Floorplan Aware Framework for Optimal SRAM Selection for Memory Subsystems

Student Name: Gaurav Narang

Roll Number: MT13125

Aug 13, 2015

Indraprastha Institute of Information Technology

New Delhi

Under the Supervision of

Dr. Alexander Fell

Prakhar Raj Gupta, STMicroelectronics

Submitted in partial fulfillment of the requirements

for the Degree of Master of Technology in Electronics and Communication Engineering,

with specialization in VLSI and Embedded Systems

This research was done in collaboration with STMicroelectronics Pvt. Ltd., Greater Noida.

Student's Declaration

I hereby declare that the work presented in the report entitled **Floorplan Aware Framework for Optimal SRAM Selection for Memory Subsystems** submitted by me for the partial fulfillment of the requirements for the degree of *Master of Technology* in *VLSI & Embedded Systems* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr. Alexander Fell**. Due acknowledgments have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

Place & Date:

Student's name

Certificate

This is to certify that the thesis titled **Floorplan Aware Framework for Optimal SRAM Selection for Memory Subsystems** submitted by **Gaurav Narang** for the partial fulfillment of the requirements for the degree of *Master of Technology* in *VLSI & Embedded Systems* is a record of the bonafide work carried out by him under our guidance and supervision at Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Dr. Alexander Fell

Indraprastha Institute of Information Technology, New Delhi

Mr. Prakhar Raj Gupta

STMicroelectronics, Greater Noida

Abstract

Embedded memories are the key contributor to the chip area, dynamic power dissipation and also form a significant part of critical path for high performance advanced SoCs. Therefore, optimal selection of memory instances becomes imperative for SoC designers. While EDA tools have evolved over the past years to optimally select standard logic cells depending on the timing and the power constraints, optimal memory selection is largely a manual process. In this thesis, a framework has been proposed to optimize power, performance, and area (PPA) of a memory subsystem (MSS) by including floorplan dependent delays and power consumption in the interconnects and glue logic in the pre-RTL stage. Considering only memory PPA metrics is not sufficient for optimal memory selection and leads to a suboptimal implementation. It has been observed that physical design parameters such as routing congestion and interconnect delays, have significant impact on the implementation of the MSS and including them into the framework leads to more accurate and optimal results. However, to reduce the run-time of the framework, the top N (user input) memory configurations are pre-selected based on the PPA values of the SRAM instances and then the floorplan related metrics are estimated on the refined results.

The framework has the capability to use different estimates, when routing congestion is important (for example, in low cost processes with less number of metal layers). It has been shown that the interconnect delays are reduced by about 68% and dynamic power by 58%, if additional metal layers are available for routing compared to a low cost 6 metal process.

Keywords: memory subsystems, floorplan aware, congestion aware, multi-objective optimization, pareto-optimal, pre-RTL estimations

Acknowledgments

This work would not have been possible without the guidance and support of several individuals who boosted me for completion of this work.

I would like to first thank my advisors Dr. Alexander Fell, Prakhar Raj Gupta, Anuj Grover for providing excellent guidance, encouragement throughout the project work. Without his indispensable guidance, this work would never have been a successful one.

Thanks to IIIT-D and STMicroelectronics, Greater Noida, for providing excellent environment and infrastructure. I would also like to thank peers in the other teams such as CPU-GPU at STMicroelectronics for their valuable suggestions and fruitful discussions. Last but not the least, I would like to thank my family for supporting me spiritually and emotionally throughout my life.

Gaurav Narang

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Many advanced System on Chip (SoC) designs utilize large memory subsystems (MSS) for various requirements such as cache memories serving processors, buffers in display drivers, embedded scratch RAMs, FIFO based storage etc. Embedded memories are usually generated by using memory compilers [1], [2], [3]. This reduces the time to market for the designs. Designers usually have access to a wide range of memory compilers optimized for different requirements. Each compiler supports multiple options for requirements such as desired aspect ratio, low power modes, speed-area optimization, and many more. Memory compilers generally serve memory needs up to 1 Mb because optimization gains dwindle when larger capacities are involved at a compiler level. Therefore, bigger logical memories are realized by stitching several smaller physical instances together instead of a large custom circuit design, increasing design time.

Using multiple memory instances offers opportunities for architectural design changes to further optimize the MSS. A designer can choose a specific memory compiler based on the application requirements and stitch physical instances considering power, area, and performance. However, a diverse memory compiler portfolio and a range of optional features, makes this task of choosing memory instances complex since these options have a significant impact on power, area, and speed.

In the absence of any formulations, a designer has to depend on prior experience to navigate

through the wide range of decisions and design choices related to the selection of memory instances. Consequently, when new choices emerge, memory selection may be suboptimal. In the past, digital circuits faced the same problem as the gate count increased and algorithms to select the best cells from the library, were implemented into CAD tools. Today, MSS also need to be mapped and optimized in the same way synthesis tools optimize standard cell logic. The difference is that the synthesis tools extract pre-characterized performance parameters data from standard cell library whereas, in case of memories, the same will come from memory compilers.

In the standard digital design flow, modification of power, performance and area (PPA) metrics in the physical design stage leads to several iterations and increased time to tape-out. Recently several tools like Design Compiler and RTL Compiler have started to support a floorplan intent in early stages of the flow. This served as a motivation to estimate physical design metrics for choosing optimal memory in the pre-RTL stage. Moreover, long simulations in the existing frameworks (due to timing checks and PnR) have to be avoided to reduce overall time-to-market. Hence, a framework which does not involve CAD tools in the flow is necessary. This requires extracting pre-characterized memory PPA from memory compilers and pre-RTL estimations of PPA parameters of glue logic.

The questions which a memory designer usually faces for an MSS, are:

- Which memory compiler to use (high density or high performance)?
- Which type of periphery to use (low leakage or high performance)?
- How many physical memories to use to achieve target words×bits capacity?
- Which multiplexer factor to use, number of banks?
- How to split the datapath of an MSS?
- How should these memories be arranged in the floorplan?
- How the choices/selection in the pre-RTL stage will affect congestion in the physical design stage?

Answers to all these questions vary with the process, voltage, and temperature (PVT) corners at which the MSS is designed. A framework to give the solution to all these questions is required.

1.2 Related Work

With plenty of parameters available, various architectural exploration techniques exist for the MSS to optimize their PPA metrics. In [4], the authors have investigated memory transformations, namely segmentation and widening, for energy reductions. Models for memory and glue logic are developed for area, energy and performance metrics. These are studied with respect to the proposed transformation methodology. However, to be accurate, the flow proposed in [4] uses standard tools for placement and routing (PnR). This increases the time required to complete multiple explorations in a functionally complex SoC. The problem of fast exploration of the architectural design space of embedded systems is addressed in [5]. Here, all the existing configurations are explored by varying the architectural level parameters without using CAD tools. The framework explores the design space using the exploration algorithms based on Monte Carlo and simulated annealing methods.

Speed, power and area are key optimization axes of a multi-dimension space in which designs can be optimized. Work in [5] is aimed at producing pareto-optimal solutions in terms of PPA metrics, which spans to a multi-dimensional space problem. A pareto-optimal solution is a solution in the design space which cannot be optimized further along a particular axis without the degradation of a design metric along another axis. However, evaluating all the points in the design space is complex and requires extensive simulations. The approach of pareto-optimal solutions is adopted in finding an optimal floorplan in [6]. The design RTL netlist is the input to their framework and a similar trade-off between energy, area, and delay is presented by varying priority along these optimization axes. Considering a wide range of pareto-optimal solutions, significant savings in the dynamic power dissipation have been achieved at the cost of a minimal increase in area. However, evaluating all the points in the design space is complex and requires extensive simulations.

An important aspect of design ignored by these frameworks is the interconnect, which introduces significant delay and dissipates dynamic power. Power dissipated in the interconnects and buffers contribute around 30% to 90% of the overall power dissipation in MSS [7], [8]. Moreover, channels are left between memories for routability of these metal wires. Hence, there is an additional area to be taken into account. Thus, considering interconnects for power and area is necessary

as studied in [9]. Authors have presented dynamic partitioning of memory space considering execution profile to achieve low power consumption in the MSS. For the low power objective, it is easily assumed that the higher the number of banks, the lesser the partition size, the lower is the power per access of memory due to lower capacitance. But, the algorithm in [9] suggests there is a bounding value to the number of partitions because with every partition, there exists an overhead due to extra glue logic and interconnects. Similar overhead in area and delay occurs due to interconnects and routing channels among the SRAM instances.

1.3 Background

MSS Organization:

Several physical SRAM instances together form larger memory assemblies, arranged as memory tiles as shown in figure 1.1. Only one of the instance is active at a time - the one which is being read/written. All the other instances are deactivated. This is implemented using a $n : 2^n$ decoder for up to 2^n physical memories. The decoder generates a chip select signal using either n MSB (most significant bits) or LSB (least significant bits) address lines. This saves active/dynamic power consumption in the MSS since only a single small physical SRAM is active. The output of all memory instances are multiplexed to give a single output. This organization is known as the banked memory architecture. To show an example of an MSS, consider a memory capacity of 8192 words $\times$ 64 bits using eight physical memories of 2048 words $\times$ 32 bits. Since the 64 bit data path is split into 2, each chip select signal (CSNi) is being shared by 2 physical memories. A 2:4 decoder is required to generate CSNi signals. In this example, two LSB address bits are used for decoding and the other MSB bits are the input to the memory macro (SRAM instance). Anything other than the memory macro is termed as the glue logic.

The chip select pin is an important input to the physical SRAM. It is an active low signal i.e. if it is low for a memory instance, the memory is inactive and the read write signals do not affect the memory. These operations are called inactive read/write memory operations. It has been observed that these operations dissipate about 10% of the active memory dynamic power. Furthermore, the chip select signal can be used for interconnect power gating to avoid switched capacitance on the data path and address path nets for the inactive memories.

Physical Architecture of SRAM:

Figure 1.2 shows the architecture of an SRAM macro. It is divided into four parts - control, I/O, decoders, and memory array. Pre-decoders and clock generation signals constitute the control block. I/O consists of sense amplifiers (SA) and multiplexers. SA control the read/write operations i.e. during write operation, the bitline (BL) or bitline_bar (BLB) is discharged, to write '0' or '1' to the selected SRAM cell through the I/O block and during read operation, SA senses the differential voltage between BL and BLB.

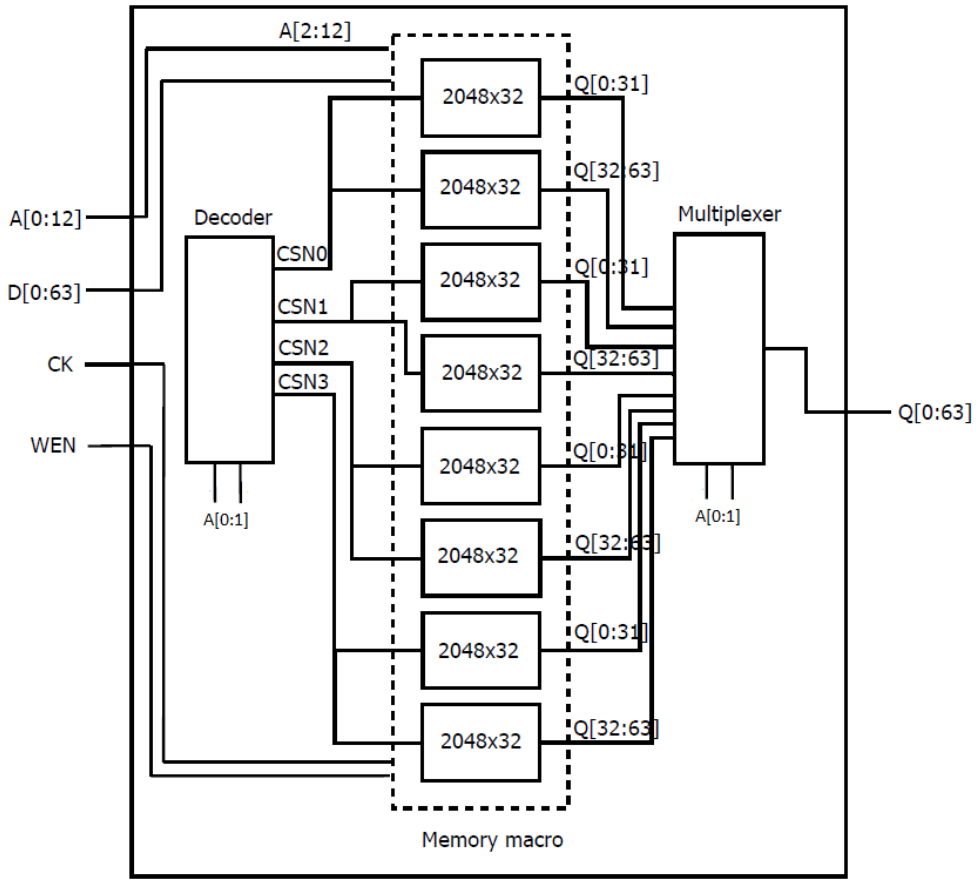


Figure 1.1: Logical view of the 8192 words×64 bits memory subsystem

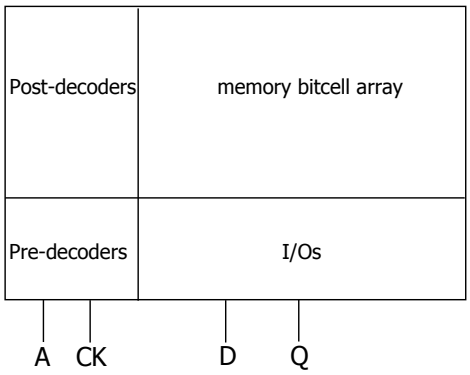


Figure 1.2: Architecture of a single port SRAM

Optimization in these internal blocks of the memory results in different memory compilers. For high density, the memory cells are made denser (lesser current drive), which makes this type of memory slower and no power saving techniques are applied to save the area. For low power design, retention circuits and the low leakage bitcells are used. Similarly, for high speed, big drivers (I/O), bigger memory cells and SA are used for the design.

Physical orientation of pins of a physical memory plays an important role for estimating length of the interconnects in the MSS. Physical view of a single port SRAM macro in figure 1.3 shows two memories with their pins, basic routing of data/address buses and clock network. A bulge is present at the center due to the I/O control. Routing is shown in figure 1.4 for an MSS for one bit data signal. All the pins are on one side of the macros and the SRAM macros are placed such that they face each other. It reduces the routing wire length through sharing. It also makes macro abutting (reducing the area between two macros on the side where there are no I/O pins) of instances possible, thus saving chip area. These assumptions have been considered for the placement and estimating interconnect length.

Floorplan and Congestion:

The aim of the floorplanner is to place the macros and the standard cells in a non-overlapping manner, achieving optimization objectives such as area, routing length and congestion. Floorplanning has a huge impact on the area, timing performance and congestion. Minimizing area has the highest priority for the floorplanner followed by routability and timing, congestion, and

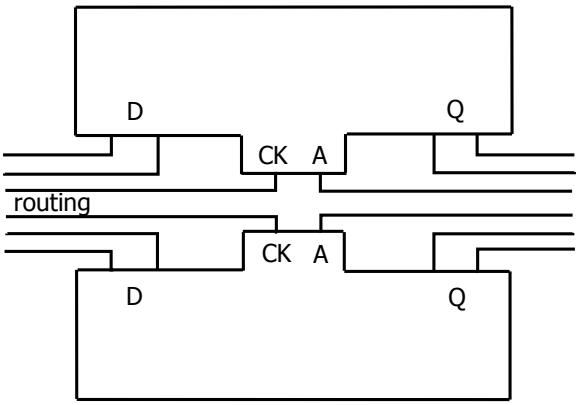


Figure 1.3: Physical view of a physical SRAM and signal routing

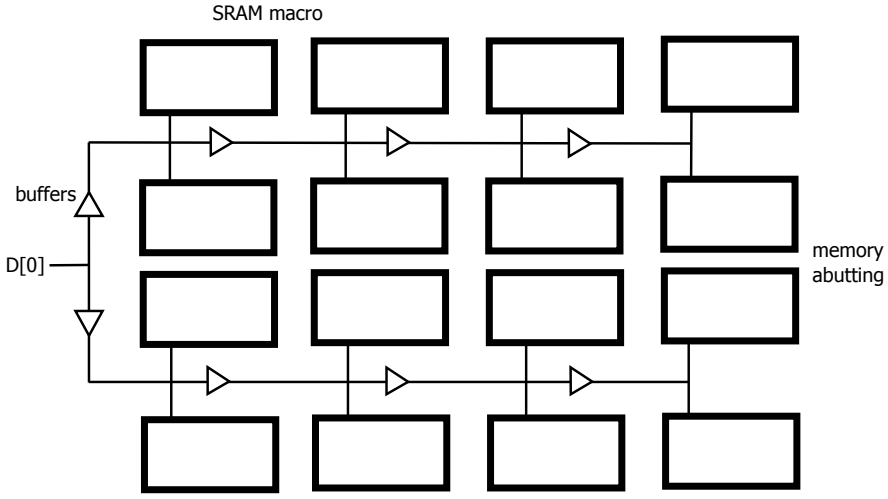


Figure 1.4: Sharing of data signal and a typical floorplan of an MSS

DRC (Design Rule Constraints) rules for the yield. Typically, a floorplan and Place-and-Route (PnR) algorithm uses equation 1.1 with higher weights to area and wire length.

$$Cost = \alpha \times Area + \beta \times Wire\ length + \eta \times Congestion \tag{1.1}$$

In an MSS, where the majority of the area is covered by the memory macros, it is important to formulate congestion models in terms of physical parameters of the SRAM macro. In this work, congestion is estimated for all available floorplan options. All the nets are routed in M3-M4 metal layers i.e. over-the-memory signals routing is not allowed. Channel width plays an important role for congestion since it defines the availability of routing resources. If a constant value of routing channel width is considered, then the congestion increases with the increase in routing wires. The number of nets in the routing channels must be considered for estimating congestion.

Chapter 2

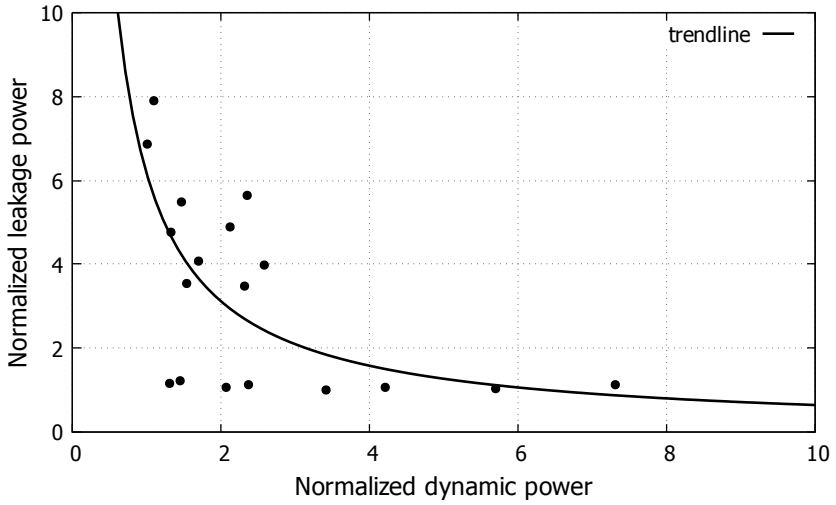
Multiobjective Design Space

To be able to do exhaustive explorations in the available design space, it is necessary to formulate architectural models for PPA that utilize physical SRAM parameters, leading to a faster exploration [10]. These PPA metrics form the optimization objectives for the memory subsystem (MSS). PPA values for different memory configurations are mapped on the axes, forming a minimization problem and a trade-off among these objectives exists due to variations in the physical parameters. In this section these physical parameters along with different compiler types, and optimization objectives for an MSS are discussed.

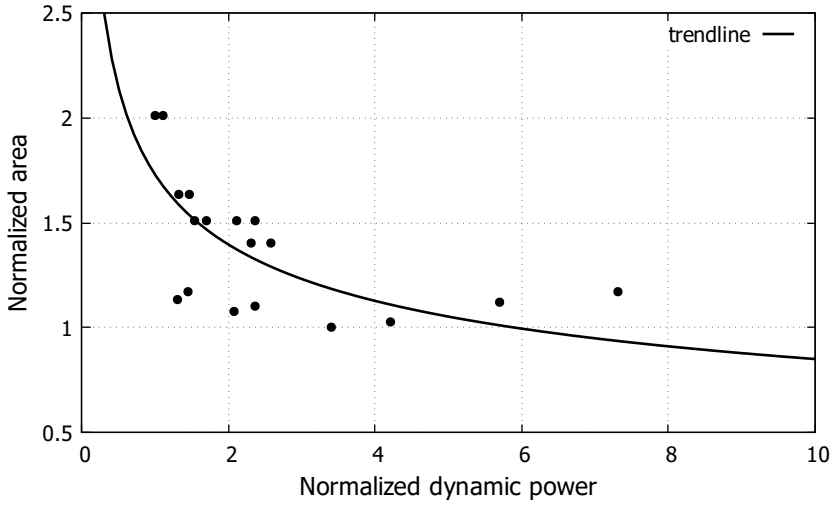
Compiler type and physical parameters together form a memory configuration. PPA values of these configurations for a 1 Mb MSS, are shown in figure 2.1. A trendline (curve fitting) drawn in both the plots, represents a convex optimization curve. This trendline is convex for all memory capacities, which suggests that the memory configurations serving the user requirements pose similar trade-offs irrespective of different word-lengths and bit-widths.

2.1 Memory Compiler Optimizations

Memory compilers are optimized for area, power and performance to serve different applications. A physical memory has two regions: bitcells and periphery. Bitcells are either optimized for area (high density bitcell) or for high performance register arrays. Further, periphery of the memory is also optimized for low leakage or high throughput using low Vt (threshold voltage)



(a) Leakage power vs dynamic power



(b) Area vs Dynamic power

Figure 2.1: Mapping of possible SRAM instances for 1 Mb MSS on various axes

and high Vt logic cells respectively. Combination of two choices each of bitcell and periphery, gives four memory compiler options characterizing different PPA values for exploration.

2.2 Physical SRAM Design Factors

In addition to compiler optimizations (e.g. high-speed, high-density), PPA metrics and instance selection are also affected by the internal organization of the physical SRAM. Four such parameters for PPA optimization have been identified.

- **Column Multiplexer (mux) Factor**: It is the number of multiplexers at the output of a physical SRAM. This factor affects the dynamic power and aspect ratio of the memories

as shown in figure 2.2. It shows an abstract view of a physical memory with three aspect ratios arising due to different mux factors. Memory width and dynamic power consumption increase proportionally with the mux factor. It is mainly used to place the memory optimally in the target floorplan. The mux factor also presents timing vs dynamic power trade-offs. Memory instances with a large mux factor give better timing performance while a lower mux factor leads to lower dynamic power dissipation.

- **Bank Factor**: A physical memory is internally split into a number of clusters, represented by the bank factor. It poses an interesting trade-off in timing, power, area, and leakage. Higher bank factor improves memory access delays (similar to use a SRAM instance with a higher mux factor). It saves dynamic power of a physical memory internally as only one bank is accessed at a time and other banks are deactivated. However, higher bank factor causes increased area and leakage power consumption.

- **Word Split Factor**: The factor by which a logical memory is split in words e.g. a logical memory requirement of 4096 words can be implemented with two physical SRAM instances of 2048 words each. It is different than the bank factor as the bank factor is internal to the physical SRAM instance.

- **Bit Split Factor**: It represents the factor by which the data path of the MSS is split e.g. a 64 bit logical memory is split into two physical SRAM memories of 32 bits each, which means each physical instance has its own pre-decoder and control circuitry. However, datapath splitting is not preferred unless the MSS bit width is more than the memory compilers can support e.g. for an MSS with a 256 bits datapath, at least two physical memories are required, having 128 bits each, since memory compilers support upto 144 bits only.

2.3 Optimization Axes

Optimizing an MSS is often a multi-objective optimization problem [11]. For example, objectives such as minimum area and minimum dynamic power consumption are often required at the same

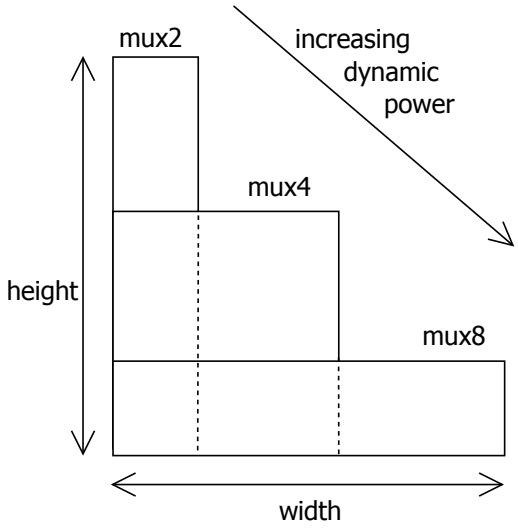


Figure 2.2: Three memories with different aspect ratios due to varying mux factor

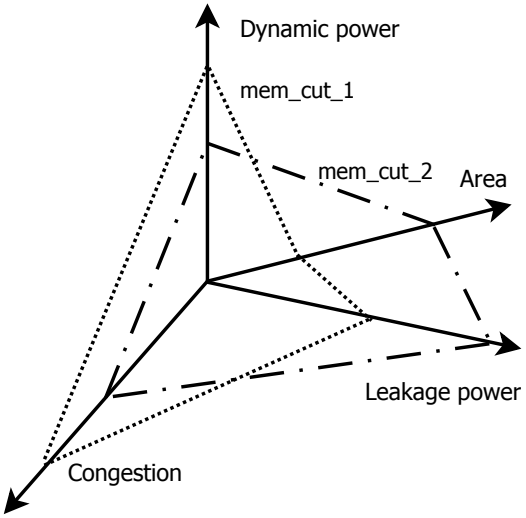


Figure 2.3: Spider-web representation of two memory configurations along optimization axes

time. The frequency of operation is a minimum imperative to be met by an MSS and optimize PPA only after the timing requirement is satisfied. For further optimization, four axes have been identified as shown in figure 2.3. Two MSS configurations are mapped in the form of spider-web diagram with the various radial lines representing the different optimization axes. Configuration 1 (mem_cut_1) is optimized for high density and low leakage but dissipates more dynamic power. Configuration 2 (mem_cut_2) has lesser dynamic power consumption but has a lower congestion (refer to section 2.3.4).

2.3.1 Area

Since the cost of a SoC is proportional to the silicon area, reducing area always remains the essential target for a designer. MSS area increases due to word split and bit split factors since some memory circuits like the control block, pre-decoders, I/Os etc. are repeated for every physical SRAM instance.

2.3.2 Dynamic Power

Memory internal power consumption and power consumed in the glue logic are considered as another optimization axis. Overall power dissipation increases as physical memories with higher mux and bit split factors are instantiated. However, it is inversely proportional to the word split factor.

2.3.3 Leakage Power

Low stand-by power consumption is a key requirement in many applications [12], where the device is in standby state for most of the time. For such applications, memories with lower bit split and word split factors achieve the objective of low leakage.

There are several types of leakage power consumption in memories:

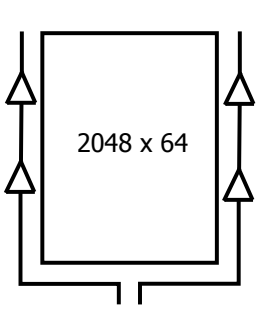
- **Active mode leakage**: Leakage power dissipated when the memory is accessed for a read/write operation.

- **Inactive mode leakage**: Static power consumed when the row decoder is inactive but the SRAM bitcell is active.
- **Retention mode leakage**: When the row decoder is inactive and the bitcell is at the retention level voltage. The retention level voltage is a lower voltage than the normal operating voltage, at which a SRAM cell can retain the bit value stored in it.

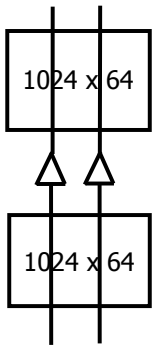
2.3.4 Congestion

The first three design metrics mentioned above are the primary optimization objectives and the MSS is optimized along these axes according to an application. However, an MSS design with optimized PPA must be easily routable in the physical design stage and therefore should lead to the non-congested design [13]. Lower congestion improves speed and reduces power dissipation in interconnects due to decreased wire length [14]. Hence, routing congestion is considered as the fourth optimization objective. Physical memories having higher mux factor, can be adopted to overcome congestion. Bit split also reduces congestion due to lower pin density around an SRAM macro. The desired MSS floorplan has an influence on the congestion too e.g. splitting the memory and stacking multiple SRAM instances vertically, degrades routability.

Apart from the local congestion issue stated above, global congestion also poses a design problem. For larger physical memories, it is not possible to route signals over them, even if over-the-memory routing of the signals is allowed. Since the buffers have to be placed after a fixed length, the interconnects have to be routed around the SRAM macro as shown in figure 2.4. This further increases the length, resulting in more buffers around the SRAM instance. This scenario increases routing congestion and is a typical case for the hard macros (such as shown in 2.4a) and a high fan-in multiplexer (e.g. 512:1 mux). To reduce the congestion, multiple instances of smaller physical memories have to be placed so that buffers are fabricated in the routing channels between these instances as shown in figure 2.4b.



(a) Routing around a SRAM instance



(b) Over-the-memory routing

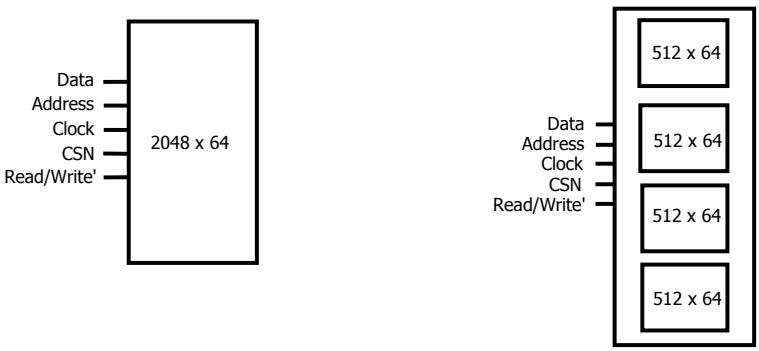
Figure 2.4: Global congestion reduction with over-the-memory routing

Memory configuration	Dynamic power	Area	Leakage power	Congestion factor
2048×64	1.00	1.00	1.00	1.00
4×512×64	0.47	1.22	1.34	4.00

Table 2.1: Trade-off in two possible configurations

2.4 Trade-off: An Example

Figure 2.5 shows two possible implementations of a memory instance with 2048 words and 64 bits. In 2.5a the MSS is realized with a single physical SRAM instance and in 2.5b four smaller SRAM macros are used. In spite of the same SRAM compiler implementation, these two configurations have different PPA values. When a memory is segmented into multiple instances, dynamic power dissipation in the MSS is reduced because only one SRAM memory instance is accessed at a time while the other physical instances are deactivated. On the contrary, area and leakage power increase in direct proportion to the word/bit split factor. At this stage, routing congestion worsens with the increase in word split factor due to memory stacking. Normalized pre-characterized results for the two configurations are shown in table 2.1. Configuration 2 i.e. 4×512×64 consumes about 53% lower dynamic power at the area cost of 22% and has about 34% higher leakage power. Moreover, routing congestion worsens with the word split factor due to increased interconnects in configuration 2.



(a) Single SRAM instance

(b) Memory assembly of four SRAM instances

Figure 2.5: Two possible memory configurations

Chapter 3

The Framework

In this section, the proposed framework for optimal SRAM instance selection for memory subsystems (MSS) is described. It considers the PPA parameters discussed in the previous section. The framework is independent of the EDA tools and evaluates different memory instance options. Reasonable accuracy is targeted in the pre-RTL estimations to suggest the best memory configuration (compiler type, number of physical instances, physical factors such as multiplexers, and banks) and a preferred floorplan. As shown in figure 3.1, the selection flow is divided into four major selection stages: exploration, logic aware selection, floorplan aware selection, and RTL generation.

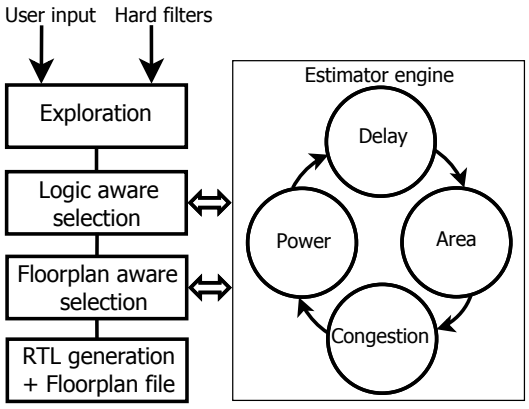


Figure 3.1: Framework flow for optimal SRAM selection

Parameter	Possible values
Memory architecture	single port or dual port
Words	as required e.g. 16384
Bits	as required e.g. 256
Target frequency	in MHz
Optimization objective	low dynamic power or low leakage or high density
Routing channel widths	e.g. 20 micron

Table 3.1: Framework inputs

3.1 Exploration

As the first step to explore MSS configurations, the user provides the memory capacity (words× bits) and the optimization objective intent (low dynamic power or low area). Target frequency of operation along with the desired process, voltage, and temperature (PVT) corner is also an input. Operating frequency is used as a hard filter for discarding configurations which do not satisfy this requirement. Floorplan related information such as routing channel width, buffer placement are optional parameters with preset defaults, if left unconfigured.

Memory requirements captured from an user input file (shown in table 3.1) are forwarded to the memory exploration stage. This stage retrieves relevant design metrics of interest (dynamic power, area, leakage power etc.) for all possible memory instances across the entire range of available memory compilers from a pre-characterized database (formed by SPICE simulations). The primary function of this step is to retrieve all the possible combinations of physical SRAM macros (either assemblies or singular memory instances) which can satisfy the user design objectives. This step is expected to suggest many possible instance combinations, each of which represents a unique contour in the spider-web diagram.

3.2 Logic Aware Selection

Once all possible combinations of SRAM instances (either singular instances or assemblies) are available, the filter for frequency of operation is applied. Other hard filters such as SRAM cell choice, or compiler optimization type (density, low power etc.) can also be implemented to reduce the selection pool. Filtered configurations are ranked using a memory cost function Mem_cost

defined in equation 3.1. It is defined as a weighted sum of the normalized values of primary PPA metrics such as area A, dynamic power dissipation P, and leakage power consumption L [15], [16]. Weights W_a, W_p, and W_l are assumed based on the relative importance of the optimization objectives. This stage is only logic aware i.e. dynamic power, leakage power, and area of available configurations correspond to the pre-characterized data of physical memories. Based on the ranking, the top N configurations are selected for the further processing, N being a user input.

$$Mem\_cost = \frac{W_a \times A + W_p \times P + W_l \times L}{W_a + W_p + W_l} \tag{3.1}$$

3.2.1 Algorithm for Logic Aware Selection

The algorithm for logic aware selection represents a modular approach used in the framework. Pre-characterized PPA values (area, dynamic power, and leakage power) of all the explored physical SRAM options are normalized using a `normalize` function with respect to the minimum PPA value. Next, the `sort` function sorts all the explored memory options based on the cost calculated. N configurations with the least cost are filtered out from this stage using a `filter` function. Height and width of the filtered SRAM macros are stored for floorplan related estimations.

Algorithm 1 Algorithm for logic aware selection

1: Filter out based on frequency of operation
2: $normalize(area)$
3: $normalize(dynamic\ power)$
4: $normalize(leakage\ power)$
5: $cost \leftarrow (Wa * area + Wp * dynamic\_power + Wl * leakage\_power)/(Wa + Wp + Wl)$
6: sort(cost)
7: filter(N)
8: **for** $i \leftarrow 0$ to $results$ **do**
9: $\quad h\_file \leftarrow H[i]$
10: $\quad w\_file \leftarrow W[i]$
11: **end for**

3.3 Floorplan Aware Selection

The floorplan requirement in the form of the physical aspect ratio (defined as ratio of height over width of the floorplan boundary) of the MSS is known to the designers. Floorplan information is used as an additional input for the framework at this stage. Dynamic power dissipation, MSS area, and timings are estimated based on this input. If a designer has no specific floorplan for MSS initially available, the framework evaluates every MSS configuration selected by the logic aware stage, with a number of floorplan schemes dictated by equation 3.2. It checks two conditions: first, if the MSS is a single SRAM macro and second, if the square root of the number of SRAM instances belongs to set of natural numbers. For example, for 16 physical memory tiles (instances), three configurations are evaluated - 4×4, 8×2, and 2×8, targeting aspect ratios of 1, 0.5, and 2 respectively, where $X \times Y$ configuration means Y memories are arranged in X rows. Similarly, for 8 memory instances, two configurations are considered: 4×2 and 2×4. Other floorplan options such as 8×1, 1×8 are not evaluated since they have skewed aspect ratios (not close to 1).

$$num\_fp = \begin{cases} 3 & \text{if } \sqrt{I} \in \mathbb{N}, I \neq 1 \\ 2 & \text{if } \sqrt{I} \notin \mathbb{N}, I \neq 1 \\ 1 & \text{if } I = 1 \end{cases} \tag{3.2}$$

where num_fp is the number of floorplan options and I is number of SRAM instances

At this stage of instance selection, power dissipation, delays due to the interconnects and buffers, density loss due to routing channels, and anticipated losses due to routing congestion are estimated and included in Mem_cost. Data/address buses run along all memory instances and contribute to a large amount of the overall capacitance [7]. Moreover, additional area is required for buffer placement between the memories. Also, interconnects and buffers increase the overall delay in the critical path (read path of the MSS).

Slack metric is used for the filtering process and is estimated using distributed RC models. Memory configurations which do not satisfy the required frequency target (due to additional

routing delays) are filtered out. It is possible that for the same logical memory, timing is not met in one floorplan, but in another one the slack is positive. After filtering, the remaining memory configurations are ranked in the same way as in the logic aware selection stage with an additional secondary objective axis for the congestion. The top M memory configurations with suggested floorplan, congestion, and slack metrics are given to the designer for a final examination and selection.

3.3.1 Dynamic Power Estimation

Components of the MSS which dissipate dynamic power, are as follows:

- **Physical SRAM instance**

  As stated earlier in the chapter 1, both inactive and active memories consume dynamic power. Although, dynamic power consumed by an inactive physical memory is only 10% of the dynamic power dissipated in active SRAM instance, inactive components become significant for large number of assemblies such as 16 macros (where 15 memories are inactive, each dissipating 10% of the dynamic power of one active SRAM).

- **Interconnect and buffers**

  In an MSS, three types of buses are present: data (D) bus, address (A) bus, and output (Q) bus. Data bus and output bus have higher dynamic power consumption than the address bus due to wider bit width. Interconnects are routed in metal 3 and metal 4 in a low cost process (6 metal process) through the routing channels. Due to the long wire lengths, routing delays become so high that setup time failures occur in the timing paths with reduced signal strength. Both these issues are mitigated through buffers [17] [18]. The framework uses fixed interval buffer placement methodology i.e. buffers are placed along the bus after every 200 micrometer [19]. Switching interconnect capacitance, gate capacitance of the buffers and intrinsic dynamic power consumption of the buffer cell contribute to the total dynamic power dissipation. One segment of the interconnect with the associated capacitance is shown in figure 3.2.

  The total interconnect length is divided into segments. Dynamic power dissipation in

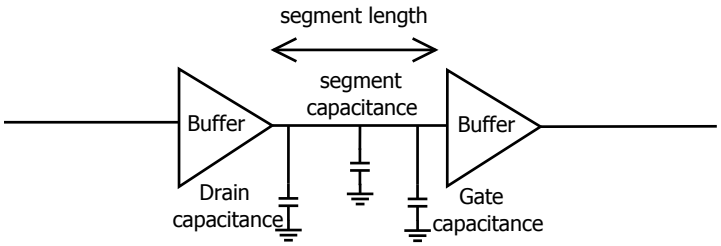


Figure 3.2: Capacitances in one segment of the interconnect

one segment ($P\_1\_segment$) is calculated and multiplied with the number of segments (num_seg) as in equation (3.4). Dynamic power dissipated in one single segment is based on the capacitance values. The intrinsic dynamic power dissipation of a buffer cell P_int_buff is taken from the 28nm FDSOI library and the unit capacitance of metal 3 layer is estimated separately by SPICE simulations. This value multiplied with the parametrized bit width ($data\_width$) gives the dynamic power dissipated in the data bus ($P\_databus$).

$$P\_1\_segment = \alpha \times f \times (P\_int\_buff + 0.5 \times C \times V^2) \tag{3.3}$$

$$P\_1\_bit = P\_1\_segment \times num\_seg \tag{3.4}$$

$$P\_databus = P\_1\_bit \times data\_width \tag{3.5}$$

where α is the switching activity, f is the frequency of operation and V is the operating voltage.

The estimations of the dynamic power dissipation in the output bus are different due to the presence of multiplexers. Here, two varying approaches are adopted. In one approach, no logic (standard cells) is placed in the routing channels i.e. the outputs of all the memories are routed to the outside of the assembly area and are multiplexed. In another approach, logic placement soon after the memories is allowed. Significant power is saved in the later approach due to lesser routing length. However, the former approach allows less routing channel width, thus saving overall area. In this framework, the former approach has been

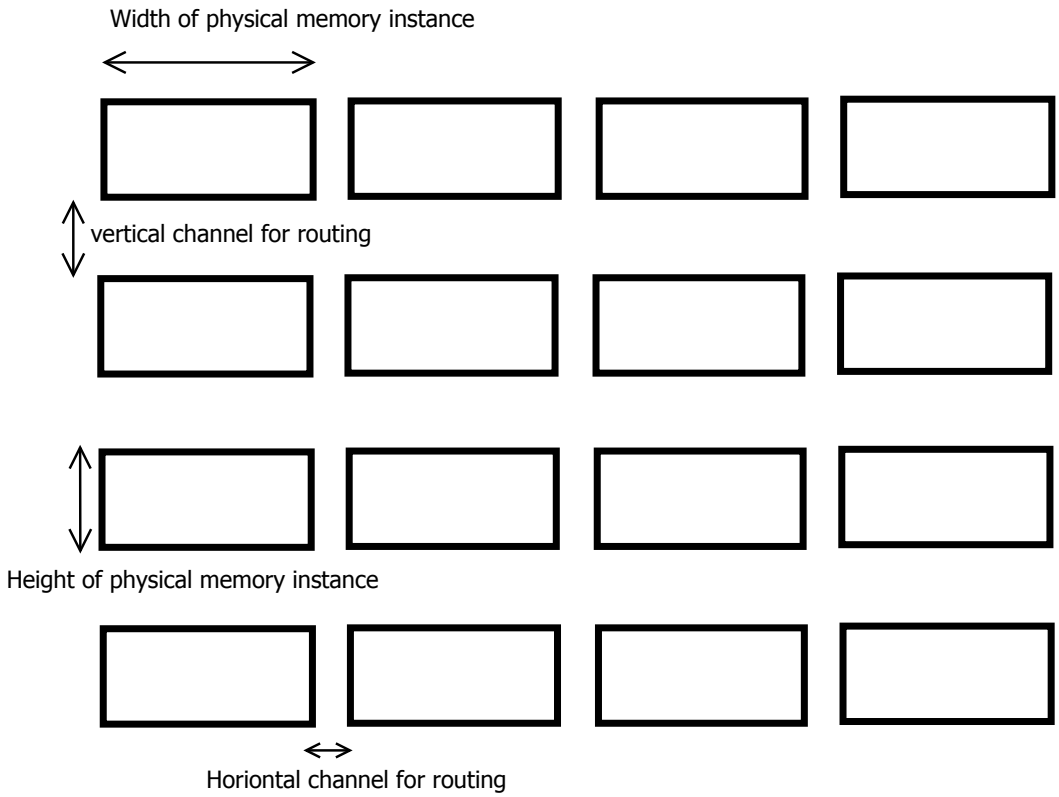


Figure 3.3: Post-floorplan area estimations

used. For estimations, multiplexing logic is assumed at the end. Dynamic power consumed in the interconnect and the internal dynamic power of multiplexers (several stages of 2:1 multiplexers are assumed) are added.

- **Glue logic**

  Combinational logic such as decoder and multiplexers also contribute to the overall dynamic power consumption. The larger the number of physical SRAM instances, the higher is the gate count in the decoder and multiplexer, consuming more dynamic power.

3.3.2 Area Estimation

Post floorplan area is different than the one estimated by the front-end tools. Routing channels are left between the SRAM instances for routing and buffer placement as shown in figure 3.3. Width of the horizontal channel is calculated based on the number of power/ground grid stripes and vertical channel width is estimated based on the number of signals to be routed and the pitch of the floorplan area. Increase in the chip area due to the routing channels is about 10% whereas the logic cells cover around 1%-2% area of the total MSS area.

3.3.3 Leakage Estimation

Leakage power is estimated using the pre-characterized values of the physical memories. In addition, the leakage power consumption in the glue logic, interconnects and buffers is also estimated. The transistor count is much higher in the memory macros when compared with the glue logic. Therefore, the leakage power consumption in the glue logic is negligible. Hence, in the framework, leakage due to the memories only is considered. For memories, there are several types of leakage powers as explained in the chapter 1. However, we choose retention mode leakage component at the worst PVT corner (best process, high voltage, and high temperature).

3.3.4 Delay Estimation

Designs must not fail the timing checks even in the worst case. Memories with larger slack are preferred since increased design margins help in easy and faster routing. Memories with cycle time T_{cycle} not satisfying the required throughput, are filtered out in the logic aware stage. Since before the physical design, the interconnect delays are not taken into account, the results are optimistic in the logic aware selection process i.e. memory configurations just passing the filtering process might fail in the physical design stage due to the interconnect delays.

Input delay D_{in} is the delay incurred by the routing from the input ports (data and address) to the memory instance input (D). Similarly, the output delay D_{out} is the delay incurred from the memory output (Q) to the output port of the MSS. A physical memory instance has two timing parameters: data/address setup time T_{setup} at the input port and access time T_{access} at the output port. Let S_{in} and S_{out} be the worst case slack at input and output ports respectively. Then, the slack metric for the input and output is calculated using equations 3.6 and 3.7. From these equations, it can be observed that the slack can be improved by either increasing the T_{cycle} or by reducing D_{in}/D_{out} (routing delay at input/output pins). T_{cycle} is fixed for the MSS, however D_{in} may be reduced by introducing more number of buffers, up to a limit, till the intrinsic delay of the buffer exceeds the decrease in interconnect delay introduced by them.

$$S_{in} = T_{cycle} - D_{in} - T_{setup} \quad (3.6)$$

$$S_{out} = T_{cycle} - D_{out} - T_{access} \quad (3.7)$$

Routing path for the output slack S_{out} estimation for one bit $Q[0]$ is shown in figure 3.4. As shown, the output of all memories have to be multiplexed outside the memory array since the routing channels between memories are not wide enough for multiplexer logic placement. Delay is caused by the interconnects, buffers in the path (not shown here), and the complex multiplexer logic. Thus, the read path becomes a critical timing path and equation 3.7 (and not 3.6) is considered for slack metric estimation in the framework.

The formulations for delay estimations are shown in equations 3.8 to 3.13. Delay of a buffer ($Propagation\_delay$) depends on the load C_{load}, which is the sum of the capacitance of an interconnect segment and the gate capacitance of the next buffer (drain capacitance is not significant). The overall path delay is the sum of the delays in the buffers and interconnects stated in the equation 3.11 where R, C is resistance and capacitance of total interconnect length respectively.

$$C_{segment} = unit\_capacitance \times L \quad (3.8)$$

$$C_{load} = C_{segment} + C_{gate} \quad (3.9)$$

$$Propagation\_delay = delay\_per\_load \times C_{load} \quad (3.10)$$

$$delay = \frac{R \times C}{num\_seg} + (num\_seg - 1) \times Propagation\_delay \quad (3.11)$$

$$R = unit\_resistance \times interconnect\_length \quad (3.12)$$

$$C = unit\_capacitance \times interconnect\_length \quad (3.13)$$

where L is length between two consecutive buffers

and num_seg is the number of interconnect segments

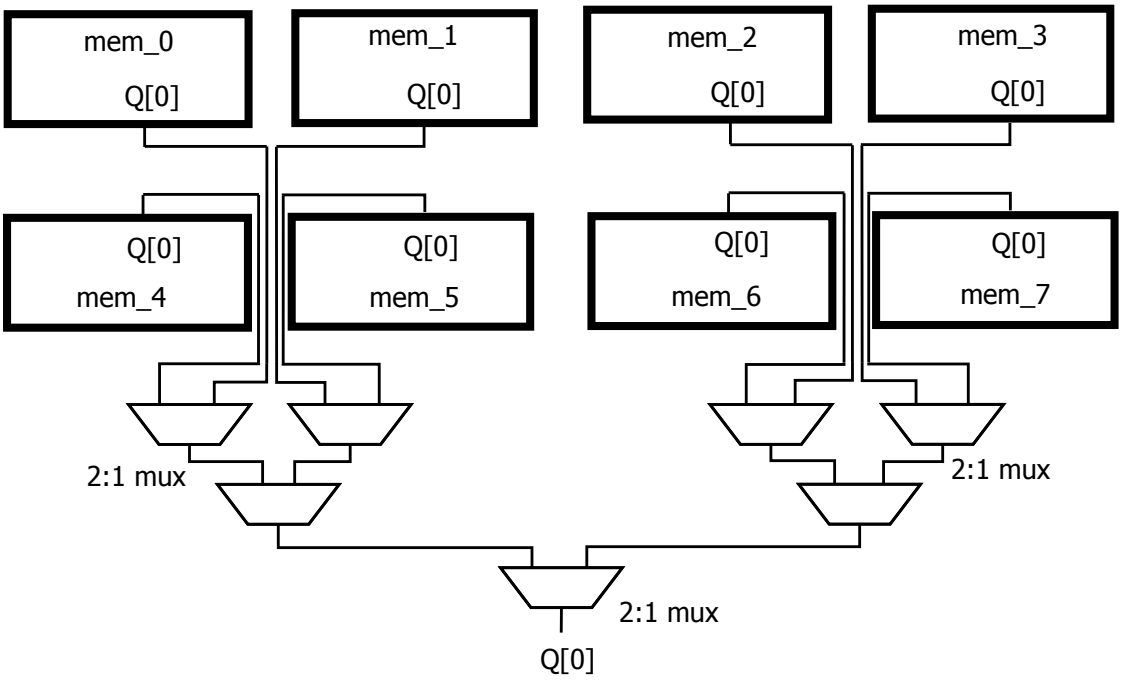


Figure 3.4: Delay estimations

3.3.5 Congestion Factor Estimation

In an MSS, the majority of the area is covered by the memory macros. Hence, the congestion model is formulated in terms of physical parameters of the memories. In this work, the congestion is estimated for all floorplan options available. It is assumed that all nets are routed in M3-M4 metal layers i.e. over-the-memory signals routing is not allowed. Congestion factor has to consider the number of nets in the user-defined routing channels too since routing and congestion are related.

Estimation of congestion needs to be relatively accurate i.e. accuracy of congestion is not matched with the EDA tools but the correct ranking is desired. The lower the value of congestion factor, the better is the design. Congestion depends on the aspect ratio of the physical SRAM along with the port orientation. If output port is on the top/bottom of the SoC, then the wider the physical instance, the less is the congestion since less routing resources are required. Another physical parameter to be considered, is the pin density (ratio of the number of pins to the width of the macro). For any macro, less number of pins means less routing around the macro. Thus, with higher bit split factor, number of data bits per macro decrease, so does the congestion. For instance, assuming data width to be 64 and bit split factor as 2, each macro has 32 data pins

and thus less routing congestion than a macro with a 64 bits data bus.

3.3.6 Algorithm for Floorplan Aware Selection

The estimations for dynamic power, leakage power, floorplan area, and congestion factor are the key constituents of the floorplan aware selection. In algorithm 2, height and width of N configurations after logic aware selection are stored in their respective arrays $H[i]$ and $W[i]$, for floorplan related calculations such as overall MSS area. For every configuration, several floorplan options exist. For each floorplan option, the length of the interconnects is estimated using the `calc_length` function. It estimates the interconnect length based on the arrangement of instances (number of rows num_row and columns num_col). Similarly, slack, dynamic power, and congestion are estimated based on the equations presented in the previous sections. Total dynamic power is the sum of the power dissipated in the active memories, inactive memories, and interconnects. Floorplan area is a function of the physical dimensions of the SRAM macro and the floorplan arrangement (num_row and num_col). Based on the slack violations, configurations are filtered out. For the remaining configurations, floorplan area, dynamic power, leakage power are normalized and the cost function is evaluated. Results are sorted with respect to Mem_cost and top M configurations are presented to the designer.

3.4 RTL Generation and Floorplan Output

The configuration selected by the designer is fed to the register-transfer level (RTL) generator. The final output of the framework is the RTL code of an MSS along with the suggested floorplan. Thus, framework aids the designers by generating the synthesizable RTL code with best physical memories including a floorplan.

Algorithm 2 Algorithm for floorplan aware selection

1: **for** $i \leftarrow 0$ to N **do**
2: $H[i] \leftarrow height\ of\ i^{th}\ SRAMoption$
3: $W[i] \leftarrow width\ of\ i^{th}\ SRAMoption$
4: **for** $floorplan\_options \leftarrow 0$ to $num\_of\_floorplan\_options$ **do**
5: $length \leftarrow calc\_length(num\_row, num\_col)$
6: $slack \leftarrow estimated\ input\ and\ output\ slack\ using\ equations\ 3.8\ to\ 3.13$
7: $total\_dynamic\_power \leftarrow active\ mem\ power + interconnect\ power + inactive\ mem\ power$
8: $floorplan\_area \leftarrow estimate\_area(H[i], W[i], num\_row, num\_col)$
9: $congestion \leftarrow 1/(mux\_factor \times bit\_split\_factor)$
10: **end for**
11: **end for**
12: $filter(options\ violating\ slack)$
13: $normalize(floorplan\_area)$
14: $normalize(total\_dynamic\_power)$
15: $normalize(leakage\_power)$
16: $Mem\_cost \leftarrow \frac{(Wa*floorplan\_area+Wp*dynamic\_power+Wl*leakage\_power+Wc*congestion)}{(Wa+Wp+Wl+Wc)}$
17: $sort(Mem\_cost)$
18: $filter(M)$

Chapter 4

Results

The framework is implemented and validated using the 28nm UTBB FDSOI SRAM compiler database and libraries from STMicroelectronics Ltd. In this section, the key results are presented elucidating the role of physical design on the overall PPA parameters of the memory subsystems (MSS) and hence the selection of an optimal physical memory.

4.1 Initial Analysis for Formulations

The contribution of glue logic to the overall dynamic power, leakage power and floorplan area is estimated by the synthesis and power simulations. Initial analysis is required to see whether the contribution of the glue logic on the PPA metrics is significant or not. It is critical to find the PPA metric to which the glue logic contributes. For example, the multiplexer logic does not dissipate significant dynamic power and leakage power, hence the power dissipated in it, is not included in the total dynamic/leakage power formulations. However, it does incur significant delay and hence needs to be considered in the delay estimations. Thus, the initial analysis helps in building the cost function.

4.1.1 Dynamic Power

The power dissipation in the glue logic is calculated using VCD (value change dump) based power simulations. The procedure for the simulations is as follows:

- Write a verilog description instantiating the physical memories with the associated glue logic.
- Synthesize the verilog code to get the gate level netlist using a synthesis tool e.g. Design Compiler.
- Simulate the gate level netlist with a test-bench to extract the switching activity.
- Switching activity data along with the synthesized netlist is read by Primetime tool to calculate the dynamic and leakage power.

The power contribution of the glue logic differs in the way the test-bench is written i.e. how read/write operations are performed on the memory (due to different switching activity). For example, to read/write X memory locations, all X locations can lie in a single SRAM instance or the addresses are such that a different physical memory is accessed randomly in every cycle.

Results for the dynamic power simulations for an MSS of 1 Mb memory capacity showed that the dynamic power consumed by the combinational standard cells is less than 10% of the total dynamic power dissipation.

4.1.2 Area

Area occupied by the combinational cells (glue logic) is reported during synthesis using Synopsys Design Compiler. For a 4 Mb MSS, the following report shows that the area covered by the glue logic (`Combinational area`) is only 0.2% of the total MSS area (`Total cell area` as shown in the following report). Based on this result, glue logic is not considered in the overall area estimation.

```
Report : area
Design : assembly_SP_1RW_16384x256
Version: I-2013.12-SP5-4
Date   : Sat May  2 13:17:24 2015
****************************************
Number of ports:                          786
```

```
Number of nets:                         4513
Number of cells:                        1934
Number of combinational cells:          1930
Number of sequential cells:                3
Number of macros/black boxes:              0
Number of buf/inv:                       369
Number of references:                      2
Combinational area:                1759.731245
Buf/Inv area:                       553.356804
Noncombinational area:                7.507200
Macro/Black Box area:            803174.375000
Net Interconnect area:      undefined  (Wire load has zero net area)
Total cell area:                 804941.613445
**************************************************
```

4.1.3 Leakage Power

The results for leakage power estimations showed that the leakage power consumed in the memories is of the order of 100 times the leakage power dissipated in the interconnects, buffers, and the standard cells. Thus, for the leakage power metric, only the pre-characterized leakage power of the physical memories is considered in the framework.

4.2 Effect of Physical Design Factors

In this section, the area penalty due to the routing channels is discussed for a 1 Mb MSS. The results also demonstrate the effect of choice of floorplan on the slack metric.

The overall MSS area is a function of the routing channel width, the chosen floorplan and MSS implementation. Area estimations for a 1Mb memory capacity are shown in table 4.1. If the MSS is implemented with 8 instances, overall MSS area differs by 5% depending on the floorplan. This is attributed to the different area of the channels and the glue logic. Penalty in the area is

Number of instances	Prefloorplan area (normalized)	Floorplan	Postfloorplan area (normalized)
8	1.00	2×4	1.14
		4×2	1.20
16	1.09	2×8	1.38
		4×4	1.37
		8×2	1.32

Table 4.1: Area Gain Post Floorplan for 1 Mb Memory Capacity

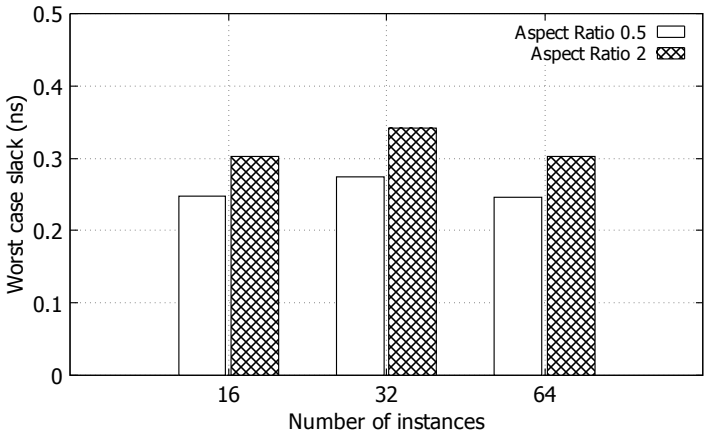
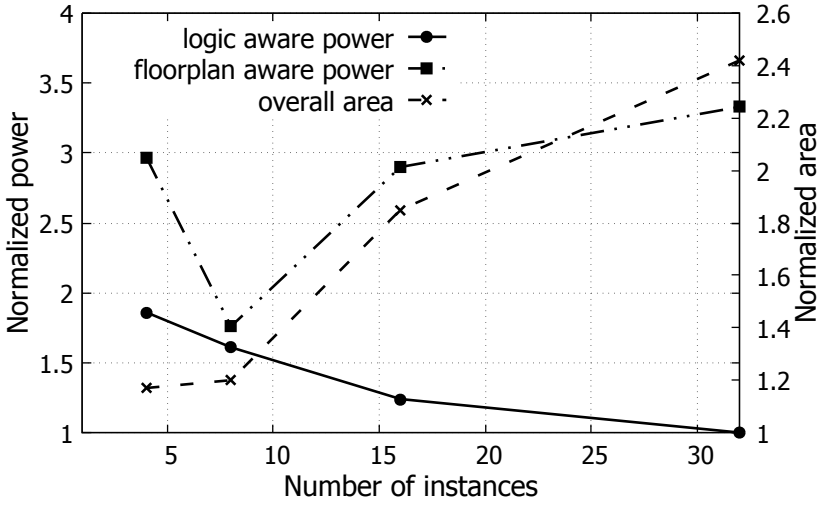


Figure 4.1: Slack improvement with change in floorplan for three implementations for 1 Mb memory capacity. Slack increases by 24%, if physical aspect ratio is 2.

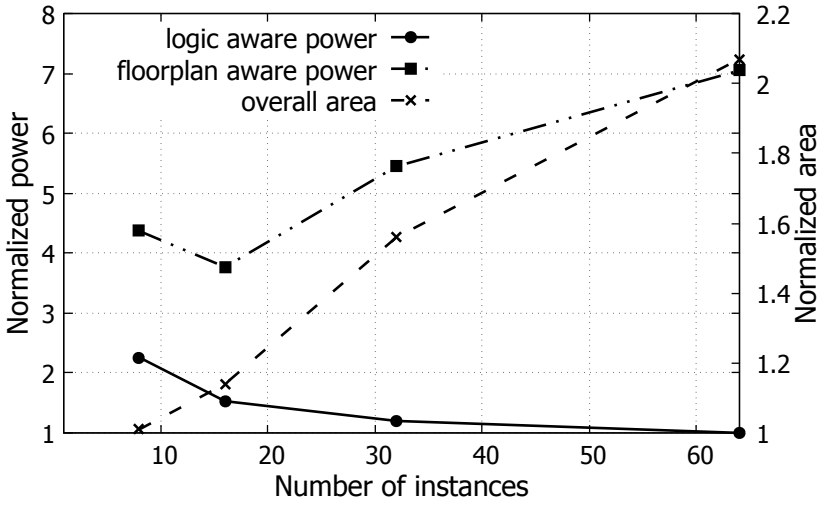
higher, if 16 SRAM instances are used (38% for the 2×8 floorplan).

In figure 4.1, the slack at the output of the MSS is estimated for a 1 Mb memory capacity with different physical aspect ratios of 0.5 and 2 for 16, 32, and 64 SRAM tiles. The slack and the design margins increase by approximately 24%, if the aspect ratio is 2, prioritizing one floorplan above another for the same logical memory requirement.

Figure 4.2a and 4.2b show the area and dynamic power results for 1 Mb and 4 Mb MSS. As higher number of instances are used to implement the required capacity, dynamic power consumption reduces monotonically if only internal power of the memory is considered (because smaller SRAM instances consume lesser power). However, when interconnect and buffer power are added in the floorplan aware selection stage, the curve is no longer monotonous, and an optimal minimum can be found. For the same number of physical SRAM instances, the dynamic power is different for different floorplans. In figure 4.2a and 4.2b, as the number of instances used to implement the MSS is increased, the area increases monotonically. However, closer analysis in figure 4.2a shows



(a) Area and dynamic power variation for 1 Mb memory capacity



(b) Area and dynamic power variation for 4 Mb memory capacity

Figure 4.2: Area and dynamic power results for two different memory capacities across different implementations

that the rate of area increase changes significantly between the second (8 instances) and third sample (16 instances). The analysis shows that if fewer instances are used, the total wire length is less and hence, lesser buffers are introduced in the data path. However, MSS with a higher number of physical instances (such as the one with 16 instances), the interconnect becomes more complex resulting in longer critical paths. To compensate the degradation in performance of the overall MSS, high speed instances are selected by the framework.

4.3 Exploration Results for a 4 Mb Memory Subsystem

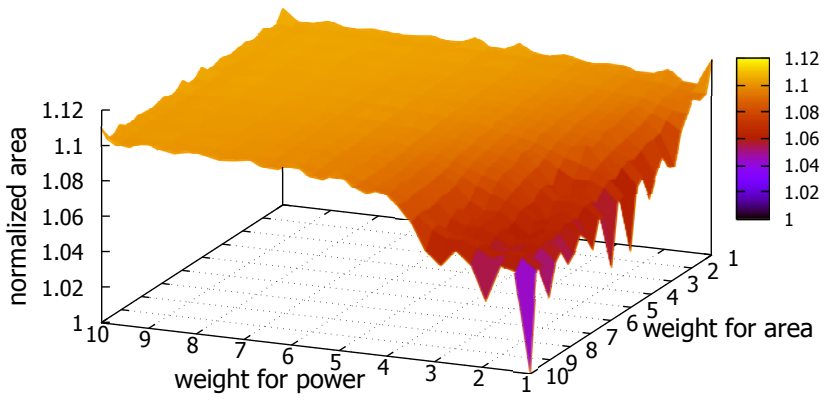
In this section, the effect of design objectives (low power or high density) on the PPA metrics are demonstrated. The framework results for a 4 Mb MSS are discussed and routing delay in the critical path is analyzed. The design implications and gains in over-the-memory routing are also explored.

4.3.1 Translation of Design Objectives on PPA Metrics

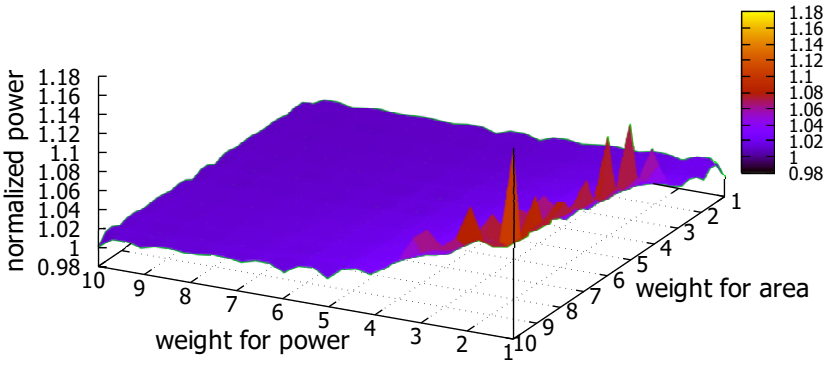
The weights for the dynamic power and area are varied from 1 to 10 for a 4 Mb memory capacity. As shown in figure 4.3, the xy plane shows the variation in the weights of the dynamic power and the area (assuming weight for leakage power as 0), the z (vertical) axis shows the normalized values of PPA metrics (dynamic power and area) for the implementation. In 4.3b, it is observed that for a low dynamic power application, dynamic power consumption is reduced by 18% at the cost of increased MSS area by 12% and vice-versa for a high density application. These plots also let the user decide the weights for dynamic power and area for relative importance of these two objectives.

4.3.2 Estimation of an Optimal MSS

Normalized PPA values for different memory configurations from logic and floorplan aware selection stages for a 4 Mb capacity MSS are shown in table 4.2 and 4.3 respectively. For applications targeting low dynamic power, memories are shown as ranked by the logic aware selection stage such that the best configuration has the least Mem_cost. Thus, configuration C1 with 64 physical memories seems to be most optimal (minimum cost), even though it has 73% more area and a 2.68 times higher leakage power consumption than configuration C9 (with 8 instances).



(a) Area reduction by 12% at weights Wa=10, Wp=1



(b) Reduction in dynamic power dissipation by 18% at weights Wa=1, Wp=10

Figure 4.3: Variation in weights for different target objectives at frequency 500 MHz

However, in the floorplan aware selection stage, C1 consumes 78.4% more dynamic power than C3 and is ranked last with the highest cost in table 4.3. Configuration C3 which has 27% more dynamic power dissipation than C1 in the logic aware stage, becomes the best solution in the floorplan aware selection process. Finally, the chosen memory configuration C3 consumes 44% less dynamic power, 71% leakage power and is 49% denser than the configuration selected by the logic aware selection stage. The change in the results of the two selection processes is due to the significant power dissipation in the interconnects and inactive memories. As shown in figure 4.4, interconnects consume about 30% to 70% and inactive memories consume around 20% of

Number of instances	Physical memory configuration	Dynamic power	Leakage power	Area	Mem_cost Wp=10,Wl=0,Wa=1
64	C1	1.00	3.68	1.73	1.06
64	C2	1.11	4.24	1.74	1.17
16	C3	1.27	1.07	1.04	1.24
16	C4	1.44	1.05	1.07	1.40
64	C5	1.42	3.00	1.59	1.43
64	C6	1.58	3.51	1.59	1.58
32	C7	1.63	2.51	1.46	1.61
32	C8	1.81	2.89	1.46	1.78
8	C9	2.08	1.00	1.00	1.98
32	C10	2.20	2.86	1.47	2.14

Table 4.2: Normalized logic aware selection results for a 4 Mb MSS for low dynamic power application ($N = 10$)

S. No.	Number of instances	Physical memory	Floorplan	Congestion factor	Dynamic power	Leakage power	Area	Slack	Mem_cost
1	16	C3	2×8	0.52	3.98	1.07	1.26	2.79	3.73
2	8	C9	2×4	0.06	4.15	1.00	1.15	1.51	3.88
3	16	C4	2×8	0.52	4.18	1.05	1.29	1.78	3.92
4	16	C3	4×4	0.13	4.52	1.07	1.29	3.98	4.23
5	16	C4	4×4	0.13	4.62	1.05	1.35	2.97	4.32
. .	. .	. .	. .	. .	. .	. .	. .	. .	. .
17	64	C1	4×16	1.00	7.10	3.68	2.47	5.57	6.68

Table 4.3: Normalized floorplan aware selection results for a 4 Mb MSS for low power application ($M = 5$)

the total dynamic power consumption in the MSS. Interconnect power is lowest in C9 with 8 physical memories and highest in C1. If the MSS has an assembly of 64 instances, then the internal power consumption of an active SRAM instance reduces (small instance size), whereas the interconnect power increases (due to complex routing). Percentage contribution of dynamic power dissipation in inactive memory is constant throughout but it becomes more than the dynamic power consumed by the active instance in configuration C1.

Optimal memory selection process is further refined along the congestion objective axis. The memory ranked 4 has 13% higher Mem_cost than the most optimal choice, but offers the preferred maximum slack and minimum congestion. Interestingly, both these configurations are logically the same, but have different metrics due to different floorplans. Similarly, for configu-

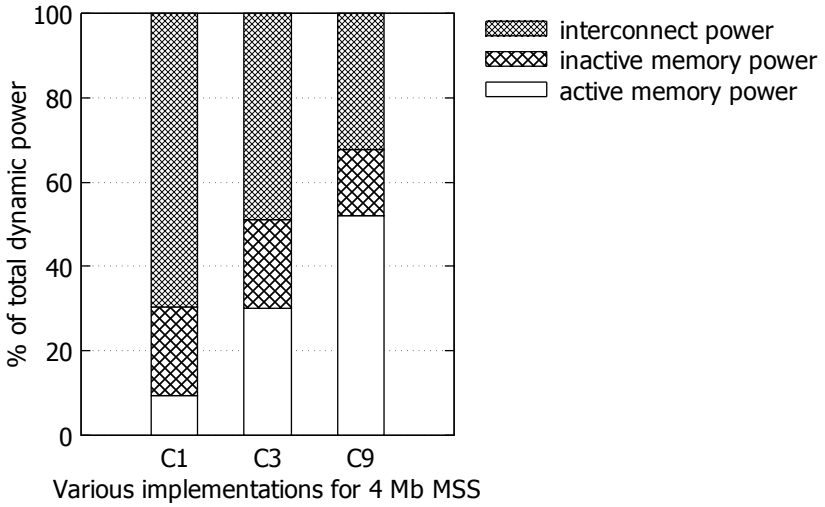


Figure 4.4: Contribution of interconnects and inactive SRAMs to the total dynamic power dissipation in a 4 Mb MSS

ration C4, the floorplan aware selection shows that area reduces by 5%, if a 2×8 layout is chosen instead of a 4×4 arrangement. Hence, the floorplan aware and the congestion aware memory selection suggest not only an optimal memory configuration, but also the best floorplan for the MSS, an associated congestion factor, and design margins available in terms of slack, all at the pre-RTL stage itself.

The framework also allows estimations for the costlier 10 metal process. If more number of metal layers are available, signals can be routed over-the-memory, improving the PPA metrics as shown in figure 4.5. Routing delays decrease by 68%, while the interconnect dynamic power and the overall MSS dynamic power reduce by 58% and 49% respectively. Similar reductions in the dynamic power are observed in the cheaper processes with lesser metal layers, if instances with a larger mux factor are used because of higher porosity. However, internal power consumption of memories is higher, if a large mux factor is used. This offsets some dynamic power gains partially.

4.3.3 Analysis of Routing Delay

The routing delay has been estimated using the RC distributed model and the slack metric was used earlier to show that the few configurations despite qualifying the frequency hard filter, cannot be implemented in the physical design stage due to the interconnect delays (negative

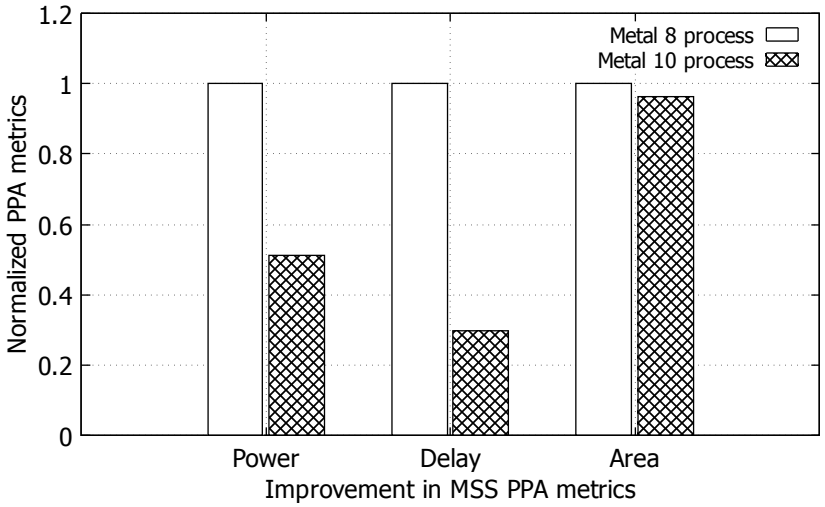


Figure 4.5: PPA improvements due to over-the-memory routing. Dynamic power reduces by 49%, interconnect delay and overall MSS area by 68% and 4% respectively.

Number of instances	Memory configuration	Floorplan	Glue logic delay (in ps)	Routing delay (in ps)	Total output delay (in ps)
64	C1	16×4	246.6	944.80	1191.40
64	C1	4×16	246.6	354.80	601.40
64	C1	8×8	246.6	491.20	737.80
16	C3	8×2	164.4	641.80	806.20
16	C3	2×8	164.4	201.40	365.80
16	C3	4×4	164.4	304.00	468.40
8	C9	4×2	123.3	520.20	643.50
8	C9	2×4	123.3	201.40	324.70

Table 4.4: Routing delays in a 4 Mb MSS

slack). In table 4.4, the total output delay is the delay from the SRAM instance output to the MSS output and it constitutes of the delays incurred in the glue logic (Glue logic delay) and in the interconnect (Routing delay). The contribution of various components in the routing path explains the reasons for any timing violation i.e. whether the negative slack is due to the interconnects and the glue logic or the memory itself. The delay in the interconnects and buffers is about 55% to 80% of the total output delay for different memory configurations. The delay due to the glue logic reduces with the number of physical memories. Similarly, routing delay through the interconnect and buffers is a function of the interconnect length and reduces with the bit split and word split factors.

Chapter 5

Conclusion

In this thesis, a framework has been presented to design optimal memory subsystems (MSS) in the pre-RTL stage by exploring multiple architectural and floorplan options and selecting suitable SRAM configurations aligned to the targeted power, performance, and area (PPA) positioning of the MSS. Distributed RC models have been used to estimate delays and power dissipation in the interconnects to estimate PPA. The proposed framework is fast and scalable because of the empirical estimations. The thesis demonstrates the impact of the floorplan, interconnect power, and congestion on the choice of optimal configuration from a pool of SRAM compilers and options available to the designer. The results show that the floorplan dependent timing and power estimation for a 4 Mb MSS aids in choosing a configuration which is 49% denser and consumes 44% lesser dynamic power than the choice a designer would make, in the absence of these formulations. When large number of metal layers are available for routing signals, delays and area penalty involved with interconnects are reduced. In a 10 metal process, interconnect power reduces by 58% and delays reduce by 68% if compared with a 6 metal process where routing of signals over SRAM instances is not possible. Thus, this floorplan and congestion aware framework enables the designers to choose optimal configurations by optimizing PPA in the pre-RTL stage itself.

Chapter 6

Future Work

6.1 Heterogeneous Memory Subsystems

This work aimed at optimizing the memory subsystem (MSS) by estimating the most optimal physical instance and then generated the RTL instantiating an assembly of the optimal physical memory. Thus, the MSS is called a homogeneous MSS. However, more optimal solutions exist, if heterogeneity is explored. For example, consider a scenario in figure 6.1, where due to the high performance target, the presented framework forms an MSS with the assembly of high performance memories (despite having more area) because the high density memory configuration had negative slack due to the routing delays. However, the area can be saved with the following scheme. SRAM macros in close proximity to the input port (mem_1 and mem_2) can possibly still meet the timings (less routing delay) and if replaced by high density SRAM tiles would save significant area (30% for this example). Farther memories use a different compiler (high performance) and are split (smaller instances are faster). So, the resultant MSS becomes heterogeneous, with high performance memory instances in the critical timing path and high density memories in less critical timing paths.

6.2 Power Gating and Pipelining

Interconnect gating is another design technique to reduce dynamic power dissipation and it has its own impact on the trade-off in the design PPA values. As observed from the estimations that for a large MSS assembly, interconnects dissipate around 60% of the total dynamic power. It is inevitable that the designers use power gating techniques in such a scenario to control the overall power dissipation. Ranking of the presented framework will change in the case of low dynamic power applications. This is one aspect, where this framework can improve to meet the dynamic power budget of the design.

Pipelining is another implementation technique used by the designers. Pipeline stages are introduced in the implementation to increase the throughput at the cost of latency cycles. As shown in figure 6.2, critical paths are identified and a number of pipeline stages are inserted. As observed from the static timing analysis (STA), output from the memory (mem_0/mem_1) to the MSS output Q is a critical path, which includes access time of the memory and delay in the multiplexer logic. Hence, 1-stage pipeline is inserted after the SRAM macro. Addition of pipeline stages introduces a new parameter: latency cycles.

Dynamic power contribution of the clock network and the sequential cells is negligible in the current framework. However, the number of sequential cells increases from 3 to 2054 for a 4 Mb MSS. Power analysis reports that the dynamic power dissipation of the MSS increases from 1 mW to 2 mW due to a 1-stage pipeline while the dynamic power of the active SRAM instance, is reduced to 40% of the total dynamic power. However, area reports show sequential cells cover 0.6% of the overall area. According to the timing analysis, high density memory despite having less throughput runs at twice the frequency by adding a 1-stage pipeline.

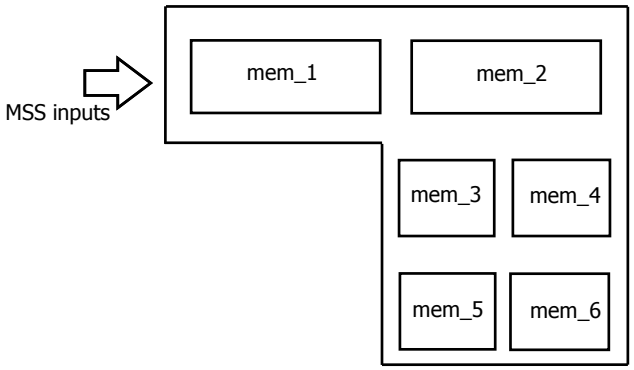


Figure 6.1: A heterogeneous memory architecture

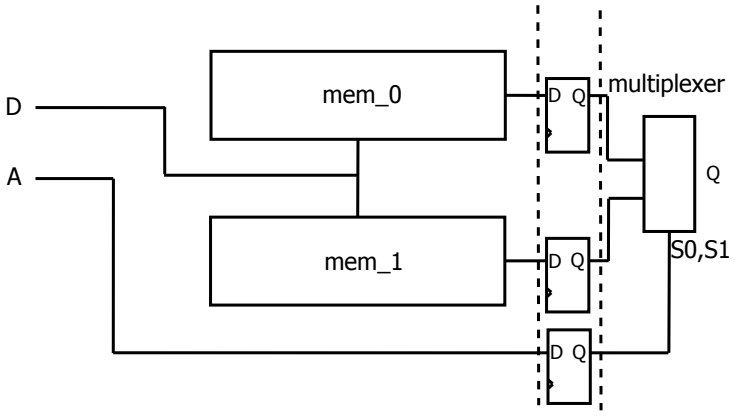


Figure 6.2: Pipeline implementation and critical path identification

Bibliography

[1] C. Ming and B. Na, "An efficient and flexible embedded memory ip compiler," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2012 International Conference on.* IEEE, 2012, pp. 268–273.

[2] R. Capraro and L. Ben Ammar, "A proven qualification methodology for embedded cmos memory compilers," in *Design and Technology of Integrated Systems in Nanoscale Era, 2008. DTIS 2008. 3rd International Conference on.* IEEE, 2008, pp. 1–6.

[3] A. C. Cabe, Z. Qi, W. Huang, Y. Zhang, M. R. Stan, and G. S. Rose, "A flexible, technology adaptive memory generation tool," *University of Virginia*, 2006.

[4] S. L. Coumeri and D. E. Thomas, "An environment for exploring low power memory configurations in system level design," in *Computer Design, 1999.(ICCD'99) International Conference on.* IEEE, 1999, pp. 348–353.

[5] G. Palermo, C. Silvano, and V. Zaccaria, "A flexible framework for fast multi-objective design space exploration of embedded systems," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation.* Springer, 2003, pp. 249–258.

[6] J. Guo, A. Papanikolaou, P. Marchal, and F. Catthoor, "Energy/area/delay trade-offs in the physical design of on-chip segmented bus architecture," in *Proceedings of the 2006 international workshop on System-level interconnect prediction.* ACM, 2006, pp. 75–81.

[7] A. Chhabra, H. Rawat, M. Jain, P. Tessier, D. Pierredon, L. Bergher, and P. Kumar, "Falpem: Framework for architectural-level power estimation and optimization for large

memory sub-systems used in graphics applications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[8] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0," in *Microarchitecture, 2007. MICRO 2007. 40th Annual IEEE/ACM International Symposium on*, Dec 2007, pp. 3–14.

[9] L. Benini, L. Macchiarulo, A. Macii, and M. Poncino, "Layout-driven memory synthesis for embedded systems-on-chip," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 10, no. 2, pp. 96–105, 2002.

[10] S. Pasricha, Y.-H. Park, F. J. Kurdahi, and N. Dutt, "Capps: A framework for power–performance tradeoffs in bus-matrix-based on-chip communication architecture synthesis," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 2, pp. 209–221, 2010.

[11] K. Deb, "Multi-objective optimization," in *Search methodologies*. Springer, 2014, pp. 403–449.

[12] S. R. Sridhara, M. DiRenzo, S. Lingam, S.-J. Lee, R. Blázquez, J. Maxey, S. Ghanem, Y.-H. Lee, R. Abdallah, P. Singh *et al.*, "Microwatt embedded processor platform for medical system-on-chip applications," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 4, pp. 721–730, 2011.

[13] C. J. Alpert and G. E. Tellez, "The importance of routing congestion analysis," *DAC Knowledge Center Online Article*, 2010.

[14] T. Taghavi, F. Dabiri, A. Nahapetian, and M. Sarrafzadeh, "Tutorial on congestion prediction," in *Proceedings of the 2007 international workshop on System level interconnect prediction*. ACM, 2007, pp. 15–24.

[15] I. Y. Kim and O. De Weck, "Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation," *Structural and Multidisciplinary Optimization*, vol. 31, no. 2, pp. 105–116, 2006.

[16] Y. Ding, S. Gregov, O. Grodzevich, I. Halevy, Z. Kavazovic, O. Romanko, T. Seeman, R. Shioda, and F. Youbissi, "Discussions on normalization and other topics in multiobjective optimization," in *Fields-MITACS, Fields Industrial Problem Solving Workshop*, 2006.

[17] C. J. Alpert, A. Devgan, and S. T. Quay, "Buffer insertion with accurate gate and interconnect delay computation," in *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*. ACM, 1999, pp. 479–484.

[18] C. Alpert and A. Devgan, "Wire segmenting for improved buffer insertion," in *Proceedings of the 34th annual Design Automation Conference*. ACM, 1997, pp. 588–593.

[19] V. Adler and E. G. Friedman, "Uniform repeater insertion in rc trees," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 47, no. 10, pp. 1515–1523, 2000.