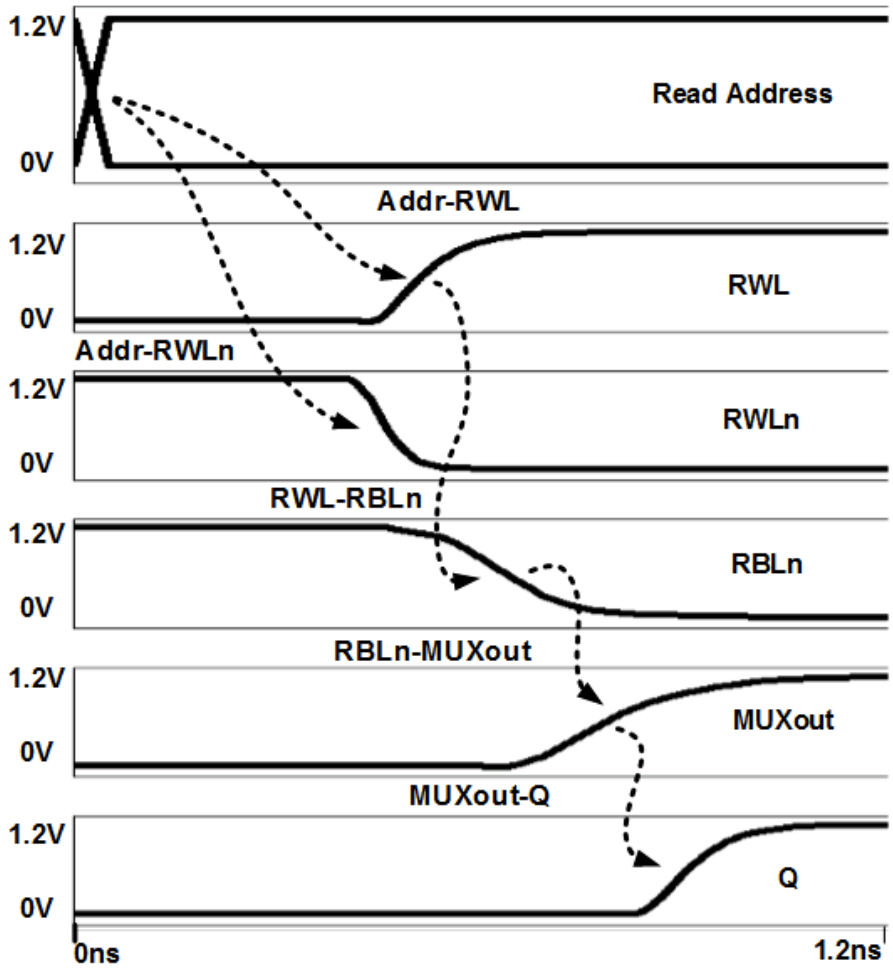


Fig. 2.8 Waveforms showing read timing critical path for the static RF used in the cache's sub-banks.

A read occurs when a new address arrives at the read decoder input (see Fig. 2.7) and after being decoded, turns the read port tri-state inverters for an entry—corresponding to the decoded read address—on by RWL assertion and RWLN de-assertion. The RBL either charges or discharges depending upon whether the cell stores a 0 or a 1, respectively. Based on the read address MSB, the inverting read multiplexer propagates the value of either RBLNtop or RBLNbot to the output, which is then further buffered. This read timing is shown in Fig. 2.8. The access time t_{ACC} of the static RF is given from read address to Q.

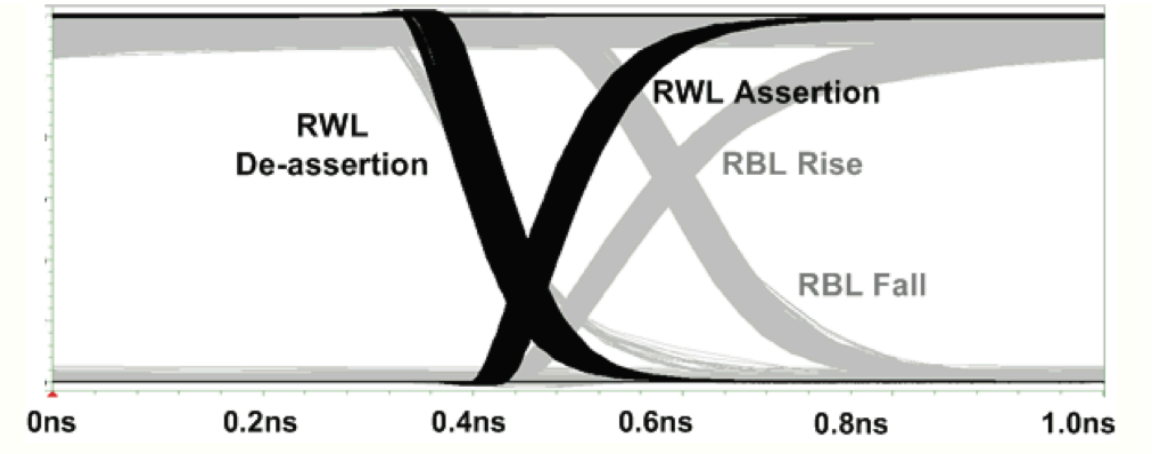


Fig. 2.9 Static RF read timing variation due to decoder variation in assertion and de-assertion timings from RWL to RWL.

The RWLs are mutually exclusive since the decoding is one-hot. Also, RWLs and WWLs are mutually exclusive because a read operation is not permitted while writing an entry due to the uncertainty in data that is read-out. The two aforementioned constraints hold true for both static and dynamic RFs.

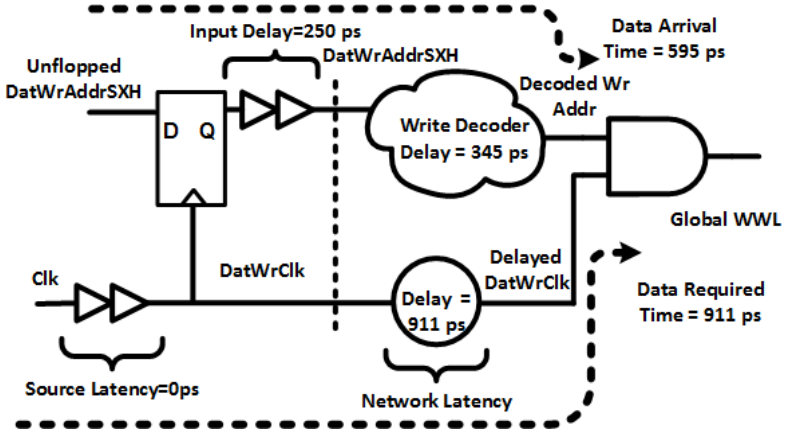
No read enabling is required since the RF design is static. The absence of enabling circuitry means that RWLs asserted during a previous read are not de-asserted completely before the current read and since path delays in the decoder vary, RWL assertion corresponding to the decoded address may slightly overlap another row's RWL de-assertion. This has a negative impact on the power, as it causes a brief DC contention path through the BL. A simulation with 64 read operations shown as an eye-diagram in Fig. 2.9 demonstrates the RWL and RBL timing variation, even with no transistor variability included.

The top/bottom multiplexer select signal MUXsel must be delayed to coincide with the RWL/RWLN pair to be selected for a correct read. Otherwise, data from the

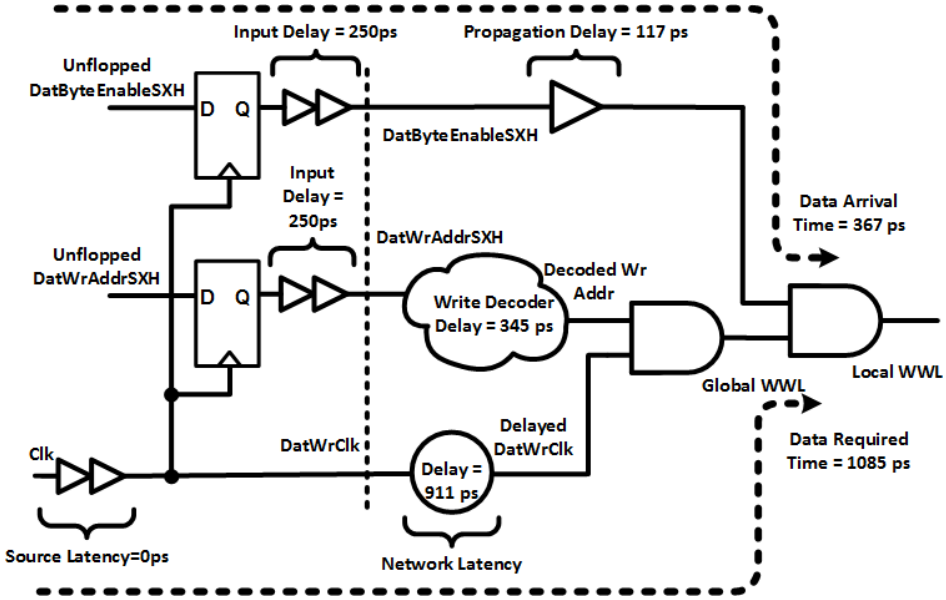
RBL that was not intended to be read, may propagate to the output if MUXsel for the current read is different from its value during the previous read. Since the correct RBL is eventually read, this does not affect the functionality. However, the undesired read does cause additional power dissipation that is preventable by delaying MUXsel.

2.2.5.2. Write Races

Flip-flops (FF), instead of latches are chosen as the sequential elements for gating the addresses and enable signals. This introduces races when performing write operations in



(a)



(b)

Fig. 2.10 Write races in the cache arrays. (a) Shows the race between the decoded write address and the delayed write clock. (b) Shows the race between the byte write enable and the delayed write clock.

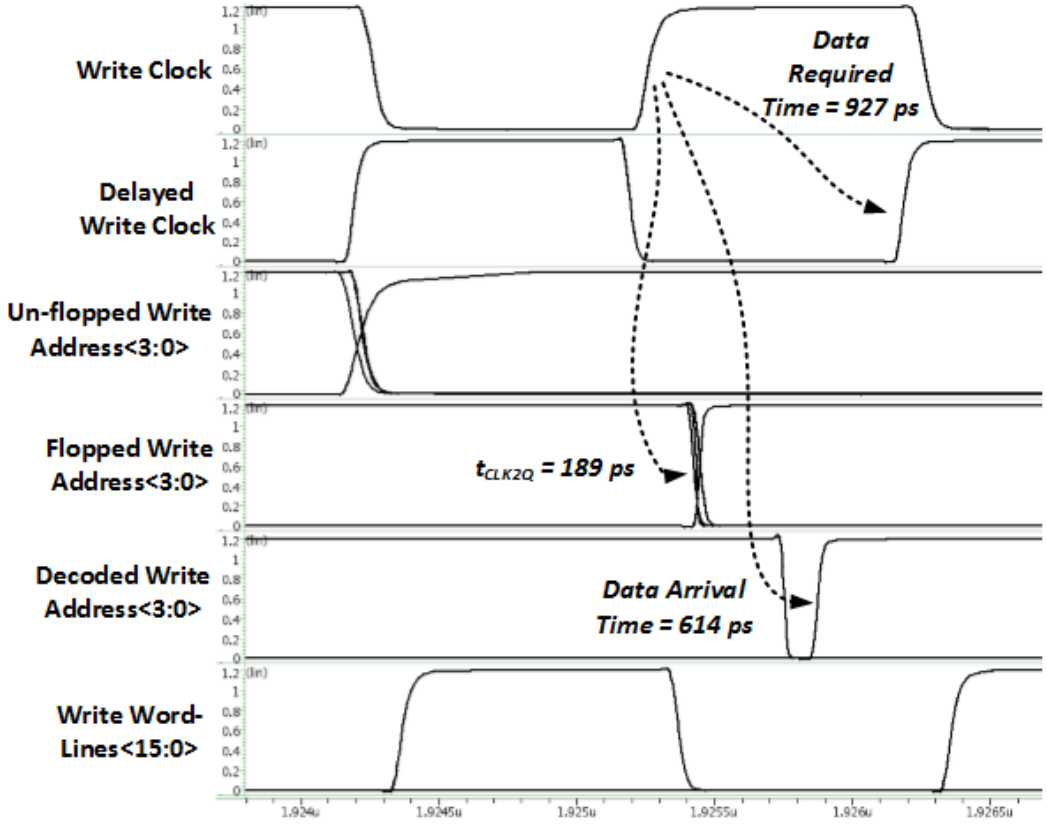


Fig. 2.11 Post-layout waveforms for the cache showing the successful write race mitigation.

the cache data and tag sub-banks, since the clock used for FFs is also used for enabling the decoded write addresses and byte write enables. This creates a point of divergence and a point of convergence with regards to the clock at the clock gates (see Fig. 2.10). As a result, races arise due to the following timing conditions in the design.

Decoded write address from previous clock cycle must de-assert before the current write data clock rising edge to prevent incorrect entry from being written to. The byte write enable from previous clock cycle must likewise de-assert before the current GWWL. Both these constraints require that the write clock be delayed. NanoTime static

timing analysis ascertained the optimal write clock delay in order to alleviate the races. The race condition between the decoded write address and write data clock, as well as that between byte write enable and write data clock, becomes evident when gated clock setup check is performed in the same clock cycle (see Fig. 2.10 (a) and (b)). Thereupon, a delay of 911 ps is added in the write data clock path to ensure decoded write address set-up to the delayed write data clock, assuming a 250 ps input delay that includes the FF t_{CLK2Q}. This also guarantees that the byte write enable sets up ahead of the GWWL since the propagation delay in its path, and thus the associated data arrival time, is less than that for the decoded write address. Besides adding delay in the clock path, an additional margin of 250 ps is added to the setup time corresponding to write address pins in the liberty file (.lib) to guarantee correct post-APR timing. Waveforms (Fig. 2.11) obtained from the simulations run using UltraSim (a fast SPICE circuit simulator) on the post-APR cache at 25^{o}C TT in mixed-signal mode, show that the amount of delay added to the write clock is sufficient for successful mitigation of write races. The proximity of delay numbers obtained from Nanotime static timing analysis and those from dynamic simulations using UltraSim shows that static timing analysis can be used as a reliable method to determine critical timing aspects in a design, such as races. The t_{CLK2Q} delay of 189 ps in Fig. 2.11 does not include the additional RC delay incurred due to the signal propagation from FF outputs to the array address inputs. Therefore, the decoder delay should not be considered as 425 ps (614 ps-189 ps), which far exceeds the 345 ps delay estimated with static timing analysis.

2.3. Cache Sub-Bank Description

2.3.1. Data Sub-Bank

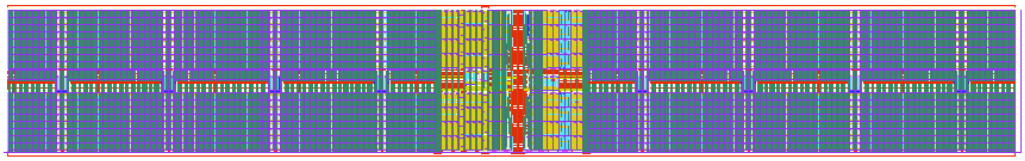


Fig. 2.12 Data sub-bank layout (metal 2, metal 3 directions are horizontal and vertical, respectively)

The data sub-bank (layout shown in Fig. 2.12) has 16 entries, each 128 bits wide (16 bytes), organized as four 32-bit words. Each of these words belong to a different cache way. A data sub-bank read results in an entire row being read. By using the byte write enable scheme described in 2.2.3, each of the 16 bytes in a row can be written independently, with at most 4 bytes of the same word in an entry written at any given time. The write operation occurs in the high phase of the clock. Referring to Fig. 2.12, the read and write decoders lie in the center, and the RF columns are to their left and right.

2.3.2. Tag Sub-Bank

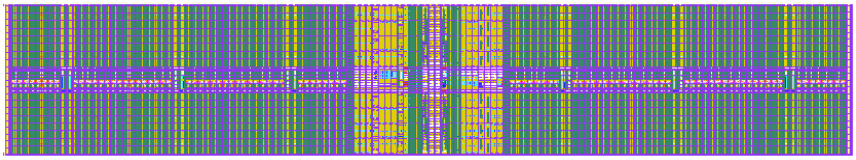


Fig. 2.13 Tag sub-bank layout (metal 2, metal 3 directions are horizontal and vertical, respectively).

The tag sub-bank, shown in Fig. 2.13, has a floorplan that is identical to that of the data sub-bank. It uses the same read and write address decoders as the data sub-bank since it also has 16 rows. However, it is 96 bits wide and is organized as four 24-bit tags where each tag belongs to a different cache way. At most one tag in a row is written at a time by employing the byte write enable scheme. Each tag consists of 21 tag address bits, a valid bit, a lock bit and a least recently filled (LRF) bit. It thus uses three 8-bit groups sharing a single write enable. The valid bit cell is resettable via an additional NMOS transistor that turns on when a global invalidation signal is asserted to perform an invalidation of the cache.

CHAPTER 3. CACHE SYNTHESIS AND APR

3.1. Overview

Cache constitutes of high speed memory, such as SRAMs or RFs, and forms the first level in a microprocessor memory hierarchy. It contains the most frequently accessed information from the main memory that lie outside the microprocessor. This helps ease the bottle-neck between the processor and the slow main memory by reducing the time it takes to access the data. The concept behind storing frequently accessed information is related to the principle of locality, which states that information with addresses in close range of each other tend to be referenced close together in time (temporal locality) and that recently accessed information has a higher likelihood of being accessed imminently [Patterson 2007]. The presence of information in a cache is referred to as a ‘hit’ and its absence as a ‘miss’.

3.2. Cache Architecture

The microprocessor cache in this work, named ‘CacheSD’, is 4-way set associative, which implies that information from the main memory can only be placed within a particular set comprising of four blocks. The cache is used in both data and

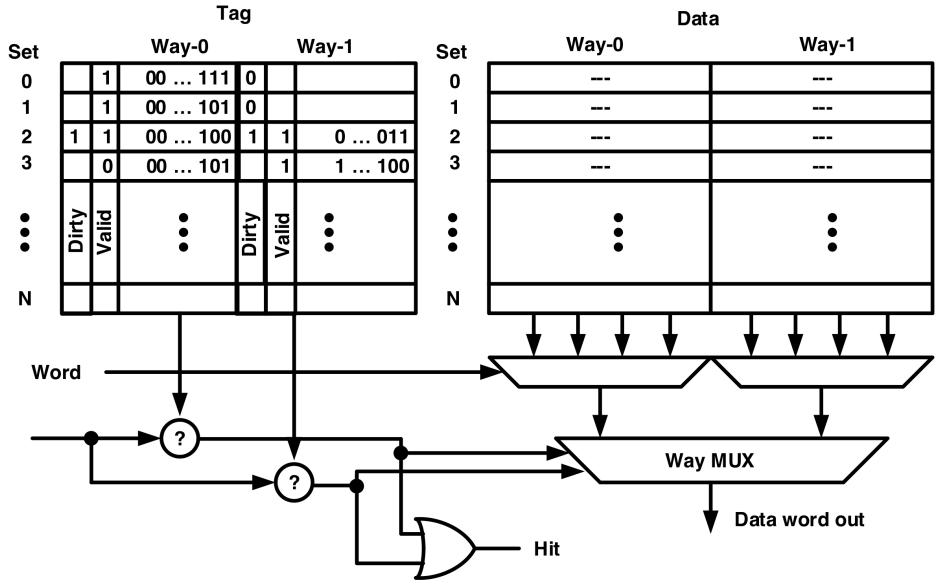


Fig. 3.1 A basic 2-way cache.

instruction cache which, as the name indicates, are used to store data and instruction, respectively. The data cache uses a write-through type write policy, whereby information is written to both the cache as well as the main memory in order to ensure higher cache coherency. This allows for the reinstitution of information to the cache from the main-memory in the event of an error in the former. The data cache is also read-allocate, implying the allocation of memory space in the cache when a read miss occurs.

CacheSD is comprised of 8 groups, each containing 1KB of cache data and the logic to determine the group being accessed so as to toggle inputs to only the selected sub-banks for power savings. The 1KB group is called ‘CacheClusterSD’ that contains

one 16-entry × 96-bit tag sub-bank, each comprised of four tags corresponding to four ways, and four 16-entry × 128-bit data sub-banks, each associated with one of the four words in a cache line and each containing four ways. Thus, the ways are interleaved. The group is divided into tag and data sub-banks so as to allow for their independent addressing in order to speed-up the hit generation and data read-out.

The cache can perform four operations, viz. lookup, read, write and global invalidation [Yao 2009]. During a lookup, all four ways are read and the tag address is compared against the physical address from translation lookaside buffer (TLB). In the case of a match, the selected way is read along with the generation of a hit signal. Only the data sub-banks for the target word is addressed and the word, associated with the way that is hit, is read. If a mismatch occurs, then the miss signal is generated. The read operation allows for reading information from an addressed set and way. The write operation permits an information write to an addressed set and way of data and tag sub-banks. The minimum number of bits that can be written is eight. In the event of an error, power-up or reset, the whole cache can be invalidated using global cache invalidation in order to prevent hitting the fallacious data. This is accomplished by marking the cell invalid by pulling down on the bit cell.

3.3. Cache Design Flow

The cache described in this work was designed using a conventional ASIC flow for faster design realization, which is described in the following sub-sections.

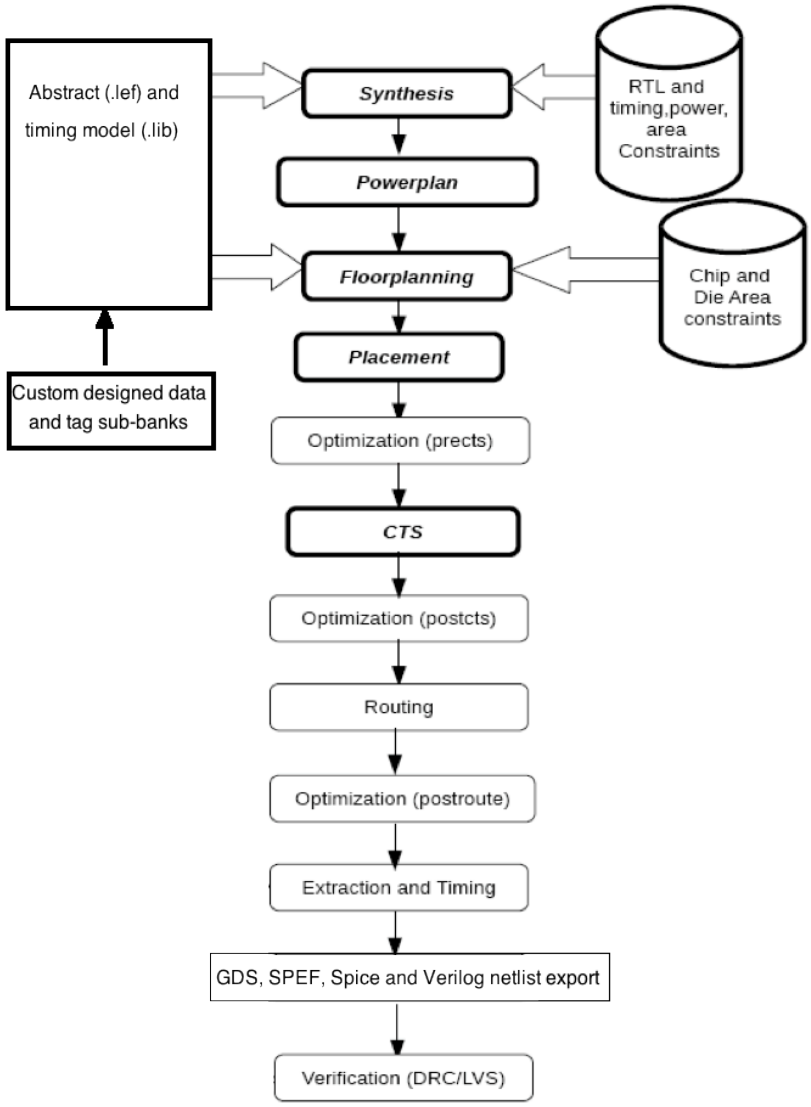


Fig. 3.2 Conventional ASIC design flow (after [Ramamurthy 2013]).

3.3.1. Sub-Bank Custom Design

First, the data and tag sub-banks are custom built (see Chapter 2) using Cadence design suite. Custom design of these blocks enables exerting tight control over the layout, which allows achieving a high density and speed. In addition to interleaving the ways, the four bytes belonging to a single way in a word are also interleaved. This is done in a manner so that any two bytes affiliated to the same way are four bytes apart. Thus, the four continuous groups of 32 bits in a word are each comprised of four bytes, each belonging to a different way. By doing this, length of the horizontal routes created by swizzling of read output metal tracks associated with the same byte of all four ways that must be brought-in closer for multiplexing, is reduced.

Upon design completion and physical verification, each sub-bank is tested by utilizing a set of test vectors in the form of a digital vector file that are used to provide inputs to the cache arrays in dynamic simulations run with UltraSim. The simulations outputs are then compared with the expected values within the digital vector file to verify the design veracity.

3.3.2. Sub-Bank Abstraction

For the arrays to be used during synthesis and APR, they must be used as macros—which unlike the foundry supplied standard cells, are user-designed blocks. Using custom designs as macros requires timing and physical abstraction of these designs. To this end, a timing model describing the design's timing behavior, and an abstract containing the physical locations of various back-end layers, are created.

The timing model is extracted using Synopsys's NanoTime static timing analysis tool by a process known as timing characterization. This process is explained in more

detail in chapter 4. A liberty file (.lib) embodies the timing model and contains a design's timing characteristics, such as input-to-output delays, setup and hold constraints, as well as output slews. A .lib file is required by synthesis and APR tools, so that any logic connected by these tools to the design's inputs and outputs, does not violate the constraints necessary for the correct design functionality in addition to ensuring the optimal design performance.

A physical abstract—often referred to as simply the 'abstract'—of the design is necessary during the APR process. It is represented in the form of a library exchange format (.lef) file, and contains a design's physical information, such as the names of the inputs, outputs, and power pins, the metal layers that they connect to, the obstructions created by any metal layers that are not pins, location of the place-and-route boundary, as well as the antenna gate and diffusion areas. The pins and obstructions are defined as the co-ordinates—with respect to a block's origin—of the metal layers that they are associated to. These provide the APR tool with the correct information about the metal layers that must be routed to while avoiding obstructions and without creating any shorts. Cadence's abstract generator was used to create a .lef file for the designs presented in this work.

3.3.3. Cache Cluster Synthesis

As per the flow in Fig. 3.2, the process of synthesis is carried out using Cadence's RTL compiler (RC) following the abstract and timing model generation. Two RTL codes, one for CacheSD and another for CacheClusterSD are used for cache synthesis. Both data and tag sub-banks are instantiated as components within the latter code and regardless of their extant RTL, the respective .lib files are instead used by setting the attribute

'hdl_use_techelt_first' to true, which ensure the .lib file precedence over the RTL during synthesis. Any violating paths must be carefully scrutinized to ascertain whether they are legitimate paths or if they are false. Setting false paths during synthesis only works if the entire logic in the path is created by the synthesis tool. If the .lib file for the macro already contains a false path, then it must be excised by regenerating the timing model. Pertaining to tag sub-banks, all paths from its invalidate pin—which assists the global cache invalidation by providing a means to nullify the valid bits at a lower hierarchy level—to all its outputs are declared as false during timing characterization since the functionality related with the signal is actualized upon invalidation and no output transition is associated with it. Constraints from the lookup signal to tag sub-banks' outputs are constricted for speeding up the path. At first, only CacheSD is synthesized. During the process a standard timing constraint (.sdc) file for CacheClusterSD is generated. This file is then used as an input during the synthesis of CacheClusterSD that is set in motion after the first synthesis run. This is done because CacheSD is comprised of eight identical CacheClusterSD and synthesizing each unit independently allows for better timing optimization besides ensuring design symmetry and ease of synthesis control. Finally, a synthesized gate level verilog netlist and an .sdc file are exported upon synthesis completion.

3.3.4. Cache Cluster APR

The .sdc file and the verilog netlist thus generated for CacheClusterSD are then used to commence with the APR, during which, the data and tag-subbank .lib and .lef files are also provided. Cadence's Encounter is used for APR. First, careful power-planning is carried out to ensure that the metals tracks allocated to power supply lines

coincide with those for the chip. Wide metal layers are used for the purpose of routing power supply lines to ensure robustness and lower the resistance. Following this, floorplanning is done to ensure least amount of metal routing, congestion, and delay. The macro locations are specified during this step since the arrays form the biggest components of the floorplan and the manner in which they are arranged affects the design timing tremendously. Several iterations are required to achieve an optimal floorplan. After this, standard cells are placed during the placement step. The relative location of standard cells with respect to the macro also determines the design timing and density. Pre clock tree synthesis (CTS) optimization is thus required and is done over several iterations by determining the worst-case negative slack (WNS) and then minimizing the total negative slack (TNS), which is the summation of all the WNS in the design. Once a satisfactory TNS is achieved, CTS is then undertaken to build a balanced clock tree comprising of repeaters and wires that route clock to all of the design's components requiring a clock. Optimization is again required after the CTS to fix any remaining violations. Thereafter, the entire design is routed using Encounter's in-built Nanoroute and then post-route optimization is carried out to fix all the timing and design rule violations. The latter is the final step in the APR process and emphasis is placed on getting as clean a design as possible following this step. A graphic database system II (GDSII, or simply GDS), is then exported from Encounter. This contains all the back-end and front-end layer information for the design. Additionally, a post-APR verilog netlist is also generated for various purposes, one of them being the design's spice netlist creation using the v2lvs tool. The GDS is then imported in Virtuoso as a layout and any unresolved design rule violations are fixed. Physical verification, including layout versus

schematic (LVS) check against the generated spice netlist is then carried out. The CacheClusterSD thus obtained is

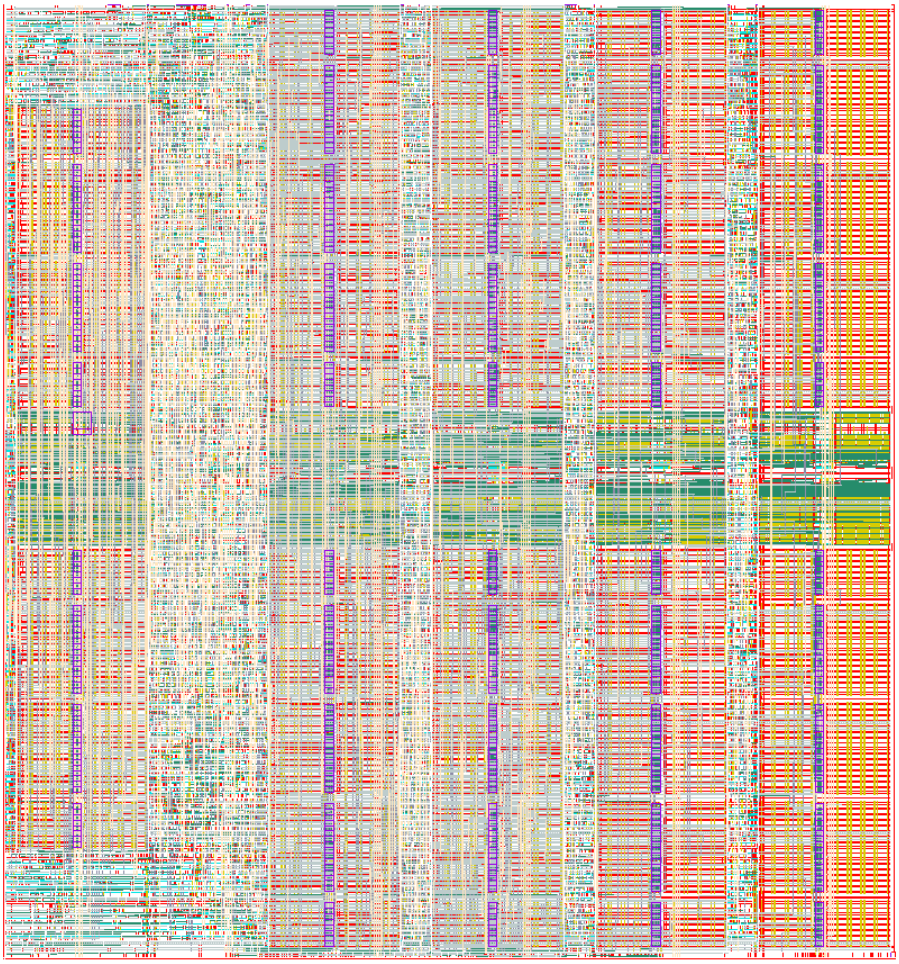


Fig. 3.3 Cache cluster layout.

shown in Fig. 3.3. The smaller tag sub-bank's location is evident to the left of the layout and can be easily distinguished from the four data sub-banks to its right. The large area between the tag and data sub-banks encloses the word and way multiplexers besides the auxiliary logic. The .sdc, .lef files and the verilog netlist thus generated are then used for the cache synthesis and APR at the microprocessor level.

CHAPTER 4. MULTI-PORT STATIC REGISTER FILE

Microprocessors often use RFs with multiple read ports that allow simultaneous operand reads from it. The number of operands that may be read at any time is equal to the number of the RF's read ports. The Alpha 21264 is an example of a microprocessor with multiple read ports [Gieseke 1997].

The microprocessor targeted in this work requires multi-port RF cells for its register file. The target processor has a DMR pipeline. Every copy of the RF has 32 entries and each is 40 bits wide. Out of these 40 bits, 32 are data bits and the other eight are parity bits—one for each nibble of data [Clark 2011]. Once again, static RF cells are used for the multi-port RF design. The RF employs error detection circuits for additional protection. Checker circuits, used for checking write data as well as WWL assertions, check for mismatch between the redundant WWLs/write data. Any disparity indicates an error and triggers the correction circuits that stall the pipeline and reinstate a prior architectural state by re-writing the previous write data to the register entry. The general-purpose register file thus contains the DMR RF arrays, as well as the associated control and error-checking logic. This chapter discusses the multi-port RF design and addresses the challenges associated with static read-out in such a design.

4.1. Multi-Port RF Cell Design

The multi-port RF cell is shown in Fig. 4.1. The cell has three static read ports (Rs, Rt, and Rt/Rd) that, like the single read port RF cell described in chapter 2, use tri-state inverter for read-out.

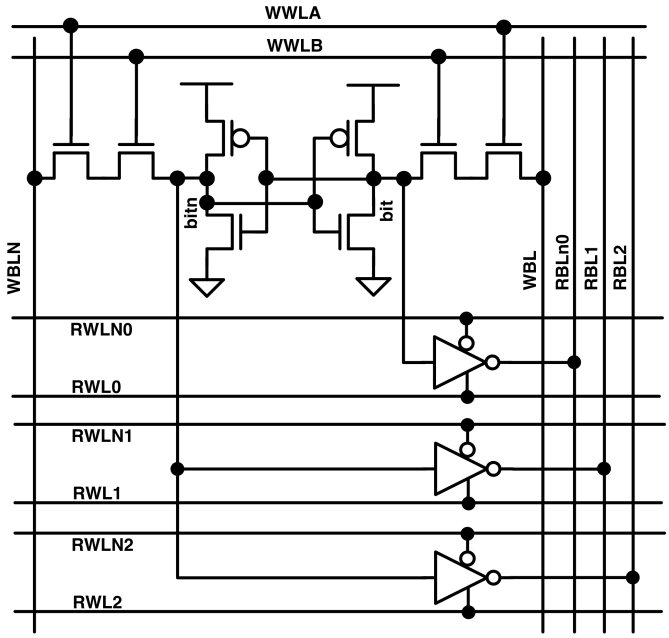
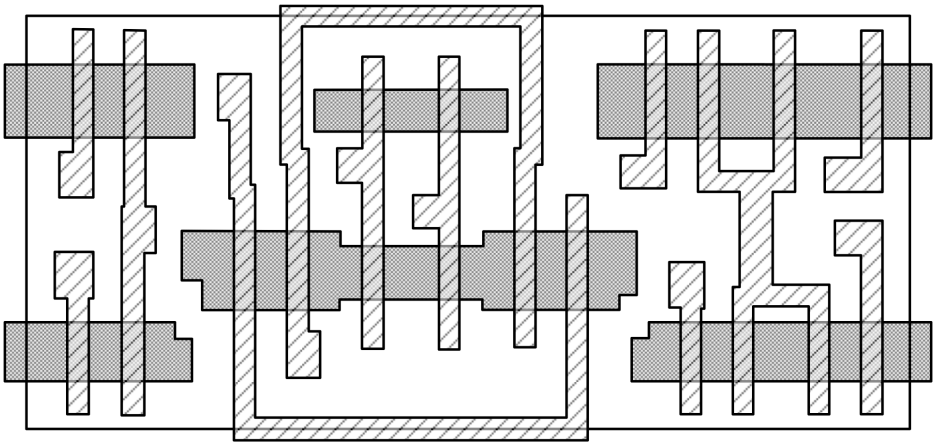


Fig. 4.1 Diagram for the static 1-W, 3-R RF cell.

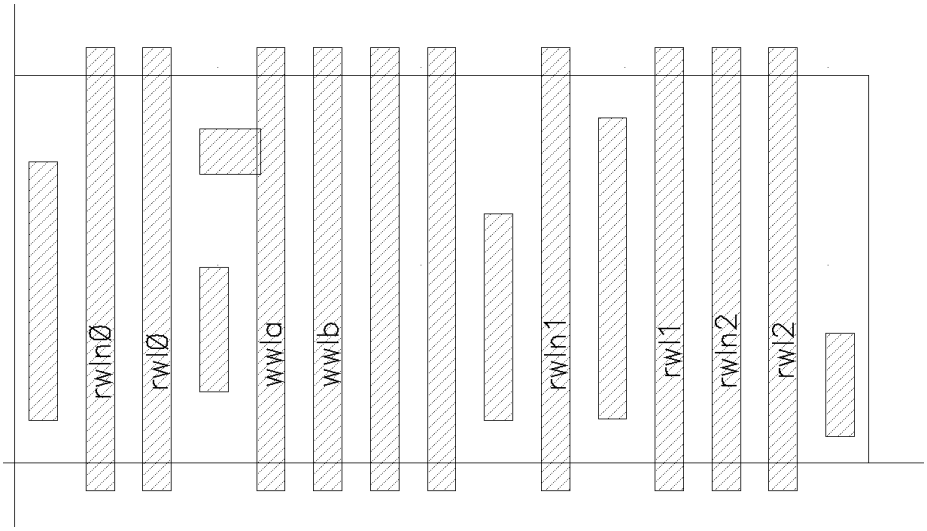
Two of these three read ports are connected to the same storage node, which results in extra capacitance at that storage node. Attaching the third port to the other side increases the storage cell overall Q_{CRIT}. The cell contains a total of six RWLs, two each for every read port, and two WWLs.

A salient feature of this RF cell is that data can be written to it only when both WWLs assert. This is done to ensure so that if one WWL erroneously asserts due to a radiation particle strike, the write port (Rt/Rd) is not activated as the other WWL

maintains the correct state of being low and the cell is not written to when it is not meant to be. The WWLs in the design are thus DMR.



(a)



(b)

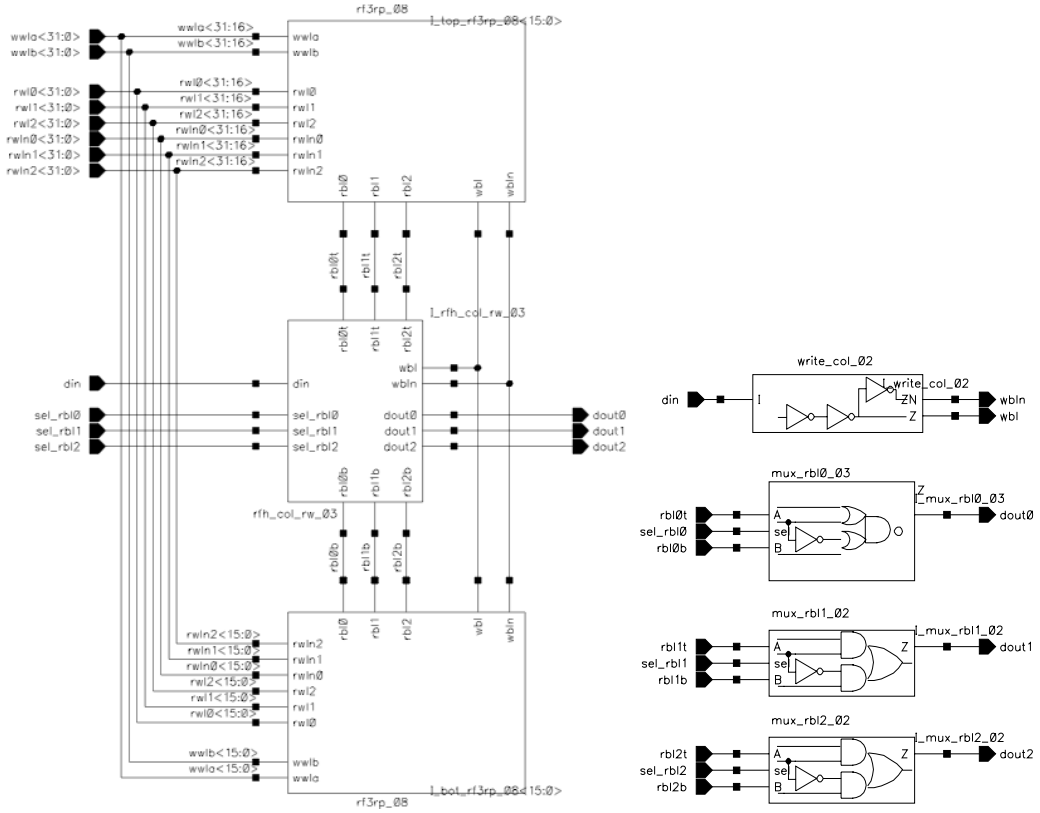
Fig. 4.3 Diagram showing the 20-T multi-port RF cell layout. (a) Only diffusion and poly-silicon layers are shown. (b) Metal 2 along with the place-and-route boundary are shown.

The multi-port RF cell (layout diagram shown in Fig. 4.3(a)) has an area of 8.232 μm^2, and is the same height as a standard cell. It does not use the thin-cell modern layout as it

has been designed to be compatible with the foundry 7-track standard cell library. However, due to the large number of NMOS transistors, and out of the necessity to maximize the write port access transistors' widths for sufficient write margin, it becomes necessary to make the N-well smaller than it is in the standard cells. This feature necessitates the use of edge cells within the column for seamless integration with other standard cells. Well tap cells, which must be used for latch-up prevention as per the design rules, are therefore designed in a manner so as to facilitate this transition from the non-standard cell N-well height to the standard cell N-well height, providing an interface between the RF cell and other standard cells in the RF column. It can be seen from Fig. 4.3(a), that diffusions for two of the three read ports are shared in order to lower the parasitic capacitance from disjointed diffusions. The high metal density in the cell entails that both WWLs be routed in poly-silicon. These protrude from the PR boundary since it is not possible to restrict them as per standard cell rule given other poly-silicon geometries within the cell. However, this does not result in any area increase as they are shared with the adjacent columns. Nevertheless, this does require the use of columns with special cells at the array boundaries just like the single port static RF used in the microprocessor cache. Again, metal 2 is used for WL, power, as well as passing into the cell, and metal 3 is used for BL routing.

4.2. Multi-Port RF Column Design

The multi-port RF column contains a total of six RBLs (see Fig. 4.4(a)), two each per read port. Bifurcation of the RBLs improves the read delay. The column thus contains two groups of RF cells, each comprised of 16 cells.



(a) (b)

Fig. 4.4 (a) Multi-port RF block-level column schematic. (b) Multi-port RF read and write circuit block-level schematic.

The WBLs are continuous since write delay is not a critical factor and the write data is gated outside the RF for power saving. The write circuit is identical to the single port static RF, except for the use of wider gates to drive the larger WBL load and consists of inverters that drive data onto the two WBLs for dual-ended write. The read circuit (block-level schematic shown in Fig. 4.4(b)) consists of three multiplexers, each one of

which selects between the top and the bottom RBL for a given read port. The select for each multiplexer is a buffered version of the read address MSB corresponding to a particular read port. The multiplexers are implemented using static AO22 and OAI22 gates. The multiplexer for the port connected to the ‘bit’ node (see Fig. 4.1) employs an OAI22 gate and those connected to node ‘bitn’ employ AO22 gates. Due to this disparity in the number of inversions in the read-out path, read delay for the port connected to the node ‘bit’ is smaller than that for the other two ports.

4.3. Multi-Port RF Floorplanning

The original RF used a dynamic version of the multi-port RF cell which, unlike the static RF cell that contains six RWLs and two WWLs, had three RWLs in addition to the two WWLs that belonged to a single RF copy.

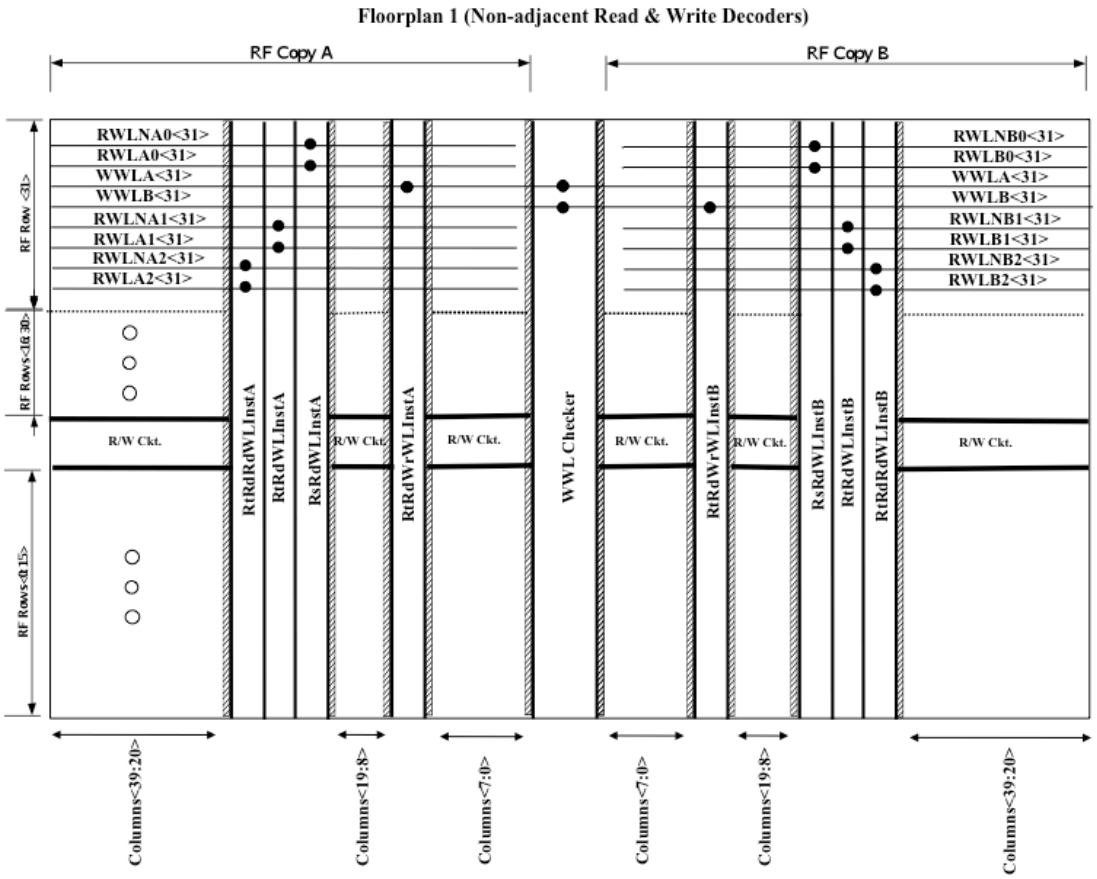


Fig. 4.6 Multi-port RF floorplan 1 with non-adjacent read and write decoders. The stippled bands represent columns containing interface cells. A black dot implies WL connection to a particular decoder.

This allowed the routing of three RWLs from the other RF copy within the given cell dimensions, making it possible to interleave the RF columns from the two copies. However, in the static RF cell, the three additional metal 2 tracks required to route the complementary RWL (RWLn) utilized by

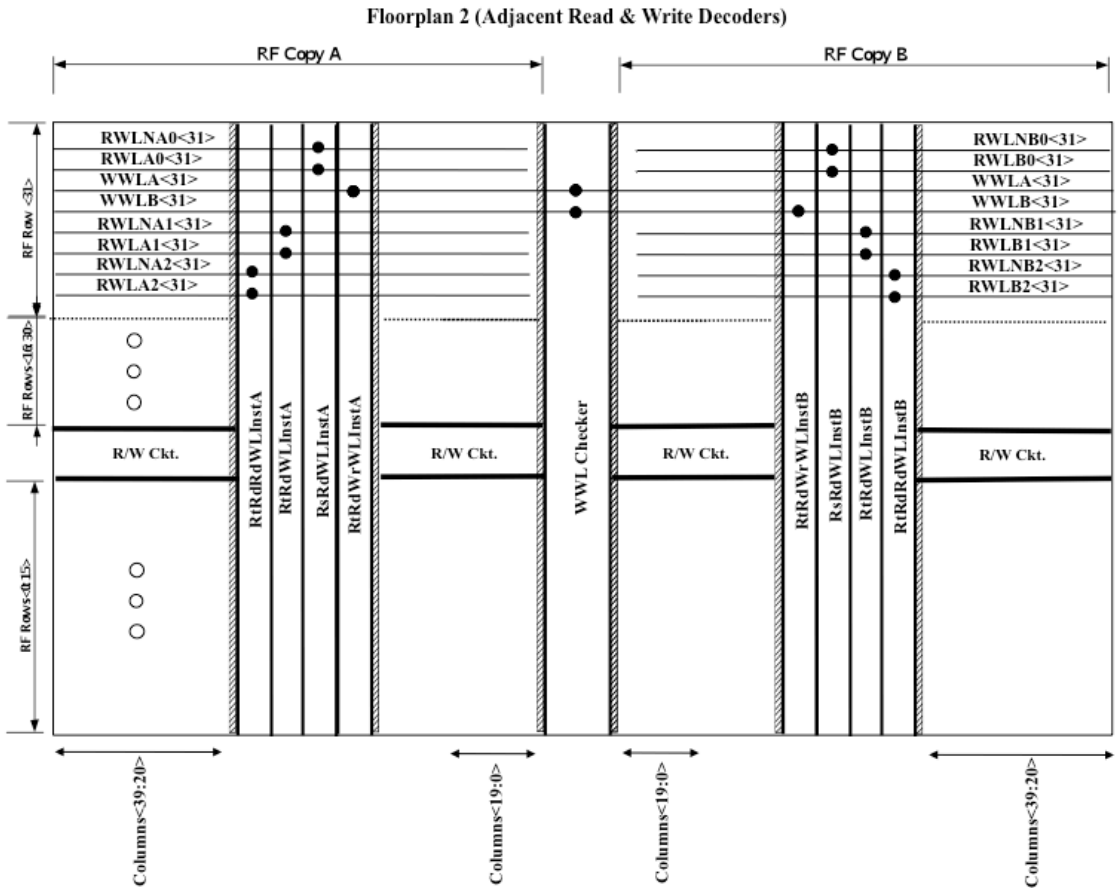


Fig. 4.7 Multi-port RF floorplan 2 with adjacent read and write decoders. The stippled bands represent columns containing interface cells. A black dot implies WL connection to a particular decoder.

the static read port's pull-up, along with those used for power and internal routing, occupy all the available metal 2 tracks in the cell besides the required RWL and WWL tracks. This precludes any further WL routing through the cell and hence any prospect of RWL interleaving. Two possible floorplans are shown in Fig. 4.6 and Fig. 4.7. The decoders, located at the middle of each RF copy can be identified by their names that correspond to the write (RtRd) and read port names (Rs, Rt, RtRd) followed by Rd or Wr to indicate a read or a write decoder, respectively. Both of these floorplans dispense with

the interleaved columns and instead use two distinct regions for the RF A and B copies that are separated by the WWL checkers. The RWLs do not interleave and terminate

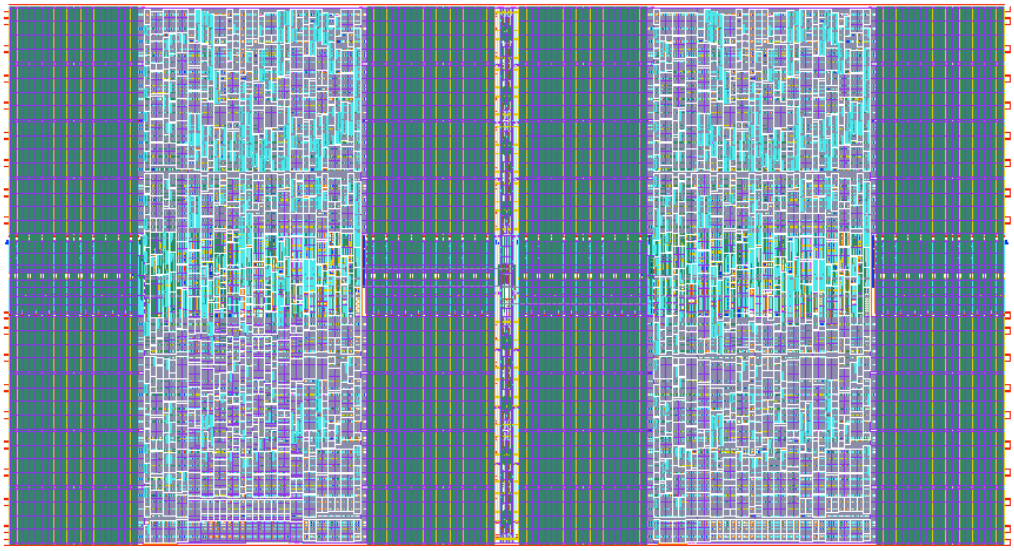


Fig. 4.8 Multi-port RF layout (metal 2, metal 3 directions are horizontal and vertical, respectively).

within a single RF copy while the WWLs do interleave. The floorplan 2 was chosen since it required four fewer columns containing interface cells. Fig. 4.8 shows the DMR register file layout based on this floorplan.

4.4. Multi-Port RF Decoder Design

The DMR multi-port RF contains six read address decoders, three each for every copy of the RF, and two write address decoders common to each RF. Considering the large number of decoders required for the RF, all decoders are designed using synthesis and APR to reduce the design time, but the area impact is evident in Fig. 4.8.

4.4.1. Multi-Port RF Decoder Synthesis

Cadence's RTL compiler was used for synthesis, with two RTL codes as inputs—one for the six read decoders and another for the two write decoders. Distinct RTL was used since each read decoder generates a RWL and a RWLn that are not clock gated. The write decoders generate a single polarity WWL and use clock gating to enable the decoded write addresses. All nets are preserved during the synthesis to keep the decoder structure depth constant. This ensures that all input to output paths have nearly the same propagation delay.

4.4.2. Decoder Pin and Blockage Assignment

A necessary criteria, while designing a memory decoder, is that the WLs must pitch map the WLs in the memory array.

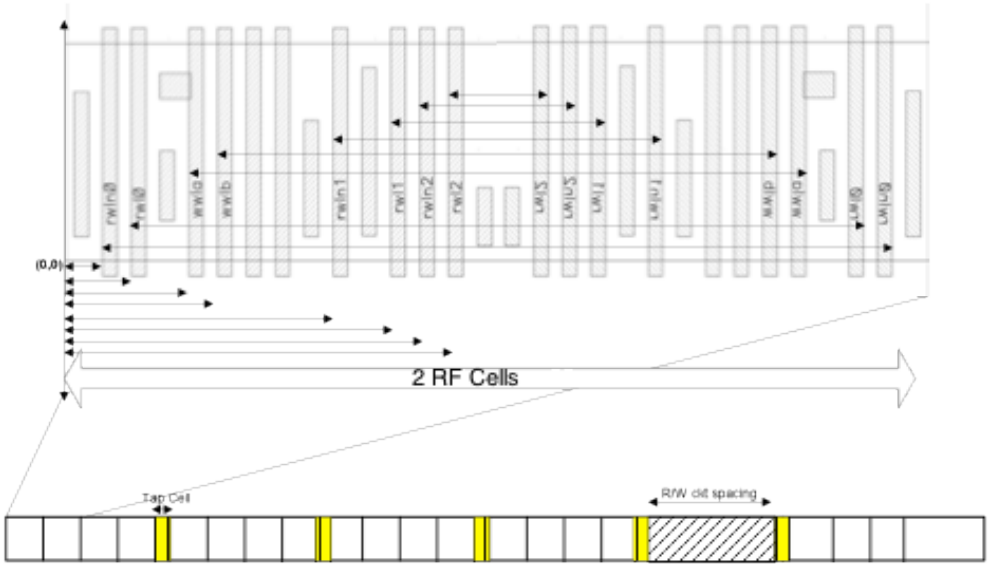


Fig. 4.9 WL offset determination for DEF file generation.

Metal tracks, corresponding to the WLs, were therefore specified as pin locations and blockages in a design exchange format (DEF) file for decoder. Pin locations for a decoder are the RWL/WWL locations corresponding to that particular decoders. The blockages for a decoder are the RWL/WWL locations belonging to all other read as well as write ports, and specifying these prevents shorts to the other WLs passing through the decoder. A perl script generated the DEF file. Inputs to the perl script are two offsets, namely the WL to origin and WL to the same WL when a copy of the RF cell is flipped along the Y-axis and abutted to it (see Fig. 4.9). Additional parameters, such as well taps and read-write circuit locations breaking the RF cell grouping, are also specified as inputs to the Perl script.

Pins and blockages are generated using the following method. For a decoder corresponding to read word-line (RWL) 0, only the tracks allocated for RWL0 are declared as pins in a DEF file. All the other RWL/WWL are declared as blockages. Additional pins/blockages are also specified for auxiliary signals that are routed across the decoders. The same process is repeated for the other two read and two write decoders. As a result, a total of five DEF files are generated. A DEF file for a read address decoder contains 64 pins (32 RWL and 32 RWLn) and 192 blockages, and a DEF file for a write address decoder contains 32 pins and 224 blockages. DEF files are not generated for eight decoders since three read decoders are identical for both RF A and B copy. The write address decoders are distinct since WWLs are interleaved.

4.4.3. Multi-Port RF Decoder Auto Place-And-Route

Cadence's Encounter SOC is used for APR. Once the DEF file is generated, blockage and pin lengths are changed to the floorplan height iteratively since early estimation does not give accurate floorplan dimension.

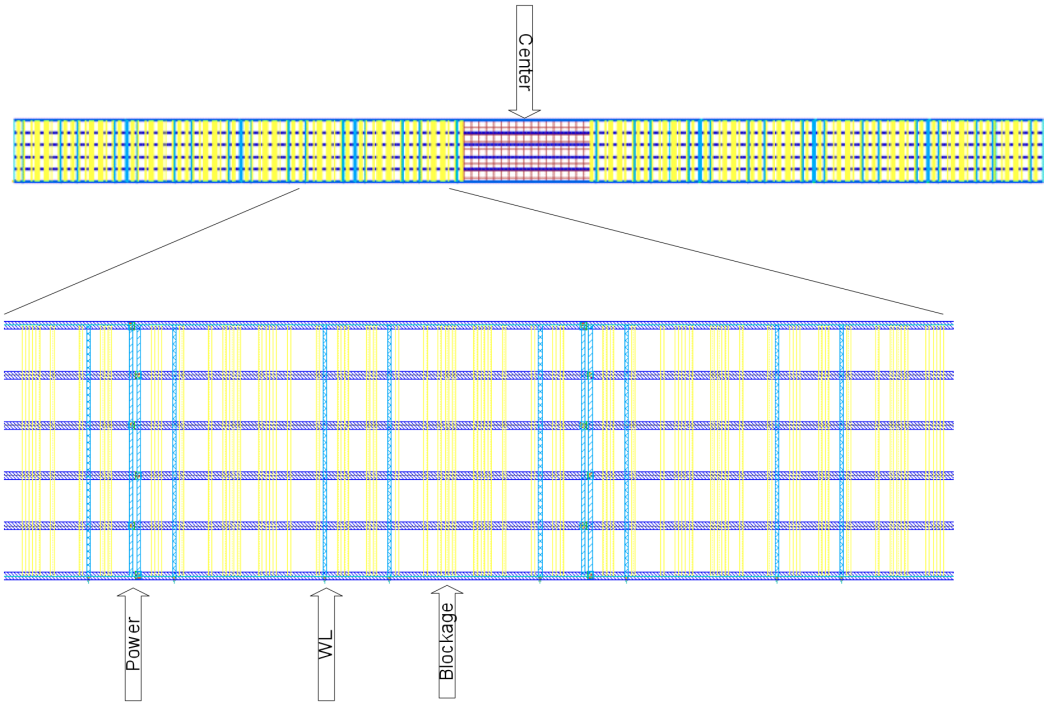
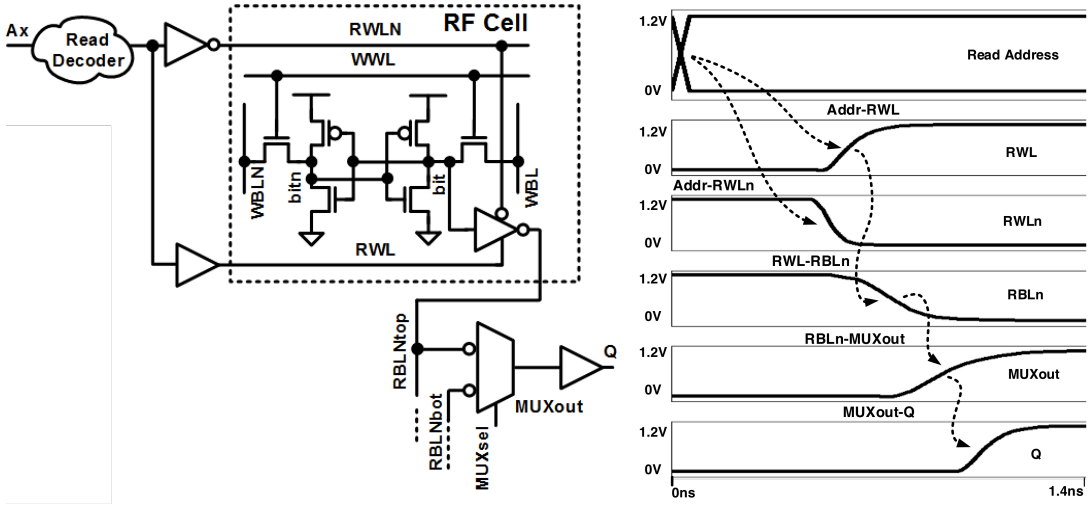


Fig. 4.10 Blockages, pins, and power routes in a decoder floorplan.

Width of course, is fixed to that of the RF. The high metal 2 density precluded WL routing in metal 2 and therefore, the WLs are routed in metal 4 (see Fig. 4.10).

4.5. Multi-Port RF Read Timing

Read operation for the multi-port RF is exactly identical to that of the single read port RF used in its cache's sub-banks. Fig. 4.11(a) and (b) show the read critical timing path diagram and waveforms, respectively. Additional read ports have not been shown for brevity.



(a) (b)

Fig. 4.11 (a) Multi-port RF critical read timing path diagram (additional read ports excluded for brevity). (b) Representative critical read timing path waveforms.

Referring to Fig. 4.3(a), tri-state inverters in the multi-port RF cell have poor pull-up capability due to N-well notching mentioned earlier. This results in poor low-to-high transitions on all the RBLs.

		Multi-port RF access time (ps) from UltraSim		
Read Port		Rs	Rt	RtRd
Value being read	0	1050	781	681
	1	803	1170	1030

(a)

		Multi-port RF access time (ps) from NanoTime		
Read Port		Rs	Rt	RtRd
Value being read	0	1135	847	760
	1	893	1265	1148

(b)

Table 4.1 Multi-port RF worst-case access times for the three read ports obtained using (a) UltraSim, and (b) NanoTime.

Table 4.1 demonstrates how the poor RBL rising slews impact the multi-port RF access time. Delay numbers in Table 4.1(a) are obtained from dynamic simulations with UltraSim in mixed-signal mode for higher accuracy and those in Table 4.1(b) are obtained from static timing analysis using NanoTime. The latter exceed the former by 8-12%. Dynamic simulations as well as static timing analysis are carried-out at 25^{o}C TT.

The Rs read port connects to node ‘bit’ and therefore, rising RBL transition is slower when a 0 is read (see Fig. 4.13(a)) from the cell. The read ports Rt and RtRd connect to node ‘bitn’ and the rising transitions on the corresponding RBLs are poor when reading a 1 (see Fig. 4.13(b) and (c)) from the cell. Thus, reading a 0 becomes the worst-case delay for the Rs read port, whereas reading a 1 becomes the worst-case delay for the Rt and RtRd read ports.

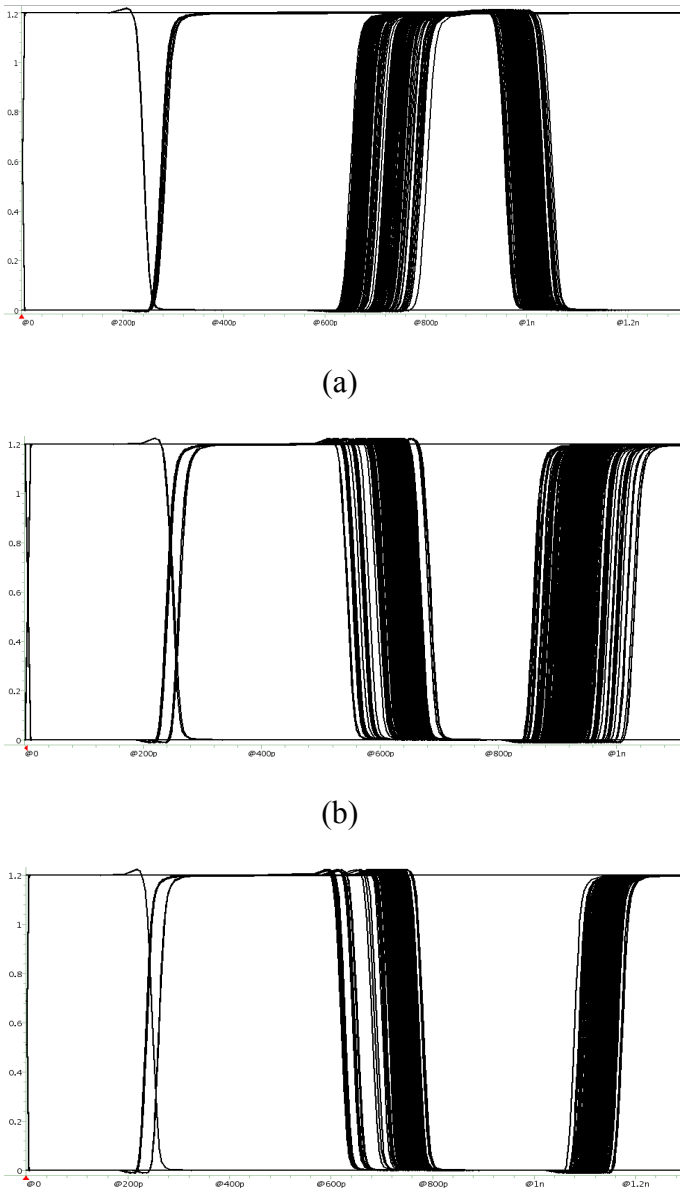


Fig. 4.13 Multi-port static RF read-out data timing variations for the read ports (a) Rs, (b) RtRd, and (c) Rt.

Simulation waveforms, with 32 read operations for each read port, are shown as eye-diagram in Fig. 4.13 and demonstrate the worst-case assertion and de-assertion read data timings for the read ports Rs, RtRd, and Rt.

4.6. Static Timing Analysis And Timing Characterization

4.6.1. Timing Characterization Overview

A timing model for a block, such as the RF described in this chapter, provides a means to supply the cell's timing characteristics to the synthesis and APR tools that use the cell in a CAD flow. Timing characterization is the process used for timing model extraction that culminates in the generation of a liberty (.lib) file, which is a representation of a cell's timing parameters.

Synopsys's NanoTime is used for the purpose of timing characterization and static timing analysis of the various RFs presented in this work. Timing characterization is only described for the multi-port RF for brevity. Although a typical .lib file also contains power parameters, NanoTime does not support this feature. However, this does not impede the design flow, since a .lib file with timing parameters alone is sufficient for synthesis and APR.

A .lib file contains timing arcs in the form of I/O path delays, the associated output transitions and timing check values, such as setup and hold checks. All of these are described in a look-up table format supported in a non-linear delay model (NLDM) that is used in this work.

```
pin("RtRdReadDataSMLB[39]") {
        direction : output ;
        max_capacitance : 0.491052 ;
        min_capacitance : 0.008252 ;
        capacitance : 0.003852 ;
        rise_capacitance_range (0.003852, 0.003852);
        fall_capacitance_range (0.003852, 0.003852);
        timing () {
                related_pin : "RtRdReadAddrSMLB[1]" ;
                timing_type : combinational ;
                timing_sense : positive_unate ;
                /* comment : block combinational (combinational), path 30991, path 31420; */
                cell_rise( f_itrans_ocap ){
                        index_1 ( "0.003840, 0.010400, 0.023440, 0.049520, 0.101680, 0.206000, 0.414720");
                        index_2 ( "0.008252, 0.015912, 0.031242, 0.061892, 0.123152, 0.245752, 0.491052");
                        values ( "0.931709, 0.942477, 0.960362, 0.996787, 1.070552, 1.218519, 1.514891",\
                                 "0.934200, 0.944969, 0.962854, 0.999278, 1.073043, 1.221010, 1.517382",\
                                 "0.938812, 0.949581, 0.967466, 1.003891, 1.077655, 1.225622, 1.521994",\
                                 "0.946006, 0.956775, 0.974660, 1.011084, 1.084849, 1.232816, 1.529188",\
                                 "0.955298, 0.966066, 0.983951, 1.020376, 1.094140, 1.242107, 1.538479",\
                                 "0.967845, 0.978613, 0.996498, 1.032923, 1.106688, 1.254655, 1.551026",\
                                 "0.985347, 0.996115, 1.014000, 1.050425, 1.124190, 1.272157, 1.568529");
```

(a)

```
pin("WriteDataCWHA[0]") {
        direction : input ;
        max_transition : 0.414720 ;
        capacitance : 0.004209 ;
        rise_capacitance_range (0.004209, 0.004209);
        fall_capacitance_range (0.004209, 0.004209);
        timing () {
                related_pin : "RtRdWriteClkB" ;
                timing_type : setup_falling ;
                /* comment : reference path 20485, checked path 40735, reference path 20486, checked path 40839; */
                rise_constraint( f_dtrans_ctrans ){
                        index_1 ( "0.003840, 0.010400, 0.023440, 0.049520, 0.101680, 0.206000, 0.414720");
                        index_2 ( "0.003840, 0.010400, 0.023440, 0.049520, 0.101680, 0.206000, 0.414720");
                        values ( "0.122003, 0.120251, 0.116350, 0.108532, 0.090784, 0.059985, 0.016290",\
                                 "0.124030, 0.122279, 0.118378, 0.110560, 0.092811, 0.062012, 0.018318",\
                                 "0.128395, 0.126643, 0.122742, 0.114924, 0.097175, 0.066377, 0.022682",\
                                 "0.137791, 0.136040, 0.132139, 0.124321, 0.106572, 0.075774, 0.032079",\
                                 "0.155137, 0.153386, 0.149485, 0.141667, 0.123918, 0.093119, 0.049425",\
                                 "0.180337, 0.178585, 0.174684, 0.166866, 0.149117, 0.118319, 0.074624",\
                                 "0.214181, 0.212430, 0.208529, 0.200711, 0.182962, 0.152164, 0.108469");
                }
                fall_constraint( f_dtrans_ctrans ){
                        index_1 ( "0.003840, 0.010400, 0.023440, 0.049520, 0.101680, 0.206000, 0.414720");
                        index_2 ( "0.003840, 0.010400, 0.023440, 0.049520, 0.101680, 0.206000, 0.414720");
                        values ( "0.074560, 0.072808, 0.068907, 0.061089, 0.043341, 0.012543, -0.031151",\
                                 "0.076518, 0.074767, 0.070866, 0.063048, 0.045299, 0.014501, -0.029192",\
                                 "0.080512, 0.078760, 0.074859, 0.067042, 0.049293, 0.018495, -0.025198",\
                                 "0.088350, 0.086598, 0.082697, 0.074879, 0.057130, 0.026333, -0.017361",\
                                 "0.107234, 0.105482, 0.101581, 0.093764, 0.076015, 0.045217, 0.001523",\
                                 "0.138378, 0.136627, 0.132726, 0.124908, 0.107159, 0.076361, 0.032668",\
                                 "0.182215, 0.180463, 0.176562, 0.168744, 0.150996, 0.120198, 0.076504");
                }

        } /* end of arc RtRdWriteClkB_WriteDataCWHA[0]_stupf*/
```

(b)

Fig. 4.15 Sample timing arcs from the multi-port RF .lib file. Example of (a) I/O delay and (b) setup and hold timing checks.

Each entry of a table for I/O delay and output transitions contains a value that is computed for a particular combination of transition time at the cell input, i.e. the input slew, and the capacitive load at the cell output. Fig. 4.15(a) shows a timing arc consisting of a table for address-in to data-out delay, where index_1 is comprised of inputs slews

and index_2 of output loads. In case of the timing checks, these values are a function of the data and clock input slews. Fig. 4.15(b) shows a timing arc composed of a table corresponding to write data setup with respect to the clock. Here, both index_1 and index_2 are comprised of data and clock input slews.

Some of the more noteworthy commands required to perform timing characterization and static timing analysis for the multi-port RF are discussed in the following sub-sections.

4.6.2. Completing Parasitic Annotation

Timing analysis and characterization are more accurate when parasitic elements along device interconnections are considered [NanoTime 2013]. Parasitic data extracted in the form of standard parasitic exchange format (SPEF) or detailed standard parasitic format (DSPF) can be back-annotated on the design with the 'read_parasitics' command. Mentor Graphics' xRC is used for extracting parasitics for all designs in this work, which generates PEX files that contains spice netlists that include parasitic data. When a PEX file is read by NanoTime, the parasitics are not completely annotated and the command 'complete_net_parasitics' must be used to complete partial parasitics annotated on all nets of the current design. The switch 'complete_with_zero' is used in conjunction with the command to complete the annotation by inserting small capacitances and resistances.

4.6.3. Defining Logic Constraints

Setting logic constraints, such as mutually exclusive conditions, restricts the scope of analysis to the specified logic conditions and improves the topology identification in order to help NanoTime recognize more topologies [NanoTime 2013]. Several logic constraints, with regards to the RWL and WWL, exist in the multi-port RF described in this chapter. For instance, only an RF single entry may be written at any given time and therefore, the WWLs need to be declared mutually exclusive. This is accomplished by using the 'set_logic_constraint' command as follows:

set_logic_constraint –at_most_one_hot [get_nets –hierarchical –top_net_of_hierarchical_group {n_wwla[*]}]*

The WWLs are 'at_most_one_hot', since at times write clock may not be asserted high and as a result, all WWLs may be turned off. Also, all RWLs—and all RWLNs—for

the same read port are mutually exclusive because only one entry corresponding to a particular read port can be read at a time. Contrary to the WWLs, the RWLs and RWLNs must be declared as 'one_hot', as at least one of them is always asserted high owing to the fact that read is not clocked. Yet another logic constraint dictates that for the same entry, the RWL must be mutually exclusive to the WWL, since an entry may not be both written to and read from at the same time. 'At_most_one_hot' is the correct switch to use when defining this logic constraint as WWL and RWL for the same entry may both be off at a given time.

4.6.4. Specifying Direction of Bi-Directional Transistors

The access transistors in an RF cell are regarded as bi-directional by NanoTime and cause errors when the topology check is performed. These bi-directional transistors are then defined by reading-in the pattern for a single RF cell and then setting the direction for a given access transistor towards a particular net. The command 'read_pattern', followed by a spice netlist name containing the RF cell sub-circuit, is used to read-in the RF cell pattern. The following command is then used to set the access transistor direction in all the cells matching the pattern 'rf3rp_08'—multi-port RF sub-circuit name—in the design towards the storage nodes.

foreach_match -pattern rf3rp_08 -command {set_transistor_direction -to_net qn mm2}

In total, four such commands, corresponding to all the four access transistors in the multi-port RF cell, are required.

4.6.5. Declaring False Paths

False paths must be declared by using the command 'set_false_path' so that they are not considered during path tracing and timing model extraction. For instance, false

paths are declared in the following manner from write addresses to read data ports since write and read operations are mutually exclusive.

set_false_path -from rtrdwriteaddrsala -to rsreaddatasila**

NanoTime allows for the use of wild cards and therefore, the asterisk is used in the above example to encompass all the possible bus bits. Not declaring all false paths results in timing arcs being broken up into fragments that are associated with indiscernible internal pins and can also result in correct timing arcs not being generated.

4.6.6. Setting Special Attributes

4.6.6.1. Enabling WWL Checker Topology Identification

The WWL checkers are incorrectly identified during topology recognition since they use a pulse generator. This issue is related to the attribute 'topo_clock_gate_strict_checking' that specifies whether strict checking for re-convergent clock gate recognition is enabled or not. Setting it to 'false' forces NanoTime to perform a more aggressive clock gate recognition and mark any such downstream clock gate to be a resolved pulse shaper or resolved pulse generator, which ultimately results in the correct identification of the WWL checker topology.

4.6.6.2. Enabling Same Phase Gated Clock Check

By default, when a reference clock is propagated to a control gate of a gated clock, and if the same reference clock triggers the control signal of the gated clock, NanoTime performs a gate-clock timing check in the next clock cycle (no-same-phase checking). To have NanoTime perform the timing check in the same clock cycle (same-phase checking), the variable 'timing_same_phase_gated_clock' is set to true. This is

important due to the presence of the AND gates that qualify the decoded write address against the write clock.

4.6.7. Timing Model Extraction

Before extracting the timing model, input slews and output loads for all inputs and outputs are specified using the command ‘set_model_input_transition_indexes’ and ‘set_model_load_indexes’, respectively. The ‘extract_model’ command generates a .lib file when used in conjunction with the switch ‘debug’, since NanoTime only generates the .lib file for debugging purpose, as the primary model extraction format is .db.

The usability of the timing model is context independent for the input arrival time and transition time, as well as output required time and network load. This implies that the model is accurate even when these conditions change.

The usability of the model is context dependent—implying the model is accurate only under these fixed conditions—for the clock frequency, duty cycle, power rail voltages, and operating conditions.

CHAPTER 5. STATIC AND DYNAMIC REGISTER FILE COMPARISON

Chapter 3 discussed the static RF used for the microprocessor cache. The static RF architecture was chosen for ease of design and portability across disparate designs. However, the timing was not as fast as desired, particularly in a microprocessor 1st level cache. Therefore, a dynamic version - which offers faster readout speeds - of the RF was designed, and its performance was compared against the static RF to provide a guideline for choosing the most suitable RF architecture for use in future designs. This chapter briefly describes the dynamic RF design and discusses the specific tradeoffs in performance, i.e., delay, and power, of size variations of the fabricated static RF and the aforementioned dynamic RF. Both static as well as the dynamic RF designs share the common target process, cell pitches, memory sizes, and when possible — decoding, to ensure a fair comparison.

5.1. Static and Dynamic RF Comparative Pipeline Timing Placment

Due to the differences in the way they operate, static and dynamic RF are located differently in a pipeline. Unless replica timing is used to delay the clock, a dynamic RF must be placed on a clock boundary since all read operations begin at the clock. The time-borrowing decoding (further discussed in 5.2.3 and 5.2.4) used in such a design creates a large setup time, albeit with a faster t_{CLK2Q}.

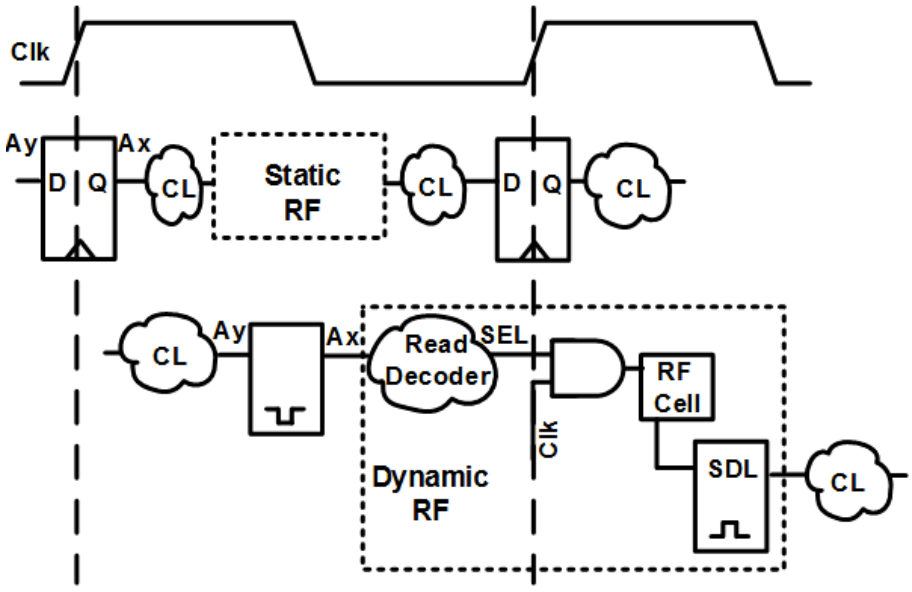


Fig. 5.1 Comparative pipeline timing placement for the static RF (top) and dynamic RF (bottom). The latter must straddle a clock edge. CL indicates combination logic in the timing path.

A static RF is often preferred in SOC designs due to the great flexibility in placement with respect to the surrounding logic timing. Since the read path is entirely combinational logic, a static RF can reside inside any pipeline stage as shown in Fig. 5.1(top). The timing flexibility of the static RF can be approximated with a dynamic

circuit by using timing circuits to provide t_{SETUP}. However, the required extra timing margin further subtracts from the dynamic circuit's delay advantage. Static RFs have been used in high performance microprocessor designs. Multiple generations of the PowerPC employed a static read-out using CMOS transmission gates to drive RBLs [Brondax 1994]. Very short RBLs, i.e., four cells on each, allowed high performance and faster decoding [Franch 1997].

5.2. Dynamic RF Design

Most RF designs have historically used dynamic read-out for high speed. Large high performance processor RFs frequently use multiple domino stages, i.e., local and global read bit lines (RBLs) to minimize delay with a large number of entries. Fetzer, et al., sacrificed decode power for low decode latency using NOR type decoding [Fetzer 2006]. However, like all dynamic circuits, the domino read operation must begin at a clock (Fig. 5.1 (bottom)). The Cell processor pipelined the RF read path into three stages to achieve high frequency operation [Riley 2007]. The following sub-sections describe the design and operation of a dynamic RF and further elaborate the reasons for its faster read speed.

5.2.1. Dynamic RF Cell

The 8-T generic dynamic RF cell shown in Fig. 5.2 was designed, for the purpose of comparison against the static RF by removing the PMOS transistors that comprised the tri-state inverter in the single port static RF cell.

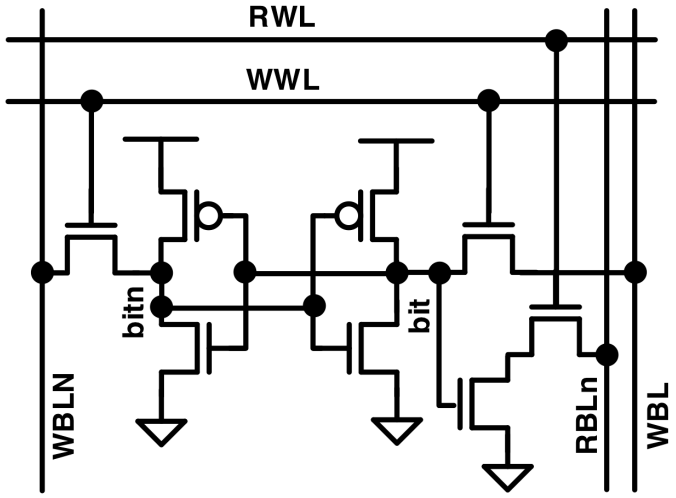


Fig. 5.2 Diagram showing 8-T dynamic RF.

This results in RWLN elimination but the cell remains identical in size and in both front as well as back end aspects to the static RF cell (see Fig. 5.4), which makes the dynamic RF cell design compatible with the 7-track library similar to the single port static RF. For speed, the read-out transistor sizes are maximized within the well dimensions. Once again, similar to the static RF cell, the write in this dynamic RF cell is dual-ended whilst the read is single-ended. The cell reads when the RWL is asserted high during the clock high phase, while in the clock low phase the RBLs are either restored to, or maintained at, the power rail voltage. The former is referred to as the evaluate phase and the latter as the pre-charge phase. The read operation is discussed in more detail in the next sub-section.

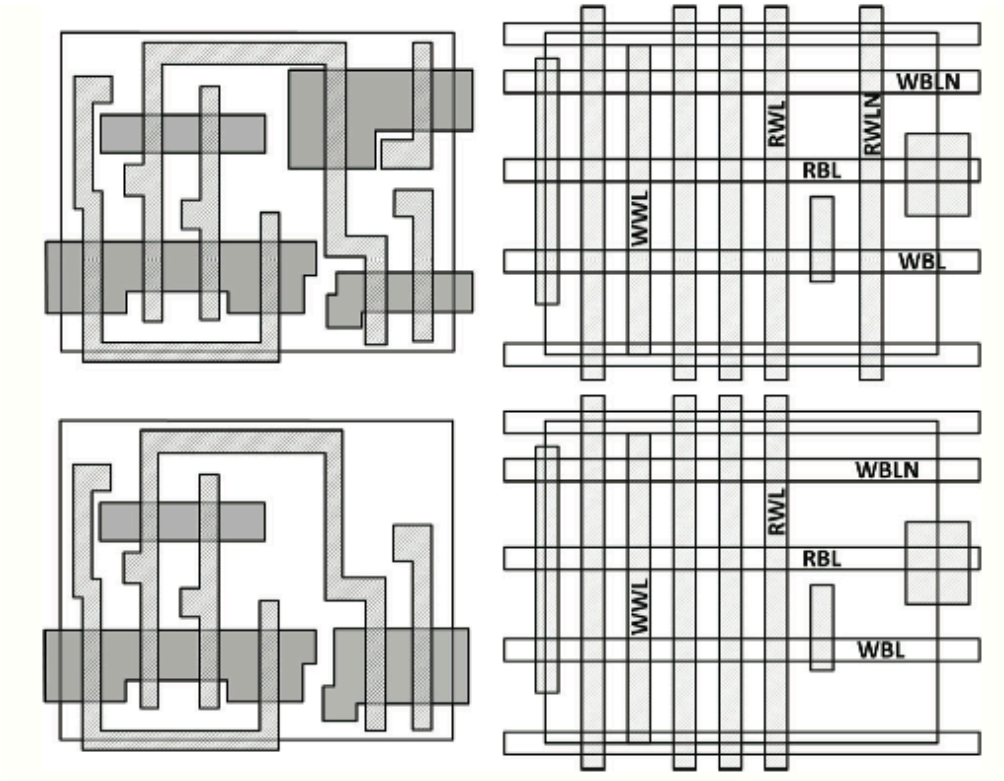


Fig. 5.4 RF cell layouts with the static at top and dynamic at the bottom. Poly and diffusion are similar (left). At right, metal routes on M2 (verical) and M3 (horizontal) are labeled. One WBL and the RBL diffusion are shared.

5.2.2. Read and Write Operations in a Dynamic RF

The write circuit for the dynamic RF is exactly identical to that of the static RF described earlier and is comprised of inverters and buffers that drive data of opposite polarity onto the WBLs for a dual-ended write. Just like the static RF, the write in a dynamic RF is accomplished by asserting the WWL for a single clock phase, which gates the two access transistors connected to the WBLs, high and thereby enabling a data write into the cross-coupled inverter pair used to store the data.

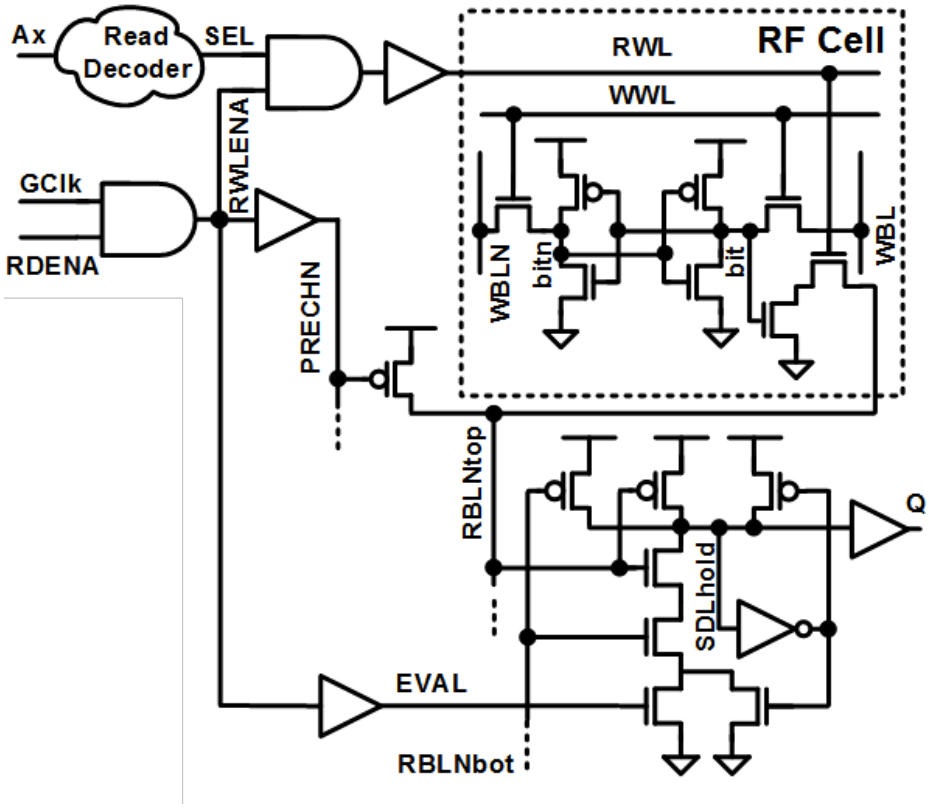


Fig. 5.6 Dynamic RF storage cell and read path diagram.

The read operation for a dynamic RF is different from that for the static RF and is comprised of two distinct phases that correspond to the low and high clock phase. During the low phase of the clock, referred to as the 'pre-charge' phase of operation, an active low signal 'PRECHN' (see Fig. 5.6) is generated to turn on the pre-charge circuit comprising of two PMOS transistors — one each for the top and bottom RBL within a column — that charge the RBLs, which are bifurcated in the same manner as their static RF counterparts, to the power rail voltage. If no read occurs, then the read enable signal (RDENA) is not asserted high during a clock high phase.

In the ‘evaluate’ phase, when a read occurs, RDENA is asserted high before the clock edge, and as a result the RWL enable signal (RWLENA) is asserted high during the clock high phase. The RWLENA then enables the decoded read addresses (SEL) that causes the RWL corresponding to a read address, to be asserted high. This has the effect of turning on all the read ports in an addressed row, which allows a RBL to discharge through the NMOS transistor stack in the read port. Depending on which row is being read, only the top or the bottom RBL in a given column discharges. A key design consideration is that the pull-up PMOS transistor in the keeper circuit—used to maintain an unselected RBL at V_{DD}—must be weak enough so as to be easily overpowered by the read port transistors. This is achieved by designing the keeper PMOS transistor to have a small width or a long channel (or both). The time that it takes to pre-charge the RBLs is not a part of the read timing.

Unlike the static RF, where the read operation is independent of the clock so that the output of the multiplexer—which propagates the RBL value to the output—does not change until a new read address is issued, the value read during the evaluate phase in a dynamic RF may be lost in the pre-charge phase if the value on the pre-charged RBL propagates to the output and if that value is disparate from the value read during the evaluate phase. Therefore, the following two conditions must be met during the pre-charge phase—propagation of the RBL value to the output must be turned off and the value read during the evaluate phase must be held. For this purpose, we use a non-inverting set-dominant latch (SDL) that consists of a footed NAND gate that is turned off during the pre-charge phase, followed by a latch to provide dynamic-to-static conversion. This latch is followed by a buffer to drive the output. The SDL is clocked by the EVAL

signal, which is a buffered version of the RWLENA and it turns off the RBL value propagation to the SDLhold node during the pre-charge phase. EVAL is analogous to the MUXsel signal in the static RF design and it too must be sufficiently delayed to prevent an unintended read resulting in additional power dissipation. A dynamic RF's column is thus composed of the following read circuits—an SDL and a buffer as well as the pre-charge and keeper circuits. The dynamic RF array also supports byte writes and similar to the static RF, consists of a column group with local write select gates for every eight columns.

One of the chief advantages in dynamic RF design is the ability to skew the PMOS/NMOS (P/N) ratios of the transistors that favor the monotonic discharging of the dynamic nodes in the RF read path, which helps to reduce the read circuit delay in the evaluate phase. Thus, the read-out transistor sizes are maximized for faster discharging of the RBLs where a falling transition is critical, while the footed NAND in the SDL is high-skewed to favor the charging of SDLhold node corresponding to a RBL falling transition. Inverters in the buffer, which is placed after the SDLhold node to drive the output, are similarly skewed low and high to favor a rise transition at the output. Skewing the gates and the use of the SDL, as well as custom pre-charge and keeper circuits, dictates that custom gates must be used in the dynamic RF read circuit design instead of the ones available in the standard cell library. This increases the design effort for a dynamic RF and each gate must be carefully sized to avoid contention and obtain the optimum benefit from the P/N ratio skewing.

5.2.3. Dynamic RF Decoder Design

Both read and write addresses are decoded using N-to-2^N decoders. The write decoder is exactly identical to that used in the static RF. However, the dynamic RF design requires a clocked RWL since the read operation is clock dependent. Thus, just like the write decoder, the select signals generated by the read decoder are also clock gate. The RWLENA, which is generated by ANDing the clock with the RDENA signal, is used for gating the selects. Using the RDENA signal ensures power savings by only enabling the RWLs during clock high phases that use the read. The RWL drivers are skewed to favor the RWL rise transition to minimize delay.

A transparent low latch is used at the read address decoder input to ensure the address hold time is met during the evaluate (clock high) phase. Failure to meet the hold time would result in SEL changing to a different entry than the one being read, thereby corrupting the read operation. The same transparent low latch requires that decoded read address (SEL) sets-up ahead of the rising RWLENA. This is important since it allows the decoding process, which is typically the largest component (~322 ps at 25^o C TT for this particular dynamic design) of the total access time (832 ps at 25^o C TT), to take place during the clock low phase. This time borrowing decode scheme is further related to other read timing parameters in the next sub-section.

5.2.4. Dynamic RF Read Timing

Waveforms in Fig. 5.7 show the signal propagation during a dynamic RF read. PRECHN has not been shown for brevity, but it can be seen that the RBLs are pre-charged prior to the beginning of a read operation.

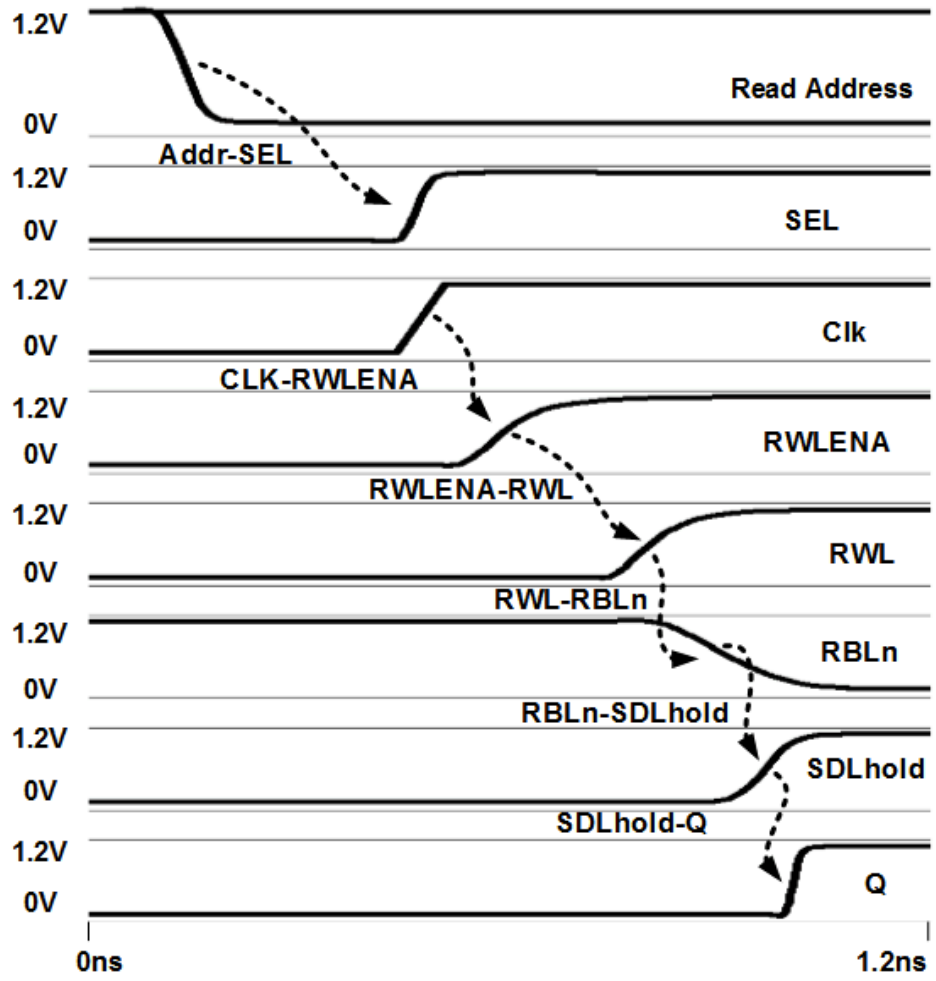


Fig. 5.7 Dynamic RF read timing critical path waveforms. Decoder outputs SELx set up to the RWLENA, which is a clock, leaving a t_{CLK2Q} timing.

The access time of the dynamic RF begins before the clock edge, with the address setup to the following rising clock edge that begins the read operation. This setup time (t_{SETUP}) not only includes the setup time of the read address latch, but also the amount of

time, ahead of which the read address must arrive to ensure that the cell's data is successfully written to the setup node of the SDL before it turns off at the end of the evaluate phase for the SDL to hold the data during the pre-charge phase. As mentioned earlier, decoder delay forms the most of this setup time and it is for this reason that a time-borrowing scheme is used to ensure that the decoding operation concludes in the clock low phase, thereby ensuring a fast data read-out during the clock high phase. This data read-out is equivalent to a t_{CLK2Q} timing and this, in addition to the t_{SETUP} time mentioned earlier, forms the total access time of the dynamic RF.

5.3. Area and Performance Analysis

We designed static and dynamic RF arrays with varying number of entries and word bit-width to study the effects of array dimensions on delay and energy per operation. Both read and write decoders are designed for the baseline 16 × 128 static RF as well as the 16 × 128 dynamic RF derived from it.

			RF Array Area (μm^2)								
	Number of read ports		1			2			3		
	Columns		32	64	128	32	64	128	32	64	128
Rows (WLs)	Dynamic	16	4081	7049	12985	7191	11301	19520	9964	14653	24031
		32	7365	12721	23434	12355	19414	33534	16234	23873	39151
		64	13933	24066	44332	22681	35641	61562	28772	42312	69392
	Static	16	3434	6051	11284	5536	8943	15757	8677	13155	22111
		32	6569	11574	21584	10331	16688	29403	16026	24298	40841
		64	12839	22620	42184	19919	32177	56693	30725	46583	78299

Table 5.1 Comparative total array areas for the dynamic and static RFs with various storage sizes and number of read ports.

The WL drivers and clock gaters are sized according to the load from the arrays of various dimensions. The results thus obtained are discussed in this section.

5.3.1. Area

Both static and dynamic RFs are designed to be compatible with the standard cell library, which forces metal 3 orthogonal and metal 2 parallel to the polysilicon gates as shown in Fig. 5.4. As stated earlier, both static and dynamic RF cells have identical dimensions and their area is 4.67 μm^2. For the dynamic design, more area efficient 8-T SRAM layouts are possible [Chang 2008], but are not compatible with the cell library pitch. Interestingly, our cell is still smaller than in [Hsiao 2014], where a full custom approach was used on the same process.

Area comparisons comprise Table 5.1. The dynamic to static conversion latch, pre-charger, and BL keepers in the domino RF make the dynamic I/O circuit 154% larger than that for the static RF. The clocked RWL drivers and buffers to drive the PRECHN and EVAL signals make the dynamic read decoder gate area 36% larger, although some of this merely replaces the static RF MUXsel driver.

Additional read ports each add two poly tracks to the static RF cell. Adding two at a time allows diffusion sharing and retains the sharing between adjacent cells. The resulting width increase because of adding one port for a static RF cell is three M2 pitches, increasing the cell area by 35%. Adding two ports can be accomplished in five metal pitches and allows diffusion sharing, making each cell 58% wider. Adding two read ports to the dynamic RF cell allows a different N-well structure costing only three tracks, which is the same as adding one read port. However each read port is expensive since extra read-out circuits, i.e., pre-charge, keeper, and output latch, are required. The dynamic RF has an area advantage only when three read ports are required, due to its smaller cell area.

5.3.2. Delay Comparison

Access times for the static and dynamic RF arrays of different dimensions are shown in Table 5.2. Delay numbers were obtained by running simulations with UltraSim in mixed-signal mode at 25C TT.

			Write Energy / Op (pj)			Read Energy / Op (pj)			Access time (ps)			EDP (pj * ps)			Relative EDP (%)		
	Columns		32	64	128	32	64	128	32	64	128	32	64	128	32	64	128
Rows (WLs)	Dynamic	16	0.86	1.78	3.55	1.85	3.43	6.48	685	689	696	1266	2365	4510	121.4	139.5	135.8
		32	1.37	2.69	5.33	2.11	3.86	7.99	721	728	760	1523	2813	6074	136.0	133.2	135.8
		64	2.40	4.82	9.72	2.57	4.80	9.17	807	813	818	2072	3902	7499	128.4	124.0	118.1
	Static	16	0.84	1.66	3.31	0.86	1.75	3.48	815	816	832	704	1430	2895	100	100	100
		32	1.32	2.66	5.21	1.08	2.06	4.25	903	911	918	975	1880	3900	100	100	100
		64	2.42	4.73	9.46	1.39	2.78	5.64	1066	1066	1089	1484	2968	6142	100	100	100

Table 5.2 Comparative energy per operation for write and read, access time, and EDP for the dynamic, and static RFs at various sizes.

The delay is largely unaffected when increasing the array width, since we sized the RWL driver to compensate for some of the RC contribution. The read address to SEL delay for the dynamic RF design is 1.9% larger than the static because of the extra loading due to the clock gates in the former's path. The overall delay (not including the dead-time between SEL and RWLNENA assertion) from read address to RWL assertion for the dynamic RF is 7.7% larger than the static, which is due to the wider read port NMOS transistors in the dynamic design that makes the RWL capacitance for it 24% larger than in the static design.

Since the read port transistor widths are layout constrained, it becomes difficult to compensate for the RBL capacitance that increases linearly with the array height. However, transistor widths for the single read port of the dynamic RF cell are less constrained compared to the static RF. Increasing read transistor pull down width provides RBL delay improvements ranging from 10% for columns with 16 entries to up to 90% for the ones with 64 entries, over the static design.

5.3.3. Energy Per Operation

Energy per operation comparisons are shown in Table 5.2. Write energy is essentially the same for the static and dynamic RFs, due to identical write circuits and similar overall capacitances within the cells. Energy per operation increases linearly with array width and near linearly with height. The static RF has obviously lower activity factor, 1⁄2 that of the dynamic RF. Unfortunately, the complementary WLs cancel some of this advantage, but the RWLN capacitance is 82% of the RWL capacitance. The dynamic RF RWL has greater capacitance than the static designs' RWL due to larger pull-down transistors, whose width was maximized for delay. On net, the dynamic RF has a 47% RWL energy advantage over the static RF.

REFERENCES

T. Brodnax, *et al*.,"Implementation of the PowerPC 601 micro- processor," *IBM J. Res. and Dev.*, vol. 38, no. 5, pp. 621–632, 1994. [Brodnax 1994]

J. Chen, L.T. Clark and T. Chen, "An ultra low power memory with a subthreshold power supply," *IEEE JSSC*, 41, 10, pp. 2344-2353, 2006. [Chen 2006]

G. Chen, D. Sylvester, D. Blaauw, T. Mudge, "Yield-Driven Near-Threshold SRAM Design," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol.18, no.11, pp.1590,1598, Nov. 2010. [Chen 2010]

L.T. Clark, D.W. Patterson, N.D. Hindman, K.E. Holbert, S. Maurya, S.M Guertin, "A Dual Mode Redundant Approach for Microprocessor Soft Error Hardness," *Nuclear Science, IEEE Transactions on* , vol.58, no.6, pp.3018,3025, Dec. 2011. [Clark 2011]

L.T. Clark, S. Leshner, G. Tien, "SRAM cell optimization for low AVT transistors," Low Power Electronics and Design (ISLPED), 2013 IEEE International Symposium on , vol., no., pp.57,63, 4-6 Sept. 2013. [Clark 2013]

L. Chang, *et al.*, "Stable SRAM cell design for the 32 nm node and beyond," VLSI Technology, 2005. Digest of Technical Papers. 2005 Symposium on , vol., no., pp.128,129, 14-16 June 2005. [Chang 2005]

L. Chang, *et al.*, "An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High-Performance Caches," Solid-State Circuits, IEEE Journal of, vol.43, no.4, pp.956,963, April 2008. [Chang 2008]

E. Fetzer, *et al*., "The parity protected, multithreaded register files on the 90-nm Itanium microprocessor," *IEEE JSSC*, 41, 1, pp. 246-255, 2006. [Fetzer 2006]

R. Franch, J. Ji, and C. Chen, "A 640-ps, 0.25-μm CMOS, 16× 64-b three-port register file," *IEEE JSSC,*. 32, 8, pp. 1288–1292, 1997. [Franch 1997]

B. Gieseke *et al.*, "A 600 MHz superscalar RISC microprocessor with out-of-order execution," in *Proc. IEEE Int. Solid-State Circuits Conf. Tech. Dig.*, 1997, pp. 176–177. [Gieseke 1997]

K. Govil, E Chan, and H. Wasserman, "Comparing Algorithms for Dynamic Speed-Setting of a Low-Power CPU," *1st Int'l Conf. on Mobile Computing and Networking,* Nov. 1995. [Govil 1995]

R. Heald and P. Wang, “Variability in Sub-100 nm SRAM Designs,” IEEE International Conference on Computer-Aided Design, pp. 347-352, 2004. [Heald 2004]

S.F. Hsiao; P.C. Wu, "Design of low-leakage multi-port SRAM for register file in graphics processing unit," Circuits and Systems (ISCAS), 2014 IEEE International Symposium on, vol., no., pp.2181,2184, 1-5 June 2014. [Hsiao 2014]

K. Itoh, Y. Nakagome, S. Kimura, and T. Watanabe, “Limitations and challenges of multigigabit DRAM chip design,” JSSC, vol. 32, no. 5, May 1997, pp. 624–634. [Itoh 1997]

ITRS Report 2013, http://www.itrs.net/ [ITRS 2013]

J. Xavier, “45-nm Radiation Hardened Cache Design,” Masters thesis, Arizona State University, 2012 [Xavier 2012]

R.W. Keyes, "Effect of randomness in the distribution of impurity ions on FET thresholds in integrated electronics," Solid-State Circuits, IEEE Journal of , vol.10, no.4, pp.245,247, Aug 1975 [Keyes 1975]

H. Na and T Endoh, “A compact half-select disturb free static random access memory cell with stacked vertical metal-oxides-semiconductor field-effect transistors,” Japanese Journal of Applied Physics, 2012. [Na 2012]

NanoTime User Guide, version H-2012.12-SP2. Synopsys Inc., March 2013 [NanoTime 2013]

D.A. Patterson, John. L. Hennessy, “Computer Architecture: A quantitative Approach,” 4th ed., Morgan Kaufmann Publishers, Inc., 2007. [Patterson 2007]

C. Ramamurthy, “Chip Level Implementation Techniques for Radiation Hardened Microprocessors,” Masters thesis, Arizona State University, 2013. [Ramamurthy 2013]

M. Riley, J. Warnock, and D. Wendel, “Cell broadband processor: Design and implementation,” IBM J. Res. Dev., pp. 545-557, 51, 5, 2007. [Riley 2007]

E.Seevinck *et al.*,“Static-noise margin analysis of MOS SRAM cells,” *IEEE J. Solid-State Circuits*, vol. 22, no. 5, pp. 748–754, Oct. 1987. [Seevinck 1987]

N. Weste, D. Harris, “CMOS VLSI Design: A Circuits and Systems Perspective,” Addison-Wesley Publishing Company, 2010. [Weste 2010]

X. Yao, “Radiation hardened high performance microprocessor cache design”, Phd dissertation, Arizona State University, 2009. [Yao 2009]

M. Yoshimoto, *et al.*, “A divided word-line structure in the static RAM and its application to a 64K full CMOS RAM,” JSSC, vol. SC-18, no. 5, Oct. 1983, pp. 479–485. [Yoshimoto 1983]