VLSI Datapath Choices: Cell-Based Versus Full-Custom

by

Andrew Liang Ping Chang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1998

© Massachusetts Institute of Technology 1998. All rights reserved.

Author ....
Department of Electrical Engineering and Computer Science
February 6, 1998

Certified by ....
William J. Dally
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ....
A. C. Smith
Chairman, Department Committee on Graduate Students

VLSI Datapath Choices:
Cell-Based Versus Full-Custom

by

Andrew Liang Ping Chang

Submitted to the Department of Electrical Engineering and Computer Science
on February 6, 1998, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

Traditionally, VLSI architects and designers have acknowledged the area, performance, and effort tradeoffs between cell-based and full-custom implementations of the same datapath function. However, few attempts have been made to characterize these tradeoffs in the context of contemporary fabrication processes and area place and route tools. More importantly, few attempts have been made to determine how to enable cell-based implementations to approach the density and speed of full-custom designs.

This work quantifies the limits of cell-based datapath implementations based on results derived from a detailed analysis of the density and performance tradeoffs in the implementation of two full-custom datapaths, the Integer Register-Read Datapath (IRRDP) and the 64-bit adder/subtracter (ADDSUB), employed in the multi-ALU Processor (MAP) chip.

A cell-based implementation of the IRRDP is **1.64x** larger than the full-custom original. The critical timing path for the cell-based implementation is **11%** longer. The cell-based implementation of the ADDSUB is **1.74x** larger, however, the critical timing path is only **17%** longer. Both of these implementations are *significant* improvements over a traditional ASIC approach. The traditional ASIC implementation of the IRRDP is **14.5x** larger and has **123%** longer critical path. The best ASIC implementation of the ADDSUB is **1.67x** larger and has a **170%** longer critical path.

Also, this work identifies a new physical design efficiency metric (η_{layout}) based on a linear combination of area utilization (ρ_{gate}) and device utilization (ρ_t).

Thesis Supervisor: William J. Dally
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank my advisor, Bill Dally, for supporting me and believing in my potential. Countless times, a few moments of your insight and clarity have unraveled seemingly intractable problems. Thank you, also, for picking challenging research questions, for assembling a strong team of researchers and for allowing each of us to find our own limits.

I would also like to thank my fellow M-Machine architects: Nicholas Carter, Stephen Keckler and Whay Sing Lee. Over the course of our project I have learned much from each of you and it has been a truly valuable and memorable experience working together and proving that so much can be done by so few.

Dan Hartman, Jeff Bowers and Parag Gupta, the M-Machine circuit hackers collectively and affectionately known as the "stooges", were tirelessly diligent, creative and amazingly resourceful.

Also, I especially thank Parag Gupta for his excellent detailed circuit design of both the high-performance 64-bit adder/subtractor and the integer execution unit.

Each of the past and present members of the Concurrent VLSI Architecture group has contributed to making my time in the CVA group memorable, productive and fun.

Friends make one's life happier, one's load lighter and one's way easier. Over the course of my life I have been fortunate have both Rob Kasbohm and Mukesh Jain as best friends. You've both shared my triumphs and my disappointments for over twenty years. You've laughed with me, supported me in tough times and given me a "swift kick" when I deserved it. Thanks for continually showing me the importance of true friendship.

My brother Nelson has had to "put up" with an irascible know-it-all older sibling throughout his life. Many thanks Nelson for a lifetime of friendship and fun.

Finally, I would like to thank my parents, Carl and Birdie, for their love and support. You have always provided valuable guidance throughout my life but have let me find my own way.

Contents

List of Figures

List of Tables

Chapter 1

OVERVIEW

This work proposes an intermediate design point between the full-custom and traditional standard cell methods for the implementation of datapaths. The key to this *cell-based* (**CB**) methodology is the selective creation of a small set of module-specific cells to overcome inherent weaknesses of the collection of primitives typically found in standard cell libraries. The **CB** method also combines manual cell-placement with automated net routing for global assembly.

The results of this research demonstrate that a *cell-based* implementation of datapath functions can approach both the layout efficiency and timing performance of full-custom designs. In the two selected examples, the *cell-based* approach resulted in module areas within **1.74x** of the original full-custom designs and critical timing pathes within **17%** of the original delays.

Continuing improvements in semiconductor technology drive growth in both product development and research by increasing the design space for and the potential of new architectures and systems.

Correspondingly, system architects and chip designers gain expanding opportunities for creativity. However, the resulting new innovations and new levels of function integration in VLSI systems can lead to daunting design complexity.

Simultaneously, schedule pressures and resource constraints also increase dramatically as the tempo of competition accelerates in parallel with the explosion of design possibilities.

As a result, system architects are caught between the blessing of almost unlimited opportunities for innovation and the curse of ever decreasing time to explore and understand each. Ultimately, the scope of future projects will be limited by each design team's innate ability to manage complexity, effort, and resources instead of external constraints such as manufacturing limits.

Cell-based designs have the advantages of hierarchy and reuse but have traditionally been viewed as limited in their achievable density and performance. Optimized cell-based implementation methodologies can significantly reduce design complexity, implementation effort, and resource demand without sacrificing system density or performance. An increased understanding of the practical limits of cell-based designs is therefore beneficial.

1.1 Introduction

Improvements in semiconductor fabrication technology have spurred continued growth in VLSI systems functionality and complexity. The design trends in microprocessor chips clearly reflect this development. As shown in Figure 1-1[1], the effective chip area occupied by successive generations of microprocessors has grown exponentially from 1979 to 1997. This is a direct result of the improvements in semiconductor fabrication technology over the same period. As the number of transistors available on a chip has grown, the density of available wiring resources to interconnect them has increased at a matching rate. This availability of wiring resources enables the additional transistors to be effectively utilized.

The additional available transistors and chip-area have been used to implement either larger cache structures or to add functionality in each new microprocessor generation. The physical limits to the continued scaling of VLSI technology have not yet been reached. Several vendors are predicting the realization of 100 Million and 1 Billion transistor logic chips in the near future. How best to employ these transistors

[1]The data for this graph comes from a wide range of sources [2] [4] [8] [11] [13] [16] [18] [19] [29] [30] [33] [32] [39] [41] [42] [45] [49] [46] [47] [52] [51] [55] [60] [73] [74] [76] [75].

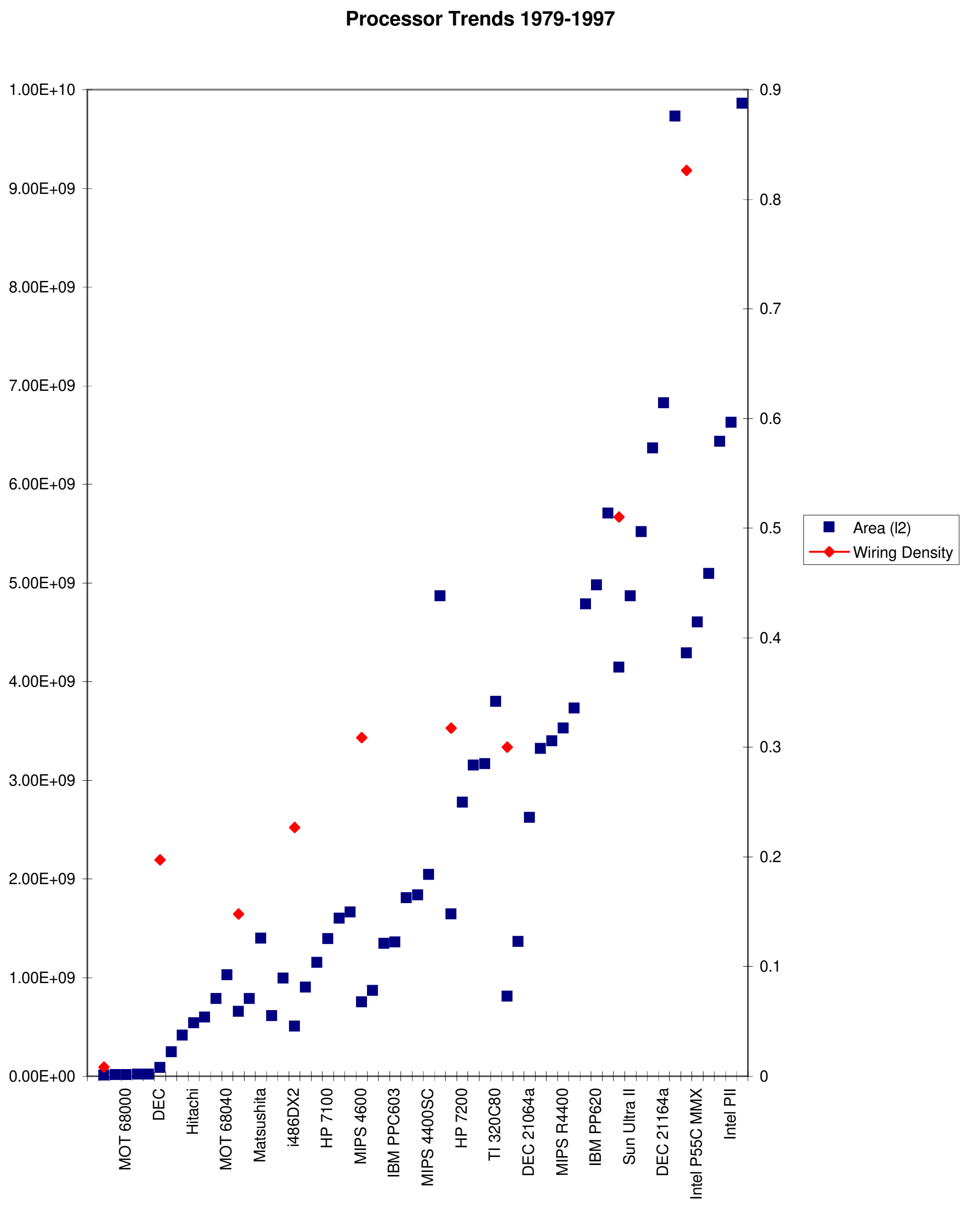


Figure 1-1: *Processor Area and Wiring Density Growth 1979-1997 (λ^2)*

is an active area of open research [22], [71], [78].

While, the physical limits of VLSI are not a near term concern, the current VLSI physical design methodologies *are* reaching noticeable limits. Over the last three years, the steady growth in chip complexity has forced most microprocessor vendors to switch from "flat" design flows to hierarchical ones [12], [28], [7].

The increased number of realizable devices on a chip has also given system architects a spectrum of design approaches, each occupies a specific point of density, performance, flexibility, design effort, and schedule risk in an overall design solution space.

For example, system architects can choose to map their designs into Field Programmable Gate Arrays (FPGA's), Mask Programmable Gate Arrays (GA's), Standard Cell based Application Specific Integrated Circuits (ASIC's), Semi-Custom chips, and Full-Custom chips.

The above-listed solutions represent points in a multidimensional design space. The amount of human effort and degrees of customization represent independent variables that dictate final chip performance, density, size, power consumption, and ease of rework or design variation. The boundaries defining these points are dynamic. As the number of transistors continues to grow the complexity of designs that can be acceptably mapped into each category grows as well. One example of this is the IBM PPC AS A10 microprocessor [9]. Created in 1996, this implementation of the IBM PowerPC architecture was successfully realized using only standard cell technology. In contrast, prior generations of the PowerPC architecture, such as the PPC 601, PPC 603 required semi-custom design methodologies.

At each technology design point, clear tradeoffs exists between the goals of customization, density and performance and the resulting costs in complexity, effort and schedule.

Traditionally, a single solution existed for high density and high performance layout - full-custom design. A common characteristic of most high performance microprocessors is a large percentage of full-custom layout. However, the full-custom methodology does not scale well with design complexity due to its inherent cost, effort and resource requirements.

There are four main advantages to a full-custom layout approach:

1. Freedom in device placement, sizing, orientation, and powering.

2. Customization of routing.

3. Support of non-static circuit styles - the smaller device sizes result in density improvements and higher performance.

4. Inherent iterative nature, detailed focus and fine tuning.

The keys to the listed advantages (especially the device variations and the support of non-static circuit styles) are the allocation of additional effort and resources dedicated to iterative circuit simulations for detailed characterization and the detailed control of both device placement and routing. The important result of this additional effort is the increased predictability of the circuit performance.

Unfortunately, the amount of effort required for full-customization scales at least linearly with the number of unique circuits in a design. The number of transistors available on a chip doubles every 2.5 years. Hence, a simple extrapolation of the 100+ person P6 team at Intel in 1996 results in a 1600+ person 2007 "P10 team". Carrying this to the extreme, in less than 70 years every inhabitant of the planet would be an employee of Intel working on the "P38" chip.

The alternative *cell-based* approach, recently pioneered by the MAP design team, is the combination of module specific cell libraries with semi-custom and full-custom cell placement and semi-custom and full-custom net routing to allow significantly smaller design teams to implement chips with near full-custom density and performance characteristics.

This work explores new points in the design space by using concrete examples to answer the following questions:

1. What are the area and performance costs in the MAP chip of the cell-based datapath implementations as compared to full-custom as exemplified by the IRRDP pipeline stage and the 64bit adder/subtracter?

2. What limits the density and performance of cell-based datapath implementations assuming custom placement and/or custom routing and both a modern multi-level metal fabrication process, and an area Place and Route tool?

3. Are there better ways to classify the layout efficiency of cells?

1.2 Prior Art

Many researchers have contributed to improving the understanding of the constraints involved in optimizing physical design. In general, prior work can be classified into five broad categories: layout synthesis and compaction; automated floorplanning and cell placement; parameter studies on cell size and performance sensitivity; case studies/implementation examples; and effort studies.

Both optimal and heuristic solutions to the topics of compaction, configuration, sizing, and geometry in layout synthesis have been investigated extensively [1] [5] [10] [14] [56] [62].

Researchers have also extensively posited, implemented and studied automated optimization of cell placement and floorplanning [3][15] [20] [31] [38] [64].

The sensitivities of circuits and physical design to a wide range of parameters including process feature size, number and type of interconnect layers has also been characterized [6] [43] [57].

Separately, both industrial and academic teams have detailed their experiences and tradeoff choices for the physical design of a wide selection of designs. [7] [17] [23] [26] [37] [40] [58] [61] [63] [65].

Finally, several studies have been performed to quantify the human effort costs of various chip implementation flows [34] [35] [36].

While this thesis draws from the collective experience of past work, it differs from past work primarily in three important points.

First, this work focuses on characterizing the design space. The tradeoff of human effort and automation is viewed as a spectrum and the primary goal is not obtaining the absolute or optimal performance goal but to characterize the space and facilitate a choice by individual designer that is appropriate to their situation.

Second, this work employs a contemporary foundation of design and process in its experiments. The MAP chip is representative of both the complexity of a state-of-the-art microprocessor and constraints and advantages of a representative multi-level metal sub-micron fabrication process.

Finally, philosophically, this thesis asserts that human intervention and direction in the physical design process is both necessary and desirable. Therefore, this work focuses on facilitating the identification of desirable intervention points and identifying improved ways to direct and guide automated tools.

The balance of this thesis is organized into six chapters. Chapter 2 provides an overview of the methodology and research approach taken in this work. Chapter 3 provides a baseline of comparison between the MAP design, fabrication technology, and selected modules with examples from the available literature. Chapter 4 summarizes insights gained from exploring cell library constraints and introduces a new efficiency metric - η_{layout}. Chapter 5 details the IRRDP experiments. Chapter 6 details the ADDSUB experiments. The final chapter summarizes the conclusions from this research and proposes future work.

Chapter 2

RESEARCH APPROACH

The approach of this research is to first establish baselines for area, density, performance, and power by characterizing the components of the MIT MAP design. Separately, the cell libraries employed in the MAP chip design are examined for density and timing constraints. Next, the layout density and performance limits of a cell-based implementation are examined in detail in two selected modules, the Integer Register Read stage Datapath (IRRDP) and the 64-bit Adder/Subtracter [44] (ADDSUB). As the IRRDP is a predominantly regular structure, exploration of its different implementations provides insights to the limits of bit-sliced cell placement. Conversely, the ADDSUB is a very irregular structure and provides information on the potential difficulties of cell-based structures. In both cases, each module is re-implemented in four different cell-based styles and the resulting layouts are evaluated for layout and timing performance.

The original cell and megacell layouts were implemented by commercial designers and represent a reasonable cross-section of expected value. Some attempts are also made to correlate the MIT MAP chip baseline with contemporary examples in the literature and, thus, further validate this baseline.

Throughout the research four design points are investigated and compared:

1. *Traditional ASIC* - design mapped to standard cell library. Assembly via Cell3 Ensemble (CELL3) automated place and route tool.

Throughout this document this approach is referred to as **ASIC**.

2. *Extended ASIC* - design mapped to standard cell library extended with specialized latch cells. Assembly via Cell3 automated place and route tool.

   Throughout this document this approach is referred to as **ASIC+**.

3. *Custom Cells* - design mapped to a small set of custom cells specifically created for the IRRDP and the ADDSUB. Automatic placement of a bit-slice via CELL3. Replicated tiling of the bit-slice to form the full datapath.

   Throughout this document this approach is referred to as **CB(Auto)**.

4. *Custom Cells and Placement* - design mapped to the previously generated custom cells. Explicit manual tiling of all cells and manual specification of all data and control I/O pin locations.

   Throughout this document this approach is referred to as **CB**.

The reference design in each experiment is the original full-custom implementation and is denoted **FC**.

2.1 MAP Chip

The MAP chip in the first implementation of the MIT M-Machine architecture and is the core processing element of a node in the M-Machine multi-computer system. The MAP chip is a contemporary 64-bit microprocessor and is composed of 5 Million transistors and was implemented on a 18.25mm x 18.35mm die in a 0.7μm-drawn, 0.5μmm-effective, 5-level-metal, CMOS fabrication process.

The MAP design is implemented in a predominantly static CMOS circuit style and is composed of the spectrum of circuit components: standard cell, datapath cells, full-custom arithmetic and datapath components and memory arrays.

A chip plot of the MAP processor is shown in Figure. 2-1.

The MAP processor contains three (3) processing clusters, a 32-KByte unified cache organized as two banks, three (3) 4-KByte instruction caches, two (2) global

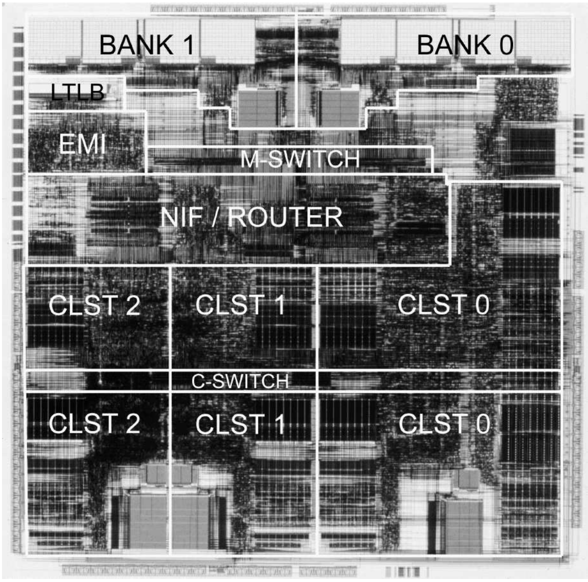


Figure 2-1: *The MAP Chip*

inter-module connection switches and a 2-D mesh router. The three processing clusters are asymmetric as CLST0 contains three execution units: integer, memory, and floating-point. The other two clusters, CLST1, and CLST2, each contain only two execution units: integer and memory.

The MAP ASIC was constructed using a diverse library of components: 7 unique megacells, 91-cell standard-cell library (*mitstdcells*) library, 106-cell composable datapath-cell library (*share library*) including 1 full-custom ADDSUB (with 5 variations) and 1 full-custom multiplier array; 5 unique memory arrays, 5 unique standard-cell based register files, and 5 full-custom I/O pad designs.

The MAP design effort, from architecture creation to tape-out, represents less than 40-man years distributed over four years. Of this effort, the implementation portion of the MAP project represents approximately 30-man years of effort distributed over the final two years. Due to the limited engineering resources available to implement the chip, the MAP design team made an early decision to attempt to obtain design "leverage" by limiting the instances of full-custom circuit design and physical design and by actively seeking opportunities to reuse all full-custom design efforts. Each custom cell was classified by its number of "unique" transistors. As an example, a static CMOS NAND2 is counted as 4 unique transistors. The global chip budget for unique transistors was 1993 devices.

As a result, the project successfully intermixes and utilizes cell and megacell libraries in conjunction with custom placement in the attempt to achieve *full-custom circuit performance and density with standard cell effort and design cycles.* The use of high degree multiplexed input latches is a recurring motif in the MAP chip design. As a result of initial place and route experiments, a set of custom N-only pass-gate multiplexed input latch cells was added to *mitstdcells* and significantly reduce the wiring congestion that had occurred from the original mapping of these specialized functions onto the traditional selection of standard cells. These cells incorporated much of the routing and placement which proved difficult to deduce automatically. As a result the global routing problem was simplified for the Cell3 Ensemble^{TM} (CELL3) tool. Also, custom pre-routing and multiple stage final routing add important additional customization to the standard ASIC flow. *With these methods, the MAP design team achieves high performance datapaths without the cost of full-custom design cycles and effort.*

2.2 Research Methods

While coupled, layout density and circuit performance are addressed separately in this research. A baseline of comparison is created for each characteristic.

To provide a broad range of baseline data for the area study, several tasks are performed.

The basic characteristics of the semiconductor fabrication process used in the MIT MAP project are translated to λ-based rules[1], facilitating comparisons of these results with past, present, and future research.

The device area utilization or, equivalently, the density of gate area (ρ_{gate}) within a

[1]The parameter λ is a normalized, process independent unit of distance equivalent to one half of the drawn gate length [66]. Thus, typically, the fabrication process used for the MAP chip would have a λ equal to 0.35μ. However, the metal rules for this process are typical of those found in a $0.5\mu m$ processes. Therefore, for the purposes of this study λ is 0.25μm.

cell, is employed as one metric of layout efficiency (η_{layout}). The practical upper bound for ρ_{gate} in the MAP project's fabrication process is also calculated as a reference. While ρ_{gate} is an obvious metric, the MAP chip design experience indicates that the layout efficiencies of some circuit forms are not accurately represented by this single metric.

To provide a meaningful supplemental metric, all reported module areas and cell areas are normalized to both λ^2 and square grids (χ^2)[2]

As a result, a transistor density (ρ_t) metric can be calculated and has units of transistors per square lambda (T's/λ^2), transistors per square micron (T's/μm^2) and/or transistors per square grid (T's/χ^2). An additional metric is defined as the inverse of ρ_t and can be viewed as transistor creation efficiency (η_t). Each of the selected cells from the MAP design database is characterized in ρ_{gate}, ρ_t and η_t.

To provide a broad range of baseline data for the performance study, several tasks are also performed.

As with the physical characteristics of the process, the timing performance properties of the process are also established and normalized to a generic set of merit metrics. The parameters τ_C, t$_{INV}$, and t_{NAND2}, are provided to simplify comparison with other fabrication processes. The metric τ_C represents the time required to charge the gate capacitance of a unit transistor to the supply voltage. The parameter t$_{INV}$ represents the basic unloaded inverter delay. The time t_{NAND2} is the propagation delay for a 2-input NAND standard cell driving two equivalent loads.

After establishing design, process, and module baselines. Several of the cell libraries used in the MAP design are characterized in detail. For each cell, the three parameters of layout efficiency: ρ_{gate}, ρ_t, and η_t, are calculated after the examination of the corresponding GDSII database.

After the density limits of the various cell libraries are identified and recorded. The cell libraries are used as targets for the mapping of both the IRRDP and ADDSUB netlists. The resulting layouts are then reviewed for η_{layout}, consumed routing re-

[2]The parameter χ [25] is a normalized, process independent unit of distance equivalent to one minimal wiring pitch for a semiconductor fabrication process. For the MAP chip, χ is 1.8μm and the conversion from λ to χ is 7.2:1.

sources and overall size.

In addition, the critical paths of the selected circuits are identified and the pre-layout schematic delays are quantified using the HSPICE^{TM} circuit simulator. Finally, the post-layout parasitics are extracted and back-annotated into the HSPICE^{TM} databases. The resulting circuit is then re-simulated. The results from the different implementations are reviewed and discussed.

2.3 Candidate Blocks

Two of the MAP chip megacells, the IRRDP and the ADDSUB, are re-implemented by mapping their designs to the M-Machine standard cell library and alternately to a set of module specific cells. The re-implementation requires the following tasks:

- The creation of a new set of schematics
- The creation of a random place and route employing standard cells
- The creation of a random place and route employing both standard cells and selected custom cells.
- The creation of a custom "bit-slice tiling" and manual tiled placement via the Cadence Smartpath^{TM} tool.
- The analysis of the size, aspect ratio and density of the resulting designs and comparison to the existing full-custom implementation.

The exploration of the performance limits are done in a similar fashion. Critical paths for each circuit are identified. The loading data from the previous P&R experiments, specifically wire lengths, required drive strengths, and parasitics, are extracted for each implementation style. The effects of actual loading data on the previously identified critical paths from both the megacells and cell-based implementations are then characterized via circuit simulation. The results of the cell-based designs and megacell designs are then compared.

2.4 Tool Suite

The described research employed the following set of resources, facilities and equipment. Broadly, the requirements can be categorized as software, hardware, and reference data. The research was conducted, predominantly, in the Computer Systems Laboratory (CSL) at Stanford University.

2.4.1 Software

Several commercial Electronic Design Automation tools have been employed. Also, a new set of project specific scripts and tools have been created.

1. HSPICE, Avant!. Industry standard circuit simulator. Used for performance simulations
2. Cell3 Ensemble^{TM}, Cadence. Multi-layer, Area Place & Route tool. Used for actual place and route.
3. Smartpath^{TM}, Cadence. Automated datapath cell placement tool. Used to created automated tiling test cases.
4. Preview^{TM}, Cadence. Graphical floorplanning tool. Used to view bit-slice generation and to perform both manual and custom cell placements.
5. Virtuoso^{TM}, Cadence. Layout editor. Used for implementing new cells.
6. Dracula^{TM}, Cadence. Verification tool. Used for design rule checking and lvs
7. Verilog-XL^{TM}, Cadence. Interpreted logic simulator. Used for logic verification
8. Composer^{TM}, Cadence. Front-end schematic capture and high level language (Verilog) interface. Used for design capture.
9. Suite of GDS tools, Cadence RB. A proprietary set of gds format file manipulation tools
   - gdslist - converts binary GDSII file to ASCII text.

- gds - converts ASCI test to binary GDSII file.
- gdsmerge - merges GDSII routing data with cell libraries
- gdsroot - identifies root structure in hierarchical GDSII file.

10. Scripts, MIT. Set of MAP chip project-specific AWK and PERL scripts
    - cap_gen - translates the CELL3 RC file format into an HSPICE^{TM} deck for parasitic capacitance values.
    - convert_height - converts the height of given cell.
    - convert_all_ypins - creates a layout abstract that mamixmizes access to the I/O pins
    - convert_all_21.6_to_25.2 - generates a set of 25.2μm layout abstracts from a reference set of 21.6μm abstracts.
    - expand_pins - maximizes the length of all M1 pins in a given cell.
    - generate_iopins creates multi-bit I/O pins from a given template
    - repeat_bit creates multi-bit I/O pins from a given template
    - msclefgen creates layout abstracts compatible with the CELL3 tool from a given template
    - 1bit_to_64bit converts single bit .tf files to 64 bit .tf files
    - verify_* automates the DRC and LVS checking of a given set of cells.
    - create_*_strmout automates the generation of GDSII file from the provided Cadence dfII view.
    - trace_logic traces the path of a logical signal through two levels from a structural verilog file

11. Document Preparation, LaTeX, Microsoft Excel, Framemaker

2.4.2 Hardware

- Sun Ultra II class workstation w/ 1GB DRAM, 4GB of swap space, 9GB disk

- Sun Ultra Enterprise Server w/ 8-processors, 2GB DRAM, 32-GB of swap space, 32-GB disk

2.4.3 Database

M-Machine design database, including

- Complete MAP Verilog netlist and schematic database
- Complete MAP GDSII and LEF files
- Complete MAP floorplanning database

Chapter 3

BASELINE

3.1 Overview

The MAP chip is a state-of-the-art 64-bit microprocessor which implements the novel M-Machine architecture. The M-Machine design team's implementation of the MAP chip compares favorably with both the highest density chips previously implemented with 0.5μm-effective feature-size fabrication technology and the density of current chips implemented with leading 0.25μm and 0.35μm feature-size manufacturing processes.

This chapter quantifies both the MAP chip complexity and fabrication process in design and technology independent metrics. This chapter also computes the area efficiency and summarizes the performance characteristics of both IRRDP and ADDSUB. The employed metrics were selected to facilitate comparisons with past, present and future work.

This chapter is organized in seven sections. The second section discusses the physical design of the MAP chip and compares and contrasts the layout efficiency with a survey of industrial microprocessors released between 1995-1997. The third section summarizes the basic features of fabrication process used for the MAP chip, including λ and χ parameters. The fourth section discusses the timing parameters and constraints of the MAP chip design and provides the values for τ_C, t_{INV}, and t_{NAND2}. The fifth section characterizes the register file (IRFD) within the IRRDP and compares its density and read access time to two analytical models and two implemented examples. The sixth section characterizes the ADDSUB and compares its design with four examples from the literature. The final section provides a summary of conclusions.

3.2 Layout Efficiency and Circuit Performance

This section attempts to categorize the MAP design in the context of contemporary microprocessors released between 1995 and 1997.

3.2.1 MAP Chip

The MAP chip is composed of 5-million transistors. Of these, 2.7-million implement the on-chip caches and 2.3-million implement the control logic and execution datapaths. The die size of the MAP is 18.25mm x 18.30mm and as a result the overall η_t is 1068.7 λ^2/T, 67μm^2/T, and 21χ^2/T. As is typical, the area efficiency of the on-chip memory arrays is substantially better than any of the other function blocks. In detail, the constituent memory arrays have a η_t of 218 λ^2/T, 14μm^2/T, and 4χ^2/T. As a result, the balance of the on chip logic blocks have a considerably poorer η_t - 2067 λ^2/T, 129μm^2/T, and 40χ^2/T.

A detailed breakdown of the composition of the MAP chip is shown in table 3.1

Within CLST0, the execution datapaths occupy 44% of the total area. Within CLST1 and CLST2, the absence of the floating point unit reduces the percentage of area consumed by datapaths to 33%. In both cases, the remaining area is occupied by standard cells implementing random control logic.

3.2.2 Contemporary Microprocessors

A broad search of available literature was performed to identify the layout characteristics of a wide range of microprocessors[1] reported between 1995 and 1997.

[1] Sources for the resulting data include: [2] [11] [13] [16] [18] [19] [29] [39] [41] [42] [53] [75].

Component	Area (λ^2)	Area (mm^2)	% of Chip
CLUSTERS	2531.8 M	158.24	47.6
MEMORY	589.8 M	36.86	11.0
NETWORK	371.4 M	23.21	6.9
Local TLB	69.4 M	4.34	1.3
Global TLB	23.8 M	1.49	0.4
I/O Pads	425.6 M	26.60	8.0
I/O-CORE CHANNEL	116.8 M	7.30	2.2
External Memory Interface	62.4 M	3.90	1.2
Global Config Controller	53.1 M	3.32	1.0
Event Queue	89.8 M	5.61	1.7
Data Cache Datapath	142.1 M	8.88	2.7
CLK Drivers	91.2 M	5.70	1.7
Cluster Switch			
Drivers Only	23.4 M	1.46	0.4
Memory Switch			
Drivers Only	25.4 M	1.59	0.5
MISCELLANEOUS	727.7 M	45.48	13.6
TOTAL	5343.7 M	333.98	100.0

Table 3.1: Detail - MAP Chip Composition

Table 3.2 provides a summary of the die area, number of transistors, and resulting η_t for several designs.

Table 3.3 lists both the size of the on-chip cache and the clock rate for each design. As memory arrays have the best η_t, a large amount of on-chip memory would skew the computation for *overall* η_t. In addition, a high clock speed could indicate aggressive full-custom circuit techniques.

The target clock rate of the MAP processor was 100MHz. Each of the MAP's datapaths is designed and validated to operate at or above 100MHz at the typical-N, typical-P, 3.0V 100° C process corner. However, due to resource and schedule limitations, no opportunity was available to tune the random logic sections of the MAP chip. As a result, the anticipated maximum clock rate for the chip is 50MHz.

For the referenced microprocessors, it was not possible to determine a detailed breakdown similar to Table 3.1. While, in general it is difficult even to divide the area and transistors for each of the reviewed microprocessors into array and processor

μProc	Year	Tech	λ	T's	Area	η_t		
		(μm)		(10^6)	(mm^2)	λ^2/T	μm^2/T	χ^2/T
MAP	1998	0.50	0.25	5.00	333.9	1069	67	21
MIPS R4700	1995	0.60	0.30	1.85	73	438	39	12
Cyrix 5x86		0.65	0.32	2.00	144	682	72	22
DEC 21064a		0.50	0.25	2.80	164	937	59	18
TI SS2		0.60	0.30	3.10	299	1072	96	30
Intel P6		0.60	0.30	5.50	306	618	56	19
MIPS R4400		0.35	0.18	2.20	108	1602	49	17
Cyrix 6x86		0.65	0.32	3.00	394	1243	131	40
DEC 21164		0.50	0.25	9.30	299	514	32	10
IBM PPC620		0.50	0.25	7.00	311	710	44	14
Sun UltraI		0.47	0.24	5.20	315	1036	61	19
HP PA7300LC	1996	0.50	0.25	9.00	259	460	29	15
Sun Ultra II		0.35	0.18	5.40	149	900	28	12
HP PA8000		0.50	0.25	3.90	345	1415	88	45
Intel P6		0.35	0.18	5.50	195	1158	35	24
DEC 21164a		0.35	0.25	9.30	209	733	22	10
MIPS R10000		0.35	0.18	6.80	298	1430	44	20
IBM PPC750	1997	0.25	0.12	6.35	67	675	11	8
IntelP55CMMX		0.35	0.18	4.50	141	649	31	22
Sun UltraIII		0.35	0.18	5.40	156	943	29	13
Cyrix 6x86MX		0.35	0.18	6.00	197	1072	33	10
Intel PII		0.35	0.18	7.50	203	884	27	19
DEC 21264		0.35	0.25	15.20	302	649	20	14

Table 3.2: Area/Density: Microprocessor Chips 1995-1997

logic bins, for five of the microprocessors a more detailed analysis was successfully performed.

The partition of area and transistors between logic and memory is reported [67] [13] for both the HP PA7300LC and the HP PA8000. The HP PA7300LC possesses a 128-KByte on-chip cache that consumes 40% of the chip area. The HP PA8000 does not have an on-chip cache, therefore all its transistors implement “logic” and it is possible to directly compare its processor logic density to the MAP chip's processor logic density.

Estimates of array η_t and processor η_t were made for the MIPS R10000, DEC 21164, and IBM PPC 620 based on detailed photomicrographes for each [72] [27] [68].

μProc	Year	Cache(KB)	Freq (MHz)
MAP	1998	35	50/100
MIPS R4700	1995	32	175
Cyrix 5x86		16	100
DEC 21064a		32	300
TI SS2		36	75
Intel P6		16	133
MIPS R4400		32	250
Cyrix 6x86		16	100
DEC 21164		112	333
IBM PPC620		64	133
Sun UltraI		32	167
HP PA7300LC	1996	128	160
Sun Ultra II		32	250
HP PA8000		0	180
Intel P6		16	166
DEC 21164a		112	437
R10000		64	195
IBM PPC750	1997	64	266
IntelP55CMMX		32	166
Sun UltraIII		32	250
Cyrix 6x86MX		64	150
Intel PII		32	233
DEC 21264		128	667

Table 3.3: Design Trends: Microprocessor Chips 1995-1997

The area devoted to on-chip cache for the R10000, 21164, and PPC620 was estimated to be 17%, 18%, and 13% respectively.

Note: in all five cases, a 20% transistor overhead in addition to the transistors required specifically for the arrays was included to account for decoders and array logic.

The resulting data is displayed in Table 3.4.

Dally [21] proposes an empirical set of ranges, in units of λ^2/T, useful for gauging layout efficiency for different types of circuit blocks and implementation styles. These basic ranges, along with the corresponding χ^2/T values for the MAP process, are summarized in Table 3.5. The calculated value of η_t for each of the arrays fall within the range proposed by Dally. For each, the balance of the logic fall in the standard

μProc	λ^2/T Array	Proc	μm^2/T Array	Proc	χ^2/T Array	Proc
MAP	218	2067	14.0	129	4	40
HP PA7300LC	220	1714	14.0	107	7	55
HP PA8000	0	1415	0.0	88	0	45
MIPS R10000	438	2670	13.4	82	6	36
DEC 21164	130	1456	8.1	91	3	28
IBM PPC620	171	1342	10.7	84	3	26

Table 3.4: η_t for Selected Microprocessors

cell range. One cause of this is maybe that the η_t was calculated for the logic portion includes the conglomerated effect of both control and datapath styles.

	λ^2/T	χ^2/T
Memory Arrays	100-500	1.9- 9.6
Full Custom Datapath	500-800	9.6-15.4
Standard Cells (ASIC)	1000-2000	19.3-38.6

Table 3.5: Empirical η_t Ranges

For both the PA7300LC and the PA8000, the MAP chip has a poorer η_t as measured by λ^2/T and μm^2/T. However, in both cases it is **more efficient** in terms of χ^2/T. Two possible explanations for these anomalies are:

1. The MAP chip's micro-architecture is more hierarchical then either PA7300LC or the PA8000 and thus the resulting MAP floorplan requires less global communication and signal bussing.

2. The PA7300LC and the PA8000 design both employ substantially greater amounts of dual-rail signaling than the MAP design. With the doubling of the number of I/O wires per logic function, a lower wire utilization efficiency is expected.

The estimated η_t for the 21164, and the PPC620 are 41% and 54% better then the MAP, respectively. The estimated η_t for the R10000 is worse than for the MAP. However, this may be partially due to the presence of only 3-levels of metal in the fabrication process used for the R10000.

3.2.3 Summary

The MAP has slightly lower physical design efficiency in comparison to most microprocessors implemented between 1995 and 1997. The inclusion of a much larger internal cache in many of these microprocessors accentuates the difference between the industrial chips and the MAP chip. Overall the MAP chip has substantially worst timing performance than the surveyed designs. Even at the original targeted clock rate, the MAP chip still lags the majority of industrial designs reviewed.

Also, the data in Table 3.4 indicate that, individually, neither λ^2/T nor χ^2/T is the unequivocal gauge of layout efficiency. Chapter 4 proposes a new metric which is a combination of their inverse (ρ_t) with ρ_{gate} to generate a new metric η_{layout}.

3.3 Normalization: Fabrication Process

When the implementation phase of the MAP chip began in 1995, access was obtained to a then cutting-edge 0.5μm-effective feature-size semiconductor fabrication process.

3.3.1 Process Details

The effective gate width in the selected process for the MAP chip is 0.5μm resulting in $\lambda = 0.25\mu$m. The process can accommodate five levels metal routing with a minimum wire pitch which is approximately equal to 7 λ. Thus, the "grid" pitch for the selected process is 1.8μm (ie. $\chi = 1.8\mu$m). Also the process design rules allow the arbitrary stacking of vias.

The overall MAP chip area is 5.3Gλ^2. The available min-pitch routing resource for each layer is 103Mχ_2.

3.3.2 Maximum ρ_{gate}

An additional characteristic of the fabrication process is the maximum ρ_{gate}. For each fabrication process it is possible to calculate the density limit for ρ_{gate} from the design rules. This limit can be expressed in the following equation and assumes that transistor folding is employed extensively:

$$MAX\rho_{gate} = \frac{N(cell_{height} - min_{SpaceNdiffToPdiff})L_{eff}}{cell_{height}((N+1)((2\times overlap_{diff-gate}) + width_{contact}) + N\times L_{eff})}$$

The parameter N is equal to the number of transistor folds.

For the MAP chip datapath cells, MAX ρ_{gate} equals 32%. And for the MAP chip standard cells, MAX ρ_{gate} equals 30%. The difference between the datapath cells and the standard cells in the MAP design is the cell$_{height}$. Each datapath cell has a height equal to 25.2μm. While, each standard cell has a height equal to 21.6μm.

3.4 Normalization: Process Performance

For the MAP project, the following process parameters and assumptions were used:

1. V_{DD} = 3.3V (defined)

2. t_{OX} = 125$\mathring{A}$ (from public domain)

3. c_{OX} = 2.76 $\frac{fF}{\mu^2}$ (calculated)

4. L_{eff} = 0.5μm (from public domain)

5. μ_n = 500 $\frac{cm^2}{V-s}$ (estimated)

6. μ_p = 180 $\frac{cm^2}{V-s}$ (estimated)

7. V_T = $\pm$0.7V (estimated)

8. β_n = 138$\frac{W}{L}$ μA/V_2 (calculated)

9. β_p = 50$\frac{W}{L}$ μA/V_2 (calculated)

The following estimates were checked against the typical delay data provided by the vendor: t_{NAND2} = 220ps (2-input NAND with FO = 2).

Note: All the data presented in this section was either gleaned from open literature or independently derived,

3.4.1 τ_C

τ_C: A useful time characteristic of a semiconductor fabrication process is the time required to charge the gate of a transistor or discharge the gate of a transistor from the supply voltage. This parameter is deemed τ_C [21]. The parameter τ_C provides a normalized basis for comparison of the timing performance of different semiconductor processes and different chip implementations.

$$\tau_c = \frac{C_g V_{DD}}{I_{ds}} = 16ps$$

An alternate timing characterization constant is the time required for an electron to drift from the source to the drain of a transistor. This parameter is the transit time, τ_t. For the selected process, $\tau_t \approx$ 5ps. Note, in this case, τ_t is limited by velocity saturation.

3.4.2 Inverter Delay

t_{INV}: Another commonly used timing metric for semiconductor processes is the propagation delay of a nominal inverter. The t_{INV} for this process is calculated as follows [79]:

$$t_f = k\frac{C_L}{\beta_n V_{DD}} \approx 170ps$$
$$t_{df} = \frac{t_f}{2} \approx 85ps$$

$$t_r = k\frac{C_L}{\beta_p V_{DD}} \approx 156ps$$
$$t_{dr} = \frac{t_r}{2} \approx 78ps$$

$$t_{INV} = \frac{t_{df} + t_{dr}}{2} \approx 82ps$$

3.4.3 NAND2 Delay

$\mathbf{t}_{NAND2}$: Many ASIC fabrication vendors employ t_{NAND2} as a basic reference parameter.

$$t_{NAND2} = \frac{4t_{df} + 2t_{dr}}{2} = 248ps$$

The original target cycle time for the MAP chip is 10ns cycle which represents 40 gate delays.

3.4.4 Contemporary Fabrication Processes

This section briefly tabulates timing performance information for a set of contemporary semiconductor fabrication processes. The purpose of this section is to provide a reference in comparison with the fabrication process selected in the MAP project. Note, all of the data list in Table was obtained from public domain sources. The majority of the data was found in [48], [50] and [54]. Tables 3.6, 3.7, 3.8 summarize the physical properties:

As shown in the previous section, in 1995, mainstream microprocessors were predominantly implemented with 0.5μm feature size fabrication technology. Table 3.6 summarizes the physical data for a sample of processes.

Process	$L_{eff}(\mu\text{m})$	t_{ox} ($\mathring{A}$)	Layers	Min Pitch (μm)	V_{DD}
MAP	0.50	125	5	1.8	3.3
DEC CMOS4S	0.46	100	3	2.3	3.3
TI EPIC-2BE	0.50	120	3	2.0	4.8
HP HP14	0.37	98	3	1.8	3.3
HP HP10	0.37	76	4	1.8	3.3
Intel "0.60"	0.37	80	4	1.4	3.3
Fujitsu CS-50	0.45	110	4	2.1	3.3
Toshiba "0.60"	0.60	240	3		5.0
Toshiba "0.50"	0.50	230	3		3.3
Toshiba "0.40"	0.40	170	3		3.3
NEC "0.50"	0.50	220	3		3.3

Table 3.6: 0.5μm Fabrication Processes: Physical Data

In 1996, both 0.35μm and 0.5μm-effective feature-size fabrication processes were used to produce mainstream microprocessors. By 1997, chip manufacturers had largely completed their upgrade from 0.5μm feature-size technology. As a result, most mainstream microprocessors were fabricated with 0.35μm feature-size technology. Table 3.7 summarizes the physical data for a sample of processes:

Process	$L_{eff}(\mu m)$	t_{ox} ($\mathring{A}$)	Layers	Min Pitch (μm)	V_{DD}
MAP	0.50	125	5	1.8	3.3
IBM CMOS5S	0.36	90	5	1.4	3.3
DEC CMOS5	0.37	90	4	1.5	3.3
TI EPIC-3	0.35	80	4	1.5	3.3
HP HP14	0.37	98	3	1.8	3.3
HP HP10	0.37	76	4	1.8	3.3
Intel "0.60"	0.37	80	4	1.4	3.3
IDT CMOS7+	0.30	90	3	1.3	3.3
Fujitsu CS-55	0.35	80	4	2.1	3.3
Toshiba "0.30"	0.30	111	3		2.5
NEC "0.35"	0.28	85	3	1.5	3.3

Table 3.7: 0.35μm Fabrication Processes: Physical Data

In the latter half of 1997 and into 1998, microprocessor designs continued to migrate to smaller feature-size technology. Table 3.8 summarizes the available physical data for a sample of current 0.25μm processes.

Process	$L_{eff}(\mu m)$	t_{ox} ($\mathring{A}$)	Layers	Min Pitch (μm)	V_{DD}
MAP	0.50	125	5	1.8	3.3
IBM CMOS5X	0.25	77	5	1.2	2.5
IBM CMOS6S	0.20	55	5	1.1	2.5
DEC CMOS6	0.25	65	5	1.2	2.5
TI EPIC-4	0.22	57	5	1.2	2.5
TI EPIC-5	0.18	60	6		1.8
Intel P854	0.22	60	4	0.9	2.5
NEC "0.25"	0.18	93	3		2.5

Table 3.8: 0.25μm Fabrication Processes: Physical Data

A summary of the collected timing properties is presented in Tables 3.9 3.10 and 3.11: As with the physical data, the processes are organized by feature size. The average τ_C for the listed 0.5μ processes is 13.7ps. The average τ_C for the listed 0.35μ processes is 12.2ps. The average τ_C for the listed 0.25μ processes is 8.3ps. While the τ_C improvement from 0.5μm technology to 0.35μm technology stems primarily from the shortened channel length, in the transistion from 0.35μm to 0.25μm technology, the additional voltage scaling from 3.3V to 2.5 volts also assists directly in lowering τ_C.

Process	τ_C (ps)	τ_t (ps)	t_{INV} (ps)	t_{NAND2} (ps)
MAP	16	5	82	248
DEC CMOS4S	16	5		
TI EPIC-2BE	16	5		
HP HP14	13	4	143	210
HP HP10	13	4		155
Intel "0.60"	13	4		
Fujitsu CS-50	16	5		210
Toshiba "0.60"	17	6		240
Toshiba "0.50"	16	5		230
Toshiba "0.40"	15	4	50	170
NEC "0.50"	16	5		220

Table 3.9: 0.5μm Fabrication Processes: Timing Data

Process	τ_C (ps)	τ_t (ps)	t_{INV} (ps)	t_{NAND2} (ps)
MAP	16	5	82	248
IBM CMOS5S	12	4	75	180
DEC CMOS5	13	4		
TI EPIC-3	13	4		
HP HP14	13	4	143	210
HP HP10	13	4		155
Intel "0.60"	13	4		
IDT CMOS7+	10	3		
Fujitsu CS-55	13	4		
Toshiba "0.30"	12	3	42	111
NEC "0.35"	10	3		113

Table 3.10: 0.35μm Fabrication Processes: Timing Data

A wide range of fabrication processes are currently available to the system architect. The minimum effective feature size of these processes range from 0.18μm to 0.60μm. In comparison with the full spectrum of fabrication processes, the MAP chip is fabricated on a relatively slow technology. However, amongst exclusively 0.5μm processes, the technology employed in MAP chip is competitive.

Process	τ_C (ps)	τ_t (ps)	t_{INV} (ps)	t_{NAND2} (ps)
MAP	16	5	82	248
IBM CMOS5X	12	4	60	145
IBM CMOS6S	7	2		
DEC CMOS6	10	3		
TI EPIC-4	7	2		
TI EPIC-5	7	2	19	60
Intel P854	8	2		
NEC "0.25"	7	2		93

Table 3.11: 0.25μm Fabrication Processes: Timing Data

3.5 Survey of Register File Structures

Area: Two analytical models [21], [69] and one detailed implementation [58] were reviewed to provide normalization data for the register file portion of the IRRDP. For the first model [21], the minimal area required for a register file cell is:

$$min_{A(\lambda^2)} = [8(cell_{min} + P)]^2$$

$$\text{P} = \#\text{ of ports}$$

$$cell_{min} = 5$$

The resulting required area for the complete register file is calculated as follows:

$$Area_{RF(\lambda^2)} = (\#Registers)(Bits/Word)(min_A)$$

For the second model [69], the minimal area required for a register file cell is calculated as a multiple of "rbe". A rbe is defined to be the area of a static 6T storage cell.

$area(rbe)$

$$= (Bits/Word)(1 + (0.1)(\#Ports))(\#Registers + L_{dec})(1 + 0.25(\#Ports - 2))$$

L_{dec} was determined empirically to be equal to six (6).

Irrissou [58] reported a 3-ported register file bit-cell with an area of 2624 λ^2/bit. This cell and associated register file were developed for use in a high-performance datapath study.

Table 3.12 summarizes the three datapoints and the full-custom IRRDP register file cell (RFDCELL). Note, for the MAP chip process, one rbe is equal to $907.2\lambda^2$. An individual RFDCELL has two write ports and one read ports (ie. P = 3). However, the full register file has 7 total ports (4 read and 3 write). Both the MAP architecture and the organization of the register file guarantee that any specific register can have a maximum of two unique writers and one unique reader each cycle.

	λ^2	T's	η_t
RFDCELL	2540	7	362
Dally Model	4096	7	585
Mulder Model	10319	7	1474
Irrissou Cell	2624	7	375

Table 3.12: Actual versus Reference; RFDCELL

Table 3.12 summarizes the three datapoints and the complete full-custom IRRDP register file (IRFD). Note, the IRFD consists of 75 registers. The Irrissou implementation includes an additional reported 724 λ width required for the sense amplifiers and write drivers.

	$M\lambda^2$	T's	η_t
IRFD	20.3	43200	470
Dally Model	19.7		
Mulder Model	18.0		
Irrissou	15.7		

Table 3.13: Actual versus Reference: IRFD

While both analytic models provide inaccurate estimates of the individual RFDCELL storage bit size, the Irrissou implementation correlates to within 3% of the MAP RFDCELL. In the case of the Mulder model, the 6 rbe offset for decoders significantly skews the estimate for the single bit register case.

The actual IRFD is between +3% and +11% of the analytic models references for the full 75-register 64b word file. As the Irrissou implementation has only three ports, it is significantly smaller when scaled to the full register file size.

A supplemental reference from the literature was also identified. Jolly reports a 17-ported register file [59]. This file was implemented in a 1μm CMOS fabrication process and was composed of a basic 11-transistor cell occupying 30.5μm by 36.5μm area (4453λ^2, 178χ^2). The resulting η_t's are 405λ^2/T and 16χ^2/T. Also the full register file was reported as containing 79000 transistors and occupying 2.94mm x 4.66m. This results in a total η_t of 696λ^2/T and 27.8χ^2/T.

Performance: Both Jolly and Irrissou report timing data for their respective register files.

The read access of the register file reported by Jolly is 9ns representing 288 τ_C for the 1-μm process. The read access time of the MAP register file is 1.8ns representing 112.5 τ_C. Also, in comparison, the register file reported by Irrissou has a read access time 57τ_c. The relative speed improvement in comparison to the RFDCELL is due its simpler overall structure, smaller total array size (32 regs vers 75) and use of dynamic circuit techniques.

3.6 Survey of Adders

The 64-bit adder/subtractor [44] used in the MAP chip is a hybrid carry-lookahead/carry-select design implemented in a static CMOS circuit style and includes N-only pass gates.

Many adder designs have been reported in the literature. Four [58] [28] [70] [77] are summarized in Tables 3.14 and 3.15 for comparison to the MAP ADDSUB.

In addition, an analytic model [21] was used to estimate both the area and performance of the ADDSUB. The model estimates the area of a 64-bit carry select adder to be 1.3Mλ^2 and the t$_{PD}$ to be equal to 400 τ_C. For the MAP chip the 400 τ_C is equal to 6.4ns.

Process	Year	Width	Tech (μm)	T's	Area M$\lambda^2 t$
MAP	1998	64	0.50	4906	4.15
HPCD	1997	32	0.80	6000	13.30
HP'96	1996	64	0.50	7000	3.94
IRIS'93	1993	64	1.20	3392	20.20
ALPHA	1992	64	0.75		

Table 3.14: Summary of Adders - Area

Process	Year	Delay (ns)	# τ_C	Latency (cycles)
MAP	1998	4.0	250	1
HPCD	1997	1.8	284	3.5
HP'96	1996	1.0	77	1
IRIS'93	1993	5.5	589	3
ALPHA	1992	3.0-5.0	214-357	1

Table 3.15: Summary of Adders - Speed

The MAP ADDSUB is 60% faster than estimated by the analytical model but occupies 319% of the predicted area.

Table 3.15 contrasts the timing performance of the MAP ADDSUB with a set of selected adders detailed in the literature. As compared to the HPCD, HP'96, IRIS'93, and Alpha adders, the MAP ADDSUB is 13% faster, 210% slower, 135% faster, and between 15% slower and 43% faster, respectively. A possible source of the difference between the HP'96 and the MAP ADDSUB is that the HP'96 adder implemented in a dual rail CVSL circuit style while the MAP ADDSUB is implemented in a static CMOS style. Also, as the ADDSUB implements subtraction, one additional gate delay exists in its critical path due to the XOR used to invert the B input.

For both the HPCD and IRIS'93 adders, the reported MAP chip's advantage is a bit overstated as both of these adders are pipelined and therefore include additional latch delays along their critical pathes

The inexactness of the MAP ADDSUB's performance comparison with the ALPHA adder stems from a lack of available data on the exact t_{PD} of the ALPHA adder. The existing sources indicate that the adder has a single cycle latency for a 5ns cycle time, but do not provide additional detail on timing margin, the latch setup/hold times or latch propagation delays.

3.6.1 Summary

Table 3.16 summarizes the data related to η_t. The layout efficiency of the HP'96 adder is clearly within the range proposed by Dally for datapath cells. The layout efficiency of the MAP ADDSUB is within 6% of the range specified for datapath cells, but is 67% less efficient in comparison with the HP'96 adder.

The balance of the adders have layout efficiency comparable to standard cell designs.

	$M\lambda^2$	T's	η_t
MAP	4.15	4906	846
HPCD	13.30	6000	2217
HP'96	3.94	7000	563
IRIS'93	20.20	3392	5955
ALPHA			

Table 3.16: Summary of Adders η_t

Overall, the 64-bit adder/subtracter designed for the MAP chip has comparable timing performance to the ALPHA adder. However, due to the MAP ADDSUB's static CMOS implementation it is substantially slower than the HP'96 dual rail CVSL implementation.

3.7 Conclusion

The targeted fabrication process for the MAP chip represented the state-of-the-art at the inception of the implementation phase of the project. The overall complexity of the MAP design and the resulting density of the physical layout compare favorably to contemporary microprocessors. At the module level the density of both the IRRDP and the ADDSUB compare favorably to their counterparts in the literature. While the overall timing performance of the MAP components is lower than the average set by industrial microprocessors, the delays in both the IRRDP and ADDSUB are comparable to several examples reported in the literature.

Chapter 4

RESULTS: CELL LIBRARIES

4.1 Overview

This chapter briefly describes the characteristics of several cell libraries created during the M-Machine project. In the process of analyzing these libraries, some inherent limits to the use of *traditional* CMOS cell libraries as the targets for mapping datapath functions are identified and discussed. In addition, a new model for layout efficiency (η_{layout}) is proposed.

Area overhead occurs when datapath functions are mapped to standard cells. This growth in area has been quantified and is equal to: upto +21% cost due to the alignment of all cell geometries to the global grid; approximately +1% per extra diode contact and between 10%-20% due to the mismatch of N:P ratio between traditional standard cells and datapath cells.

A new layout efficiency metric was defined based on a linear combination of ρ_t and ρ_{gate}.

$$\eta_{layout} = \alpha \frac{\rho_{t-n0}}{\rho_{tMAX}} + \beta \frac{\rho_{gate-n0}}{\rho_{gMAX}}$$

This chapter is organized into three sections. The second section briefly describes both the *mitstdcells* and *share* libraries. It compares these "aligned-to-grid cells" to both a selected set of full-custom cells (**FC**) from the M-Machine project database and also with the cells created for the cell-based re-implementations of the IRRDP and ADDSUB. The third section identifies and discusses limits to density and circuit performance in traditional CMOS cell libraries and also discusses the derivation of η_{layout}, a new efficiency metric to cell density.

4.2 Cell Libraries

Two key cell libraries were created as a part of the M-Machine project: *mitstdcells* and *share*. The cells from both these libraries are compatible with the Cell3 Ensemble^{TM} place and route tool. Specifically, in each cell, all geometries, including every input and output pin, are aligned to the global routing grid ("on-grid"). The two libraries differ in cell height. The standard height for *mitstdcells* cells is 12 χ and the standard height for *share* cells is 14 χ. More importantly, the cells differ in their intended application. The purpose of the *mitstdcells* cells is, primarily, to be the target of the Synergy^{TM} logic synthesis tool. The *mitstdcells* are used predominantly for the random control logic in the MAP chip. The purpose of the *share* cells is to provide a common and reusable set of datapath primitives. The *share* cells are the target for the cell-based datapath implementation employed throughout the MAP chip design flow.

The design of the MAP chip employed several additional cell-libraries: input/output pads (*pads*), used for the requisite interface functions and output drive; clock drivers (*clock*), used to drive and distribute the single phase chip clock; and ram array building blocks (*ram*), used to construct the five different on-chip memory arrays. Whenever possible, cells from each of these libraries were reused to help save design effort. In addition, in several instances, full-custom cells were shared between similar modules as well. An example of this exists within each MAP cluster, as most of the full-custom IRRDP cells were reused either without modification or with slight changes to implement both the memory unit register-read datapath and the floating-point-unit register-read datapaths (MRR_DP and the FRR_DP).

This section focusses primarily on analyzing the *mitstdcells* and *share* libraries. The section begins with a brief description of the *mitstdcells* cell library, *share* cell library, a selection of the Full-Custom **FC** cells and the **CB** cells used in the cell-based re-implementation of the IRRDP. The characteristics of these four collections of cells is then compared to gain insights on the constraints surrounding the mapping of datapath functions onto cell-libraries.

4.2.1 Standard Cell Library: *mitstdcells*

The basic *mitstdcells* library is composed of 91 cells. Each cell has a height of 12χ and the average cell width is $8\ \chi$.

At its inception, this library contained only three latch primitives: PDFF_SCAN, NLTCH_SCAN, PLTCH_CELL. As a result of experience gained during the course of the project, a NDFF_CELL, REG_CELL, and eleven multiplexed input latches were added to improve layout density and/or reduce wiring congestion. The base library contains all two-input logic functions, a set of AOI and OAI complex gates, a set of inverters (6 drive strengths) and a family of buffers (5 drive strengths). A comprehensive listing of the cells is included in Appendix F.

In addition to the aforementioned standard logic functions, several unique circuits were added to *mitstdcells* during the course of the MAP chip implementation: arbiter cells (7), gated clock drivers (two varieties, 4 drive strengths each), N-only pass-gate multiplexed input latches (9), tri-state drivers (4 drive strengths), and skewed threshold inverter (1).

As alluded to above, most gates have only one drive strength (equivalent to an INV_2). However, as a result of experience gained during the project, increased drive versions of eleven cells, predominantly AOI's and OAI's, were subsequently created.

4.2.2 Datapath Cell Library: *share*

The MAP chip contains five types of datapaths: integer execution (IU), memory execution (MU), floating point multiplier execution (FMULA), floating point divide-square-root execution (FDSQ) and data-cache datapath (DCDP). Each of these datapaths is composed of a combination of full-custom megacells and *share* library components. The *share* library is composed of 106 cells: 82 logic, 17 stitch and 7 filler cells. The logic cells in the *share* library are divided into 66 single-bit and 16 multi-bit instances. The single-bit cells are predominantly inverters (14), latches (9), and multiplexors (19). The multi-bit cells include the 64-bit half-multiplier array, five variations of the full 64-bit adder/subtracter, and a carry-look-ahead adder primitive used in the FDSQ.

The purpose of the *share* library is to supply a common set of datapath primitives (latches, pass-gate multiplexors, inverters and drivers) to allow the maximum possible reuse in the implementation of the chip datapaths. Each of the five datapaths were implemented with a *cell-based approach*. In addition to truly "shared" cells, each datapath module employs a set of module-specific *share* cells. The logical function of these cells was determined after a detailed analysis of the parent datapath. An example of this feature, is the **bool_unit** cell in the IU which implements all of the IU's boolean logic operations in a bit-sliced manner. The creation of module-specific logic circuits allows additional customization and density. Each cell is designed with a specifically envisioned application and physical location. At a detail level, the location of input and output ports/pins can be customized for these cells. While these cells are designed for a particular datapath, their specificity *does not* preclude them from being used in other modules. For example, the **sh_bit** primitive designed for the 64-bit barrel-shifter in the IU is also used in the alignment shifters of the FMULA and FDSQ. Similarly, the **bool_unit** primitive is reused in the MU.

Of the 82 logic *share* cells, 35 are truly "shared". The distribution of customized *share* cells is as follows: nine(9) in the IU, two (2) in the MU, eight (8) in the FMULA, fourteen (14) in the FDSQ, and fourteen (14) in the DCDP. As the MU implements several IU operations, many of the IU "custom" *share* cells are reused, accounting for the lower total MU specific cells. A comprehensive listing of the share cells and their primary modules is detailed in Appendix F.

4.2.3 Full Custom Cells: FC

A collection of the full-custom cells was also analyzed to provide a reference for comparison with the *mitstdcells* and *share* libraries. A total of 50 cells were examined. This represents a fraction of the total full-custom cells employed in the map chip. For example the IRRDP, alone, contains 80 unique instances. The cells were selected from a wide variety of full-custom blocks: eighteen (18) from the Instruction Fetch Pipeline stage (IFU); twenty-three (23) from the ram array building blocks; and nine (9) from the IRRDP. The selected cells span a broad collection of functions including counter primitives, register bit-cells, pipeline latches, multiplexors, and drivers.

4.2.4 Cell-Based Implementation Primitives: CB

Eight cells were create as the basis for the cell-based re-implementation of the IRRDP. Six of these cells were new versions of cells selected from the original 80 IRRDP full-custom primitives. The remaining two are "align-to-grid" implementations of a 2-input n-only pass-gate multiplexor and a 4-input n-only pass-gate multiplexor. A more detailed discussion of the IRRDP is located in Chapter 5. The characteristics of these cells are summarized in Table 4.1. The "% N" value is equal to total width of N-transistors (W_N) divided by the sum of the total widthes for both N and P transistors ($W_N + W_P$). The heading N:P is the ratio between W_N and W_P.

Library	T's/cell	area (χ^2)	% N-trans	N:P ratio
CB_OPSZDNREG1_1	13	140	69	1:0.45
CB_RFDCELL	7	56	80	1:0.25
CB_IRFDCELLTOP	9	154	78	1:0.28
CB_DPLTCH	7	126	45	1:1.23
CB_IRFD_IWBO	24	531	37	1:1.70
CB_OPSZD_BY_BUF	6	98	41	1:1.43
MUX4_NONLY	4	70	100	1:0.00
MUX2_NONLYC	2	42	100	1:0.00

Table 4.1: Target cells for Cell-based Re-implementation of the IRRDP

4.2.5 Comparison

The characteristics of *mitstdcells*, *share*, **FC** cells and **CB** are summarized in Table 4.2.

The **base** version of *mitstdcells* contains only the set of cells found in traditional CMOS standard cell libraries. The **extend** version of *mitstdcells* includes the arbiter cells, the gated clock drivers, the multiplexed input latches, and the REG_CELL primitives unique to the M-Machine project. The average T's/cell for the *share* library is artificially low as this library contains 14 different inverters. The average T's per cell excluding the inverters is 11. The **CB** cells are reported with and without the two multiplexor cells.

The results demonstrate the inherent differences in the characteristics of traditional CMOS standard cells and datapath cells. Traditional CMOS standard cells have balanced P-transistor and N-transistor circuits The *mitstdcells* library reflects this characteristic. Conversely the libraries created specifically for use in datapath regions have a significantly lower percentage of P-transistors as exemplified by the *share* cells. In general, datapath cells are inherently larger and encompass greater overall functionality as shown by the differences in both in χ^2 area and the average T's/cell.

Library	#cells	Ave T's/cell	Ave cell area (χ^2)	% N-trans	N:P ratio
mitstdcells-**base**	68	8.5	95	39.4	1:1.54
mitstdcells-**extend**	91	14.8	173	39.2	1:1.55
share - bit	66	9.0	149	48.9	1:1.04
FC	50	13.0	266	43.0	1:1.33
CB-w/ mux	8	9.0	152	44.4	1:1.25
CB-w/o muxs	6	11.0	184	42.0	1:1.36

Table 4.2: Summary of Cell Library Properties

4.3 Analysis

This section discusses the constraints, identified in this research, that limit the area efficiency and performance of cell based implementations of full-custom datapath functions.

4.3.1 Constraints

As shown in Table 4.2, traditional CMOS standard cells and datapath cells have very different characteristics. Traditional CMOS standard cells are balanced P-transistor and N-transistor circuits, while the area of full-custom datapath cells is dominated by N-transistors. Traditional CMOS standard cells are generally smaller and encompass less functionality. Each of these characteristics has implications toward the effectiveness of mapping datapath functions onto traditional CMOS standard cells.

Specifically, this section discusses the constraints imposed by the following differences between traditional CMOS standard cells and full-custom datapath cells: N-transistor and P-transistor ratio, average transistors per cell, grid alignment, diode contacts and the use of N-only pass-gates.

Sources of Inefficiency: N/P ratio

The circuits employed in most traditional CMOS standard cells are designed to have balanced rise and fall times. Since the carrier mobility of N-type devices is more than twice the carrier mobility of P-type devices, this results in circuits with a 1:2 ratio of N-type to P-type device sizes. The structure of standard cells reflects this design goal as, generally, the N-well occupies more than half the cell area. This distribution of N-transistor and P-transistor areas is a handicap when traditional CMOS standard cells are used to implement datapath functions.

Specifically, there are two main penalties:

1. The partition of cell area in balanced CMOS standard cells restricts the size and amount of N-transistors that can be implemented. The smaller useful area for N-transistors increases the total number of cells needed to emulate a

datapath function as the included P-transistors do not contribute effectively to the layout efficiency. The resulting balanced circuit style mappings of the original N-dominated datapath circuits result in area inefficiency.

2. The balanced circuit style mappings of datapath circuits also result in a timing performance penalty as additional delay is incurred due to the loading from the originally unrequired P-transistors.

The real area cost of this characteristic varies based on both process parameters and circuit styles. However, a simple "bound" for the area growth is the ratio of normalized quantities:

$$W_{N(SC)} = \text{Average Width of a N-transistor in a standard cell}$$
$$W_{P(SC)} = \text{Average Width of a P-transistor in a standard cell}$$
$$W_{N(DP)} = \text{Average Width of a N-transistor in a datapath cell}$$
$$W_{P(DP)} = \text{Average Width of a P-transistor in a datapath cell}$$
$$\text{SCALE} = \mathrm{W}_{N(SC)} \ / \ \mathrm{W}_{N(DP)}$$
$$\text{area}_{GROWTH} = SCALE\frac{(W_{N(SC)}+W_{P(SC)})}{(W_{N(DP)}+W_{P(DP)})}$$

For the MAP process and circuit sizing area$_{GROWTH}$ ranges from 10%-20% based the relative N:P ratios of the specific circuits.

Also, most traditional standard cell libraries do not contain skewed-threshold inverters. Therefore, standard cell circuits incur additional delay when interfaced to the N-dominated datapath cells. As mentioned earlier vin this chapter, unlike most standard cell libraries, the *mitstdcells* library does contain a skewed-threshold inverter to improve matching with datapath cells. This cell is specifically used in the **ASIC+** re-implementation of the ADDSUB and is responsible for a 2.9ns improvement in the critical timing path.

Sources of Inefficiency: Grid Alignment

Both the traditional ASIC and the proposed cell-based design approaches rely on an automated place and route tool to position cells and make the final signal connections.

The automated place and route tool requires input/output ports for each cell be aligned to the routing grid. In addition, place and route tools require that all edge geometries be "on-grid" as well. Both these requirement exists to reduce the complexity of the placement and routing problem by abstracting the routing characteristics of each cell and restricting the number of valid locations seen by the tool. This compatability requirement results in an inherent handicap in layout density for the both the traditional ASIC and **CB** cell primitives.

The reduction of useful cell area is shown graphically in Figure 4-1.

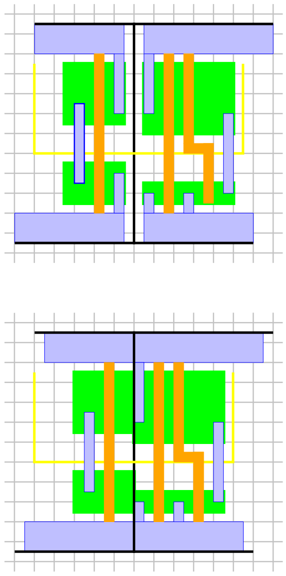


Figure 4-1: *Cell Area Overhead: Alignment to Grid*

The topmost cells in Figure 4-1 are constructed to be compatible with place and route tools and thus are aligned to the grid and have a design-rule clearance along the edge of interest. The bottom set of cells demonstrates the potential area savings when cell geometry can exist up to the edge of the cell.

The depicted situation represents only abutment in the X-direction, the actual potential for lost usable area extends also in the Y-directions. Ultimately the maximum cost is one routing track (or grid) in both the X and Y direction. Note, since the datapath cells are a fixed 14 χ height and average 11 χ width, this represents a reduction of **16%** usable cell area for gridded datapath cells in comparison to full-custom cells.

As the *mitstdcells* have a fixed 12 χ and an average 8χ width, the potential penalty is even larger - 21%.

Sources of Inefficiency: Diode Contacts

The requirement of diode contacts on silicon gates contributes a small additional inefficiency to the layout density of traditional standard cells.

Most contemporary fabrication processes are planarized. The planarization process is a combination of chemical and mechanical polishing operations. As a result diode contacts are required on each silicon gate to prevent the induced static voltages from punching through the thin-oxide gate.

As, on average, traditional standard cells encompass less functionality than datapath cells, more diode contacts are required as addtional input/output ports are created. In a larger cell these ports would be subsumed in the normal internal interconnect. The diode contacts are needed because the placement of the cells is determined by the automated tool and no guarantees exist for selection of metal layers the routing tool will use to connect ports. In the full-custom case, the designer knows that an output driven from the drain of one transistor can be used directly to connect to the gate of another, implicitly forming the required diode contact. The automated placement and routing eliminates this advantage.

The penalty due to extra diode contacts depends on the design rules of the selected fabrication process. For the MAP chip, the penalty is $1\chi^2$ of routing resource per extra contact and $1.2\chi^2$ of device area per extra contact. As the typical area of cells in the MAP libraries ranges from $98\chi^2$ to $150\chi^2$, extra diode contacts represent approximately a 1% per instance overhead.

N-only pass gate logic

An acknowledged source of the inefficiency experienced in traditional standard cell based designs is the use of static logic multiplexors in place of the N-only pass gate implementations commonly found in datapath circuits. A simple approach would estimate that a **4x** cost in area is accrued for this choice as a 2-input multiplexor can be implemented as either 2 transistors or statically in an AOI22 with 8 transistors. However, the actual penalty is actually less than **2x** as shown in Table 4.3.

CELL	T's	λ^2	χ^2	ρ_t (λ^2/T)	ρ_t (χ^2/T)
MUX2_NONLYC	2	2177	42	1088	21.0
AOI22	8	3110	60	389	7.5
MUX4_NONLY	4	3528	70	882	17.5
AOI2222	16	6843	132	428	8.3

Table 4.3: Multiplexor Implementations

Two main reasons account for the difference between perceived area cost and actual area cost.

1. Minimum N-well rules. In order to be compatible with other automatically placed cells, the multiplexor layouts need to have a minimum amount of N-well despite the absence of P-transistors in the circuit. As shown in Table 4.3, the N-only pass gate cells are much less area efficient than the AOI cells.

2. Pin-limitations. The width of the N-only pass-gate multiplexor cells is limited by the control lines needed to drive the circuit. For example, in the case of the MUX2_NONLYC, even though the transistors can be folded to occupy a smaller area, the width of the cell must be at least 3 χ to allow both selects pins and the output pin to be accessible to the place and route tool.

4.3.2 Efficiency Metrics (η)

Traditionally, layout efficiency has been measured as a function of cell area utilization. However, experience during the M-Machine project demonstrates that this is not the only valid measure of layout efficiency. In fact, an inverse relationship generally exists between number of devices within a cell and the cell area utilization. As a result a new model for layout efficiency is needed.

Figure 4-2 shows demonstrates the complicated relationship between utilization and number of transistors within a cell. With cells with lowgate density (ρ_{gate}) there is a clear inverse relationship between utilization of active area and complexity of the cell. However, a bifurcation occurs as the ρ_{gate} increases as there are *two* equally valid sources of increased active area density - tightly compacted cells with a large number of transistors (examples include AOI's, multiplexors, register bit-cells), and steadily increasing sized buffers, inverters and drivers.

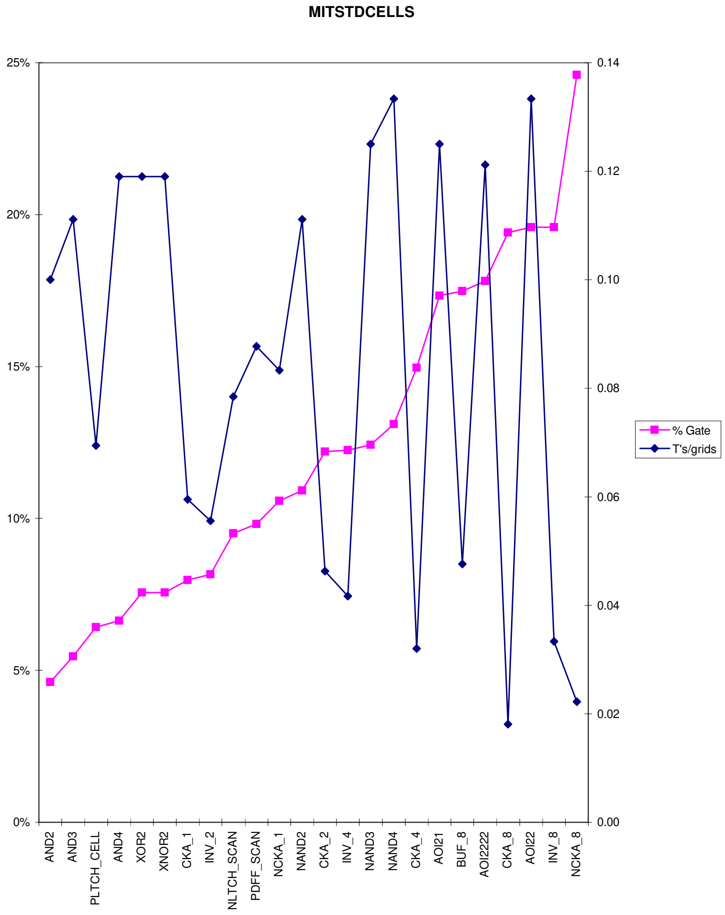


Figure 4-2: *Transistor Frequency and Density for Representative Standard Cells*

The identified relationship is valid amongst standard cells, datapath cells, and full-custom cells. Figure 4-3 shows the expected inverse relationship between buffers and complex gates used in the full-custom ICDRAM. The one anomoly in this plot is the ram BIT_CELL. The BIT_CELL has the highest possible ρ_t for the process but, as the BIT_CELL does not use large transistors and does require a great deal of interconnect, it has only has moderate ρ_{gate}. Additional data plots can be found in Appendix A.

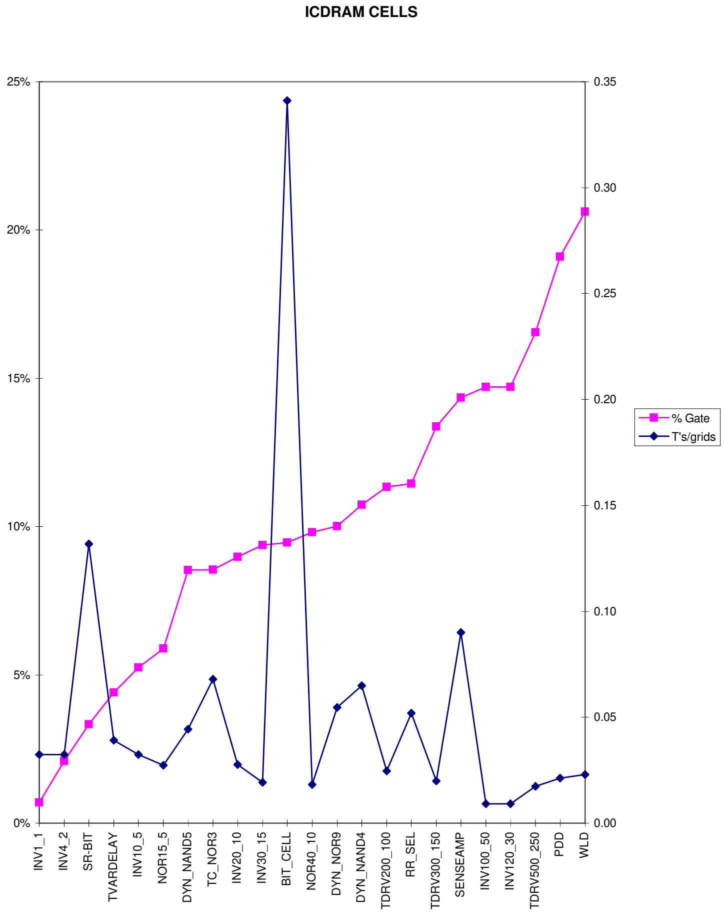


Figure 4-3: *Transistor Frequency and Density for Representative ICDRAM Cells*

As Figure 4-2 and Figure 4-3 show, there is not an single unequivical metric for judging area efficiency. A scatter plot of ρ_t versus ρ_{gate} was made of 80 cells and is shown in Figure 4-4. Two bounds for unequivically "good" layout can be easily identified - the $\rho_t = 0.34$ and $\rho_{gate} = 0.09$ point representing the BIT_CELL and the off-axis point of $\rho_t = 0$ and $\rho_{gate} = 0.32$ representing the highest area utilization cell possible for this process.

If we assume that both these points represent the limits of their respective merit metric than we can establish the following relationships for this process:

$$\rho_{tMAX} = 0.34$$

$$\rho_{gate} = 0.32$$

$$\eta_{layout} = \frac{\rho_{t-n0}}{\rho_{tMAX}} + \frac{\rho_{gate-n0}}{\rho_{gate-MAX}}$$

Note, ρ_{t-n0} is transistor density value for a given cell n0, similarly $\rho_{gate-n0}$ is the corresponding gate density value for that cell. Thus η_{layout} for the BIT_CELL is 0.28 + 1.0 = 1.28. The η_{layout} for the MAX_INV is 1.0 + 0.0 = 1.0. This result implies that the BIT_CELL is more layout efficient than the MAX_INV, which is principle **true**. The BIT_CELL contains geometries which violate the normal designed rules and require explicit waivers before processing. However, a better equation for η_{layout} adds two co-efficients to provide a method of fitting the equation to a given set of process design rules.

$$\eta_{layout} = \alpha\frac{\rho_{t-n0}}{\rho_{tMAX}} + \beta\frac{\rho_{gate-n0}}{\rho_{gate-MAX}}$$

An initial choice for α is 0.71 while and initial choice of β could be 1.0. These selections result in η_{layout} equal to "1" for both cases. The value of α could be tuned upward slightly to reflect the BIT_CELL's use of non-standard design rules.

This chapter reviewed several cell libraries from the MAP design database. The characterisitics of this libraries lead to insights about the fundamental constraints of mapping datapath functions with traditional standard cells. The area overheads of N:P ratio, geometry alignment to grid and extra diode contacts can result in between a 10% and 41% growth in area between the original **FC** datapath function and the **CB** implementation.

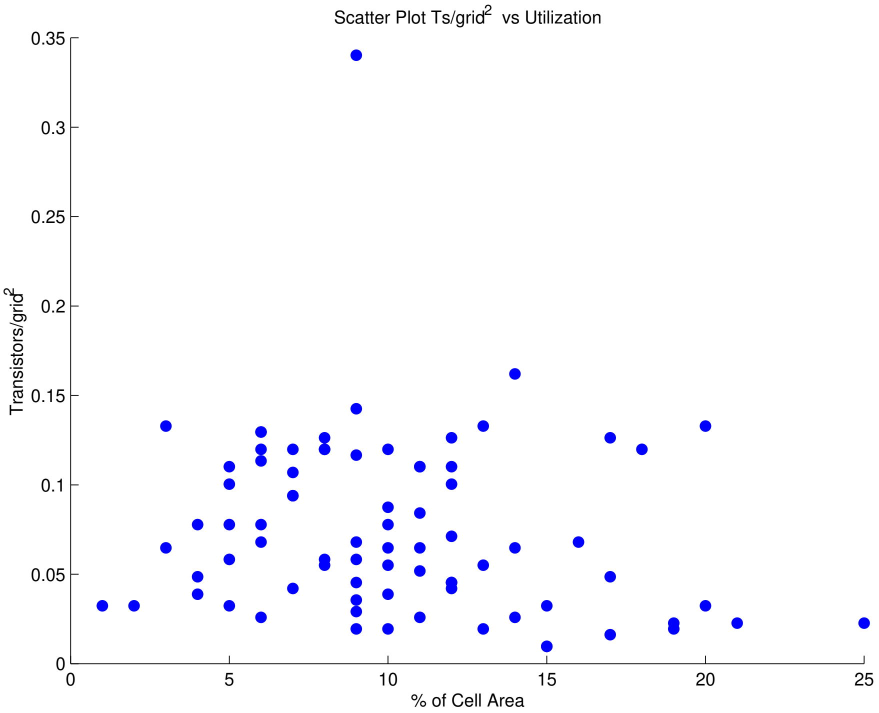


Figure 4-4: *Scatter Plot of ρ_t versus ρ_{gate}*

Chapter 5

RESULTS: IRRDP

5.1 Overview

The IRRDP was selected as one of the experimental benchmarks due to the high density and regularity of the full-custom implementation. As a target, it is intended to provide insight into the "best-case" limit on the performance of cell-based approaches.

The Integer Register Read Datapath (IRRDP) of the MAP was re-implemented in four different methodologies (**CB**, **CB(auto)**, **ASIC+** and **ASIC**) to explore the area costs, performance penalties and constraints of a cell-based approach applied to a relatively regular structure.

The re-implementation of the IRRDP consisted of seven phases:

1. Reference Verilog model creation. A new reference structural verilog model for the IRRDP was created. This model was assembled with a "bottom-up" approach. New structural verilog models were first written for constituent cells, subblocks, and bit-slices. These models were then integrated together.

2. Cell re-implementation. Seven (7) cells from the original 80 unique cells in the full-custom implementation were selected for re-implementation. A *new physical layout* compatible with the CELL3 place and route tool, was generated for *each* selected cell. These cells, along with a small set of modified *mitstdcells*, were employed in both the **CB** and **CB(auto)** implementations.

3. Logic synthesis. The reference structural verilog model was synthesized *twice* using the Cadence Synergy^{TM} tool. The first synthesis run employed an extended version of the *mitstdcells* library as a target. This version contains multiplexed-input latch primitives and is denoted as **ASIC+**. The second synthesis run employed the base version of the *mitstdcells* library as a target and is denoted as **ASIC**.

4. Baseline creation. The area and timing performance of the original full-custom design was determined. The pre-layout timing performance of each of the re-implemented schematics was simulated with the Avant! HSPICE^{TM} circuit simulator.

5. Floorplan creation. A floorplan for each case was created using the Cadence Preview^{TM} tool. In the case of **CB**, the detailed tiling and explicit placement of the constituent cells is performed in this phase.

6. Place and Route. Each design was placed and routed using the Cadence Cell3 Ensemble^{TM} tool. After successful routing, the parasitic capacitances are extracted.

7. Analysis and Evaluation. Each design was reviewed for area and each circuit was re-simulated with the back-annotated parasitic capacitance values.

The **CB** style resulted in an implementation that was **1.64x** larger than the original IRRDP. This design was *slightly slower* than the original full-custom design as the critical timing path incurred **11%** additional delay.

The **CB(auto)** style resulted in an implementation that was **1.66x** larger than the original IRRDP. This design was *slower* than the original full-custom design as the critical timing path incurred **21%** additional delay.

Both the **ASIC+** and **ASIC** styles resulted in **very inefficient** implementations of the IRRDP. Both implementations required **14.5x** the area of the original design. The timing performance of both designs was also **poor**. The **ASIC+** approach resulted in a critical timing path with **121%** longer delay. The **ASIC** approach resulted in a similar worst path as the critical timing path had **123%** longer delay.

The second section of this chapter provides a brief functional description of the IRRDP. The third section briefly describes the details of the constituent cells. The fourth section details the results and the final section of this chapter provides analysis of the data.

5.2 Functional Description

The MAP chip contains three identical integer execution units (IU's). Each unit contains an execution pipeline composed of five stages:

1. Instruction Fetch (IF) - Responsible for fetching and unpacking instructions, managing the ICache and managing the Instruction Pointers (IPs).

2. Register Read (RR) - Responsible for decoding instructions, performing sign extension on immediates, providing source operands, and managing the Register Files and Scoreboards.

3. Synchronization (SZ) - Responsible for selecting instructions to issue, arbitrating for the Network resource, and generating the thread select (TSEL) control signals.

4. Execution (EX) - Responsible for executing operations, detecting exceptions, arbitrating for CSW and MSW resources, stalling the pipeline, validating and invalidating registers, and creating Error-values.

5. Writeback (WB) - Responsible for writing results to the local register file, for writing results to the CSW for remote destinations and for updating the scoreboard.

An overview of the M-Machine pipeline is shown in Figure. 5-1.

In the M-Machine architecture [24], the integer register read stage performs four distinct functions: operation pre-decoding, provision of source operands, injection

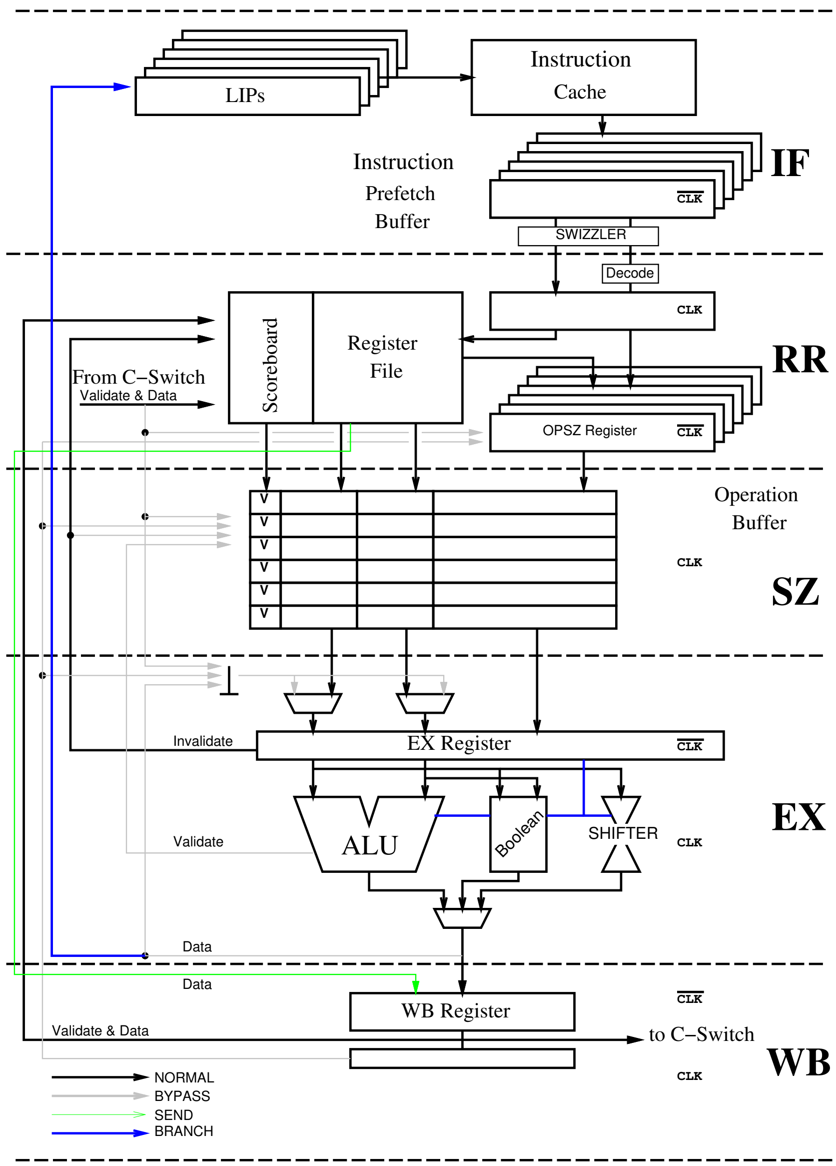


Figure 5-1: *The MAP Pipeline*

of constant values, scoreboard and data updates. The integer register read stage is divided into two modules, control (IRR_CT) and datapath (IRRDP).

The IRRDP is a very regular datapath. It consists of a 7-ported 75-entry register file, a set of operand selection multiplexors and a 6-entry pipeline register stage. While the IRRDP is predominantly bit-sliced, it contains three irregular components: 13 1-bit wide registers for condition code values, 1 62-bit path and a 18-bit path for the insertion of constant values.

In the actual implementation of the MAP chip, this datapath is a full-custom block consisting of 68361 transistors. The resulting layout occupies an area of $2.5mm^2$ (1864.8 μm height by 1344.6 μm width). The embedded power distribution network contributes 12.5% overhead to the 1864.8 μm height. Each 8-bit slice requires an additional 1-bit slice of power bussing, routing, and well/substrate contacts to interface the internal power and clock grid to the global MAP chip grid. In this study, a *reduced* height of 1612.8 μm, representing only the actual databits of the full-custom design, is used as the comparison reference for the cell-based implementation. Therefore, for the purposes of this study, the effective area of the IRRDP is $2.17mm^2$.

The IRRDP is composed of four subblocks:

IRFD, a 75-entry register file organized as five (5) 14-entry files and one (1) 5-entry file. The register file has three writers - integer execution stage (iwbD_v1), memory execution stage (mwbD_v1) and the cluster switch (cswD).

CFG_OPSZD, a 18-entry pipeline stage organized as six (6) independent groups of three latches. All of these latches are neglatches.

IMUX, a set of multiplexors allowing each of the two input operands to originate from one of seven different sources including 1-bit condition code values (**gccv1**,**gccv2**, **fwbCv_v1**) and 18-bit constant values (**k**).

MISCELLANEOUS, several miscellaneous components complete the implementation of the IRRDP including 1-bit, and 64-bit latches, buffers and tri-state drivers.

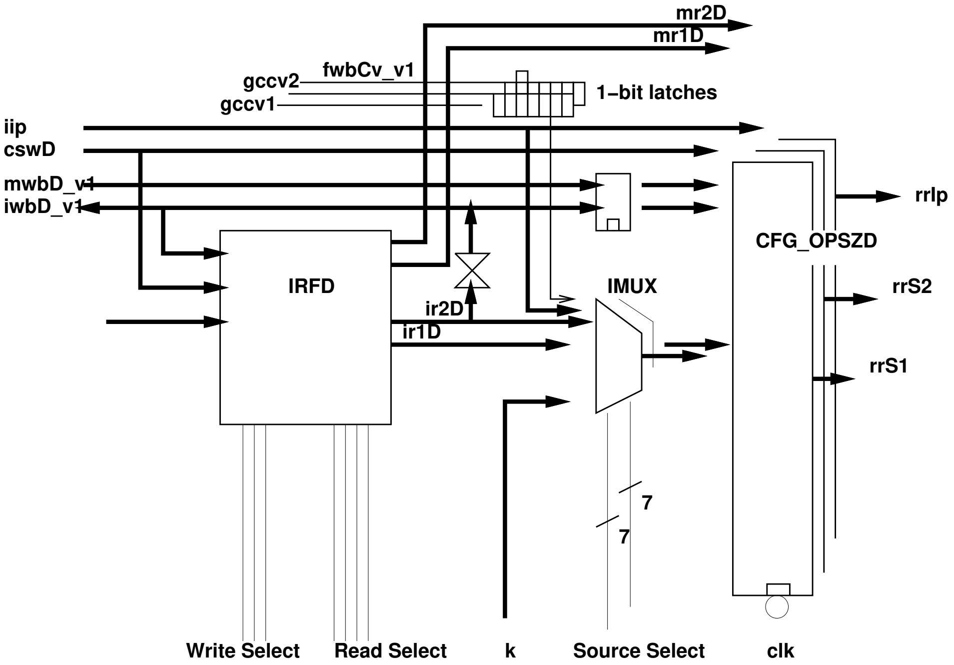


Figure 5-2: *IRRDP Block Diagram*

A block diagram of the IRRDP logic is provided in Figure 5-2.

The first two subblocks, IRFD and CFG_OPSZD, are the dominant contributors to the module area. The IRFD occupies 59% of the module area while the CFG_OPSZD accounts for 27% of the module area.

5.3 Circuits

The full-custom design of the IRRDP consists of eighty (80) unique cells. Many of these are location specific implementations of a basic set of functions. Nine (9) basic functions exist in the IRRDP: register bit (RFDCELL*)[1] , register read/write interface circuitry (IRFDCELLTOP), input and output drivers (IRFD_BUF*), tristate drivers (IRFD_IWB*), latches (DPLTCH_*), multiplexors (IMUX*_1*), output buffers (OPSZD_BY_BUF), pipeline latches with multiplexed inputs (OPSZD-

[1] The asterix denotes both multiple versions (different sizes) and multiple variations (same size, but different layout).

NREG*) and stitch cells (*_SV) The functional families that dominate the area of the IRRDP are the register file components (RFDCELL, IRFDCELLTOP) and the pipeline latches (OPSZDNREG*).

5.3.1 RFDCELL

The RFDCELL is the first basic building block of the register file. It's schematic is shown in Appendix D. It consists of a static latch with two write ports and one read port.

5.3.2 IRFDCELLTOP

The IRFDCELLTOP is the second basic building block of the register file. It's schematic is shown in Appendix D. It encompasses the interface circuits necessary to interconnect a bank of register bits.

5.3.3 IRFD_6, IRFD_5

The IRFD_6 and IRFD_5 are the basic schematic and physical units of the IRFD. The IRFD_6 and IRFD_5 are similar in implementation The IRFD_6 is composed of one IRFDCELLTOP and six RFDCELL bits while the IRFD_5 contains only five RFDCELL bits. The IRFD is composed of a two-dimensional tiling of IRFD_6 and IRFD_5 units. The tiling consists of 64 instances of either the IRFD_6 or the IRFD_5 in the "word-line" direction to form either a five or six register bank, each 64-bits in width. These "banks" represents a single logic register name. The M-Machine architecture [24] specifies 16 programmer visible registers with i0 and i1 representing the "zero" and the "instruction pointer" (IP) respectively. The 14 actual registers are crafted by replicating the IRFD_* structures in the "bit-line" direction. The complete IRFD is an array composed of 64 x 5 IRFD_6's and 64x9 IRFD_5's. Figure 5-3 shows a conceptual drawing of the IRFD.

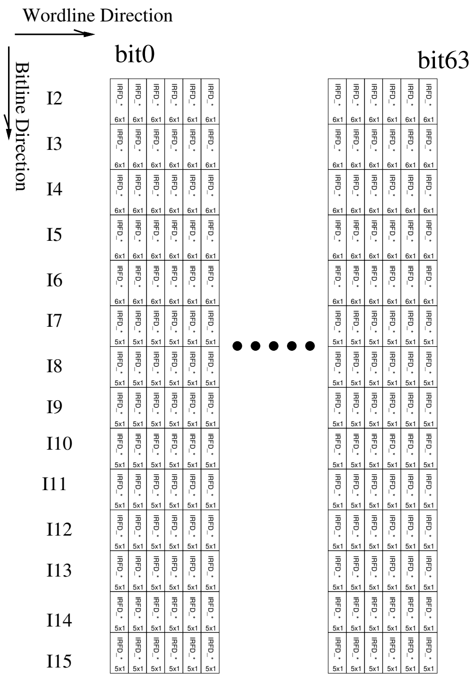


Figure 5-3: *IRFD Tiling*

5.3.4 OPSZDNREG - Family of Pipeline Registers

The M-Machine Architecture [24] can simultaneously support the state of six different instruction streams (threads). Each thread requires three state registers: instruction pointer (IP), first source operand (S1) and second source operand (S2). The M-Machine architecture supports full bypassing (forwarding) of updated values throughout the pipeline. Each pipeline register has upto seven potential sources for its value: current input value, forwarded data values from the integer unit, memory unit, cluster switch, forwarded condition code values from floating point unit, the cluster switch or the global condition-code network.

The schematic for the basic OPSZDNREG1_1 is shown in Appendix D. The basic cell is a staticized recirculating neglatch with a single input select. The explicit recirculation (hold) path allows updating decisions to be made late in the cycle as the previous value is maintained and can be restored as necessary. The full-custom implementation employs four versions of this cell: OPSZDNREG2_1, OPSZDNREG4_1, OPSZDNREG7_1. The difference in these versions is the number of different potential inputs. As a result each of the variations can be viewed as an OPSZDNREG1_1 with either one (1), three (3), or six (6) additional multiplexed inputs.

In order to reduce the re-implementation effort, the OPSZDNREG1_1 and two N-only pass-gate multiplexor primitives, MUX2_NONLYC and MUX4_NONLY, were created in place of the four unique original cells. All required OPSZDNREG#_1 cells were synthesized from a combination of these three primitives in the **CB** and **CB(auto)** designs.

5.3.5 IRFD_IWB* - Family of Tristate Drivers

The M-Machine microarchitecture requires a bidirectional path between the IRFD and the IU. During normal operation the IU writes results via the iwbD_v1 bus into the IRFD. However, during the execution of a SEND operation, the IRFD drives data values from the register file directly to the input of the IU. As a result, the iwbD_v1 bus is driven by two tristate buffers - IRFD_IWBO and IRFD_IWBI. The IRFD_IWBO circuit is detailed in Appendix D. Both the IRFD_IWBO and IRFD_IWBI differ only in the detailed device sizings. The basic circuit style of IRFD_IWBO circuit is shown in Figure 5-4.

A similarly sized tri-state driver from the *share* library was used to emulate this cell in both the **CB** and **CB(auto)** designs.

5.3.6 IMUX* - Family of Multiplexors

A set of multiplexors and associated buffers is located between the IRFD and the pipeline latches (CFG_OPSZD). In the **CB** and **CB(auto)** designs, these multiplexor functions are implemented via a combination of the MUX2_NONLYC and MUX4_NONLY primitives and inverters modified from the *mitstdcells* library.

5.3.7 Miscellaneous Cells

In addition to the previously discussed circuits, the IRRDP also employs a large variety of inverters, buffers and latches. For both the **CB** and **CB(auto)** designs, an additional primitive cell, CB_DPLTCH, was implemented to emulate the DPLTCH1_* in the original full-custom design. The collection of required inverters and buffers including IRFD_BUFCI, IRFD_BUFMI, IRFD_BUFMO, and IRFD_BUFIRO, were emulated using inverters modified from *mitstdcells*.

5.4 Results

This section summarizes the experimental results in both area efficiency and timing performance for the four re-implementations of the IRRDP. In each subsection, the baseline parameters for the original IRRDP are provided. All results from the four re-designs are presented as normalized quantities relative to the baseline parameters. The supporting background data for all these results can be found in Appendix A.

5.4.1 Area

The area efficiency of the four design styles was explored at three levels: basic cell, bit-slices, and full-design.

Cells

A structural Verilog model for each of the leafs cells was constructed. These structural Verilog models are *shared* by all four implementation styles.

Two approaches were taken to implement leaf cells. A subset of the cells was selected for new physical design. Other functions, for example the CB_OPSZDNREG7_1 block, were create using a composition of new primitives. The new primitives are:

- CB_RFDCELL
- CB_IRFDCELLTOP
- CB_OPSZDNREG1_1
- CB_DPLTCH
- CB_OPSZDBY_BUF

- MUX2_NONLYC
- MUX4_NONLY

The IRFD_IWBO was emulated using the **driver** cell from the *share* library resulting in a small additional overhead as the **driver** is composed of larger devices. The IRFD_BUFCI, IRFD_BUFMI, IRFD_BUFMO, IRFD_BUFIRO buffers were emulated using slightly modified versions of existing INV_2, INV_4, and INV_8 standard cells from *mitstdcells*. The slight modification required for these cells was *stretching* their height from 12χ to 14χ to match the standard datapath format.

In both the **ASIC+** and **ASIC** approaches, the logic for each of the above listed cells was synthesized from the common structural Verilog models.

Table 5.1 lists the baseline parameters for the original full-custom cells. Except for the IRFD_BUFCI and IRFD_BUFMI, all the cells fall into either the Array density or Datapath density range reported by Dally [21].

CELL	T's	λ^2	χ^2	η_t (λ^2/T)	η_t (χ^2/T)	ρ_t
RFDCELL	7	2540	49	363	7.0	9%
IRFDCELLTOP	9	6532	126	726	14.0	12%
OPSZDNREG1_1	13	5806	112	446	8.6	9%
OPSZDNREG4_1	16	8709	168	544	10.5	9%
OPSZDNREG7_1	19	11209	216	591	11.4	9%
IRFD_BUFCI	4	3629	70	907	17.5	15%
IRFD_BUFMI	4	3639	70	907	17.5	13%
IRFD_BUFMO	8	5080	98	632	12.2	23%
IRFD_BUFIRO	8	5080	98	632	12.2	9%
DPLTCH1	7	7258	101	746	14.4	16%

Table 5.1: Full-Custom Implementation Baseline - Cells

Table 5.2 displays the normalized experimental results. In general, a clear efficiency ordering of **CB**, **CB(auto)**, **ASIC+**, and **ASIC** exists. However, IRFD_BUFMO and IRFD_BUFIRO are anomolous. A single, "compromise", set of inverters from *mitstdcells* was used to implement both cells. The size of the compromise was more appropriate for the larger IRFD_BUFMO and as a result, its use negatively skewed the metrics for the smaller IRFD_BUFIRO.

CELL	FC	CB	CB(auto)	ASIC+	ASIC
RFDCELL	1.00	1.14	1.43	3.43	4.29
IRFDCELLTOP	1.00	1.11	1.22	5.11	11.54
OPSZDNREG1_1	1.00	1.25	1.50	5.63	7.00
OPSZDNREG4_1	1.00	1.42	1.50	4.83	8.08
OPSZDNREG7_1	1.00	1.17	1.36	6.99	8.16
IRFD_BUFCI	1.00	1.60	1.60	2.00	2.00
IRFD_BUFMI	1.00	1.60	1.60	2.00	2.00
IRFD_BUFMO	1.00	2.00	2.00	1.29	1.29
IRFD_BUFIRO	1.00	2.00	2.00	1.29	1.29
DPLTCH1	1.00	1.25	1.25	1.38	1.38

Table 5.2: Summary - Normalized Area

Bit-slices

Three bit-slices were implemented from the previously described cells: the IRFD_BIT, representing a single bit path through the register file; the CFG_OPSZD_BIT, representing a single bit path through the pipeline registers; and IRRDP_BIT, representing a full bit-slice of the complete design.

A structural Verilog model for each of these bit-slices was constructed. Once again, these structural Verilog models were *shared* by all four implementation styles.

Two approaches were taken to implement the bit-slices. In the **CB** approach, the locations for both the data I/O and control I/O pins were specified explicitly. Then each leaf cell was manually instantiated and tiled to its desired location.

In the **CB(auto)** and **ASIC** approaches, only the locations for the data I/O pins were specified explicitly. The QPLACE^{TM} tool was employed to optimize leaf cell placement via its simulated annealing algorithm.

In both the **ASIC+** and **ASIC** approaches, the logic for each of the bit-slices was synthesized from the common structural Verilog models. In floorplanning, both ASIC approaches required a single bit cell height of **63**χ. The **CB**, and **CB(auto)** layouts required one additional track resulting in a 15χ height. In comparison, the original **FC** module requires only 14χ.

Table 5.3 lists the baseline parameters for the original bit-slices. All three fall into either the Array density or Datapath density range reported by Dally [21].

SLICE	T's	Kλ^2	χ^2	η_t (λ^2/T)	η_t (χ^2/T)
IRFD_BIT	675	317.7	6128	471	9.0
CFG_OPSZD_BIT	303	159.3	3077	525	10.2
IRRDP_BIT	1068	542.1	10458	507	9.8

Table 5.3: Full-Custom Implementation Baseline - Bit-slices

Table 5.4 displays the normalized experimental results. There is a clear efficiency ordering between the cell-based approaches and the ASIC approaches. A surprising result was the significant difference between the **ASIC** and **ASIC+** area efficiency in the case of the CFG_OPSZD_BIT. A possible explanation for this result is that during the synthesis of the reported **ASIC** implementation, logic based multiplexors with "don't care" cases were enabled in place of tristate based multiplexors. These "don't care" states may have been exploited by the logic synthesis tool to optimize and further reduce the logic. The N-transistor dominated nature of the IRFD_BIT circuit contributed to the poor mapping of both the **ASIC** and **ASIC+** approaches. Part of the explanation of the relatively poor performance of both the **CB** and **CB(auto)** styles with the CFG_OPSZD_BIT is the choice to exclusively deploy the CB_OPSZDNREG7_1 cell instead of a mix of CB_OPSZDNREG7_1 and CB_OPSZDNREG4_1 cells. The CB_OPSZDNREGD7_1 is 3χ wider than the CB_OPSZDNREG4_1 cell. Also, the preliminary layout of the CB_OPSZDNREG7_1 needed to be expanded by 3χ to enable the router to access every port. The resulting total overhead is 72χ (18% of the total 391 χ required for the CFG_OPSZD_BIT).

SLICE	FC	CB	CB(auto)	ASIC+	ASIC
IRFD_BIT	1.00	1.33	1.63	7.23	7.23
CFG_OPSZD_BIT	1.00	1.91	1.91	5.37	3.67
IRRDP_BIT	1.00	1.64	1.68	5.43	5.25

Table 5.4: Summary - Normalized Area

FULL IRRDP

A structural Verilog model for the full design was constructed. This structural Verilog model is *shared* by all four implementation styles.

As in the case of the bit-sliced designs, two approaches were taken to implement the full design. In the CB approach, the locations for both the data I/O and control I/O pins were specified explicitly. Then each leaf cell was manually instantiated and tiled to its desired location.

As before, in the **CB(auto)** and **ASIC** approaches, only the locations for data I/O pins were specified explicitly. The QPLACE^{TM} tool was employed optimize leaf cell placement via its simulated annealing algorithm.

In both the **ASIC+** and **ASIC** approaches, the logic for each version of the IRRDP was synthesized from the common structural Verilog model. As with the bit-slice, both ASIC approaches required **63**χ per bit in height. The **CB** and **CB(auto)** layouts each required 15χ/ bit. In comparison, the original full-custom layout required 14χ/bit. Table 5.5 lists the baseline parameters for the original full-custom version of the IRRDP. The original design has high area efficiency and is within the datapath density range reported by Dally [21].

MODULE	T's	Mλ^2	χ^2	η_t (λ^2/T)	η_t (χ^2/T)
IRRDP	68361	34.7	669312	508	9.8

Table 5.5: Full-Custom Implementation Baseline - IRRDP

Table 5.6 displays the normalized experimental results. Both cell-based approaches are clearly more area efficient than the ASIC approaches. The difference between the **CB** and **CB(auto)** implementations is the additional vertical routing track per CB_IRFDCELLTOP (14 total) required in the CB(auto) case to enable violation free routing after cell placement. The layouts generated from both the **ASIC+** and **ASIC** approaches were very inefficient. Both of these designs were **14.5x** larger than the original. Much more surprising, both the designs were **3x** larger than their own bit-slice cases.

Clearly the placement tool was not able to automatically extract the bit-slice regularity from the full design. An irregularly-placed regular datapath results in substantial congestion. In both cases, the row utilization of the final placed design was approximately 33%. Place and route flows regularly yield utilization in excess of 55%. The random control logic in the MAP chip was placed with utilizations is between 45% and 55%. Low row utilization is usually a sign of high areas of congestion.

Note, also in both cases the placement problem was larger than typically recommended for automatic placement tools. The **ASIC+** approach resulted in 30464 cells while the **ASIC** approach required 31065. Also, for reference, the cell-based approach results in 10375 cells.

MODULE	FC	CB	CB(auto)	ASIC+	ASIC
IRRDP	1.00	1.64	1.66	14.5	14.5

Table 5.6: Summary - Normalized Area

Even if it were possible to achieve 100% row utilization with the **ASIC+** and **ASIC** approaches, the net area occupied by the constituent cells, would still be **3.63x** and **3.65x** the **FC** baseline, respectively.

Other hybrid design assembly approaches also exist. For example, a fully automated placement of the CB cells was attempted (**CB(full-auto)**) with disappointing results. This design required **10.5x** the baseline **FC** area. While, an obvious improvement to both the **ASIC+** and **ASIC** methodologies would be to use the results from the bit-sliced placements as components and replicate the placement over the full 64-bits (**ASIC+(tiled)** and **ASIC(tiled)**), the resulting layout would still be atleast **5.25x** larger than the baseline.

The key result of the area efficiency experiments is that the **CB** approach was the only style to have near full-custom area efficiency.

5.4.2 Performance

As with the area efficiency, the timing perfromance of all four design styles was explored at three levels: basic cell, bit-slices, and full-design. All timing measurements were made at the typical-N, typical-P, 3.0V 100° C (TTL) process corner. Also the nominal output load capacitance was 0.20pF.

Timing Paths

Three timing paths were selected to represent the performance of the IRRDP. In all cases, results for reading/writing both a "1" and a "0" are reported.

End-to-end Delay (t_{PD}): This path represents the cumulative logic delay through the full design as shown in Figure 5-5. This is an *artificial* metric as this path is never valid in the actual operation of the design. Nevertheless, t_{PD} does provide one indication of the logic depth of the design.

Write-through (t_{crit}): In the IRRDP design, the actual operational critical path is the write-through path from the iwbD_v1 write port of the IRFD to the input port of the OPSZDNREG7_1 pipeline latch in the CFG_OPSZD block. The length of this path determines the maximum clock frequency of this stage. Figure 5-6 highlights this path.

Read Access (t_{access}): The final representative timing path selected for characterization was the read access path as shown in Figure 5-7. Note, the input read select signals were driven by a *mitstdcells* INV_32 identical to the actual operation.

The timing performance of the original IRRDP was characterized for each of the timing paths. The resulting data was converted to four different representations: nanoseconds (ns), τ_c, t_{INV}, and t_{NAND2}. Tables[5.7],[5.8],[5.9], and [5.10] present the baseline timing data for the full-custom IRRDP.

Path	Pre-layout (ns)		Post-layout (ns)	
	"1"	"0"	"1"	"0"
t_{PD}	4.60	4.50	5.30	5.10
t_{crit}	3.10	3.00	3.80	3.70
t_{access}	1.15	0.88	1.82	1.32

Table 5.7: Full-Custom Implementation Baseline (ns)

Path	Pre-layout (τ_C)		Post-layout (τ_C)	
	"1"	"0"	"1"	"0"
t_{PD}	287.5	281.2	331.2	318.8
t_{crit}	193.8	187.5	237.5	231.2
t_{access}	71.9	55.0	113.8	82.5

Table 5.8: Full-Custom Implementation Baseline (τ_C)

Timing Comparison

The following tables summarize the relative timing preformance of each implementation style. The tables are divided into two sets. Tables [5.11],[5.12] represent the pre-layout schematics. Tables [5.13],[5.14] reflect the post-layout schematics with back-annotated capacitances.

Pre-Layout: A HSPICE^{TM} netlist was generated from the bit-slice schematics for each of the four design styles. The cell-based implementions are 4% and 34% slower than the original design. This difference is caused by the selection of a subset of the 80 full-custom cells to reimplement as targets for the cell-based approach. In several

Path	Pre-layout (t_{INV})		Post-layout (t_{INV})	
	"1"	"0"	"1"	"0"
t_{PD}	56.1	54.9	64.6	62.2
t_{crit}	37.8	36.6	46.3	45.1
t_{access}	14.0	10.7	22.2	16.1

Table 5.9: Full-Custom Implementation Baseline (t_{INV})

Path	Pre-layout (t_{NAND2})		Post-layout (t_{NAND2})	
	"1"	"0"	"1"	"0"
t_{PD}	18.5	18.1	21.4	20.6
t_{crit}	12.5	12.1	15.3	14.9
t_{access}	4.6	3.5	7.3	5.3

Table 5.10: Full-Custom Implementation Baseline (t_{NAND2})

cases the limited sizes of the available inverter primitives (INV_2, INV_4, INV_8, INV_16, INV_32) does not cover the range of original sizes used in the design. In addition, the *share* library cells substituted for the IRFD_IWBO and the assembly of OPSZDNREG4_1 and OPSZDNREG2_1 and OPSZDNREG7_1 from a collection of OPSZDNREG1_1 and MUX2_NONLYC and MUX4_NONLY cells results in an incomplete match. *This is a clear example of cost and limits in the cell-based approach as the iterations expended tuning the exact sizes of component cells are not a part of this flow.*

Path	FC		CB		CB(auto)	
	1	0	1	0	1	0
t_{PD}	1.00	1.00	1.07	1.11	1.07	1.11
t_{crit}	1.00	1.00	1.10	1.03	1.10	1.03
t_{access}	1.00	1.00	1.04	1.14	1.28	1.34

Table 5.11: Summary - Normalized Timing Comparison - Cell-Based (Pre-layout)

The ASIC style implementions are 50% to 123% slower than the original design. This difference is caused by a combination of the limited drive strengths available in the *mitstdcells* and the extra standard cells required to implement the more complex primitives used in the IRRDP design. Also, in the case of t_{access} a distinct asymmetry exists between the delay for high-to-low transitions and the delay for low-to-high transitions. As with the cell-based approach and unlike the original full-custom case, no attempt was made to balance both paths. *This is another inherent limit in both the cell-based and ASIC approaches.*

Path	FC		ASIC+		ASIC	
	1	0	1	0	1	0
t_{PD}	1.00	1.00	1.50	1.70	1.85	1.78
t_{crit}	1.00	1.00	2.00	1.87	2.23	2.13
t_{access}	1.00	1.00	1.76	1.06	2.16	1.08

Table 5.12: Summary - Normalized Timing - ASIC (Pre-layout)

Post-Layout: After the completion of place and route, the resulting parasitic capacitances are extracted and back-annotated into the four HSPICE netlists.

Some additional wiring overhead is incurred in the cell-based approach as, unlike the full-custom methodologies, routing is not pre-planned. As a result, additional delay exists in the cell-based approaches and the resulting designs are now between 9% and 33% slower.

Path	FC		CB		CB(auto)	
	1	0	1	0	1	0
t_{PD}	1.00	1.00	1.09	1.08	1.21	1.14
t_{crit}	1.00	1.00	1.11	1.08	1.21	1.18
t_{access}	1.00	1.00	1.05	1.15	1.21	1.33

Table 5.13: Summary - Normalized Timing Comparison Cell-Based (Post-layout)

As expected, with routing, the delays along the three paths increase relative to the original baseline. The automated place and route results excess wiring and longer connections.

Path	FC		ASIC+		ASIC	
	1	0	1	0	1	0
t_{PD}	1.00	1.00	1.68	1.92	1.94	1.90
t_{crit}	1.00	1.00	2.21	2.05	2.32	2.22
t_{access}	1.00	1.00	2.45	1.43	2.04	1.41

Table 5.14: Summary - Normalized Timing - ASIC (Post-layout)

5.5 Analysis

In the full-custom IRRDP each bit is 14χ, the **CB** version required 15 χ per bit to be routable. In addition to this **1.07x** area growth due to routing requirements, several other sources of the area growth were also identified and analyzed. Except for the bit-height growth, all other area cost increases can be viewed as additional vertical tracks extending the width of the module. The **FC** IRRDP has a total width of 747 χ while the **CB** IRRDP has a total width of 1146 χ. Table 5.15 itemizes the sources of the 64% increase in area for **CB** approach:

Source	Width Overhead χ	% Contribution
Cell Area Growth	276	+ 37
Irregular structure - 1 bit regs	18	+ 2
Additional Routing Tracks	83	+ 11
Vertical Well/Substrate Contacts	22	+ 3
SUBTOTAL	399	+ 53
15bit vs 14 bit		**1.07x**
TOTAL		+ 64

Table 5.15: Sources of Area Growth

Several insights can be gleaned from the experimental results:

- Both cell-based approaches (**CB**, **CB(auto)**) result in designs that are within 66% of the original full custom approach. As the η_t's for the original design are 508 λ^2/T and $9.8\chi^2$/T, this results in an η_t of 843 λ^2/T and $16.3\chi^2$/T for the **CB** approaches. Both these values are well within the expected bounds of custom layout within microprocessors as shown in both Table 3.2 and Table 3.4.

- The post-layout critical path was between 11% and 21% longer with the cell-based approach. As no design effort is expended to balance the high-low and low-high transition delays, a portion of the growth in critical path is directly due to the decision to save effort in the implementation and eliminate tuning passes.

- With the correct collection of primitive cells and preassign I/O pin locations, the automated Place and route tool can generate layout nearly as dense as achievable via manual tiling (within +2%) for **bit-slices**. The 2% increase in size is directly attributed to the need to add an additional wiring track into the CB_IRFDCELLTOP primitive to guarantee that the router can make the necessary pin connections when adjacent cells are abutted.

- While the automated placement tool can correctly order cells within a bit-slice, it is unable to correctly extract signal flow locality from the full 64-bit netilist. The **CB(fullauto)** experiment resulted in a 10.5x increase in block area. Manual tiling of auto-placed bit-slices is a workable design point.

- Both standard ASIC approaches (**ASIC+**, **ASIC**) are poorly suited for implementing high performance datapaths. The area of a fully automated placement of a datapath design is 14.5x the size of the original design and over 3x the size of the cell-based approaches. Manually tiling ASIC bit-slices created with either the ASIC+ or ASIC approaches does improve the layout density, however, the resulting design is still 5x larger than the full-custom datapath and 3x larger than the cell-based approaches.

- Both standard ASIC approaches (ASIC+, ASIC) result in a doubling of the critical path. And, as with the cell-based approach, a portion of this increase is due to imbalance in the high-to-low and low-to-high transitions. As standard cells inherently are designed for balanced thresholds they are a poor match with traditionally N-dominated datapath circuits.

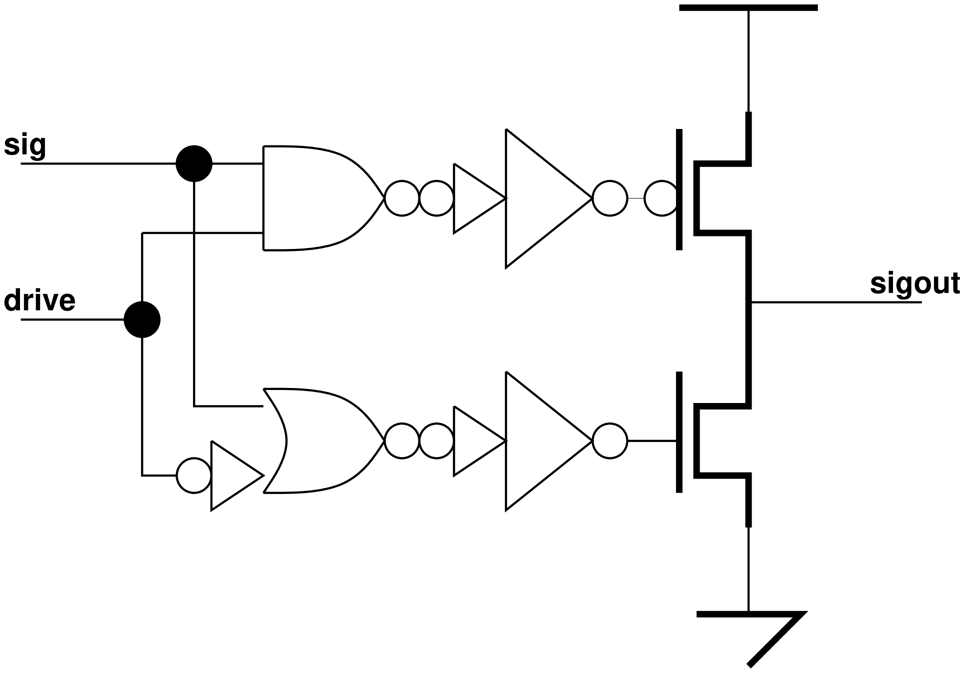


Figure 5-4: *IRFD_IWBO Circuit*

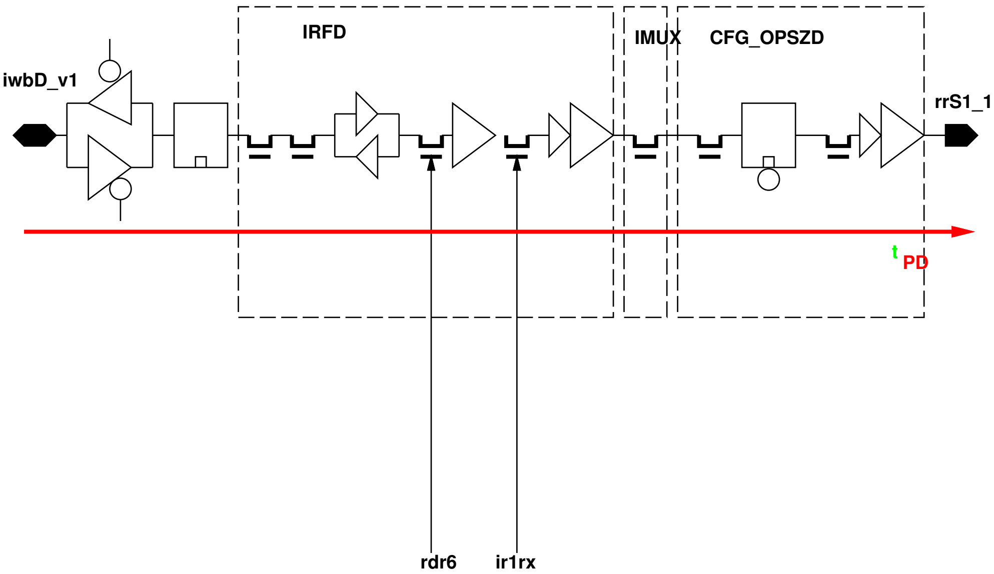


Figure 5-5: t_{PD}: *End-to-end Delay*

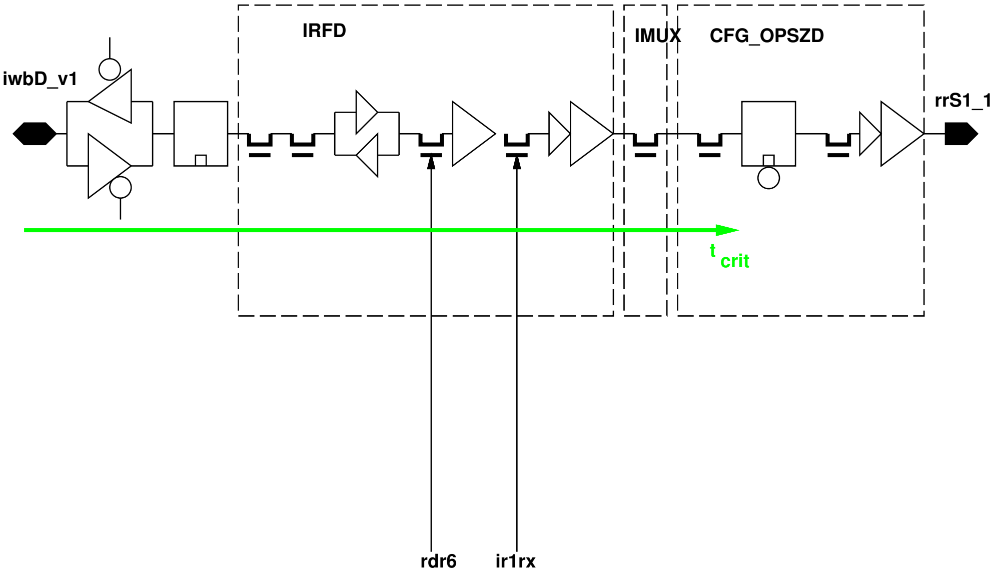


Figure 5-6: t_{crit}: *Critical Path - Write-through*

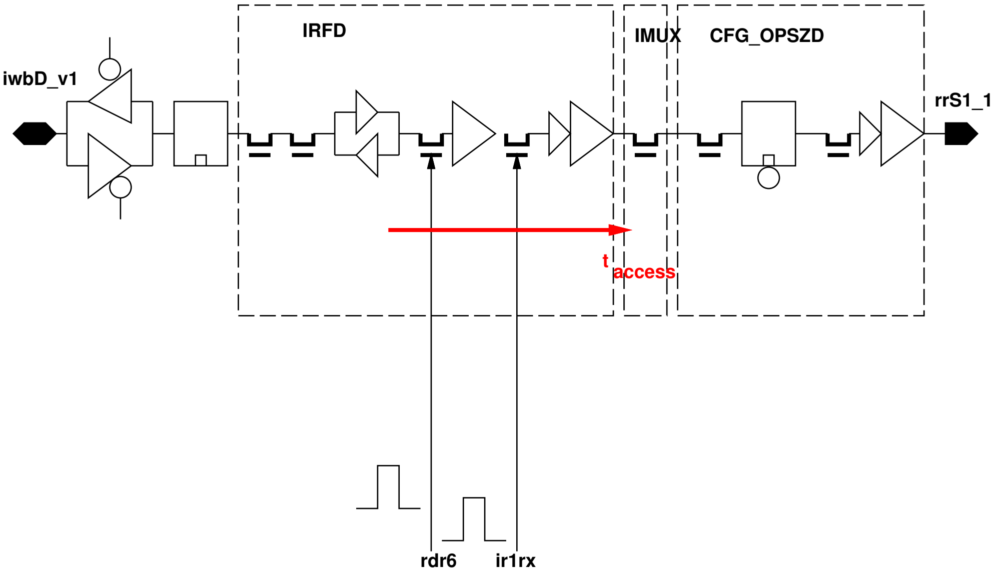


Figure 5-7: t_{access}: *Read Access*

Chapter 6

RESULTS: ADDSUB

6.1 Overview

The 64-bit Adder/Subtracter (ADDSUB) was selected as one of the experimental benchmarks because it is an irregularly structured functional block. Also, as it is an arithmetic block, the critical path timing performance is of primary importance. The cell-based implementation of the IRRDP lead to reasonable area density and modest growth in critical path. Also, the array-like structure of the IRRDP facilitated a simple identification and selection of building block primitives. This task is more difficult with the ADDSUB. As a selected re-implementation target, it is intended to test the limits of the cell-based implementation style.

In a process similar to the IRRDP, the ADDSUB was re-implemented in two different methodologies (**CB** and **ASIC**). Due to time schedule constraints, the **ASIC+** approach was not explored completely and the **CB(auto)** approach was omitted.

The **CB** style resulted in an implementation that was **1.74x** larger than the original ADDSUB. This design was *slower* than the original full-custom design as the critical timing path incurred **17%** additional delay.

Two different iterations of the **ASIC** style were created. Both resulted in **very slow** implementations of the ADDSUB. The **ASIC#1** design consisted of a completely flat synthesized netlist. While the area of the resulting layout is **0.75x** the size of the original design), this design has a **500%** longer critical path. The **ASIC#2**

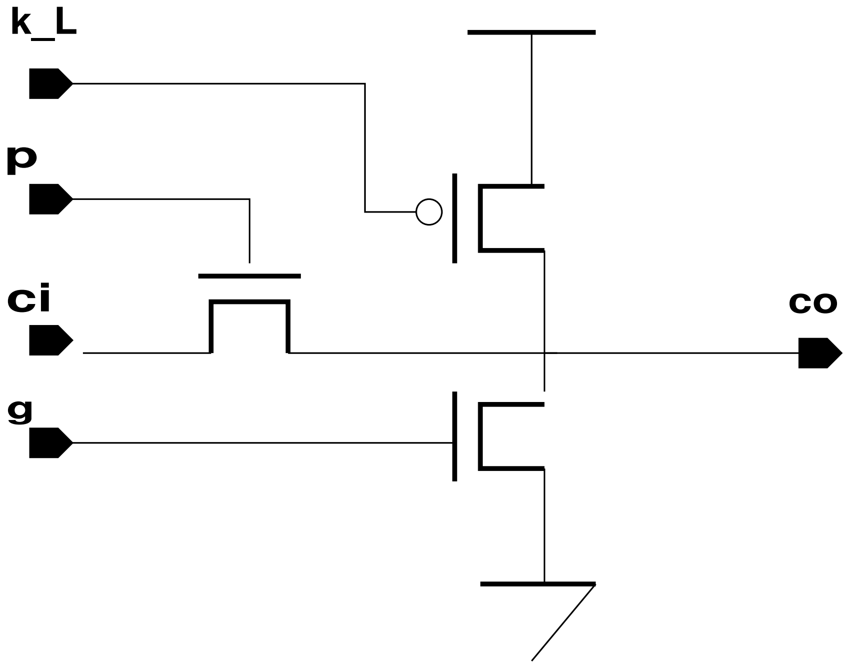


Figure 6-1: *Carry Bit - Basic Schematic*

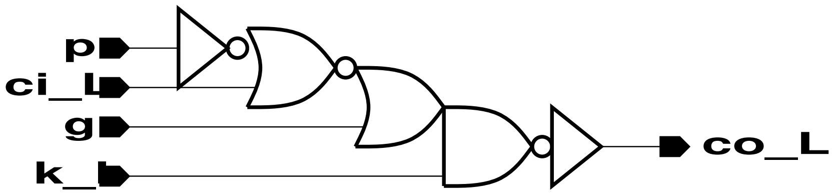


Figure 6-2: *Carry Bit - Synthesized Logic*

netlist retained one level of hierarchy by preserving the *carrybit* submodule. The functionality of the *carrybit* sub-module is shown in Figure 6-1. The logic synthesis equivalent of this function block results in four logic gates[1] and is shown in Figure 6-2.

Also, the synthesis parameters were tuned to yield larger drive strength throughout the **ASIC#2** logic As a result, the **ASIC#2** design was **1.2x** larger than the original design and **1.6x** larger than **ASIC#1**. The additional device size resulted in a small improvement in timing performance. The critical path in the **ASIC#2**

[1] One of the logic gates is an OAI21.

design was 440% longer than the original and 14% shorter than the **ASIC#1**.

A full **ASIC+** re-implementation of the ADDSUB was not completed. However, two additional cell-primitives *carrybit2* and *carrybit3* were designed. These cells were substituted for the corresponding functionality in the **ASIC#1** design and the both the area cost and the timing improvement were estimated. The addition of these two specialized primitives improved the timing performance while incurring more area cost. The critical timing path for the **ASIC+** version of the ADDSUB is estimated to be **170%** longer than the original **FC** design. The relatively large size of the *carrybit2* and *carrybit3* primitives, $300\chi^2$ and $408\chi^2$ respectively, results in a **1.67x** larger design as compared to the original **FC** version. However, the estimated **ASIC+** performance **is** a substantial improvement over both the **ASIC#1** and **ASIC#2** and results in 46% shorter and 38% shorter critical paths respectively.

The balance of this chapter is organized into four sections. The second section provides a brief functional description of the ADDSUB and describes the details of the constituent cells. The third section details the results and the final section of this chapter provides an analysis of the experimental data.

6.2 Functional Description

6.2.1 Overview

The 64-bit ADDSUB architecture is a hybrid of the carry-lookahead and carry-select approaches. It is composed of five functional blocks: **pgkblock**, **lcc**, **grouppgk**, **gcc**, and **finalcell**.

The **pgkblock** determines bit-wise propagate, generate, and kill values. The complete adder contains 64 **pgkblock's**. The **lcc** module contains two local carry-chains, one chain speculates a "true" carryin and the other chain speculates a "false" carryin. There are eight **lcc** blocks in the full adder. The **grouppgk** computes a propagate, a generate and a kill value for each set of eight bits. There are eight **grouppgk's** in the ADDSUB design. The **grouppgk** also performs speculation. The

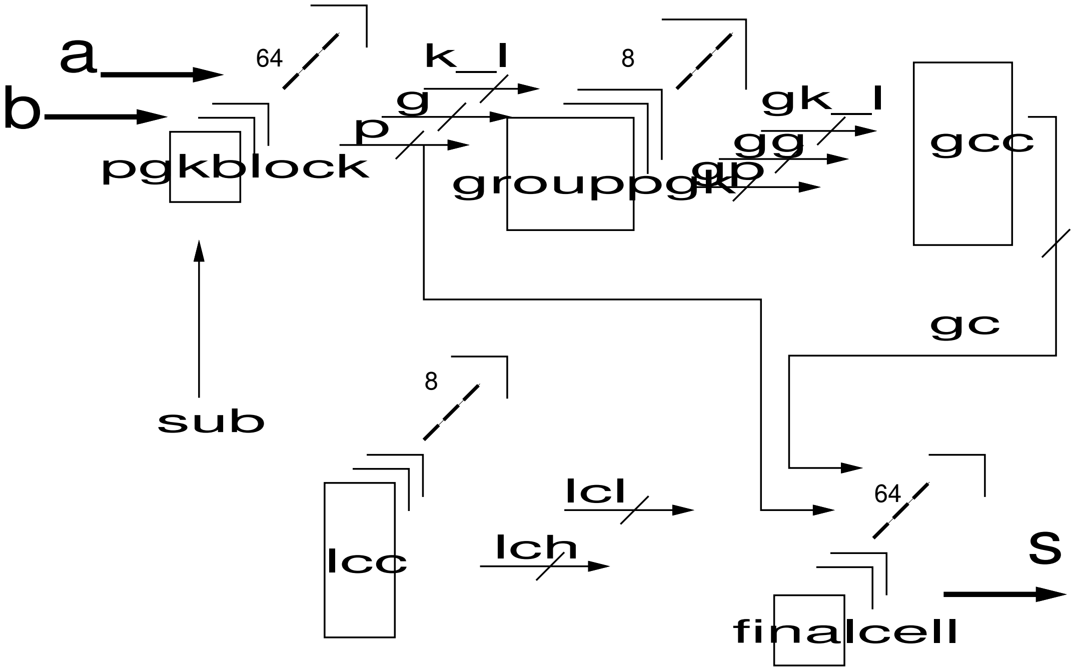


Figure 6-3: *64-bit ADDSUB Architecture*

top three bits of the internal eight bit carry chain are replicated and the results from the bottom five bits is used to select the correct set of top bits. The **gcc** (global carry chain) generates the appropriate carry for each set of eight bits based on the carryin input to the ADDSUB and global "p", global "g"and global "k" values received from each **grouppgk**. For improved performance, the **gcc** also implements speculation as the top four bits of its eight bit carry chain are replicated and, as with the **grouppgk**, the results from the low-bit chain select the correct upper bits. The actual result data is computed by the **finalcell**. This function receives inputs from the **grouppgk**, **lcc** and **gcc** blocks. The generation of the final output value is an independent operation for each **finalcell**. The ADDSUB contains 64 **finalcells**.

As a result, the critical path of the circuit includes the propagation delays through the **pgkblock**, the carry chain delays through both the **grouppgk** and **gcc** and, finally, the result computation in the **finalcell**. The critical path does not contain the **lcc** chains. An overview of the 64-bit ADDSUB architecture is presented in Figure 6-3. A detailed description of the ADDSUB can be found in [44].

6.2.2 Cell Descriptions

pgkblock

The schematic for the **pgkblock** is located in Appendix E. The circuit is a static logic implementation of the following equations:

$$p = a \oplus (b \oplus sub)$$
$$g = a \wedge (b \oplus sub)$$
$$\text{k\_L} = a \vee (b \oplus sub)$$

The pass-gate XOR circuit implements the inversion of b for the subtraction case.

lcc

The schematic for the **lcc** is located in Appendix E. The circuit is a pair of 7-stage carry chains and buffers for the outputs.

grouppgk

The schematic for the **grouppgk** is located in Appendix E. The circuit is divided into a five-stage carry chain and includes two copies of the top 3-stage portion of the carry-chain. The asymmetric split was selected to attempt to balance to overall delay of both paths. The circuit implements the following equations:

$$gp = \bigwedge(p0, p1....p7)$$
$$gg = g_{bottom} \wedge \neg kill_{top} \vee g_{top}$$
$$\text{gk\_L} = (g_{bottom} \wedge (gp \vee \neg kill_{top}) \vee (\neg g_{bottom} \wedge (gp \vee g_{top})$$

gcc

The schematic for the **gcc** is located in Appendix E. The global carry-chain takes as inputs the *carryin* and results from all the **grouppgk's**. It generates the carry for each eight bit section. The global chain is divided in half and the upper four bit chain is replicated allowing both the carryin "true" and carryin "false" results to be computed in parallel.

finalcell

The schematic for the **finalcell** is located in Appendix E. This circuit implements the following function

gc = 1 :

$$s = p \oplus lch$$

gc = 0 :

$$s = p \oplus lcl$$

6.3 Experimental Data

A single structural verilog source was used as the basis for all the re-implementations of the ADDSUB. The critical path time delay for the original **FC** ADDSUB is 4ns. The area of the original design is 80930 χ^2.

6.3.1 CB

The cell-based re-implementation was performed by converting each of the five primitives to place and route compatible layouts. The re-implemented cells were 18 χ per bit in height, a 28% increase in comparison to the original 14 χ per bit. In addition, the cells grew an average of 36% in width. The additional width was caused by the requirement of at least one additional track at the start and end of each cell to meet design rule violations and the addition of open wiring tracks to "spread-out" the cell and allow the place and route tool to access all the input and output ports. The **CB** approach resulted in 162 cells. The five cells were tiled in an identical floorplan to the original **FC** design.

The critical path time delay of the **CB** design is 4.7ns and the area consumed by the design is 141032 χ^2.

6.3.2 ASIC#1, ASIC#2

The two ASIC approaches were implemented through logic synthesis of the structural verilog netlist using the Synergy^{TM} tool. The **ASIC#1** approach resulted in 1042 gates while the **ASIC#2** approach resulted in 1803 gates due to the preservation of the carrybit sub-module. Both layouts were implemented via automated place and route with 12 χ per bit. In both cases the location of the input and output ports was defined explicitly before placement. The **ASIC#2** approach also benefited from a larger sizing of buffers within the logic. The critical path delay for the **ASIC#1** design was 19.9 ns while the critical path delay for the **ASIC#2** design was 17.4ns.

The area consumed by the **ASIC#1** design is 60873 χ^2. While the area consumed by the **ASIC#2** design is 96696 χ^2.

6.3.3 ASIC+

The **ASIC+** approach was not fully implemented. However, the circuit design for both the proposed carrybit2 and carrybit3 primitives was performed.

The *carrybit2* and *carrybit3* primitives represent two and three bit chains of the previously discussed N-only pass-gate *carrybit*. In addition, the output of each of these chains is buffered by a combination of an INV_1R, skewed threshold inverter from *mitstdcells*, and a balanced INV_4 inverter. The benefit of the skewed inverter is substantial. The delay of the *carrybit3* followed by an INV_1R is a **2.9ns** shorter than the delay of the *carrybit3* followed by a balanced inverter. As described previously, the critical path for the addition in the ADDSUB architecture consists of the PGKBLOCK computation, a five stage propagation through the GROUPPGK carry chain, a four stage propagation through the GCC carry-chain, followed by the final cell logic. In the full-custom ADDSUB the propagation through both sets of carry-chains requires 2.6ns [44]. The propagation through a carrybit2/carrybit3 implementation of the same function requires 3.3ns. The 3.3ns required for the carrybit2/carrybit3 implementation of the carry-chain's was combined with the unchanged 4.6ns required for the PGKBLOCK computation via synthesized logic and the 2.8ns delay through the synthesized FINALCELL logic to yield the 10.7ns estimated critical path timing. The area consumed by the **ASIC+** design is estimated to be 135048 χ^2. The area of the carrybit2 and carrybit3 cells was estimated to be $300\chi^2$ and $408\chi^2$ respectively. This estimate was based on the resulting device sizes and the anticipated ρ_t and ρ_{gate} metrics typical to the *mitstdcell* library.

6.4 Analysis

Tables 6.1, 6.2, 6.3, and 6.4 summarize the results from the re-implementation of the ADDSUB function. As expected, the original full-custom design is both the most area efficient and the fastest design. The **CB** implementation has comparable timing performance but is substantially less efficient in area. Some of this area inefficiency may be due to the manner in which the constituent cells were re-implemented. Unlike the IRRDP re-implmentation, no time was spent optimizing the constituent cells for the **CB** ADDSUB. As a result, the average cell is 74% larger. For comparison, the cell-area difference between the **CB** IRRDP and the **FC** IRRDP was only 37%. The results from the cell library studies discussed in Chapter 4 indicate that a 30% to 40% cell area increase instead of the experienced 74% may be achievable. Also, as the ADDSUB contains multi-bit height cells (LCC, GCC, GROUPPGK), additional overheads in cell-width have a magnified effect.

Source	FC χ^2	FC	CB
Total Module Area	80930	1.0	1.74

Table 6.1: Normalized Area Results - CB

Source	FC χ^2	FC	ASIC#1	ASIC#2	ASIC+
Total Module Area	80930	1.0	0.75	1.19	1.67

Table 6.2: Normalized Area Results - ASIC's

Source	FC t_{crit} (ns)	FC	CB
Total Critical Path Time	4.0	1.0	1.17

Table 6.3: Normalized Timing Results - CB

Source	FC t_{crit} (ns)	FC	ASIC#1	ASIC#2	ASIC+
Total Critical Path Time	4.0	1.0	5.0	4.4	2.7

Table 6.4: Normalized Timing Results - ASIC's

Chapter 7

CONCLUSIONS AND FUTURE WORK

7.1 Summary of Results

Both a regular (IRRDP) and an irregular (ADDSUB) datapath were re-implemented with *cell-based* approaches and then compared with both the original full-custom design and more traditional ASIC implementations.

In both cases, the *cell-based* approach provided a significant improvement in both area and performance over the traditional ASIC implementations. More importantly, in both cases cell-based approaches approached the density and performance of the full-custom designs.

In the case of the IRRDP, the cell-based design was less that **1.64x** larger and the resulting critical timing path grew less that **11%**. In the case of the ADDSUB the cell-based design was less that **1.74x** larger and the resulting critical timing path grew less than **17%**. In both these cases, corresponding traditional ASIC implementations were significantly worse in both area efficiency and timing performance. The ASIC design for the IRRDP was **14.5x** larger and incurred 123% more delay. While, the best ASIC design for the ADDSUB was only **1.67x** larger it resulted in a 170% increase in delay.

Several factors limit cell-based approaches from equaling the area efficiency of full-custom implementations: overhead of grid alignment, overhead due to P-transistors, overhead of additional diode contacts, and lack of N-only multiplexor structures. The combined effect of these constraints is a **1.1x** to **1.4x** growth in area.

The predominant limits for cell-based approaches with regards to timing performance are: excess P-transistors, the course granularity of drive available drive strengths, inability to employ dynamic circuit techniques, additional parasitic capacitance due to extraneous wiring, lack of skewed threshold inverters and absence of fine-tuning design iterations.

This research also proposes a simple linear model for layout efficiency (η_t) that relates the generally competing values of transistor density (ρ_t) and gate utilization density (ρ_{gate}). The model results in a family of lines whose slope and intercept(s) are process specific and easily calculated from the employed design rules.

7.2 Conclusions

This research identifies and characterizes several intermediate design points between the traditional ASIC and full-custom methodologies.

This work produced the following insights:

- Traditional ASIC approaches are severely limited in their ability to efficiently emulate full-custom datapath designs due to the balanced N:P ratios, a lack of complex primitives and lack of skewed ratio inverter primitives. As a result they both inefficiently map the circuit structures inherent to datapaths and handicap performance.

- Current automated place and route tools are effective in efficiently placing cells within defined bit-slices but are unable to extract signal flow groupings from a full design.

- The *cell-based* approach is a significant improvement in both the layout efficiency and timing performance over the traditional ASIC approach.

- The *cell-based* approach can be viewed as either an *enhancement* of the ASIC flow resulting in superior designs. Or, as a *simplification* of the full-custom approach exchanging the benefits of design reuse, reduced complexity, and hierarchy for a quantified penalty in both layout efficiency and timing performance.

- System architects can choose from a spectrum of design points that exist between the full-custom and traditional ASIC methodologies.

7.3 Future Work

While this work begins to characterize the design space between the traditional ASIC flow and full-custom methodologies, many interesting extensions to this work are possible.

This thesis has explored two design examples. A useful extension of this work would be to apply these techniques to additional regular and irregular circuits such as signal processing circuits, for example DCT and MPEG cores and additional arithmetic execution units, for example multiplier arrays and SRT dividers.

The empirical data from this research indicates that the cell-based approach represents a substantial effort savings over the full-custom approach. All physical re-design, characterization and verification for the four re-implementations of both datapaths were completed by one person over the course of seven weeks. In contrast, *each* of the original full-custom designs required two engineers and nine months of time to complete. Thus an interesting extension to this work would quantify in detail the effort savings of the *cell-based* approach with respect to the full-custom approach and, also, quantify the incremental effort growth of cell-based in comparison to traditional ASIC.

Both design intuition and the empirical data from this research indicate that just as improvements in area efficiency can be made by adding manual intervention and customization to the layout creation and floorplanning stages, significantly improved routing resource utilization may be gained from selectively incorporating custom prerouting with the automated process. The M-Machine design team successfully employed this technique to greatly reduce congestion for datapath busses in the cluster portions of the MAP design. An interesting extension of this work would characterize in detail the benefits of semi-custom and full-custom bus pre-routing on the congestion and routing resource utilization. A $\eta_{routing}$ model akin to η_{layout} could then be proposed. Also, the potential benefits of multi-pass place and route algorithms could be explored.

This work indicates that traditional ASIC cell libraries are a poor match for datapath circuits and proposes skewed ratio inverters, register primitives, n-only multiplexor primitives and the carrybit primitive as useful extensions. An investigation of the benefits of other new cell primitives and their affects on η_{layout} or $\eta_{routing}$ would be valuable.

This study did not address the effects on power consumption of the cell-based approach. Though an obvious hypothesis is that the improved layout efficiency, lower gate count, and reduced parasitics (resulting from shorter wire lengths) would make the cell-based approach more power efficient in comparison with traditional ASIC.

Another interesting extension to this work would be a quantification of the sensitivity of the reported results to a variety of parameters: range of available cell drive strengths in the targeted library, aspect ratio of constituent cells, bit-slice height in terms of χ and λ,

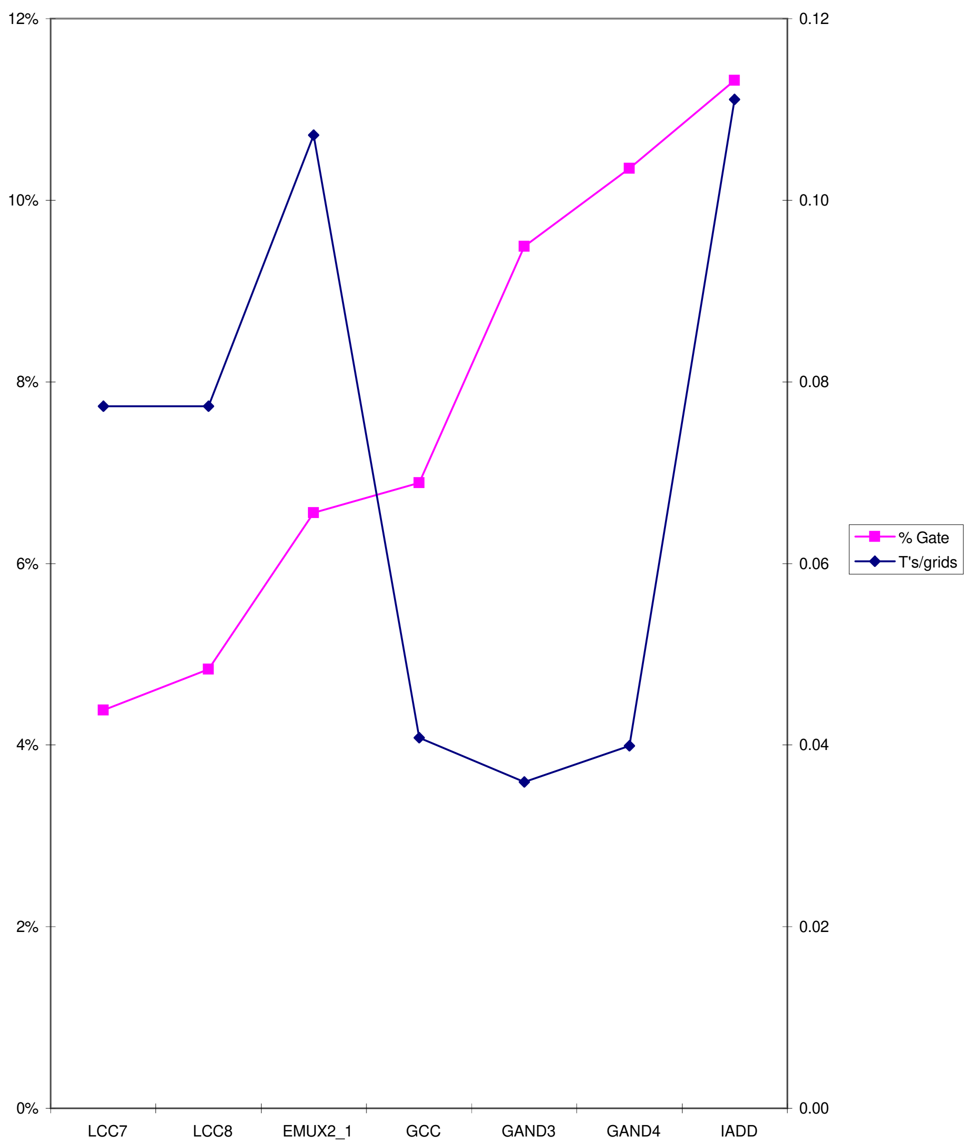
In summary, this thesis has characterized the cell-based implementations of two datapaths from a contemporary microprocessor design. The results of this research indicate that interesting design points exist between the traditional ASIC and full-custom approaches. These design points provide system architects with additional degrees of problem solving freedom and provide additional clarity into the optimal division of human and automated effort in solving the physical design problem.

Appendix A

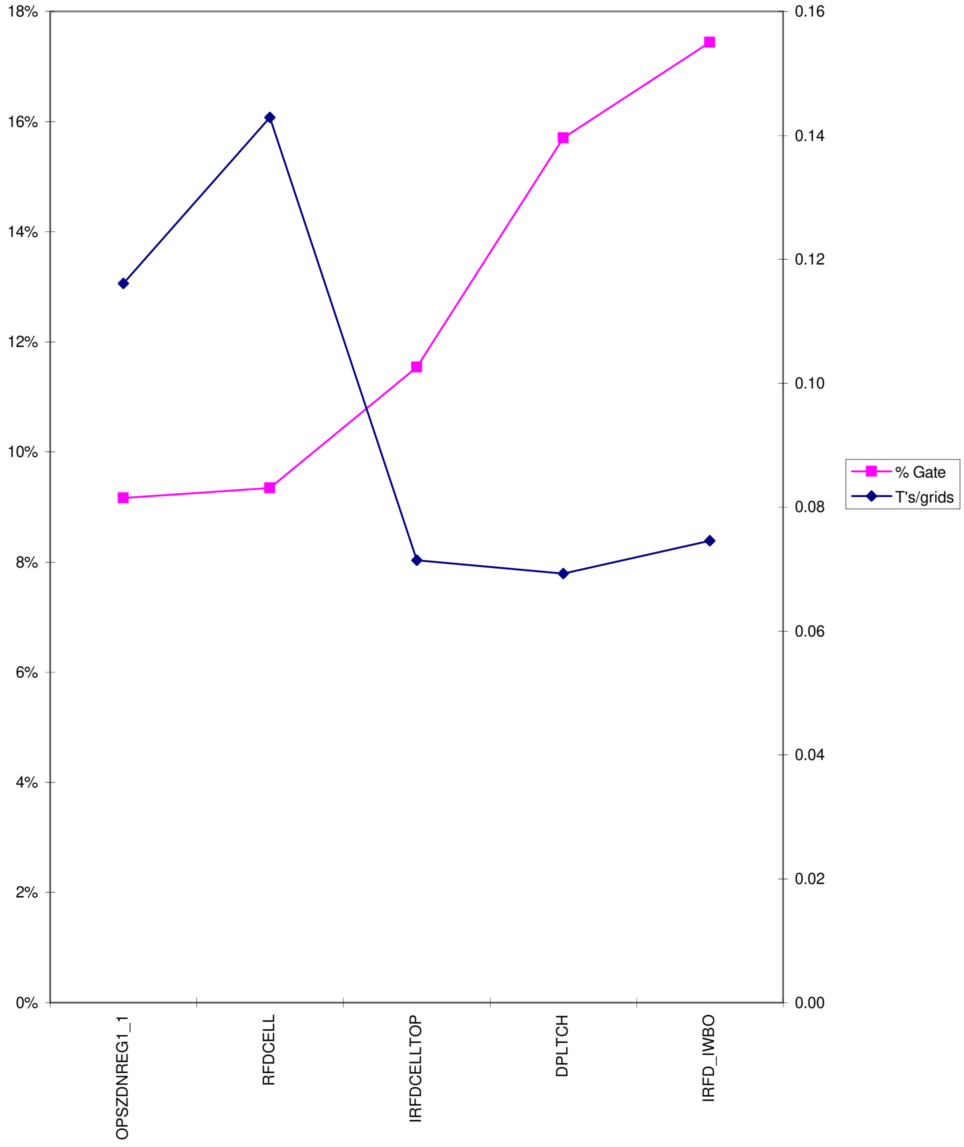
IRRDP - Experimental Data

A.1 Cell Library Data - Plots

INCREMENTOR CELLS

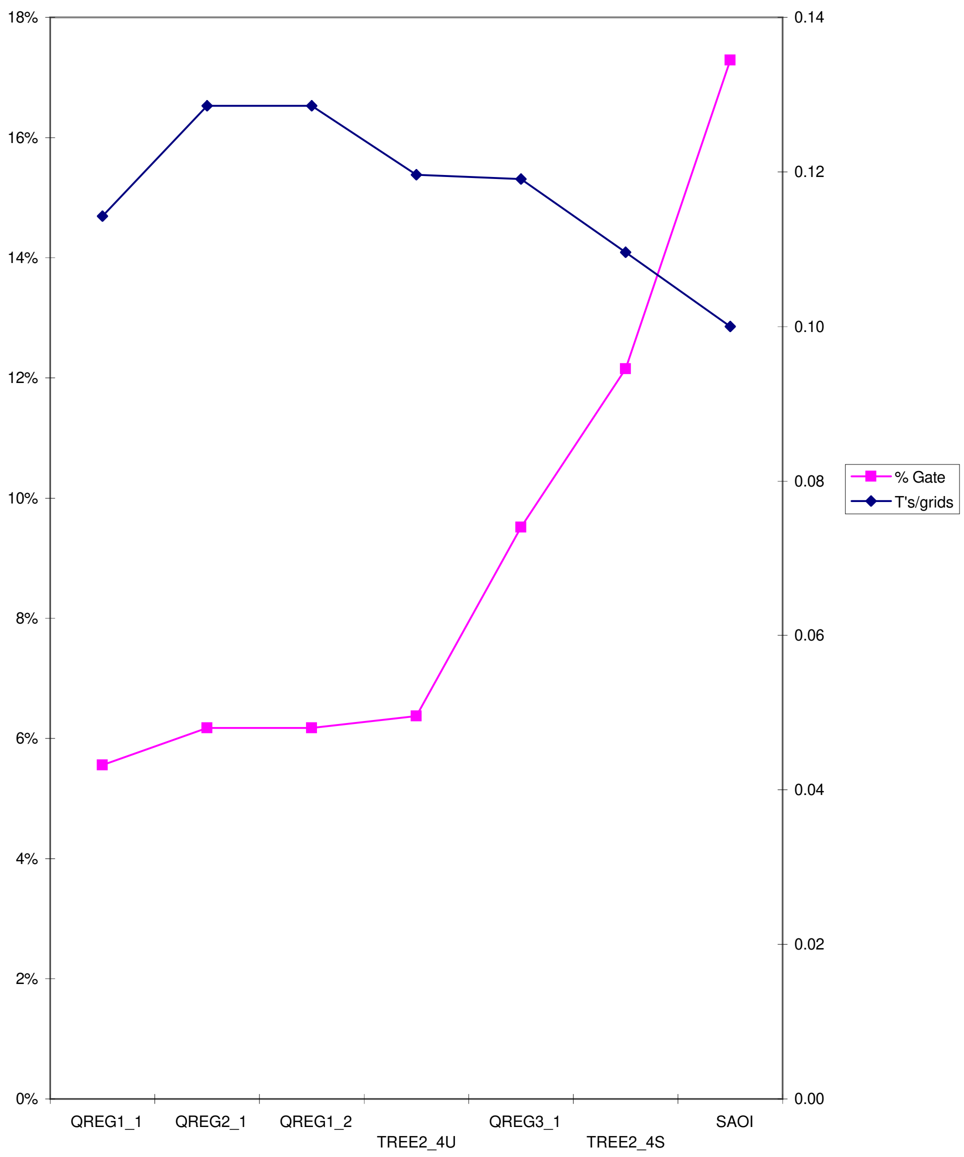


IRRDP CELLS

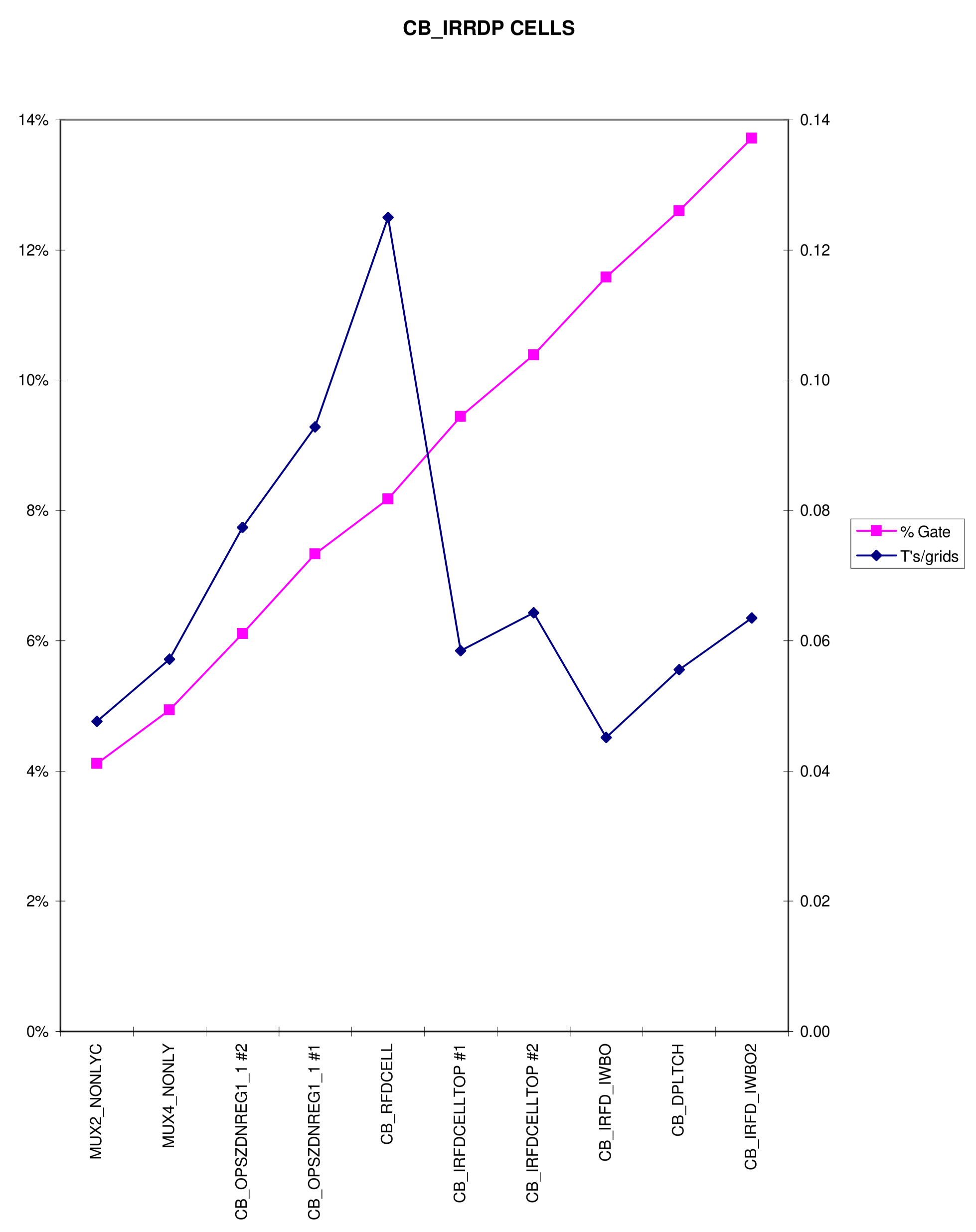


18%
16%
14%
12%
10%
8%
6%
4%
2%
0%
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
% Gate
T's/grids
OPSZDNREG1_1
RFDCELL
IRFDCELLTOP
DPLTCH
IRFD_IWBO

FIP CELLS



18%
16%
14%
12%
10%
8%
6%
4%
2%
0%
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
QREG1_1
QREG2_1
QREG1_2
TREE2_4U
QREG3_1
TREE2_4S
SAOI
% Gate
T's/grids



CB_IRRDP CELLS
14%
12%
10%
8%
6%
4%
2%
0%
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
% Gate
T's/grids
MUX2_NONLYC
MUX4_NONLY
CB_OPSZDNREG1_1 #2
CB_OPSZDNREG1_1 #1
CB_RFDCELL
CB_IRFDCELLTOP #1
CB_IRFDCELLTOP #2
CB_IRFD_IWBO
CB_DPLTCH
CB_IRFD_IWBO2

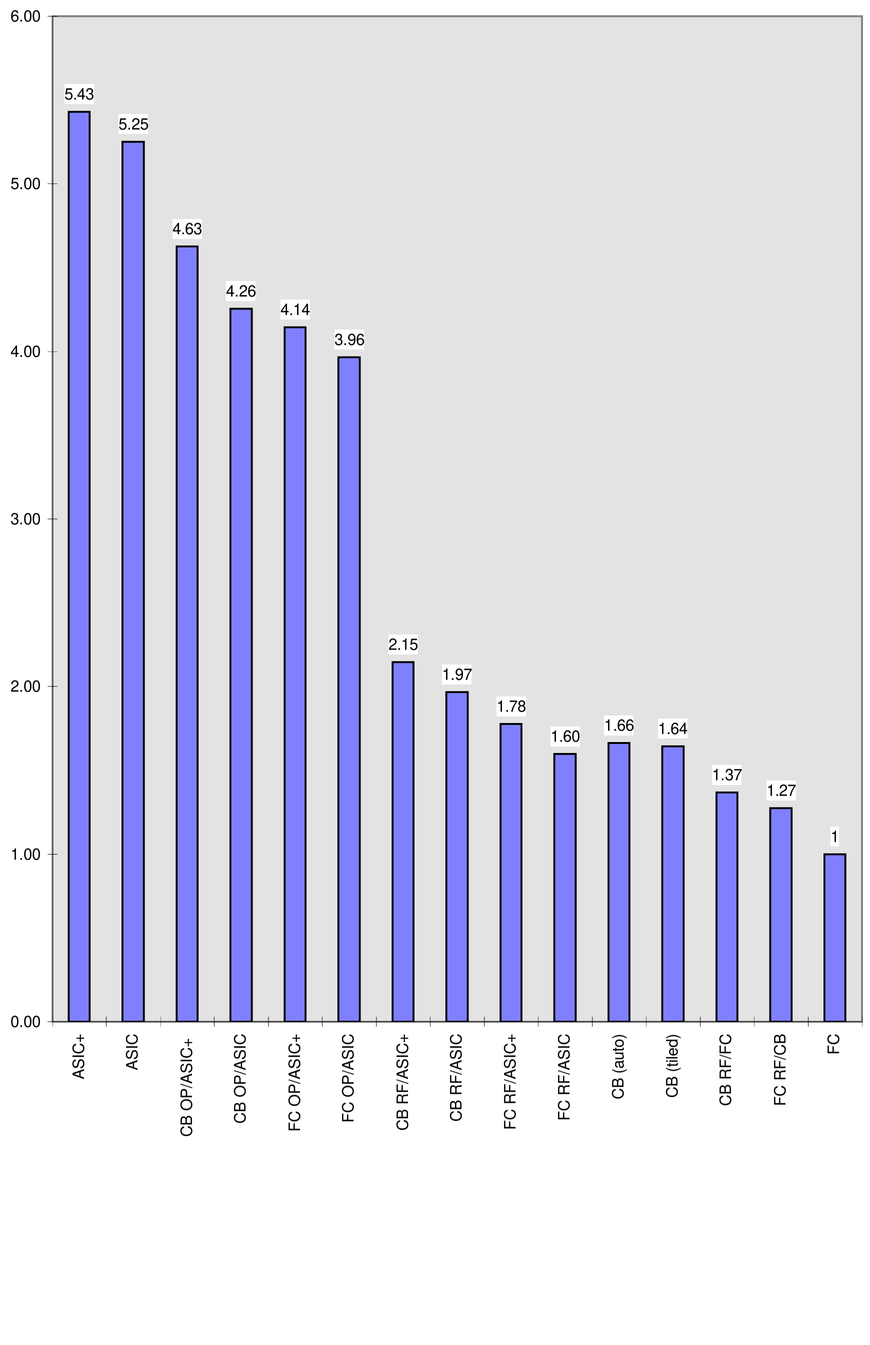
A.2 Cell Library Data - Tables

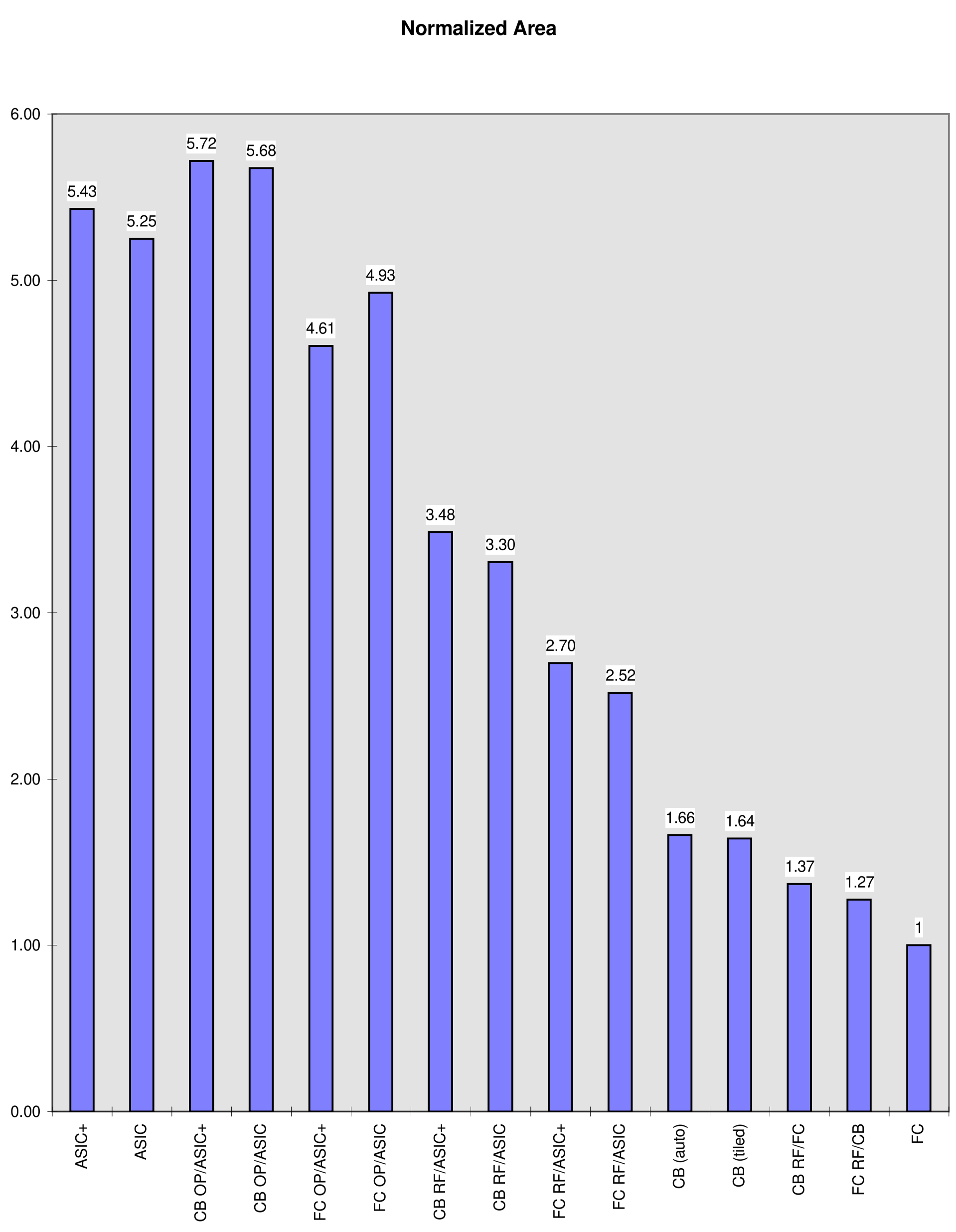
CELLS	% Gate	T's/grid2
LCC7	4%	0.08
LCC8	5%	0.08
EMUX2_1	7%	0.11
GCC	7%	0.04
GAND3	9%	0.04
GAND4	10%	0.04
IADD	11%	0.11
QREG1_1	6%	0.11
QREG2_1	6%	0.13
QREG1_2	6%	0.13
TREE2_4U	6%	0.12
QREG3_1	10%	0.12
TREE2_4S	12%	0.11
SAOI	12%	0.10
IF_XOR2A	3%	0.06
IF_XOR3	7%	0.09
IF_OP_BUF1	14%	0.16
INV1_1	1%	0.03
INV4_2	2%	0.03
SR-BIT	3%	0.13
TVARDELAY	4%	0.04
INV10_5	5%	0.03
NOR15_5	6%	0.03
DYN_NAND5	9%	0.04
TC_NOR3	9%	0.07
INV20_10	9%	0.03
INV30_15	9%	0.02
BIT_CELL	9%	0.34
NOR40_10	10%	0.02
DYN_NOR9	10%	0.05
DYN_NAND4	11%	0.06
TDRV200_100	11%	0.02
RR_SEL	11%	0.05
TDRV300_150	13%	0.02
SENSEAMP	14%	0.09
INV100_50	15%	0.01
INV120_30	15%	0.01
TDRV500_250	17%	0.02
PDD	19%	0.02
WLD	21%	0.02
AND2	5%	0.10
AND3	5%	0.11
PLTCH_CELL	6%	0.07
AND4	7%	0.12
XOR2	8%	0.12
XNOR2	8%	0.12
CKA_1	8%	0.06
INV_2	8%	0.06
NLTCH_SCAN	10%	0.08
PDFF_SCAN	10%	0.09

NCKA_1	11%	0.08
NAND2	11%	0.11
CKA_2	12%	0.05
INV_4	12%	0.04
NAND3	12%	0.13
NAND4	13%	0.13
CKA_4	15%	0.03
AOI21	17%	0.13
BUF_8	17%	0.05
AOI2222	18%	0.12
CKA_8	19%	0.02
AOI22	20%	0.13
INV_8	20%	0.03
NCKA_8	25%	0.02
MUX2_NONLYO	4%	0.05
MUX4_NONLY	5%	0.06
CB_OPSZDNRE	6%	0.08
CB_OPSZDNRE	7%	0.09
CB_RFDCELL	8%	0.13
CB_IRFDCELLT	9%	0.06
CB_IRFDCELLT	10%	0.06
CB_IRFD_IWBO	12%	0.05
CB_DPLTCH	13%	0.06
CB_IRFD_IWBO	14%	0.06
OPSZDNREG1_	9%	0.12
RFDCELL	9%	0.14
IRFDCELLTOP	12%	0.07
DPLTCH	16%	0.07
IRFD_IWBO	17%	0.07

A.3 IRRDP Data - Plots

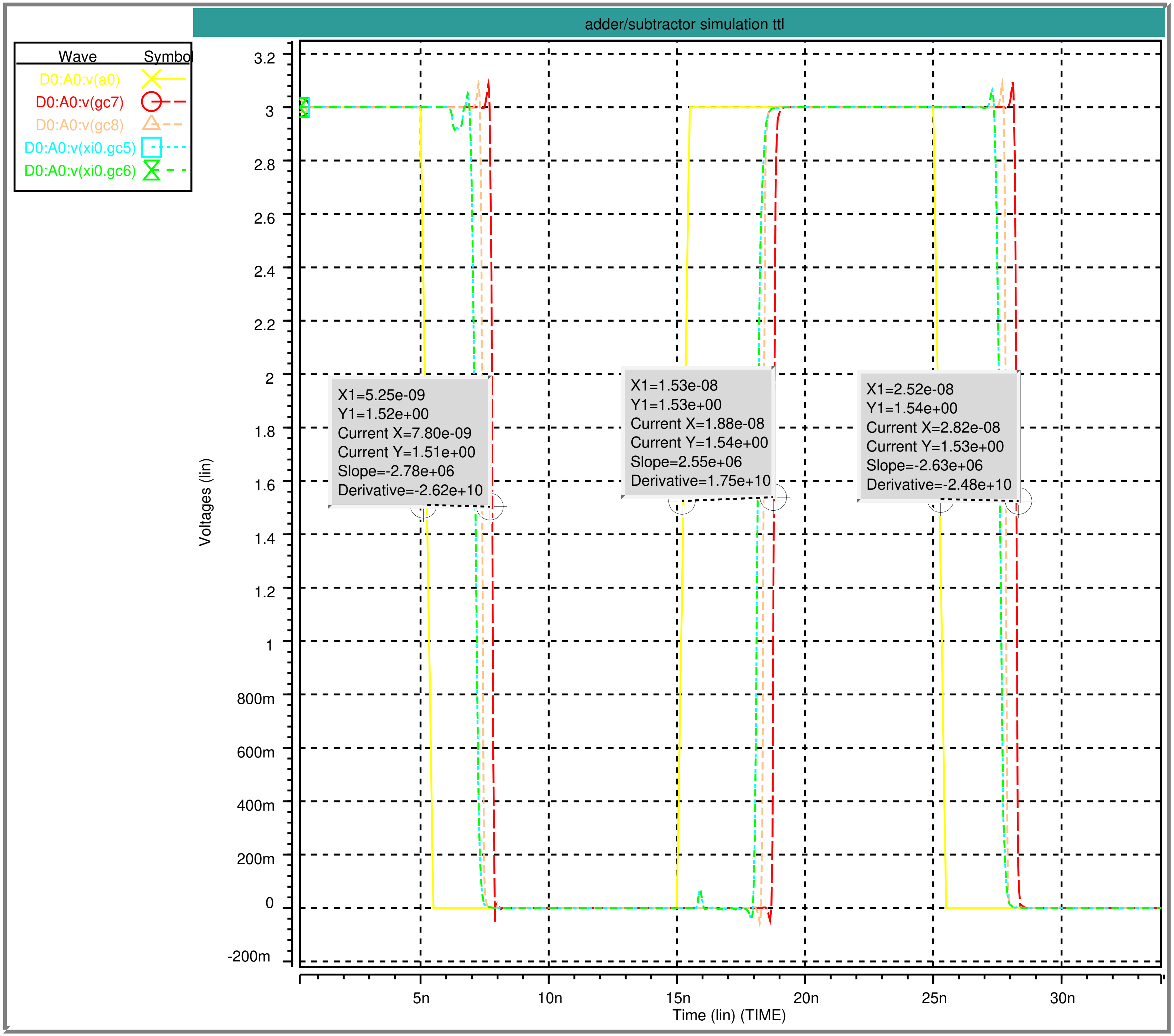
Normalized Area

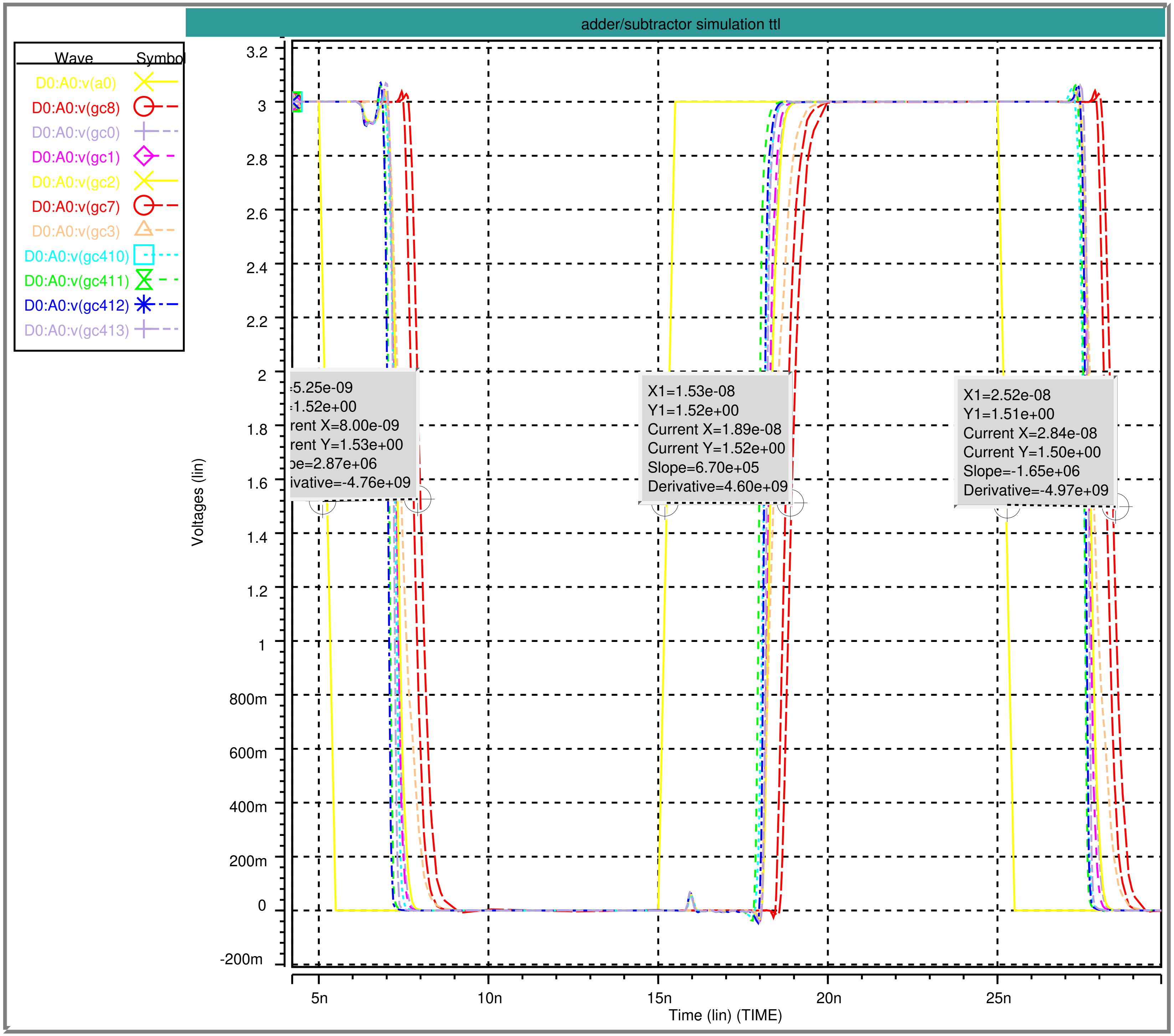



Normalized Area
6.00
5.00
4.00
3.00
2.00
1.00
0.00
5.43
5.25
5.72
5.68
4.61
4.93
3.48
3.30
2.70
2.52
1.66
1.64
1.37
1.27
1
ASIC+
ASIC
CB OP/ASIC+
CB OP/ASIC
FC OP/ASIC+
FC OP/ASIC
CB RF/ASIC+
CB RF/ASIC
FC RF/ASIC+
FC RF/ASIC
CB (auto)
CB (tiled)
CB RF/FC
FC RF/CB
FC

Appendix B

Adder - Experimental Data


adder/subtractor simulation ttl
Wave
Symbol
D0:A0:v(a0)
D0:A0:v(gc7)
D0:A0:v(gc8)
D0:A0:v(xi0.gc5)
D0:A0:v(xi0.gc6)
X1=5.25e-09
Y1=1.52e+00
Current X=7.80e-09
Current Y=1.51e+00
Slope=-2.78e+06
Derivative=-2.62e+10
X1=1.53e-08
Y1=1.53e+00
Current X=1.88e-08
Current Y=1.54e+00
Slope=2.55e+06
Derivative=1.75e+10
X1=2.52e-08
Y1=1.54e+00
Current X=2.82e-08
Current Y=1.53e+00
Slope=-2.63e+06
Derivative=-2.48e+10
Voltages (lin)
3.2
3
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
800m
600m
400m
200m
0
-200m
5n
10n
15n
20n
25n
30n
Time (lin) (TIME)

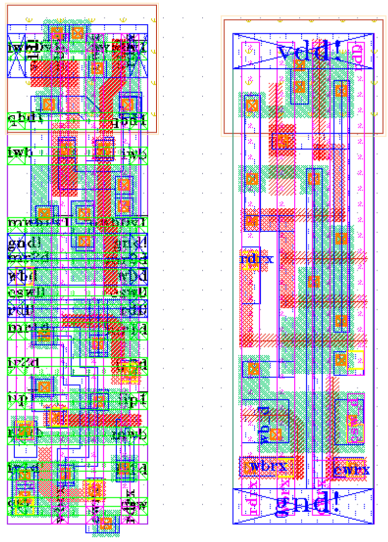


adder/subtractor simulation ttl
Wave
Symbol
D0:A0:v(a0)
D0:A0:v(gc8)
D0:A0:v(gc0)
D0:A0:v(gc1)
D0:A0:v(gc2)
D0:A0:v(gc7)
D0:A0:v(gc3)
D0:A0:v(gc410)
D0:A0:v(gc411)
D0:A0:v(gc412)
D0:A0:v(gc413)
Voltages (lin)
Time (lin) (TIME)
X1=1.53e-08
Y1=1.52e+00
Current X=1.89e-08
Current Y=1.52e+00
Slope=6.70e+05
Derivative=4.60e+09
X1=2.52e-08
Y1=1.51e+00
Current X=2.84e-08
Current Y=1.50e+00
Slope=-1.65e+06
Derivative=-4.97e+09

Appendix C

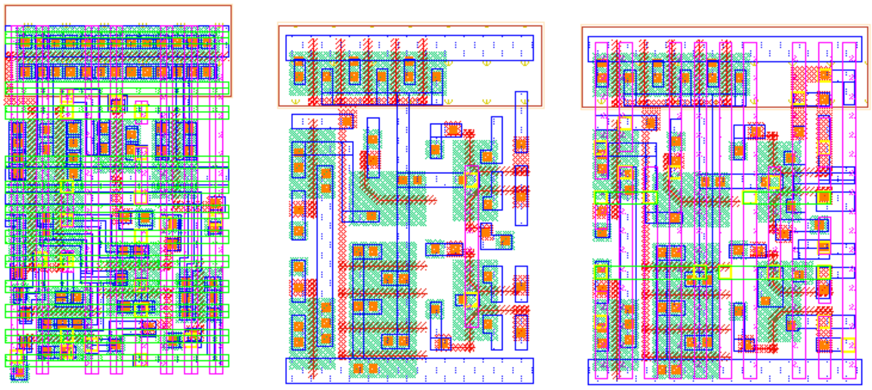
Cell Layouts

RFDCELL



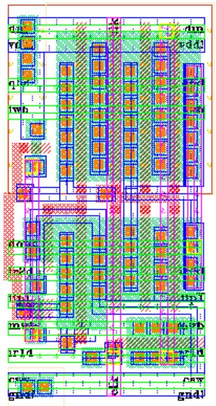
FC CB

IRFDCELLTOP

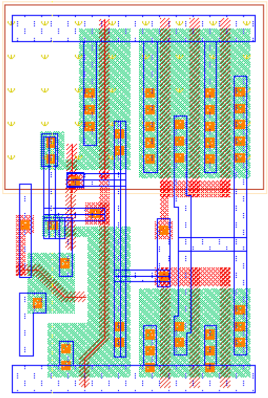


FC CB CB(auto)

DPLTCH

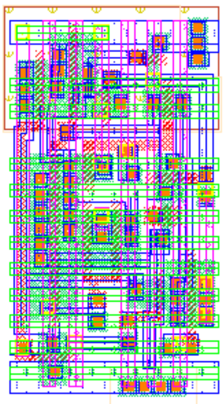


FC

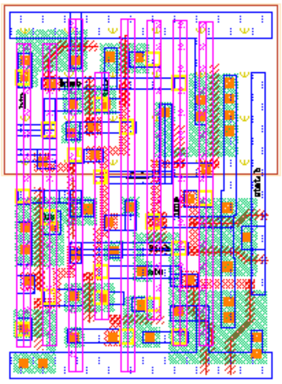


CB

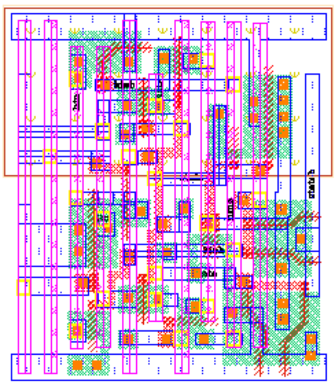
OPSZDNREG1_1



FC



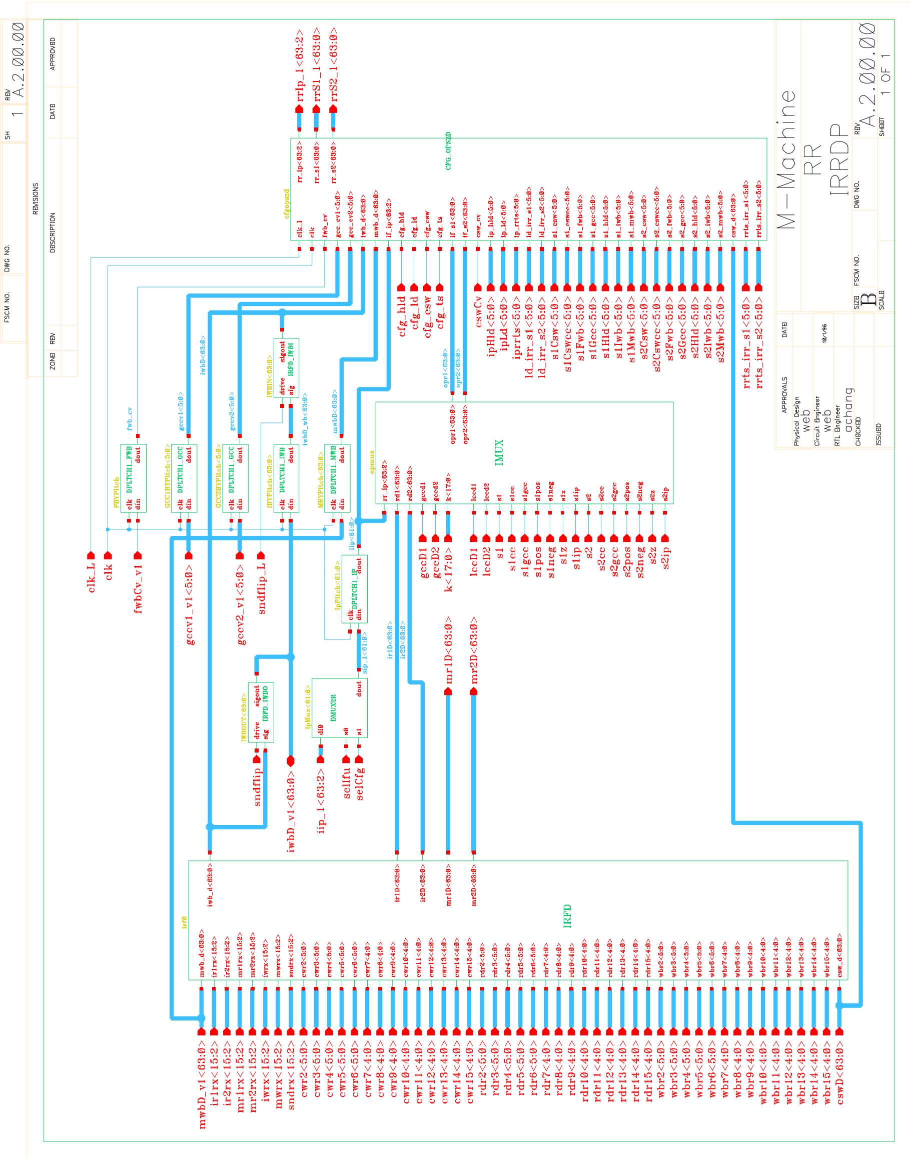
CB

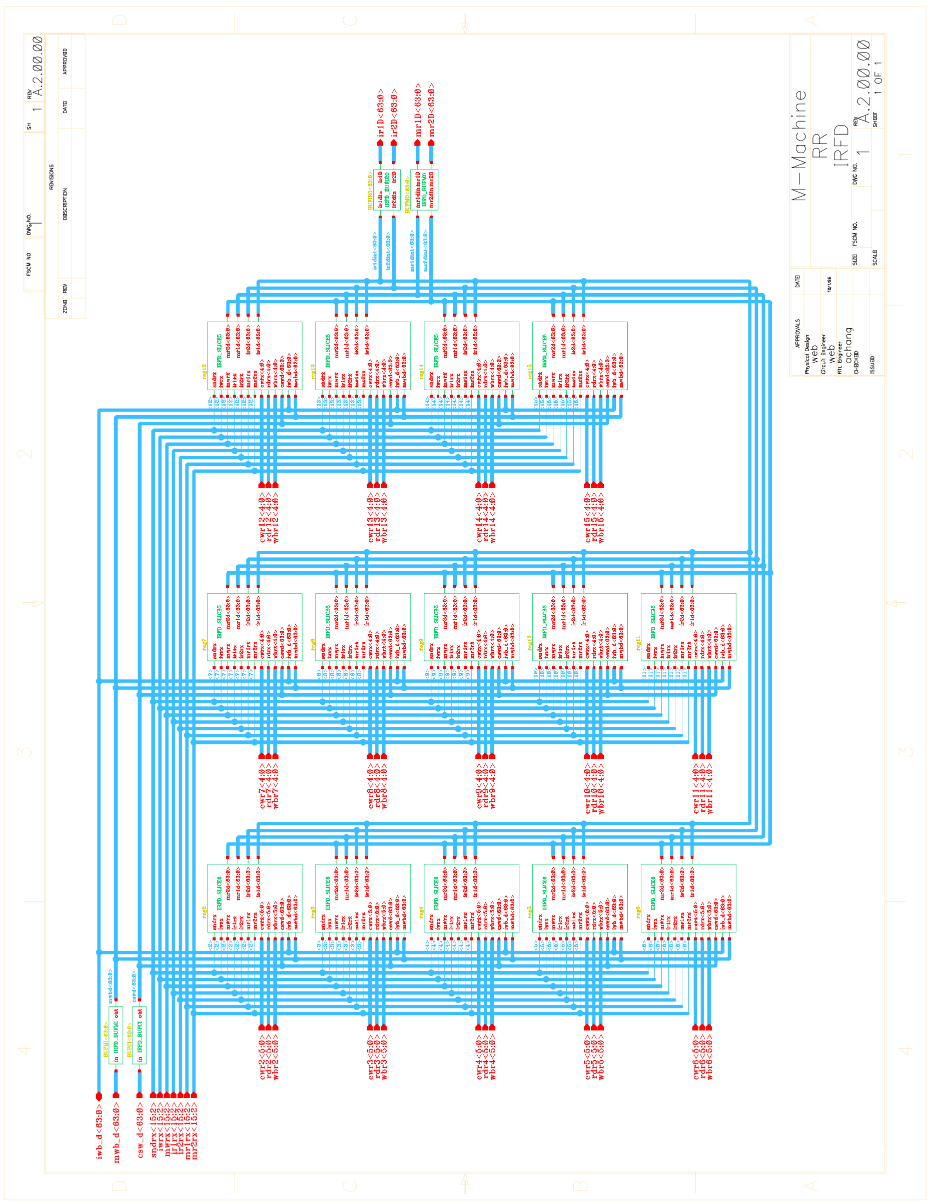


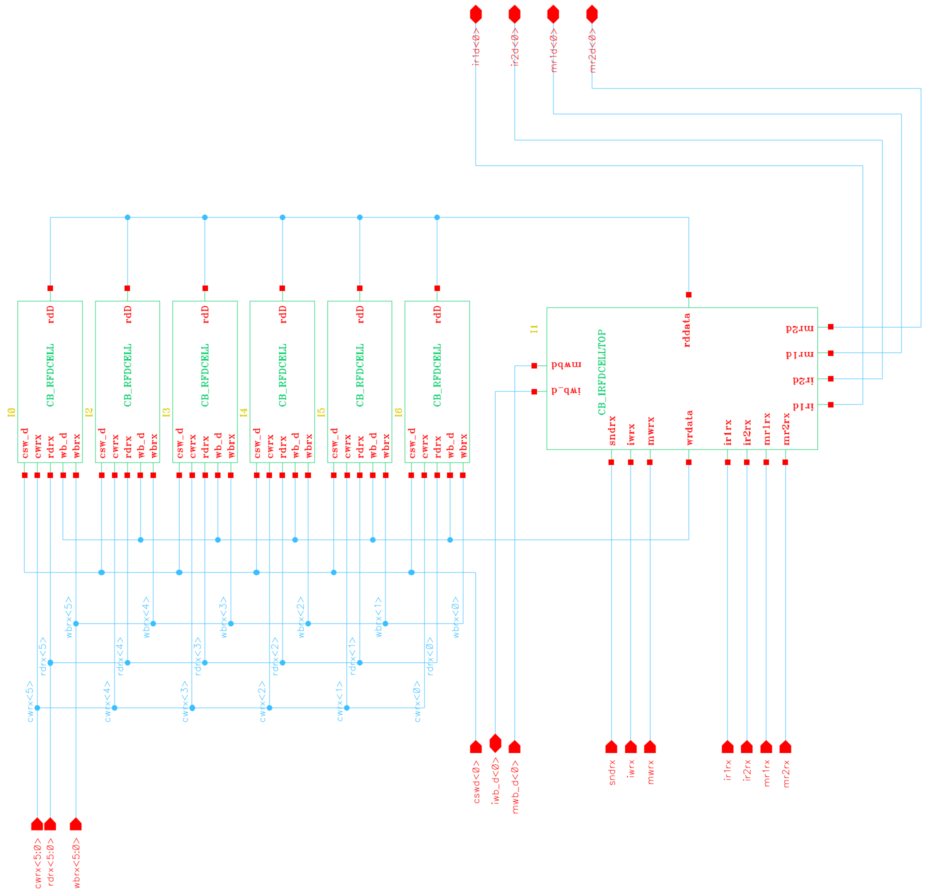
CB(auto)

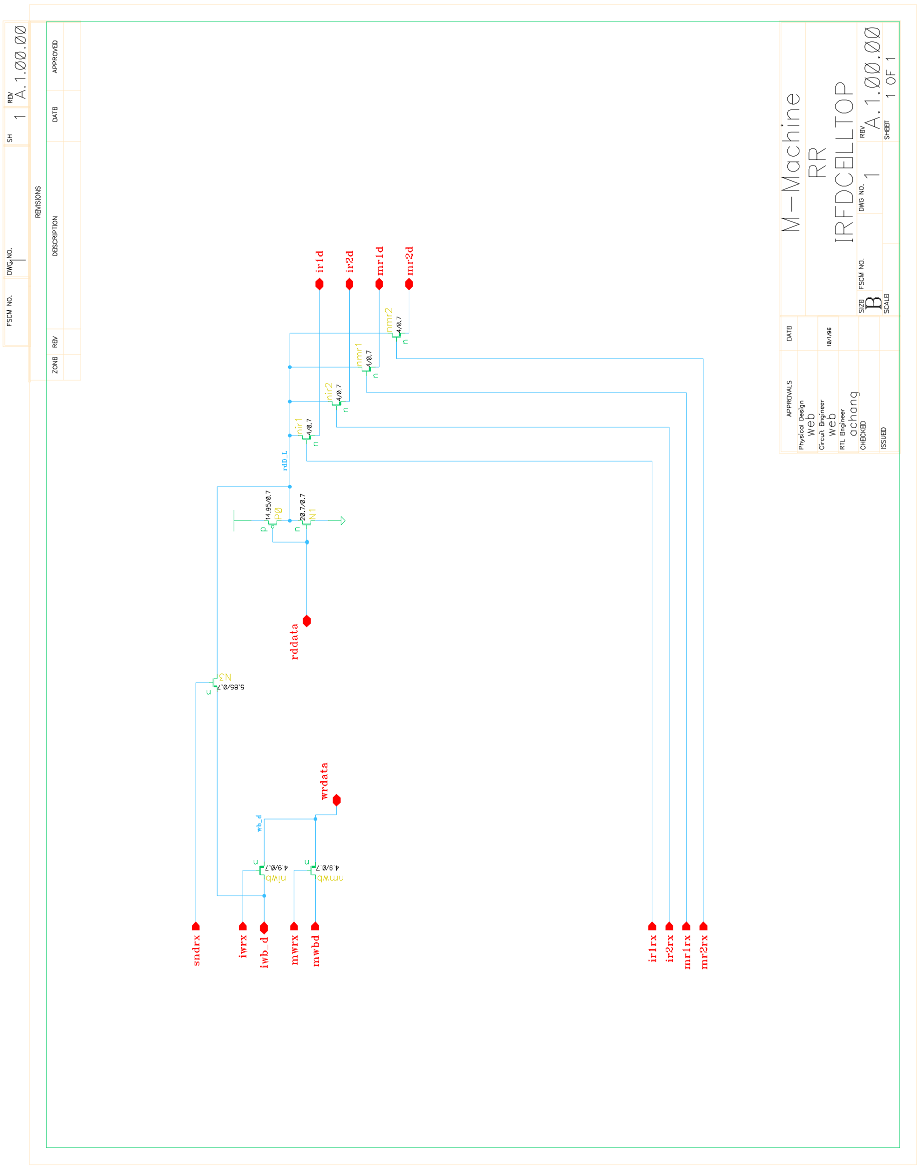
Appendix D

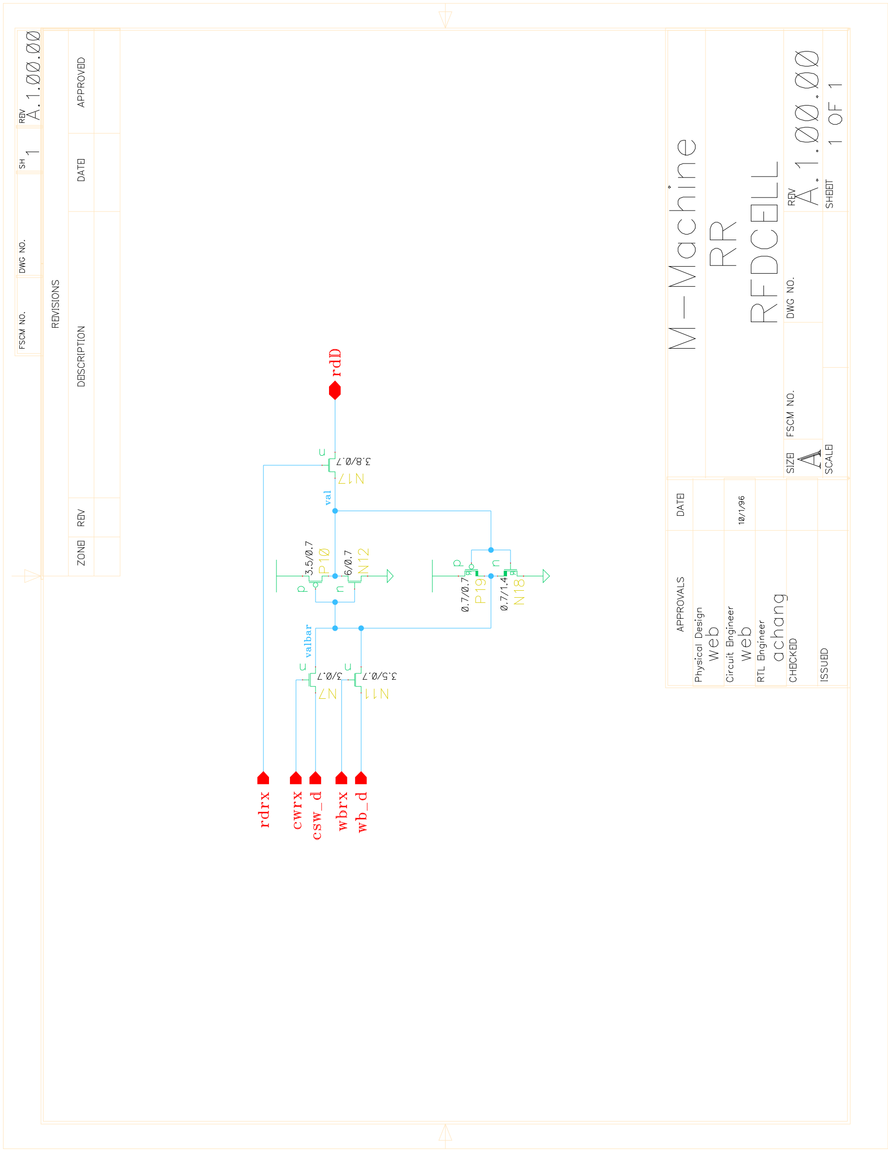
IRRDP Schematics

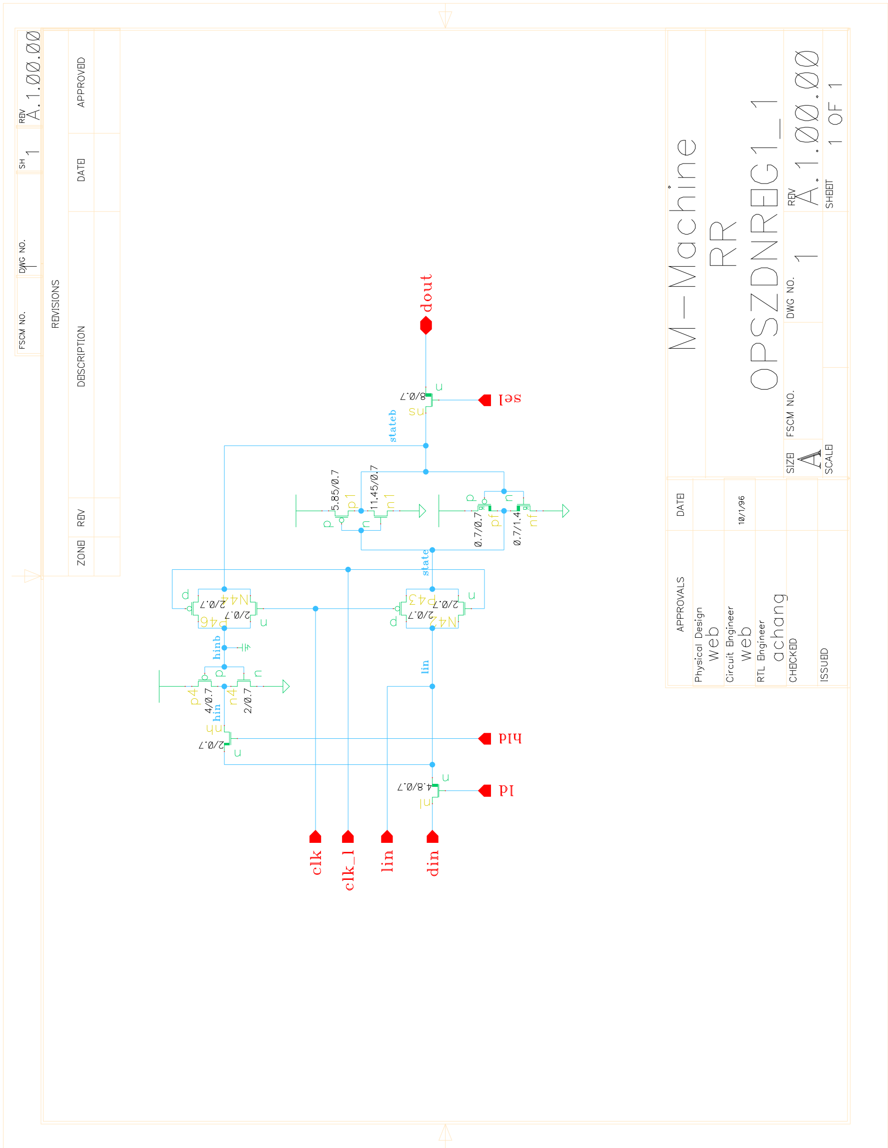

M-Machine
RR
IRRDP
A.2.00.00

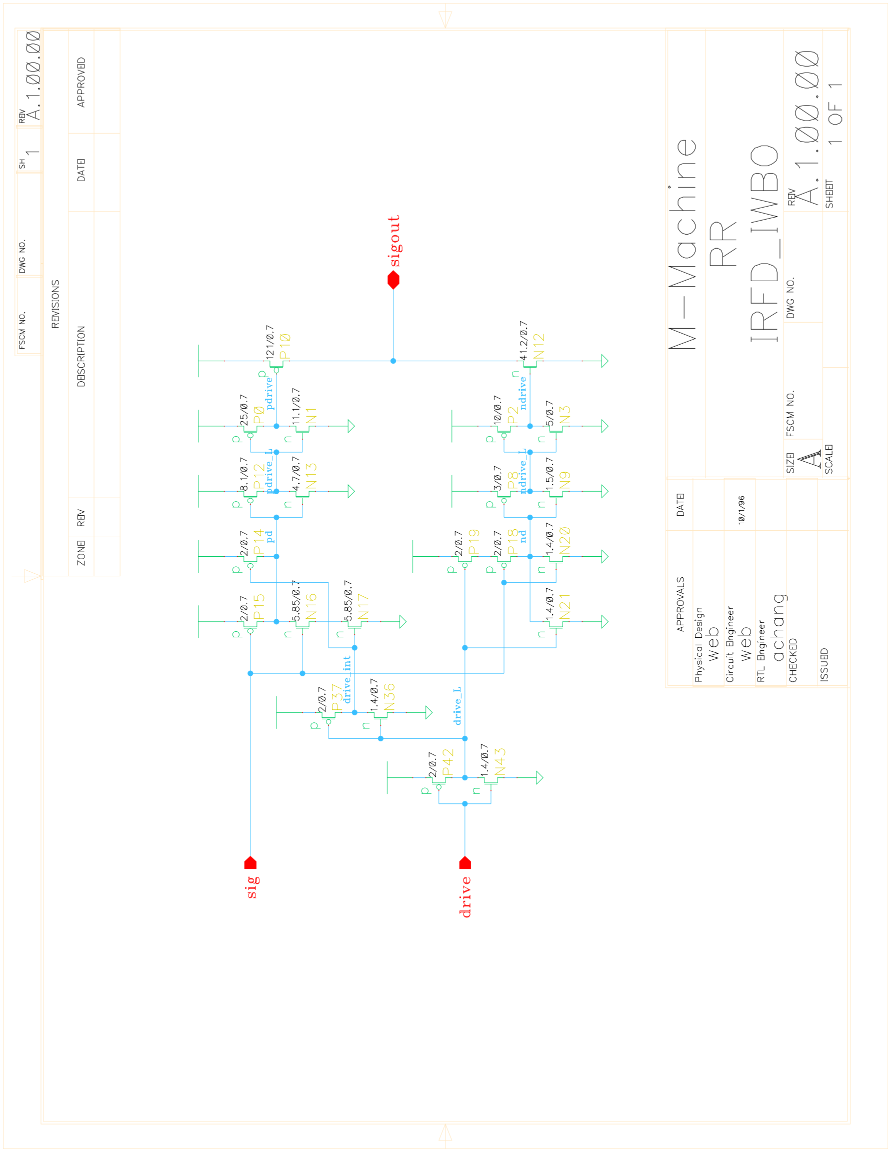

M−Machine
RR
IRFD
A.2.00.00


CB_RFDCELL
CB_IRFDCELLTOP
rdD
csw_d
cwrx
rdrx
wb_d
wbrx
rddata
wrdata
mwbd
iwb_d
mr2d
mr1d
ir2d
ir1d
sndrx
iwrx
mwrx
ir1rx
ir2rx
mr1rx
mr2rx
ir1d<0>
ir2d<0>
mr1d<0>
mr2d<0>
cswd<0>
iwb_d<0>
mwb_d<0>
cwrx<5:0>
rdrx<5:0>
wbrx<5:0>


M-Machine
RR
IRFDCELLTOP
A.1.00.00
1 OF 1
ir1d
ir2d
mr1d
mr2d
rddata
wrdata
sndrx
iwrx
iwb_d
mwrx
mwbd
ir1rx
ir2rx
mr1rx
mr2rx

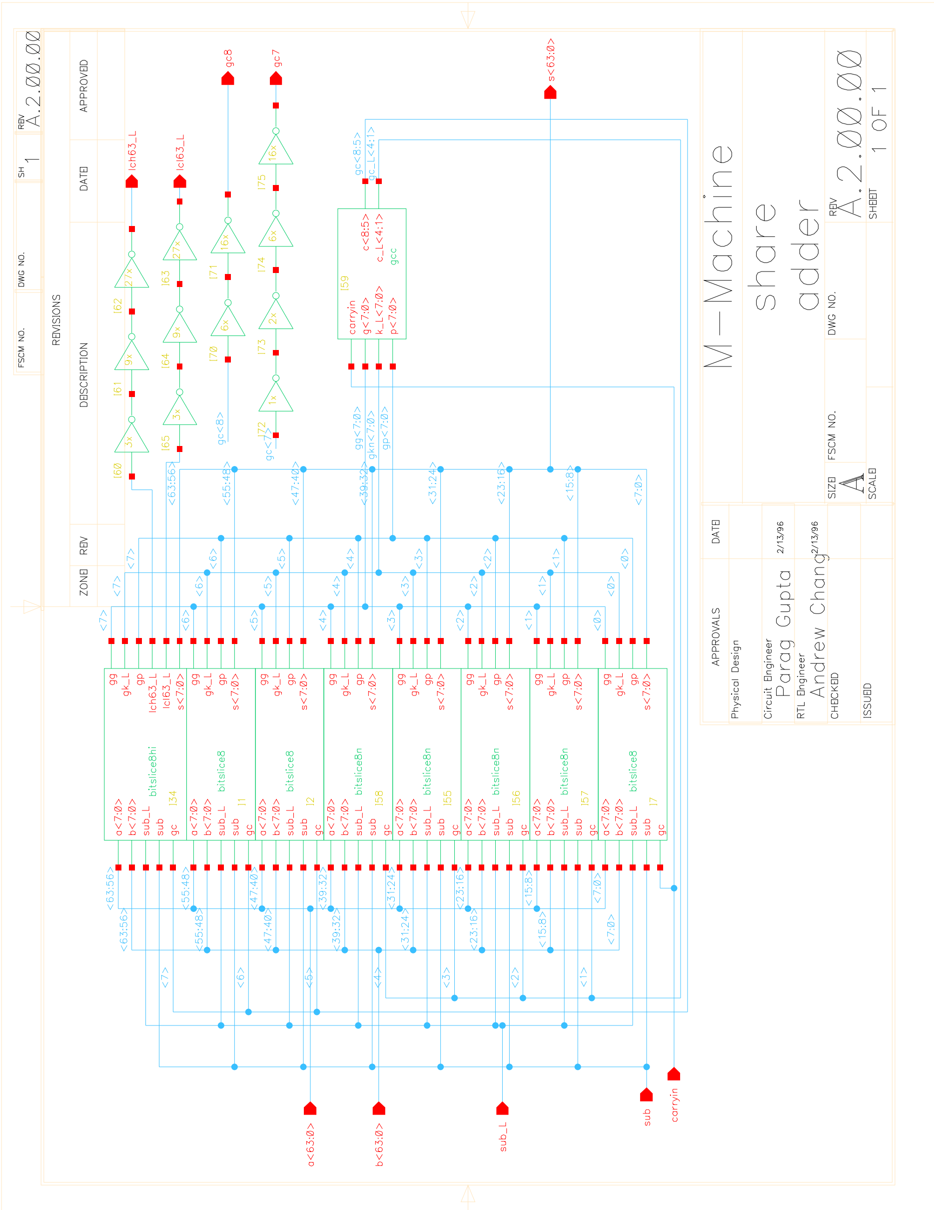

M−Machine
RR
RFDCELL
REV A.1.00.00
SHEET 1 OF 1
SIZE A
FSCM NO.
DWG NO.
SCALE
APPROVALS
DATE
Physical Design web
Circuit Engineer web
RTL Engineer achang
10/1/96
CHECKED
ISSUED
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPROVED
SH 1
rdD
N17
3.8/0.7
val
P10
3.5/0.7
N12
6/0.7
P19
0.7/0.7
N18
0.7/1.4
valbar
N7
3/0.7
N11
3.5/0.7
rdrx
cwrx
csw_d
wbrx
wb_d

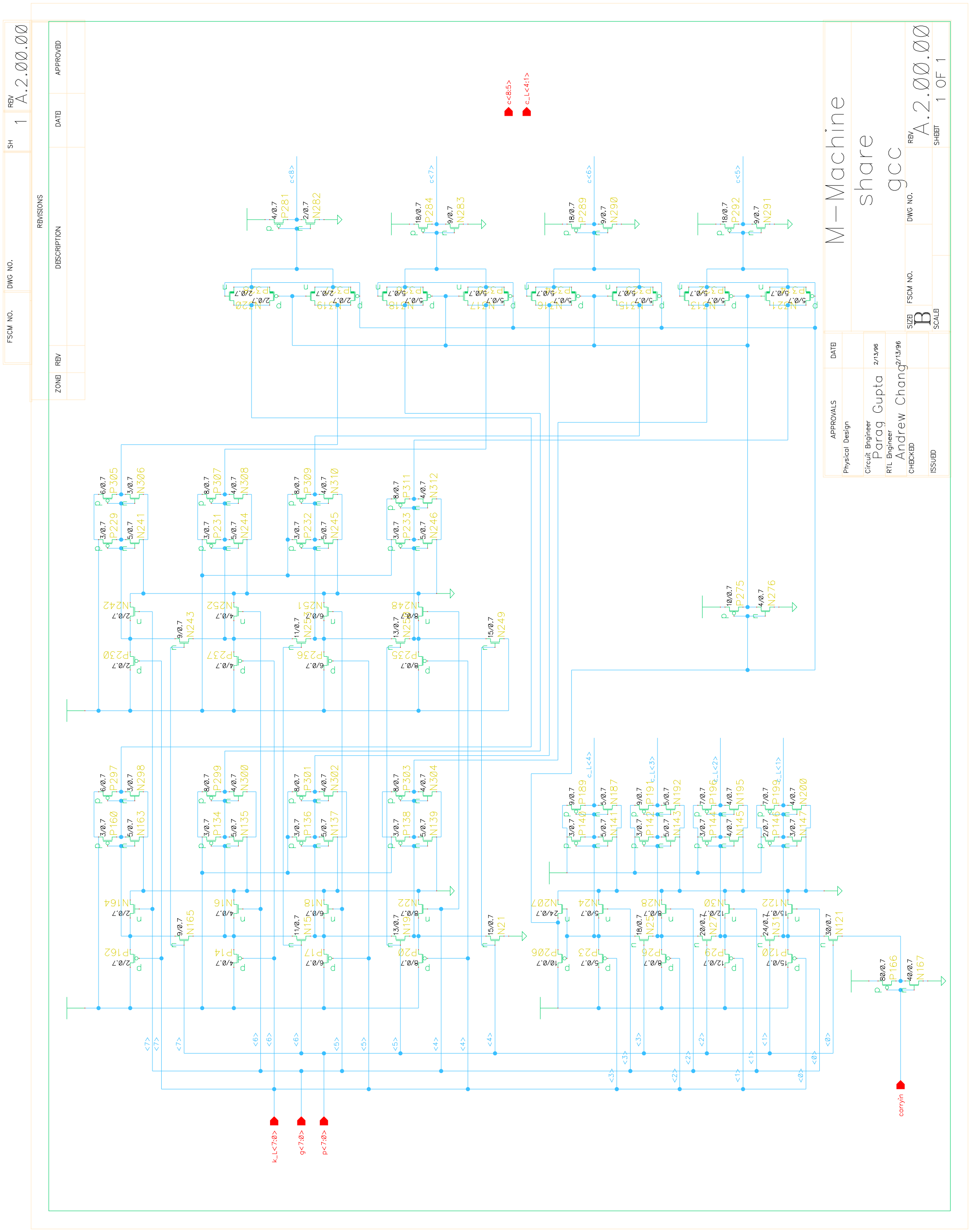

M—Machine
RR
OPSZDNREG1_1
A.1.00.00
1 OF 1
dout
sel
hld
ld
clk
clk_l
lin
din


M-Machine
RR
IRFD_IWBO
A.1.00.00
1 OF 1
sig
drive
sigout
web
achang
10/1/96

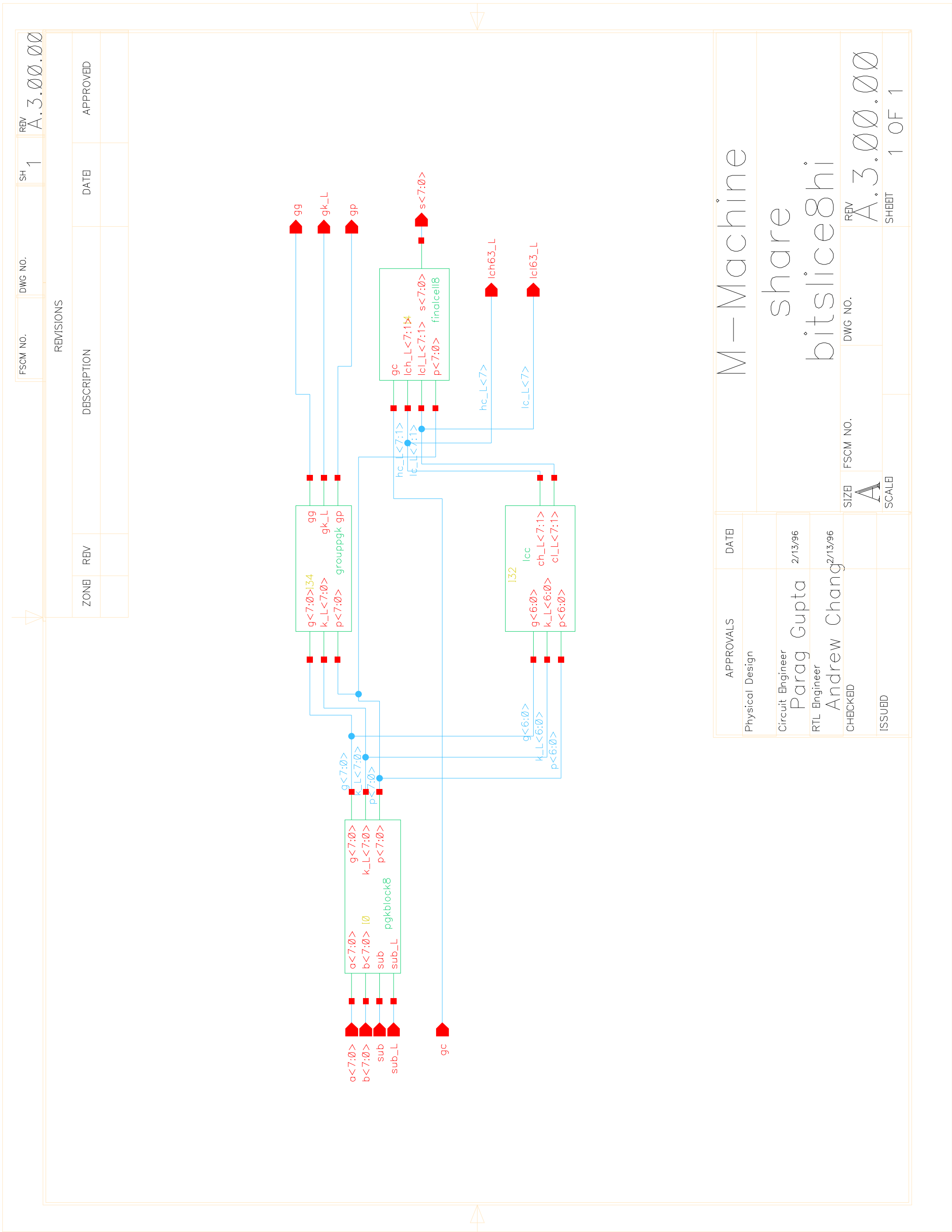
Appendix E

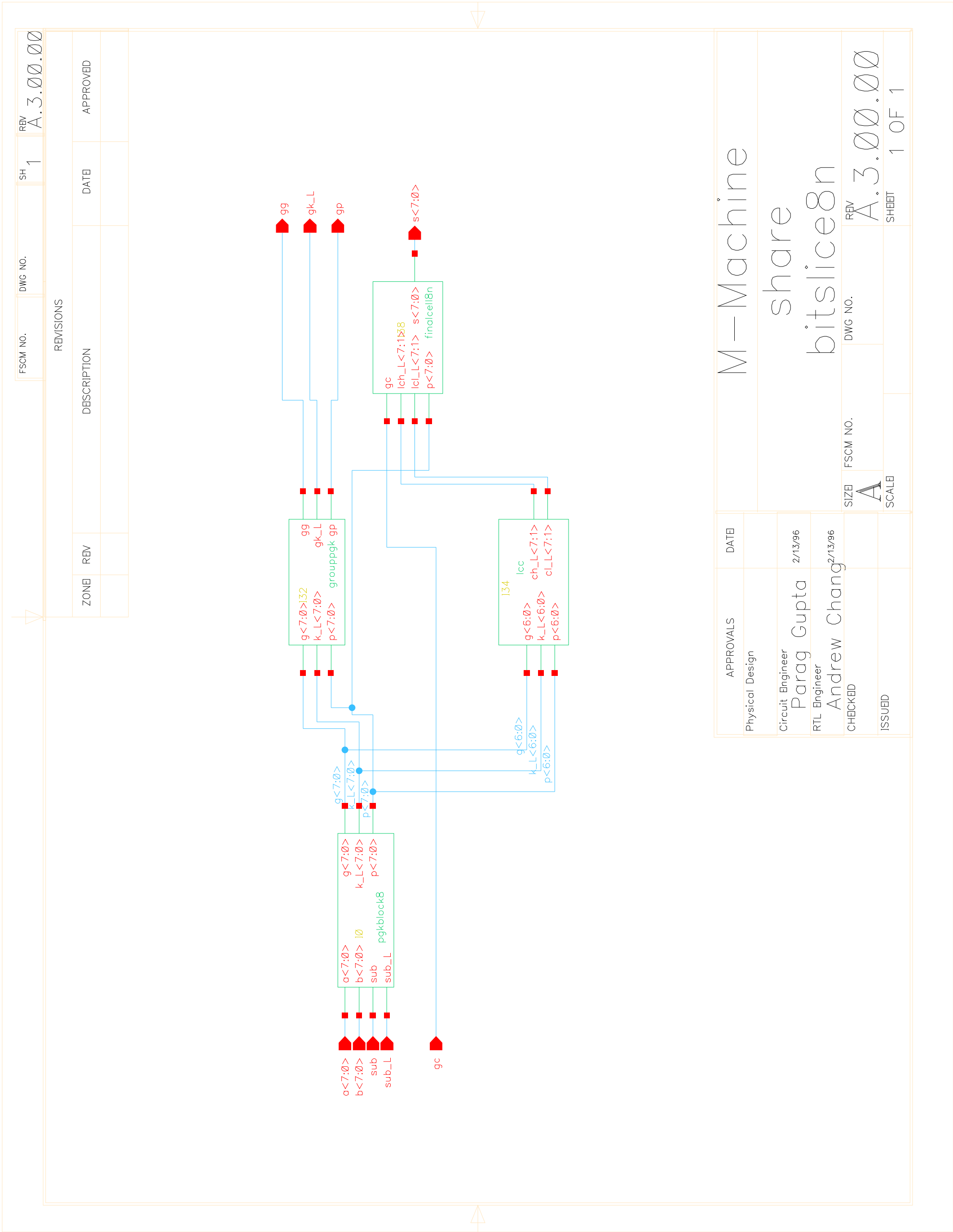
Adder Schematics

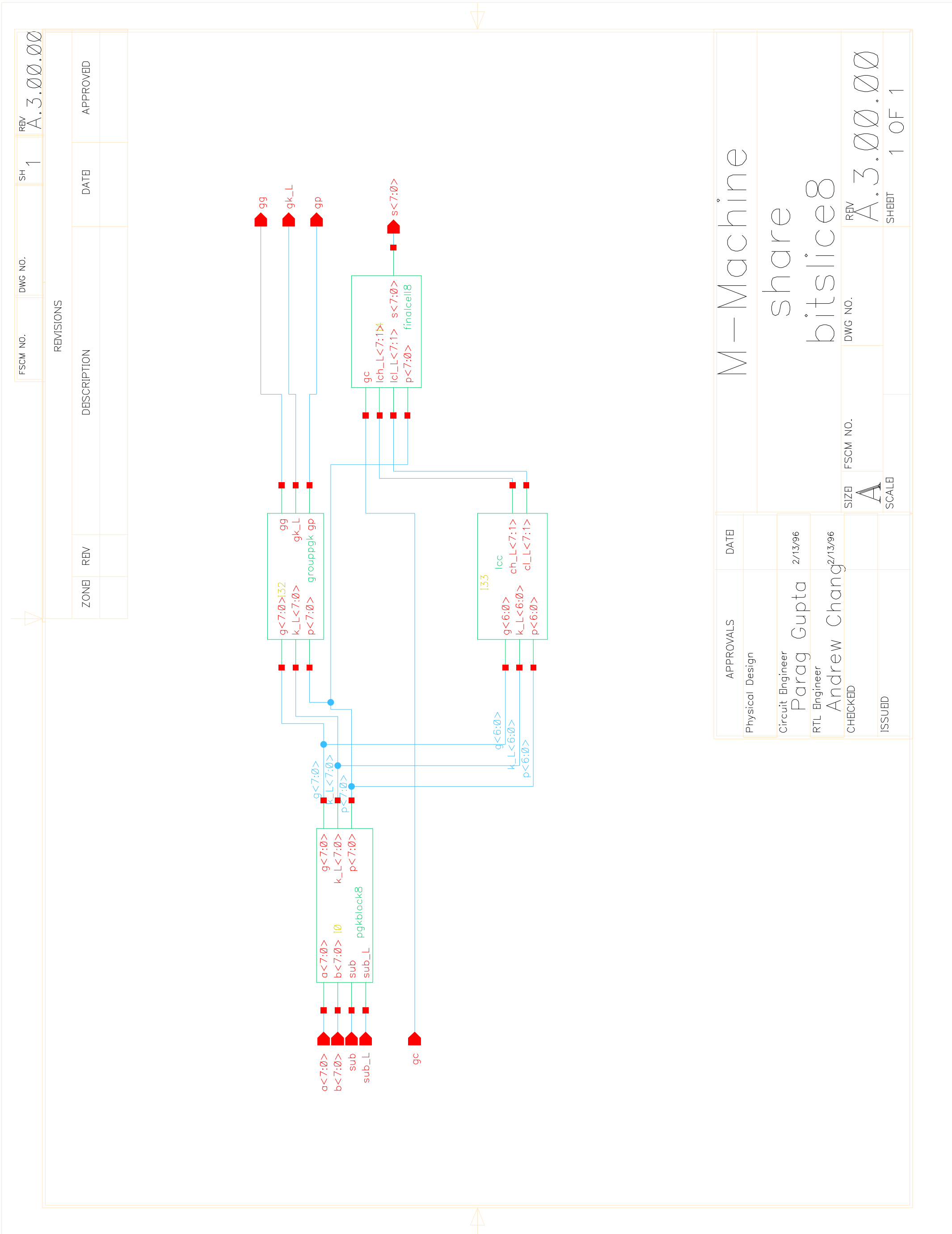

M—Machine
share
adder
A.2.00.00
1 OF 1
Parag Gupta
Andrew Chang
2/13/96
REVISIONS

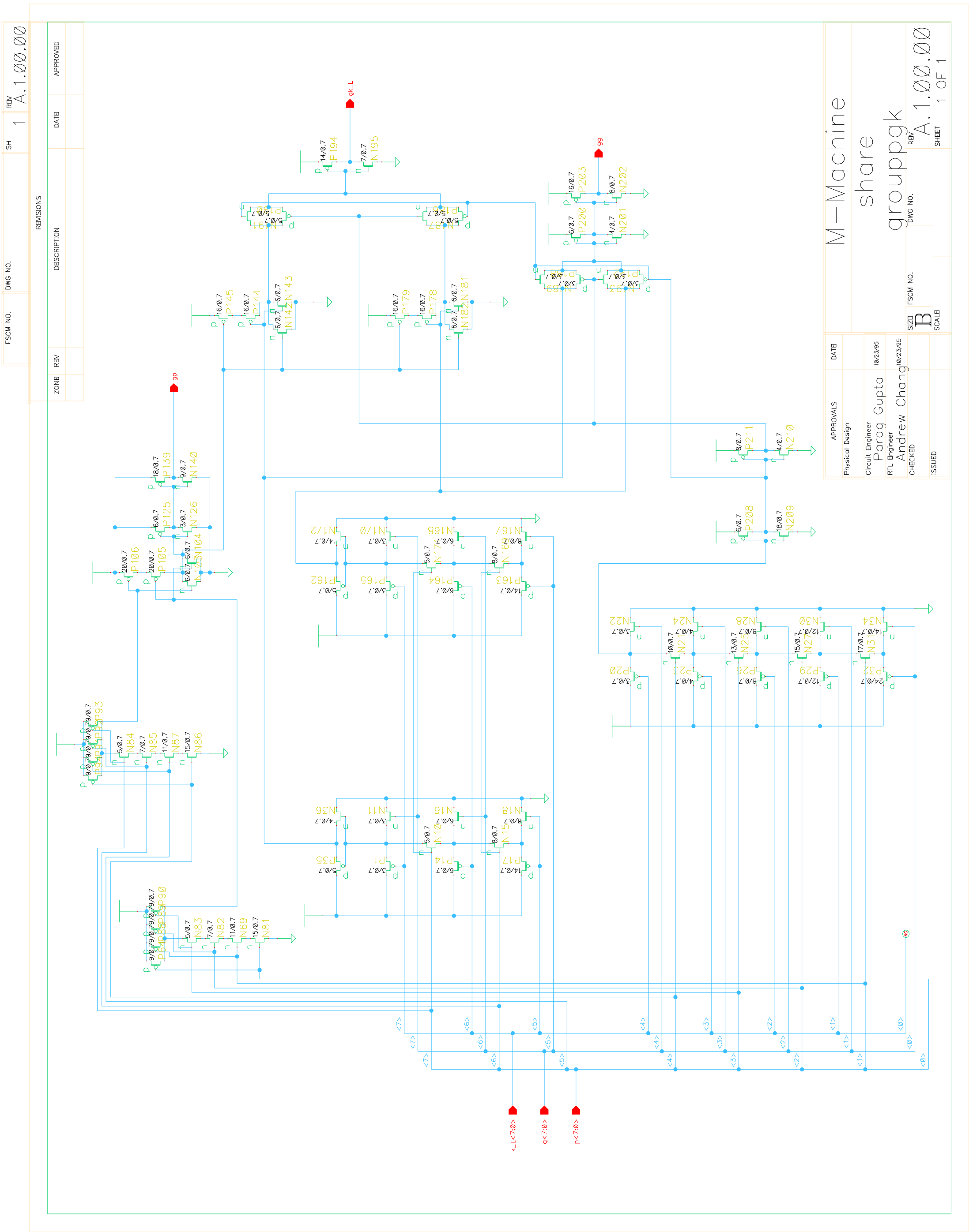


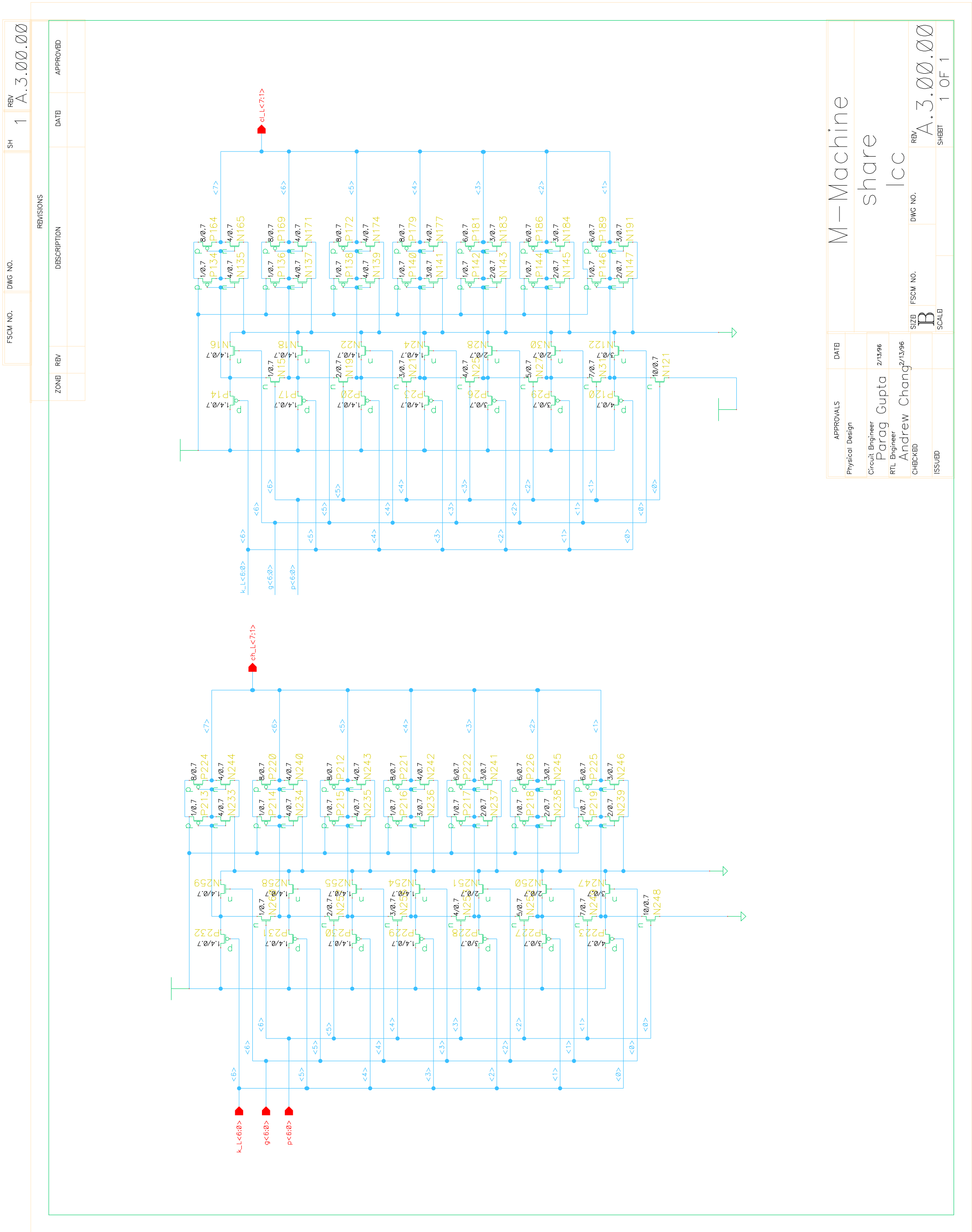
M−Machine
share
gcc
Parag Gupta
Andrew Chang
2/13/96
A.2.00.00
1 OF 1
B
carryin
k_L<7:0>
g<7:0>
p<7:0>

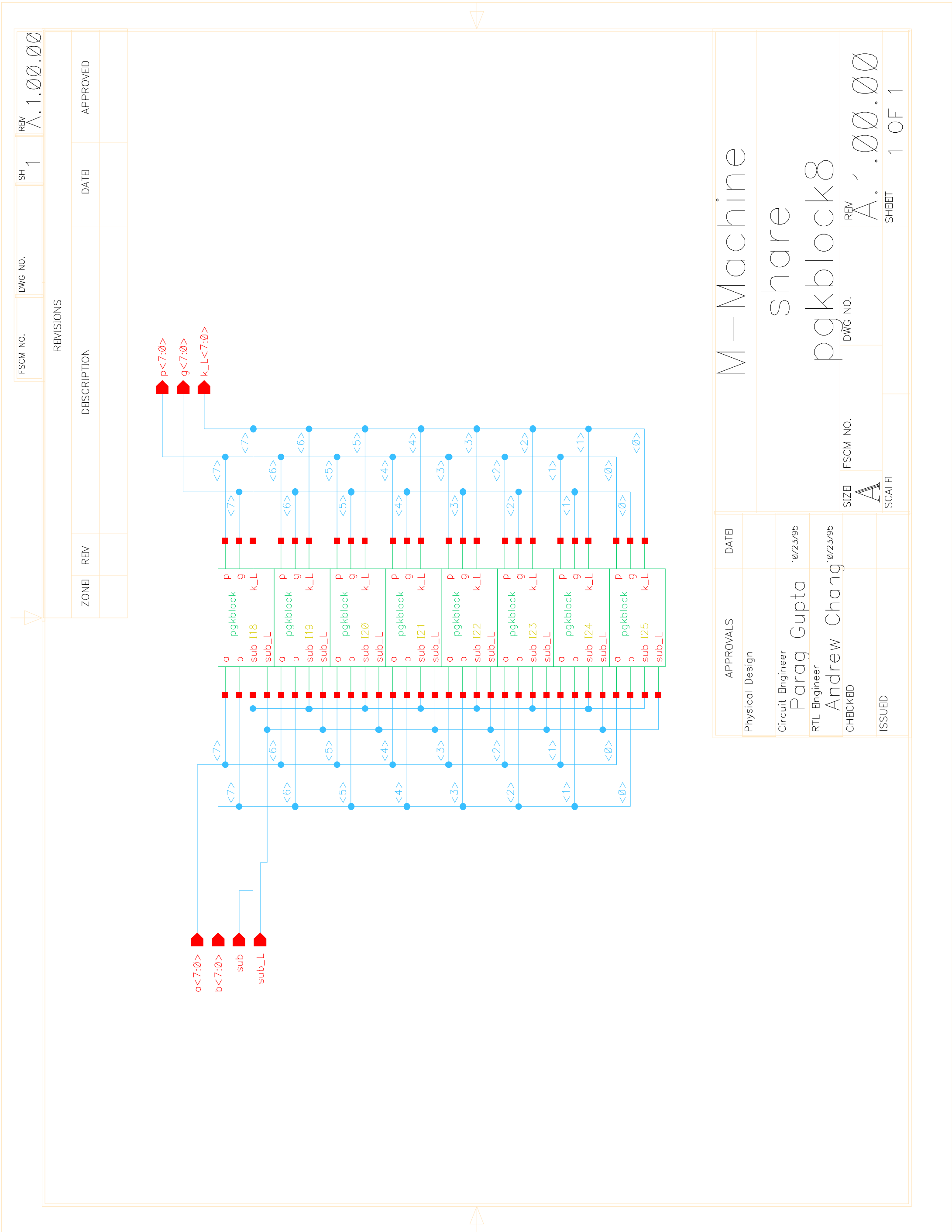

M—Machine
share
bitslice8hi
A.3.00.00
1 OF 1
APPROVALS
Physical Design
Circuit Engineer
Parag Gupta
2/13/96
RTL Engineer
Andrew Chang
2/13/96
CHECKED
ISSUED
DATE
SIZE
A
FSCM NO.
DWG NO.
REV
SHEET
SCALE
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPROVED
SH
1
pgkblock8
grouppgk
lcc
finalcell8
a<7:0>
b<7:0>
sub
sub_L
gc
gg
gk_L
gp
s<7:0>
lch63_L
lcl63_L

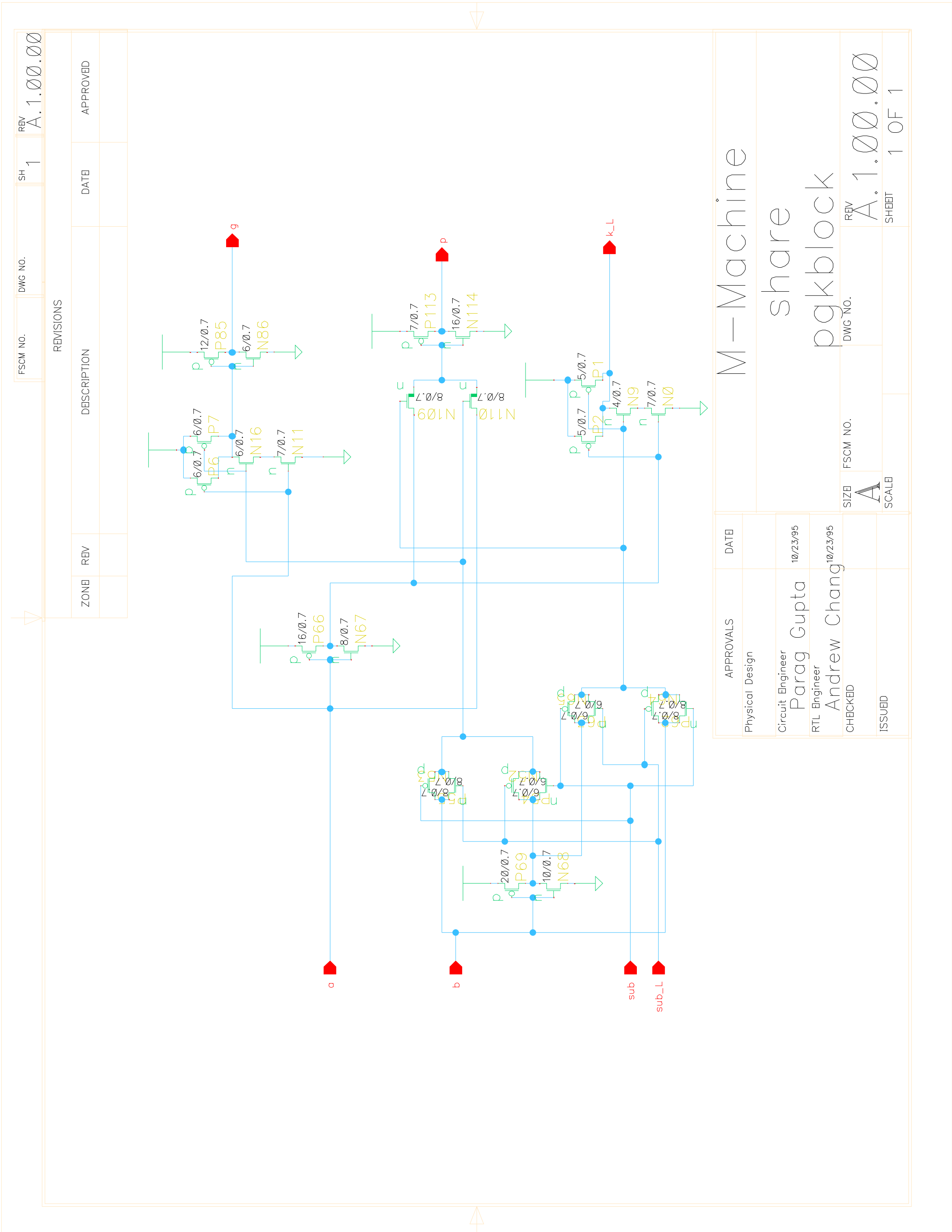

M—Machine
share
bitslice8n
A.3.00.00
1 OF 1
Parag Gupta
Andrew Chang
2/13/96
2/13/96
APPROVALS
Physical Design
Circuit Engineer
RTL Engineer
CHECKED
ISSUED
SIZE
A
FSCM NO.
DWG NO.
REV
SHEET
SCALE
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPROVED
pgkblock8
grouppgk
lcc
finalcell8n
a<7:0>
b<7:0>
sub
sub_L
gc
gg
gk_L
gp
s<7:0>

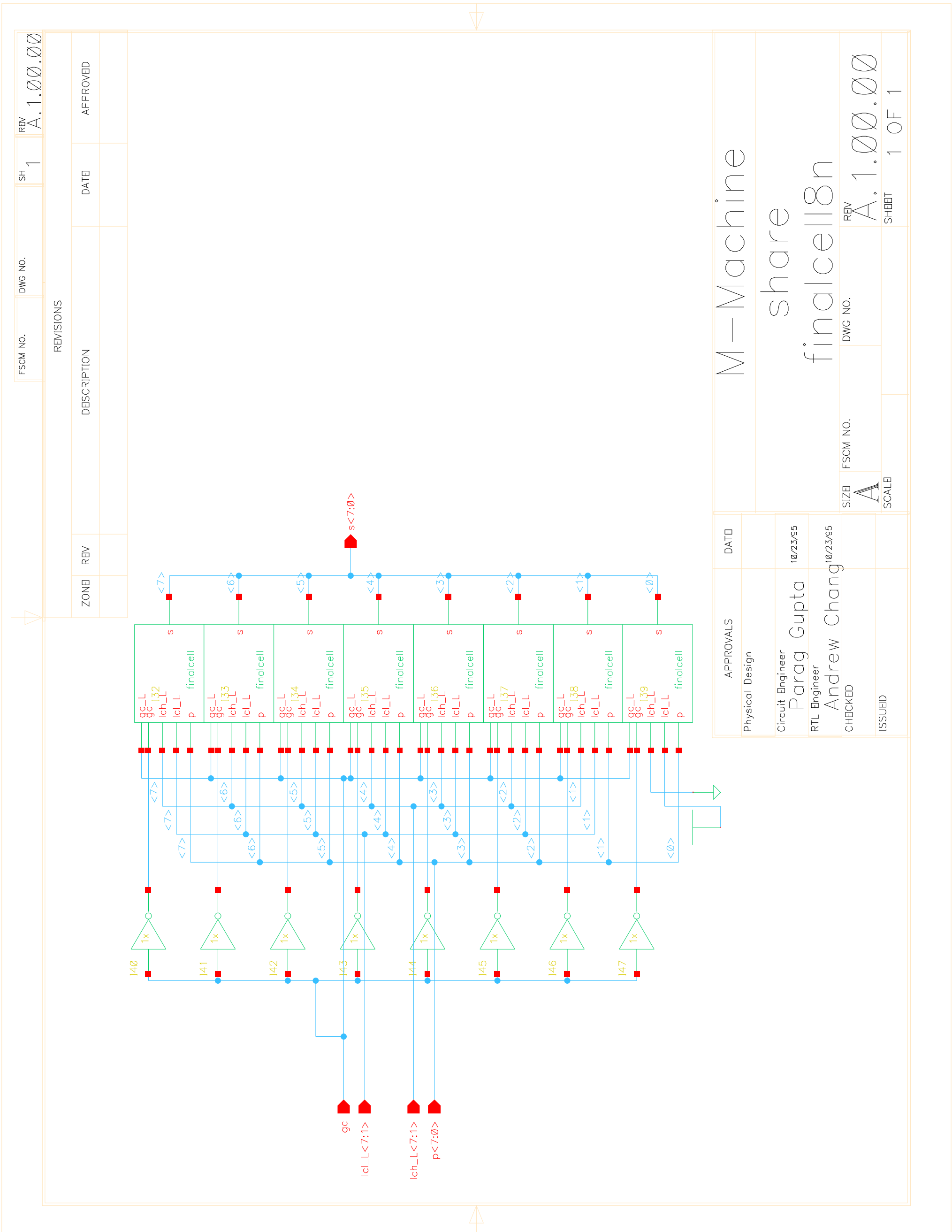

M-Machine
share
bitslice8
APPROVALS
DATE
Physical Design
Circuit Engineer
Parag Gupta
2/13/96
RTL Engineer
Andrew Chang
2/13/96
CHECKED
ISSUED
SIZE
A
FSCM NO.
DWG NO.
REV
A.3.00.00
SHEET
1 OF 1
SCALE
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPROVED
FSCM NO.
DWG NO.
SH
1
REV
A.3.00.00
gg
gk_L
gp
s<7:0>
gc
lch_L<7:1>
lcl_L<7:1>
p<7:0>
s<7:0>
finalcell8
g<7:0>
k_L<7:0>
p<7:0>
grouppgk
lcc
g<6:0>
k_L<6:0>
p<6:0>
ch_L<7:1>
cl_L<7:1>
a<7:0>
b<7:0>
sub
sub_L
pgkblock8
gc

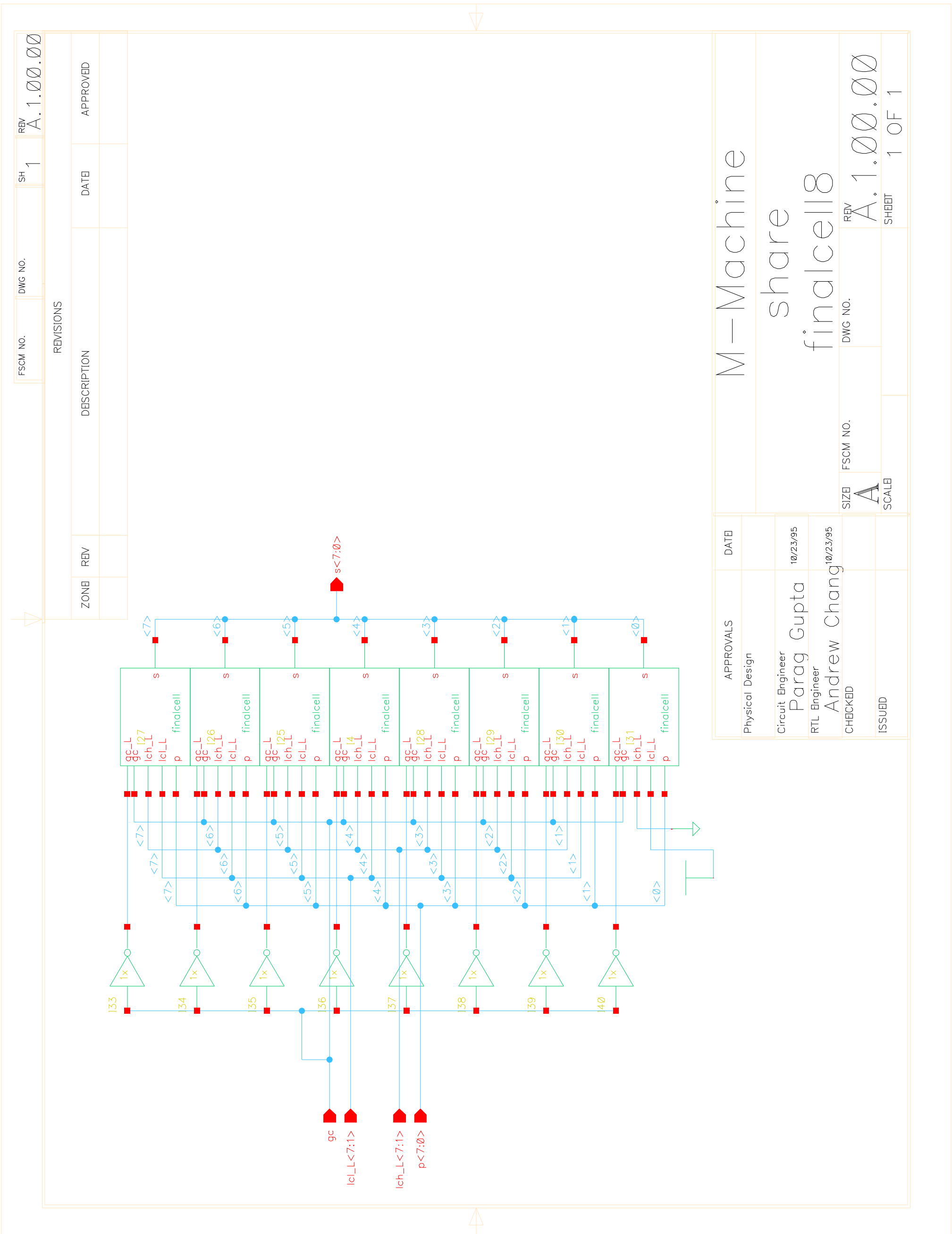

M-Machine
share
grouppgk
Parag Gupta
Andrew Chang
A.1.00.00
1 OF 1

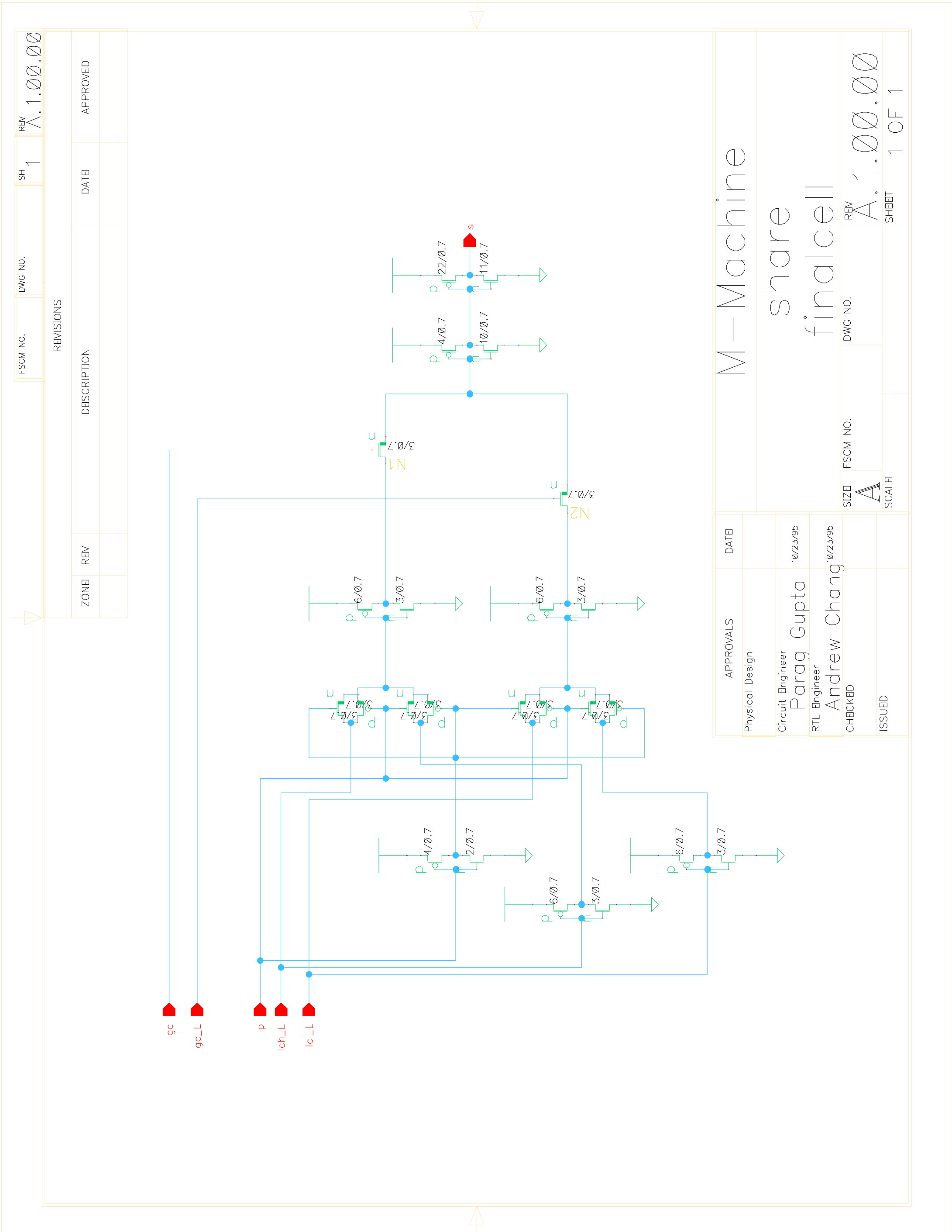

M—Machine
share
lcc
A.3.00.00
1 OF 1
Parag Gupta
Andrew Chang
2/13/96
ch_L<7:1>
k_L<6:0>
g<6:0>
p<6:0>


M-Machine
share
pgkblock8
A.1.00.00
SHEET 1 OF 1
SIZE A
FSCM NO.
DWG NO.
REV
SCALE
APPROVALS
DATE
Physical Design
Circuit Engineer
Parag Gupta
10/23/95
RTL Engineer
Andrew Chang
10/23/95
CHECKED
ISSUED
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPROVED
SH 1
REV A.1.00.00
p<7:0>
g<7:0>
k_L<7:0>
a<7:0>
b<7:0>
sub
sub_L
pgkblock I18
pgkblock I19
pgkblock I20
pgkblock I21
pgkblock I22
pgkblock I23
pgkblock I24
pgkblock I25


M—Machine
share
pgkblock
Circuit Engineer
Parag Gupta
10/23/95
RTL Engineer
Andrew Chang
10/23/95
APPROVALS
DATE
Physical Design
CHECKED
ISSUED
SIZE
A
SCALE
FSCM NO.
DWG NO.
REV
A.1.00.00
SHEET
1 OF 1
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPROVED
SH
1
g
p
k_L
a
b
sub
sub_L


M—Machine
share
finalcell8n
A.1.00.00
SHEET 1 OF 1
SIZE A
FSCM NO.
DWG NO.
REV
SCALE
APPROVALS
DATE
Physical Design
Circuit Engineer Parag Gupta 10/23/95
RTL Engineer Andrew Chang 10/23/95
CHECKED
ISSUED
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPROVED
SH 1
finalcell
s<7:0>
gc
lcl_L<7:1>
lch_L<7:1>
p<7:0>


M—Machine
share
finalcell8
A.1.00.00
1 OF 1
APPROVALS
Physical Design
Circuit Engineer
Parag Gupta
10/23/95
RTL Engineer
Andrew Chang
10/23/95
CHECKED
ISSUED
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPROVED
gc
lcl_L<7:1>
lch_L<7:1>
p<7:0>
s<7:0>



M—Machine
share
finalcell
REVISIONS
ZONE
REV
DESCRIPTION
DATE
APPROVED
FSCM NO.
DWG NO.
SH 1
REV A.1.ØØ.ØØ
APPROVALS
DATE
Physical Design
Circuit Engineer
Parag Gupta
10/23/95
RTL Engineer
Andrew Chang
10/23/95
CHECKED
ISSUED
SIZE A
FSCM NO.
DWG NO.
REV A.1.ØØ.ØØ
SCALE
SHEET 1 OF 1
N1
N2
gc
gc_L
p
lch_L
lcl_L
s

Appendix F

Cell Libraries

F.1 *mitstdcells*

ALLLIST   Tue Feb 3 01:37:22 1998   1

```
AND2
AND2_2
AND3
AND4
AOI21
AOI211
AOI2111
AOI22
AOI221
AOI2211
AOI222
AOI2221
AOI2222
AOI2222_4
AOI31
AOI32
AOI33
ARB_CHAIN3
ARB_CHAIN4
ARB_CHAIN4_BUF
ARB_CHAIN5
ARB_CHAIN5_BUF
ARB_CHAIN6
ARB_CHAIN6_BUF
BUF_16
BUF_2
BUF_32
BUF_4
BUF_8
CCSZ_CELL_BASE
CKA_1
CKA_2
CKA_4
CKA_8
CLKDEL
DIP_CELL_0
DIP_CELL_1
EMI_M2_NL
EMI_M4_NL
FRFD_CELL_MSZPTR_BASE
GRCCD_CELL_BASE
INV_1
INV_16
INV_1R
INV_2
INV_32
INV_4
INV_8
LCCD_CELL_BASE
NAND2
NAND2_2
NAND3
NAND4
NAND4_4
NCKA_1
NCKA_2
NCKA_4
NCKA_8
NDFF_CELL
NLTCH_SCAN
NOR2
NOR2_2
NOR3
NOR4
NOR4_4
OAI21
OAI211
OAI2111
OAI22
OAI221
OAI2211
OAI222
OAI2221
OAI2222
OAI31
OAI31_4
OAI32
OAI32_4
OAI33
OAI33_4
OR2
OR3
OR4
PDFF_SCAN
PLTCH_CELL
REG_CELL
RR_M0_NL
RR_M3_NL
SZ_M0_PL
SZ_M2_PL
SZ_M3_PL
TINV_1
TINV_2
TINV_4
TINV_8
XNOR2
XNOR2_4
XOR2
XOR2_4
```

F.2 *share*


```
TRUE "SHARE"
------------
LZDetect.lef
Latchi.lef
addboth.lef
and1_dp.lef
and2_dp.lef
and3_dp.lef
driver.lef
fadder_cvsl.lef
inv1.lef
inv15.lef
inv16.lef
inv2.lef
inv27.lef
inv3.lef
inv4.lef
inv4_1.lef
inv5.lef
inv6.lef
inv64.lef
inv81.lef
inv9.lef
keeper_10_15.lef
keeper_10_20.lef
keeper_16_32.lef
keeper_4_8.lef
mux2.lef
mux4.lef
mux6.lef
sh_bit.lef
sh_bit4.lef
sh_wrap_high.lef
sh_wrap_low.lef
simple_comp_latchi.l


FILLERS
--------
dpfill1.lef
dpfill2.lef
dpfill4.lef
dpfill8.lef
dpfill16.lef
dpfill32.lef
dpfill64.lef

STITCH m2
---------
stitch2x64.l
stitch2x66.l
stitch2x68.l

STITCH m3
---------
stitch3x1.le
stitch3x1h.l
stitch3x32.l
stitch3x4h.l
stitch3x8.le
stitch3x16.l
stitch3x2.le
stitch3x32h.l
stitch3x64.l
stitch3x8h.l
stitch3x16h.
stitch3x2h.l
stitch3x4.le
stitch3x64h.l

IU
-------
adder_iu.lef
bool_unit.lef
mask_byte_ctrl.lef
mask_byte_ctrl.lef
mux3.lef
muxa2.lef
muxb2.lef
muxc2.lef
sc_addr_comp.lef


MU
-----
adder_mu.lef
format_driver.lef

DCDP
----
Latch_ld.lef
Latch_ld_as_modified_b
Latch_ld_rst.lef
Latch_ld_rst_as_modifi
c_and1.lef
c_and2.lef
c_and3.lef
domino_xor.lef
mux2_d.lef
st1.lef
st2.lef
st3.lef
tribuf.lef
xxor.lef

FDSQ
-----
adder_dsq.lef
bus_drive.lef
cla_final.lef
fgen_mux.lef
inv1_dsq.lef
latch.lef
latch_dsq.lef
latch_dsq2.lef
qmux.lef
qmux0.lef
qmux1.lef
qmux2.lef
qmux3.lef
rem_drive.lef

FMULA
-----
adder_mula.lef
bencode22.lef
boundary.lef
d_mux4.lef
lz_decode.lef
m_array.lef
pcombiner1.lef
pcombiner2.lef
```

Bibliography

[1] Adams, G., and Sequin, C. Template Style Considerations for Sea-of-Gates Layout Generation. In *Proceedings of 26th ACM/IEEE Design Automation Conference* (1989), IEEE, pp. 31–36.

[2] Alpert, D., and Avnon, D. Architecture of the Pentium Microprocessor. *IEEE Micro* (June 1993), 11–21.

[3] Ammar, B. L., and Greiner, A. A High Density Datapath Compiler Mixing Random Logic with Optimized Blocks. In *Proceedings of ICCD* (1993), IEEE, pp. 194–198.

[4] Asprey, T., Averill, G. S., DeLano, E., Mason, R., Weiner, B., and Yetter, J. Performance Features of the PA7100 Microprocessor. *IEEE Micro* (June 1993), 22–35.

[5] Auvergne, D., Azemard, N., Bonzom, V., Deschacht, D., and Robert, M. Formal Sizing Rules of CMOS Circuits. In *Proceedings of ICCD* (1991), IEEE, pp. 96–100.

[6] Bachelu, C., and Lefebvre, M. A Study of the Use of Local Interconnect in CMOS Leaf Cell Design. *IEEE Transactions on VLSI Systems 1*, 4 (Dec. 1993), 566–571.

[7] Bass, M., Blanchard, T. W., Weir, D. D. J. D., and Halperin, D. L. Design Methodologies for the PA7100LC Microprocessor. *Hewlett Packard Journal* (April 1995), 23–35. PA7100LC, Design Methodologies.

[8] BECKER, M. C., ALLEN, M. S., MOORE, C. R., MUHICH, J. S., AND TUTTLE, D. P. The PowerPC 601 Microprocessor. *IEEE Micro* (Oct. 1993), 54–68.

[9] BISHOP, J., CAMPION, M. J., JEREMIAH, T. L., MERCIER, S. J., MOHRING, E. J., PFARR, K. P., RUDOLPH, B. G., STILL, G., AND WHITE, T. S. PowerPC AS A10 64-bit RISC microprocessor. *IBM Journal of Research and Developement* (September 1996).

[10] BONDONO, P., JERRAYA, A. A., HORNIK, A., COURTOIS, B., AND BONIFAS, D. Nautile: A Safe Environment for Silicon Compilation. In Proceedings of ICCD (1990), IEEE, pp. 605–609.

[11] BOWHIL, W. A 300 MHz 64b Quad-Issue CMOS Microprocessor. In ISSCC'95 Digest of Technical Papers (1995), IEEE, pp. 182–183.

[12] BRODNAX, T. B., BILLINGS, R. V., GLENN, S. C., AND PATEL, P. T. Implementation of the PowerPC 601 microprocessor. *IBM Journal of Research and Developement* (September 1994), 621–631.

[13] BURD, T. Web Page: CPU Info Center - Microprocessor Info. http://infopad.eecs.berkeley.edu/CIC/.

[14] CHEN, H. Pseudo Pin Assignment for Single-Layer Over-the-Cell Routing. In Proceedings of ICCD (1990), IEEE, pp. 343–346.

[15] CHEN, X., AND BUSHNELL, M. L. A Module Area Estimator for VLSI Layout. In *Proceedings of 25th ACM/IEEE Design Automation Conference* (1988), IEEE, pp. 54–59.

[16] CHOUDHURY, M. R., AND MILLER, J. S. A 300 MHz CMOS Microprocessor with Multi-MediaTechnology. In ISSCC'97 Digest of Technical Papers (1997), IEEE, pp. 170–171.

[17] CHU, K. M., AND PULFREY, D. L. A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic Versus Conventional Logic.

IEEE Journal of Solid State Circuits (August 1987), 528–531. CVSL, Circuit Techniques.

[18] Cohen, E. T., Ballard, J., Blomgren, J., Brashears, C. S., Moldenhauer, V., and Pattin, J. A 533MHz BiCMOS Superscalar Microprocessor. In ISSCC'97 Digest of Technical Papers (1997), IEEE, pp. 164–165.

[19] Colwell, R. P., and Steck, R. L. A 0.6μm BiCMOS processor with Dynamic Execution. In ISSCC'95 Digest of Technical Papers (1995), IEEE, pp. 176–177.

[20] Curry, D. Schematic Specification of Datapath Layout. In Proceedings of ICCD (1989), IEEE, pp. 28–34.

[21] Dally, W. J. 6.845 Class Notes, Fall 1995. CSL, Stanford University.

[22] Dally, W. J. Communication-Oriented Architecure, CVA Group Meeting Winter 1998. CSL, Stanford University.

[23] Dally, W. J., Fiske, J. A. S., Keen, J. S., Lethin, R. A., Noakes, M. D., Nuth, P. R., Davidson, R. E., and Fyler, G. A. The Message Driven Processor. *IEEE Micro* (April 1992), 23–39. MDP, J-Machine, Computer Architecture, Fine Grain.

[24] Dally, W. J., Keckler, S. W., Carter, N. P., Chang, A., Fillo, M., and Lee, W. S. M-Machine Architecture v1.0.

[25] Dally, W. J., and Poulton, J. *Digital Systems Engineering.* 1998.

[26] Dennison, L. *The Reliable Router: An Architecture for Fault Tolerant Interconnect.* PhD thesis, Massachusetts Institute of Technology, May 1996.

[27] Digital Semiconductor - DEC Alpha 21164 Die Photo. http://infopad.eecs.berkeley.edu/CIC/die_photos/21164.gif.

[28] Dobberpuhl, D. W., Witek, R. T., Allmon, R., Anglin, R., Bertucci, D., Britton, S., Chao, L., Conrad, R. A., Dever, D. E., Gieseke, B., Hassoun, S. M. N., Hoeppner, G. W., Kuchler, K., Ladd, M., Leary, B. M., Madden, L., McLellan, E. J., Meyer, D. R., Montanaro, J., Priore, D. A., Rajaopalan, V., Samudrala, S., and Santhanam, S. A 200MHz 64-b Dual Issue CMOS Microprocessor. *IEEE Journal of Solid State Circuits* (November 1992), 1555–1567.

[29] Draper, D. A., Crowley, M. P., Hoist, J., Favor, G., Schoy, A., Ben-Meir, A., Trull, J., Khanna, R., Wendell, D., Krisha, R., Nolan, J., Partovi, H., Johnson, M., Lee, T., Mallick, D., Frydel, G., Vuong, A., Yu, S., Maley, R., and Kauffmann, B. An X86 Microprocessor with Multimedia Extensions. In ISSCC'97 Digest of Technical Papers (1997), IEEE, pp. 172–173.

[30] Druke, M., Gusowski, R., Buckley, E., Carberry, D., March, R., and Feaver, R. *LSI processor mirrors high-performance minicomputer*, first ed. McGraw-Hill, New York, New York, 1981, pp. 118–125.

[31] Dutt, D., and Lakhani, G. Optimization for Automatic Cell Assembly. In Proceedings of ICCD (1988), IEEE, pp. 186–189.

[32] Feigel, C. Motorola Introduces Heir to 68000 Line . *Microprocessor Report* (April 18 1994), 10–13.

[33] Feigel, C. TI Introduces Four Processor DSP Chip. *Microprocessor Report* (March 1994), 22–25.

[34] Fey, C. Custom LSI/VLSI Chip Design Productivity. *IEEE Journal of Solid State Circuits* (April 1985), 555–561. VLSI, Effort, Productivity.

[35] Fey, C., and Parakevopoulos, D. E. Studies in LSI Techology Economics II: A Comparison of Product Costs using MSI, Gate Arrays, Standard Cells and

Full Custom VLSI. *IEEE Journal of Solid State Circuits* (April 1986), 297–303. VLSI, Effort, Productivity.

[36] Fey, C., and Parakevopoulos, D. E. Studies in LSI Techology Economics II: Models for Gate Array Design Productivity. *IEEE Journal of Solid State Circuits* (August 1989), 1085–1091. VLSI, Effort, Productivity.

[37] Fillo, M., Keckler, S., Dally, W., Carter, N., Chang, A., Gurevich, Y., and Lee, W. S. The M-Machine Multicomputer. In 28nd Annual International Symposium on Microarchitecture (1994), IEEE.

[38] Gao, T. *Performance Driven Placement and Routing Algorithms.* PhD thesis, University of Illinois at Urbana-Champaign, May 1994.

[39] Gieseke, B. A., Allmon, R. L., Bailey, D. W., Benschneider, B. J., Britton, S. M., Clouser, J. D., III, H. R. F., Farrell, J. A., Gowan, M. K., Houghton, C. L., Keller, J. B., Lee, T. H., Leibholz, D. L., Lowell, S. C., Matson, M. D., Matthew, R. J., Peng, V., Quinn, M. D., Priore, D. A., Smith, M. J., and Wilcox, K. E. A 600MHz Superscalar RISC Microprocessor with Out-of-Order Execution. In ISSCC'97 Digest of Technical Papers (1997), IEEE, pp. 176–177.

[40] Gray, J., Naylor, A., Abnous, A., and Bagherzadeh, N. VIPER: A VLIW Integer Microprocessor. *IEEE Journal of Solid State Circuits* (December 1993), 1377–1382. Microprocessor, VLSI, VLIW.

[41] Greenhill, D., Anderson, E., Bauman, J., Charnas, A., Cheerla, R., Chen, H., Doreswamy, M., Ferolito, P., Gopaladhine, S., Ho, K., Hsu, W., Kongetira, P., Melanson, R., Reddy, V., Salem, R., Sathianathan, H., Shah, S., Shin, K., Srivatsa, C., and Weisenbach, R. A 330MHz 4-Way Superscalar Microprocessor. In ISSCC'97 Digest of Technical Papers (1997), IEEE, pp. 166–167.

[42] Gronowski, P. A 433 MHz 64b Quad-Issue CMOS RISC Microprocessor. In ISSCC'96 Digest of Technical Papers (1996), IEEE, pp. 222–223.

[43] Guan, B., and Sechen, C. Large Standard Cell Libraries and Their Impact on Layout Area and Circuit Performance. In *Proceedings of ICCD* (1996), IEEE, pp. 378–383.

[44] Gupta, P. Design and implementation of the integer unit datapath of the map cluster of the m-machine. Master's project, Massachusetts Institute of Technology, EE/CS Department, May 1996.

[45] Gwennap, L. 1992 in Review: The Top RISC Processors. *Microprocessor Report* (December 30 1992), 1, 6–9, 10.

[46] Gwennap, L. Alpha Hits Low End with Digital's 21066. *Microprocessor Report* (September 13 1993), 1,6–8,16.

[47] Gwennap, L. Alpha Maintains Performance Lead in 1993. *Microprocessor Report* (December 27 1993), 1, 6–8, 10.

[48] Gwennap, L. IC Manufacturing Drives CPU Performance. *Microprocessor Report* (May 31 1993), 15–19.

[49] Gwennap, L. TFP Designed for Tremendous Floating Point. *Microprocessor Report* (August 23 1993), 9–13.

[50] Gwennap, L. CPU Vendors Deploy Half-Micro Processes. *Microprocessor Report* (April 18 1994), 16–20.

[51] Gwennap, L. PA-7200 Enables Inexpensive MP Systems. *Microprocessor Report* (March 7 1994), 12–15.

[52] Gwennap, L. PPC 604 Powers Past Pentium. *Microprocessor Report* (April 18 1994), 1,6–9,21.

[53] Gwennap, L. Integrated PA-7300LC Powers HP Midrange. *Microprocessor Report* (November 13 1995), 12–15.

[54] Gwennap, L. Microprocessors Lead the Way to 0.35 Microns. *Microprocessor Report* (July 10 1995), 16–20.

[55] Hartman, B. *16-bit 68000 microprocessor camps on 32-bit frontier*, first ed. McGraw-Hill, New York, New York, 1981, pp. 126–133.

[56] Hill, D. D., and Preas, B. Benchmarks for Cell Synthesis. In *Proceedings of 27th ACM/IEEE Design Automation Conference* (1990), IEEE, pp. 317–320.

[57] Hinsberger, U., and Kolla, R. Cell based Performance Optimization of Combinational Circuits. In *Proceedings of ICCD* (1990), IEEE, pp. 594–599.

[58] Irrissou, B. Design Techniques for High-Speed Datapaths. Technical Report UCB/CSD 93/748, University of California at Berkeley, UCB/CSD, August 1993.

[59] Jolly, R. A 9-ns, 1.4-Gigabyte/s 17-Ported CMOS Register File. *IEEE Journal of Solid State Circuits* (October 1991), 1407–1412.

[60] Katz, B. J., Morse, S. P., Pohlman, W. B., and Revenel, B. W. *8086 microcomputer bridges the gap between 8- and 16-bit designs*, first ed. McGraw-Hill, New York, New York, 1981, pp. 112–117.

[61] Keckler, S. W., Chang, A., Dally, W. J., Carter, N. P., and Lee, W. S. The MIT Multi-ALU Processor. In *HOTCHIPS IX* (1997), IEEE, pp. 30–40.

[62] Kim, J. *Automatic Layout Synthesis for High Performance Full Custom VLSI Chips*. PhD thesis, University of Illinois at Urbana-Champaign, May 1995.

[63] Klein, K., Koetzle, G., Miersch, E. F., Schettler, H., Schulz, U., and Wagner, O. Characteristics of a Set of 12.7mm Processor Chips. *IEEE Journal of Solid State Circuits* (October 1987), 783–789. Processor, VLSI.

[64] Kolla, R. A Dynamic Programming Approach to the Power Supply Net Sizing Problem. In *Proceedings of ICCD* (1990), IEEE, pp. 600–604.

[65] Kurdahi, F. J., and Ramachandran, C. Evaluating Layout Area Tradeoffs for High Level Applications. *IEEE Transactions on VLSI Systems 1*, 1 (Mar. 1993), 46–55. VLSI Area, Tradeoffs.

[66] Mead, C., and Conway, L. *Introduction to VLSI Systems.* Addison Wesley Press, 1980.

[67] Meyer, T. The PA-7300LC The First "System on a Chip". http://www.hp.com/computing/framed/technology/micropro/.

[68] Motorola SPS Motorola PowerPC^{TM} 620 32/64-Bit RISC Processor. http://www.mot.com/SPS/PowerPC/image/overlay/620.overlay.gif.

[69] Mulder, J. M., Quach, N. T., and Flynn, M. J. An Area Model for On-Chip Memories and its Application. *IEEE Journal of Solid State Circuits 26*, 2 (February 1991), 98–106.

[70] Naffziger, S. A Sub-Nanosecond 0.5μm 64b Adder Design. http://www.hp.com/computing/framed/technology/micropro/.

[71] Patterson, D., Anderson, T., Cardwell, N., Fromm, R., Keeton, K., Kozyrakis, C., Thomaas, R., and Yelick, K. A Case for Intelligent RAM: IRAM. *IEEE Micro* (April 1997).

[72] Silicon Graphics MIPS Division - R10000 Die Photo. http://www.sgi.com/MIPS/products/r10k/T5_Die.html.

[73] Shima, M. *Two versions of 16-bit chip span microprocessor, minicomputer needs*, first ed. McGraw-Hill, New York, New York, 1981, pp. 118–125.

[74] Slater, M. Second Generation i860 Premiers as i860XP. *Microprocessor Report* (June 12 1991), 1, 7–9.

[75] Slater, M. Sun and TI Preview Viking as "SuperSPARC". *Microprocessor Report* (September 18 1991), 8–9, 18–19.

[76] Slater, M. MIPS R4400 Offers 60-80% Performance Boost. *Microprocessor Report* (November 18 1992), 7–8.

[77] Srivastava, A. 500MHz High-Speed Adder. http://www.hp.com/computing/framed/technology/micropro/.

[78] Waingold, E., Taylor, M., Srikrishna, D., Sarkar, V., Lee, W., Lee, V., Kim, J., Frank, M., Finch, P., Barua, R., Babb, J., Amarasinghe, S., and Agarwal, A. Barring it all to Software: Raw Machines. *IEEE Computer* (September 1997), 86–93.

[79] Weste, N., and Eshraghian, K. *Principles of CMOS VLSI Design: A Systems Perspective*, second ed. Addison-Wesley, Reading, Massachusetts, 1993, section 4.5.1, p. 208.