ABSTRACT

THOROLFSSON, THORLINDUR. Three-Dimensional Integration of Synthetic Aperture Radar Processors. (Under the direction of Prof. Paul D. Franzon.)

In this dissertation we explore the advantages of 3D integration in the context of building synthetic aperture radar image processors through two case studies. In the first case study we demonstrate a floating point FFT processor that leverages both 3D integration and a unique hypercube memory division scheme to reduce the power consumption of a 1024 point FFT down from 5.476 $\mu J$ down to 4.227 $\mu J$. In this case study the hypercube memory division scheme lowers the energy per memory access by 59.2% while only increasing the total area required by 16.8%. In the same case study, the use of 3D integration reduces the logic power by 5.2%. In the second case study we present a full SAR processor that achieves a power efficiency of 18.0 mW/GFlop through the use of a 3D memory, 3D logic-on-logic integration and datapath reconfiguration. The 3D integrated memory reduces the memory power consumption by 70% when compared to a 2D memory. The logic-on-logic 3D integration used in the PE, decreases the power consumed in the interconnect of each PE by 15.5%, the footprint of the PE by 49.2% and allows the PE to operate at a 7.1% faster clock speed. The datapath reconfiguration reduces the number of arithmetic units required in each PE from 24 down to 10. Finally, in the second case study, we explore how a 3D system can be realized using 2D tools and propose a new algorithm for TSV assignment based on Lee’s algorithm.
Three-Dimensional Integration of Synthetic Aperture Radar Processors

by
Thorlindur Thorolfsson

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Computer Engineering

Raleigh, North Carolina

2011

APPROVED BY:

Prof. Gregory T. Byrd
Prof. William Rhett Davis

Prof. Michael Devetsikiotis
Prof. Helena Mitasova

Prof. Paul D. Franzon
Chair of Advisory Committee
DEDICATION

This dissertation is dedicated to all the deadlines that helped me finish the dissertation,
especially the ones I did not meet.
Thorlindur Thorolfsson was born in Reykjavik, Iceland. He completed the accelerated Masters program at North Carolina State University in 2005. He started his Ph.D. in Electrical and Computer Engineering the same year. His research focus is on digital circuits that take advantage of 3D integrated circuits and the tool flow required to realize them. He also maintains an active interest in network and computer security and has been a student member of the IEEE and ACM starting in 2004 and 2009 respectively.
ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor Dr. Paul Franzon for being an excellent advisor, helping me find a topic interesting enough to stay engrossed in for a few years and supporting me to realize my research goals. Second, I would like thank the following people for their contribution that made this dissertation possible: Dr. William Rhett Davis for critically analyzing my architectural designs and answering every EDA tool question I asked him, Dr. Gregory T. Byrd for his insights and helpful comments on all my written Ph.D Documents, Dr. Michael Devetsikiotis for serving on my committee and helping me see my work from a different perspective, Dr. Helena Mitasova for showing me the utility of Synthetic Aperture Radar in other research domains, my mother for teaching me programming at an early age, my father for encouraging my education and explaining the academic world to me, Steve Lipa for too many things to list from helping my with my car stereo to helping me write my EDA scripts, Kiran Gonsalves for designing the SRAM memories used in my design, Magnus Halldorsson at Reykjavik University for helping me with the math required for my memory partitioning, Ravi Jenkal for his insights and for teaching me how to use many of the tools required for my research, Evan Erickson for helping me keep my lab computers running and suggesting good music to listen to while completing this research, Samson Melamed for helping me with the thermal analysis of my architectures, Chris Mineo for providing me with numerous scripts, Shep Pitts for answering every solid state question I ever asked him, Nariman Moezzi-Madani for showing me how to automatically balance my critical paths, Peter Gadfort for explaining how to use decoupling capacitors, Ojas Bapat for tool flow advice, Sveinbjorn Thordarson for encouragement, MIT Lincoln Labs for providing me access to their 3D FD-SOI technology, and Tezzaron for providing access to their 3D integration technology.
# TABLE OF CONTENTS

List of Figures ................................................. vii
List of Tables .................................................. ix

**Chapter 1 Introduction** ...................................... 1
  1.1 Overview of the Following Chapters .................... 2
  1.2 Abbreviations ........................................... 3

**Chapter 2 Synthetic Aperture Radar** .......................... 6
  2.1 Overview of Synthetic Aperture Radar .................. 6
    2.1.1 Basic SAR Concepts .................................. 8
    2.1.2 SAR Processing Algorithms ......................... 13
    2.1.3 Summary ............................................. 20
  2.2 Computational And Memory Requirement Analysis ....... 20
    2.2.1 Basic Operations ................................. 20
    2.2.2 Range Doppler Algorithm ......................... 22

**Chapter 3 Memory Circuits** .................................. 30
  3.1 SRAM .................................................... 30
  3.2 DRAM .................................................... 31
  3.3 MRAM .................................................... 34
  3.4 Summary and 3D Implications ........................... 36

**Chapter 4 3D Integration** ................................... 38
  4.1 Overview of 3D Integration Technologies ............... 39
    4.1.1 Monolithic 3D Approaches ......................... 39
    4.1.2 3D Stacking with TSVs ............................ 41
    4.1.3 3D Packaging ...................................... 46
  4.2 Challenges for 3D ....................................... 47
    4.2.1 Power Delivery ..................................... 48
    4.2.2 Thermal Density ................................... 49
    4.2.3 Clock Tree Design .................................. 49
    4.2.4 Design for Test .................................... 50
    4.2.5 3D Floorplanning And Placement ................. 51
  4.3 Wire Length Reduction ................................... 52
    4.3.1 3D Wire Length Reduction ........................ 52
    4.3.2 3D Power Reduction ............................... 53
Chapter 5 Case Study 1: Using 3D for Memory Partitioning .......................... 55
  5.1 Memory Division Scheme .................................................. 57
  5.2 System Overview ............................................................ 67
  5.3 Tool Flow and 3D Implementation ....................................... 67
  5.4 Thermal Analysis ............................................................ 74
    5.4.1 Reducing Thermal Bottlenecks ..................................... 74
    5.4.2 Thermal Simulation ................................................... 75
    5.4.3 Thermal Discussion ................................................... 78
  5.5 Results ................................................................. 78
    5.5.1 Memory Division Power Savings ................................... 80
    5.5.2 3D Improvements Over 2D .......................................... 80
    5.5.3 Comparison to Other FFTs ......................................... 83
    5.5.4 Summary .............................................................. 85
  5.6 Test Setup and Measurement ............................................. 85
  5.7 Conclusion ............................................................... 90

Chapter 6 Case Study 2: 3D Memory and Logic-on-Logic ......................... 91
  6.1 Architecture And Operation ............................................. 92
    6.1.1 Main Memory DRAM ................................................ 92
    6.1.2 Memory Controller ................................................ 93
    6.1.3 Register Files .................................................... 93
    6.1.4 Instruction Decoder ................................................ 95
    6.1.5 Twiddle Factor ROMs .............................................. 97
    6.1.6 Processing Elements ............................................... 98
  6.2 Manufacturing Process .................................................. 104
  6.3 Tool Flow ............................................................. 105
    6.3.1 3D Placement ...................................................... 106
    6.3.2 3D LVS ........................................................... 109
    6.3.3 Overall Tool Flow ................................................ 110
  6.4 Power Delivery ........................................................ 114
  6.5 Results ............................................................... 117
    6.5.1 Memory Results ................................................... 118
    6.5.2 Thermal Profile ................................................... 118
    6.5.3 Logic-on-Logic Results .......................................... 120
    6.5.4 Comparisons To Other Works ................................... 123
  6.6 Conclusion ............................................................. 123

Chapter 7 Conclusions and Future Work ............................................ 125
  7.1 Summary of Contributions ............................................. 125
  7.2 Future Work ............................................................ 126

References ................................................................. 127
LIST OF FIGURES

Figure 2.1 Different SAR Modes. ................................................. 8
Figure 2.2 IQ Quadrature Demodulator. ......................................... 9
Figure 2.3 Samples arranged in memory. ....................................... 10
Figure 2.4 The squint angle. .................................................... 13
Figure 2.5 Flowchart for the different variants of the Range Doppler Algorithm ............................................. 15
Figure 2.6 Flowchart for the Chirp Scaling Algorithm ....................... 16
Figure 2.7 Flowchart for the two versions of the Omega-K Algorithm ... 18
Figure 2.8 Flowchart for the The SPECAN Algorithm ....................... 19
Figure 2.9 M143 data sample output .......................................... 23
Figure 2.10 The computational requirements for the various resolutions ... 29
Figure 2.11 The memory bandwidth requirements for the various resolutions ... 29

Figure 3.1 Basic six transistor SRAM. ............................................ 31
Figure 3.2 1T1C DRAM cell. ..................................................... 32
Figure 3.3 Internal DRAM organization. ........................................ 33
Figure 3.4 Open page versus close page policies. ............................ 35
Figure 3.5 1T1J MRAM cell. ..................................................... 37

Figure 4.1 Different 3D methods. ................................................ 40
Figure 4.2 The three different stacking orientations. ......................... 42
Figure 4.3 Cross sectional view of Irvine Sensors’ 3D Mint Process .......... 47
Figure 4.4 Power and critical period reduction given a certain wire length reduction. 54

Figure 5.1 The memory division tradeoffs. ..................................... 56
Figure 5.2 Four point FFT along with the basic partitioning. ............... 58
Figure 5.3 Architecture for basic partitioning. ................................ 59
Figure 5.4 Karnaugh Map for partitioning scheme for 4-point FFT. ......... 60
Figure 5.5 Karnaugh Map for eight partitions. ................................ 62
Figure 5.6 64 point FFT hypercube with the partitioning shown. .......... 63
Figure 5.7 Basic Partitioning shown for N=16 hypercube ................... 65
Figure 5.8 The divided memory architecture. ................................... 68
Figure 5.9 The block diagram of the processing element. .................... 68
Figure 5.10 Side view of the MIT Lincoln Labs’ process. .................... 70
Figure 5.11 The 3D design flow. ................................................ 71
Figure 5.12 The 3D SAR FFT processor with thru silicon vias drawn in. .... 73
Figure 5.13 The 3D floorplan. .................................................... 73
Figure 5.14 The temperature profile for all three tiers. ....................... 77
Figure 5.15 A histogram of junction temperatures for both designs. ....... 79
Figure 5.16 A closeup of hot spots caused by the clock buffers on the middle tier ... 79
Figure 5.17 The 2D floorplan for comparison. .................................. 81
Figure 5.18 Histogram of wire lengths of the SAR FFT processor. ........ 82
# LIST OF TABLES

Table 2.1  Process Summary  ................................................................. 22  
Table 2.2  Flops/frame required for base RASSP algorithm .................................. 24  
Table 2.3  Gigaflops/frame required for higher resolutions of the RASSP Algorithm ... 25  
Table 2.4  Memory size requirements for main processing grid  .................. 25  
Table 2.5  Memory size requirements for twiddle factors  ................................ 26  
Table 2.6  Memory size requirements for azimuth compression kernels ................ 26  
Table 2.7  Memory transactions for larger resolutions  ................................... 27  
Table 2.8  Memory transactions for smaller resolutions  .................................. 28  
Table 3.1  Memory Technology Summary ..................................................... 37  
Table 4.1  Average Wire Length Reduction .................................................. 53  
Table 5.1  Basic partitioning scheme for 4-point FFT  ................................... 60  
Table 5.2  Eight partition partitioning scheme  ........................................... 61  
Table 5.3  Relationship between partition scaling, memories and processing elements 64  
Table 5.4  The alternate thirty-two partition partitioning scheme  ..................... 66  
Table 5.5  Relationship between partition scaling, memories and processing elements for the 32 partition partitioning approach  .................................................... 67  
Table 5.6  Thermal simulation results ........................................................... 76  
Table 5.7  The design metrics of the undivided and the divided memory arrangements 81  
Table 5.8  Improvements from using 3D integration in the implementation of the FFT 84  
Table 5.9  Comparison between FFTs ............................................................ 85  
Table 5.10  Energy per 1024 point FFT for both 2D and 3D versions  .................. 86  
Table 6.1  Total area and aspect ratio of the three ROM options  ....................... 97  
Table 6.2  Controls for various operations .................................................... 100  
Table 6.3  Input addresses for the eight PEs in a 64 point FFT  ......................... 102  
Table 6.4  Input registers used for the different FFT stages  ............................ 103  
Table 6.5  Which PE can multiply which P and R register for complex multiplication 104  
Table 6.6  Estimated maximum currents for vias and metals in mA per micrometer 114  
Table 6.7  Summary of comparison between the 2D and 3D PE  ......................... 121  
Table 6.8  Summary of power comparison @ 31.61MHz between the 2D and 3D PE 121  
Table 6.9  Comparisons to other works ....................................................... 123
Chapter 1

Introduction

Scaling has resulted in a significant power reduction and performance improvement in digital logic systems in the past few decades. As we move towards smaller technology nodes, three key changes will occur. First, the interconnect will consume proportionally more and more of the overall power and delay budget of the system[25]. Second, an increasing amount of off-chip memory bandwidth will be required to keep up with the improved through-put of the logic[26]. Third, fabrication costs will increase drastically because both the cost of building fabrication facilities and the cost of creating lithography masks increases drastically at smaller technology nodes. For masks this is especially true if the masks will require immersion lithography, double patterning or even iterated double patterning to function correctly. One technology that has the potential to address these three key changes is 3D integration. First, 3D integration can limit the power consumption in the interconnect by reducing the total wire length in the circuit. Second, 3D integration can improve the memory bandwidth between the memory and the logic. The improvement occurs because 3D integration allows dies that are manufactured in different process technologies to communicate with each other through TSVs rather than lossy PCB traces resulting in higher speeds and lower rates of power consumption. Third, in some cases a 3D implementation can be more cost effective than a 2D implementation, especially, when Known-Good-Die is used to reduce total cost through improved yield[16].
In this dissertation we explore the benefits of 3D integration in the context of building synthetic aperture radar (SAR) image processors. A SAR processor is a particularly good candidate for this purpose because it requires a significant amount of memory bandwidth. The 3D exploration is primarily done through two case studies. In the first case study we analyze the benefits of re-architecting systems to explicitly use 3D integration to facilitate memory-on-logic. We do this by building a FFT processor in MIT Lincoln Lab’s 3D FDSOI 1.5 V process. In the second case study we look at the benefits of using 3D integration in a logic-on-logic configuration along with a 3D memory to realize a full SAR processor using Tezzaron’s technology. The goal of these case studies is not only to analyze the 3D benefits but also try to explore the road blocks and other issues that are unique to 3D designs.

1.1 Overview of the Following Chapters

In order to set the stage for the case studies, Chapter 2 gives an overview of synthetic aperture (SAR) concepts for the four most common SAR algorithms. One of these algorithms is the range doppler algorithm (RDA), which is used in both case studies. In addition, the chapter analyzes the computational, memory bandwidth and size requirements of the range doppler algorithm for real-time operation.

Since memory bandwidth is important to the SAR application, Chapter 3 explains the operation and manufacturing of three basic memory circuits along with the implications that 3D integration has on each of them.

As 3D integration is the focus of this dissertation Chapter 4 gives an overview of the three main groups of 3D technologies: monolithic, 3D Stacking with TSVs and 3D packaging along with an explanation of the processing techniques required to implement them. The chapter also surveys the literature for the of wire length reduction that can be achieved through 3D integration and looks into five key design challenges for building 3D integrated systems.

Chapter 5 contains the first case study. In this case study the benefits of explicitly re-architecting systems for memory-on-logic 3D integration is explored. This is done through
building a FFT processor in MIT Lincoln Lab’s 3D FDSOI 1.5 V process.

Chapter 6 contains the second case study, which looks at the benefits of using 3D integration in a logic-on-logic configuration along with a 3D integrated memory, all using Tezzaron’s technology.

Finally, Chapter 7 summarize the research work, results and contributions, and discusses the future work that can be done for this research.

1.2 Abbreviations

3DIC Three-dimensionally integrated circuit
ADC Analog-to-digital converter
AES Advanced encryption standard
BEOL Back-end-of-line
CAD Computer-aided design
CMOS Complementary metal-oxide-semiconductor
CORDIC COordinate Rotation Digital Computer
DC Direct current
DFT Discrete Fourier transform
DIP Dual In-line Package
DRAM Dynamic random access memory
DRIE Deep reactive ion etching
DSP Digital signal processing
EDA Electronic design automation
eDRAM  Embedded dynamic random access memory

EEPROM  Electrically erasable programmable read-only memory

ESD    Electrostatic discharge

FDSOI  Fully depleted silicon-on-insulator

FEOL   Front-end-of-line

FFT    Fast Fourier transform

FIR    Finite impulse response

Flop   Floating point operation

FM     Frequency modulation

IFFT   Inverse fast Fourier transform

IIT     Illinois Institute of Technology

IQ     In-phase and quadrature-phase

ISPD   International symposium on physical design

LDPC   Low-density parity-check

LFM    Linear frequency modulation

LVS    Layout versus schematic

MIMO   Multiple-input and multiple-output

MIT    Massachusetts Institute of Technology

MRAM   Magnetoresistive random access memory

MTJ    Magnetic tunnel junction
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSU</td>
<td>Oklahoma State University</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed circuit board</td>
</tr>
<tr>
<td>PDN</td>
<td>Power distribution network</td>
</tr>
<tr>
<td>PE</td>
<td>Processing element</td>
</tr>
<tr>
<td>PRF</td>
<td>Pulse repetition frequency</td>
</tr>
<tr>
<td>RDA</td>
<td>Range doppler algorithm</td>
</tr>
<tr>
<td>ROM</td>
<td>Read-only memory</td>
</tr>
<tr>
<td>SAR</td>
<td>Synthetic-aperture radar</td>
</tr>
<tr>
<td>SDRAM</td>
<td>Synchronous dynamic random access memory</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-noise ratio</td>
</tr>
<tr>
<td>SOI</td>
<td>Silicon-on-insulator</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static random access memory</td>
</tr>
<tr>
<td>SRTM</td>
<td>Shuttle radar topography mission</td>
</tr>
<tr>
<td>STT</td>
<td>Spin torque transfer</td>
</tr>
<tr>
<td>TSV</td>
<td>Through-silicon via</td>
</tr>
<tr>
<td>UV</td>
<td>Ultraviolet</td>
</tr>
</tbody>
</table>
Chapter 2

Synthetic Aperture Radar

In order to understand why Synthetic Aperture Radar processors benefit from 3D Integration it is important to have a good basic understanding of the SAR application. This chapter gives an overview of the application. First, Section 2.1.1 explains the basic concepts of SAR processing along with the four most commonly used SAR processing algorithms. Second, Section 2.2 analysis the computational and memory requirements of the Range Doppler Algorithm.

2.1 Overview of Synthetic Aperture Radar

Synthetic aperture radar is a type of radar, that, unlike conventional radar is primarily used for imaging. SAR makes extensive use of digital signal processing to produce a beam that is effectively very narrow but synthetically increases the size of the imaging aperture. SAR imaging has a wide variety of uses, ranging from remote sensing applications such as cartography and oceanography, to military uses such as reconnaissance, surveillance and battle damage assessment. SAR can be used in satellites or aircrafts. For the right applications, SAR has three big advantages over optical imaging. First, SAR does not require external lighting as it only uses the transmitted waves to form the image. This contrasts with traditional optical imaging where a sensor just collects light emitted by other sources. This effectively means that SAR imaging is not dependent on external lighting conditions such as time of day. The second
advantage of SAR is that the frequencies commonly used for SAR can pass through clouds and other weather artifacts with very little attenuation. This means that SAR systems can be used under almost any weather conditions. The third advantage is that many materials scatter microwave frequencies differently than visible ones. This means that SAR images can provide information that is not present in photographs, such as ice composition of glaciers.

A very notable SAR mission is the Shuttle Radar Topography Mission[17] or SRTM. The mission produced the most complete, highest resolution digital elevation model of the Earth to date using interferometric SAR. Although the data was acquired in only 9 days, it took 2 years to completely process the data. This shows how high the high computational requirements of SAR processing can be. There are several types of SAR. The first three are illustrated in Figure 2.1 and the rest are described below.

- **Stripmap SAR Mode:** The original SAR mode. In this mode the direction in which the antenna is pointed stays constant and the platform moves parallel to the strip that is being imaged.

- **ScanSAR Mode:** This mode is a variation of Stripmap. However, unlike Stripmap the direction the antenna is pointed is altered. This provides a wider imaging area at a lower resolution.

- **Spotlight SAR Mode:** This mode is a different variation of Stripmap mode, where the antenna direction is changed to target a specific area of interest.

- **Inverse SAR Mode:** Inverse SAR is basically doing SAR in reverse, where the radar stays still but target moves to allow SAR processing.

- **Bistatic SAR Mode:** Bistatic SAR has the receiver and transmitter in different locations.

- **Interferometric SAR Mode:** In this SAR mode, post processing is used to obtain the height of the terrain from two complex SAR images.
2.1.1 Basic SAR Concepts

As stated earlier, SAR is typically used onboard a satellite or an aircraft. Regardless of which application is used, the principles involved are very much the same. In order for SAR to work the radar platform has to be moving parallel to the area it is imaging. SAR essentially operates as an array of radar platforms that all combine to form one synthetic aperture. This section explains the basic SAR concepts that are necessary to understand the SAR processing algorithms explained in Section 2.1.2.

The LFM Pulse

The platform sends out a linear frequency modulated pulse that is known as a chirp signal. A chirp signal is used because fills the target bandwidth uniformly. The rate at which the base frequency changes is known as the linear FM rate, $K$. Equation 2.1 below describes the LFM pulse. Technically, the equation is for an IQ demodulated version of the pulse that is complex, although the actual transmitted pulse is purely real.

\[
s(t) = \text{rect}(e^{j\pi Kt^2})
\]  

(2.1)
This LFM pulse is sent out at a regular interval. This interval is known as the pulse repetition frequency or the PRF. When the antenna is not transmitting, it listens for reflections of the transmitted pulse. Now depending on the material that the pulse hits, it is either reflected or absorbed. If it is reflected the antenna will receive it after a length of time that is proportional to the distance to the material.

**I/Q Quadrature Demodulation**

After being collected by the antenna the received signal is then demodulated to baseband using quadrature demodulation. It is then sampled by an ADC and stored in memory as a complex number. The I channel of the quadrature modulation becomes the real component and the Q channel becomes the imaginary component.

![Diagram of I/Q Quadrature Demodulator](image)

**Figure 2.2: IQ Quadrature Demodulator.**

**Signal Storage In Memory**

Conceptually, the stored signals are organized in the following manner: the samples collected from a single pulse get stored the same column. This column is known as a range bin. If the columns are stacked together horizontally they form the two dimensional array shown in Figure 2.3. The rows of this array are known as the azimuth and represent samples that are taken exactly one pulse apart.
Figure 2.3: Samples arranged in memory.

**Matched Filtering**

If we look down a range bin (up in the figure) we see a copy of the signal for each point that has reflected the beam back. For maximum resolution in separating different reflections apart, we want a very small pulse width. Transmitting a very small pulse width is however not practical, because a smaller pulse will have a less energy than a wider one and as a result the SNR will be worse. The solution to this problem is to transmit a wide pulse and then use DSP techniques to compress it into a smaller one. This technique is known as pulse compression. Pulse compression is typically done in one of two ways, depending on the band the radar is operating in[44]. For narrow and some medium band radars a technique known as “correlation processing” is typically used. For radars operating in the wide band “stretch processing” is typically used. For the scope of this dissertation we will focus on correlation processing. Correlation processing is achieved by convolution of the received range bin with a replica of the transmitted pulse. Convolution is defined in the Equation 2.2.

$$ (f * g)(t) = \int f(\tau)g(t - \tau), d\tau $$

(2.2)
Since, there are a lot of coefficients that need to be computed, convolution is usually performed as multiplication in the frequency domain. The filter kernel that is used to compute this is known as the correlation or matched filter. Cumming et al. [12] list three approaches to generating the matched filter from the transmitted signal.

1. Taking the DFT of the zero padded, complex conjugate of the time-reversed pulse replica.
2. Taking the complex conjugate of the DFT of the zero padded pulse replica.
3. Generating the matched filtering directly in frequency domain, using the linear FM modulated characteristics.

**Windowing For Sidelobe Suppression** After applying matched filtering we end up with spikes that look like sinc functions in the range bin. The spikes correspond to reflections of transmitted pulses. The range bin that the reflection is stored in corresponds to the time it takes the pulse to be transmitted, hit a target and return. This time is also related to the distance to the reflector. This relation is hyperbolic and is defined by Equation 2.3 the range equation below.

\[
R^2(\eta) = R_0^2 + V^2\eta^2
\]  

(2.3)

Now, since the output of the matched filter is a sinc function we will see additional lobes occur periodically with the period of the sine component of the sinc function. These lobes are known as sidelobes. It can often be hard to distinguish strong reflections of the sidelobes from weak reflections of main lobes. Since this is the case it is desirable to suppress them. The most commonly used method to suppress sidelobes is to apply a smoothing window on the correlation filter in the frequency domain. The four most commonly used windows for suppressing sidelobes are Kaiser, Hamming, Hann and Bartlett, Equations 2.4, 2.5, 2.6 2.7 respectively.

\[
w(n) = \frac{I_0 \left( \pi\alpha \sqrt{1 - \left(\frac{2n}{N-1} - 1\right)^2} \right)}{I_0(\pi\alpha)}
\]  

(2.4)
Range Cell Migration

As explained above the range bin location of the transmitted pulse reflection depends on the distance between the platform and the reflector, as described in Equation 2.3. As we fly by the target, the distance to the target will initially be great. As we approach the target the distance decreases until it reaches its minimum when the target is the closest. After we have passed the target we are moving away from it again and the distance increases again. This change in distance will cause the target to move to a different range bin and is known as range cell migration. Range cell migration does significantly complicate processing but it is an inherent part of SAR. The way a given algorithm compensates for range cell migration is usually what sets it apart from other processing algorithms.

Squint Angle

The squint angle ($\theta_{sq}$) is determined by the difference between the direction of the line-of-sight from the radar and a ray that is perpendicular to the heading of the aircraft. This angle can also be thought of as the beam yaw angle as shown in Figure 2.4. The angle is important because it is the angle that the slant range vector makes with the zero Doppler plane. To better explain the angle, we consider the two extremes. If the radar beam is facing perfectly forward (forward-look SAR) then $\theta_{sq} = 90^\circ$. If the radar beam is pointed perfectly perpendicular to the heading of the radar platform then $\theta_{sq} = 0^\circ$. The squint angle is very important because it can
have a significant impact on the processing required to generate the final display image. When the squint angle is great, the range equation is more hyperbolic than parabolic. This means that range migration will exhibit more non-linearity and has two effects on the SAR processor. First, the range used to correct range cell migration and the azimuth filter have to be modified slightly. Second, a filter needs to be applied to fix range to azimuth coupling. This filter is known as secondary range compression.

\[ \theta_{sq} = \tan^{-1} \left( \frac{x - s_0 V}{R_0} \right) \]  

(2.8)

![Squint Angle](image)

Figure 2.4: The squint angle.

### 2.1.2 SAR Processing Algorithms

Apart from the ideal processing of SAR data, there are four algorithms that are most commonly used to process SAR data, the Range Doppler Algorithm, the Chirp Scaling Algorithm, the Omega-K Algorithm and the SPECAN Algorithm. These algorithms are outlined below.

**Range Doppler Algorithm**

The Range Doppler Algorithm (RDA) [32] was the first algorithm to be used for SAR processing. It does range cell migration by interpolation in the time domain. Targets at the same range
but in different azimuth locations are transferred to same location in the azimuth frequency domain. When range cell migration is applied, it fixes a whole set of trajectories at once. The range Doppler algorithm can do secondary range correction. This allows the algorithm to handle much higher squint than it would otherwise be able to do.

The Range Doppler Algorithm has several advantages. First, it achieves block-processing efficiency by operating on a single dimension at a time. Second, it is very well suited for a pipelined architecture. However, the algorithm has several disadvantages. First, this algorithm does not work well with a high squint angle or a wide aperture. Second, the algorithm does not handle the dependance of the azimuth frequency on secondary range compression. Finally, the algorithm has a high computing load when long filter kernels are used. There are essentially three variants of the algorithm. One version that does secondary range compression, one version that does approximate secondary range compression and one version that does neither.

The Chirp Scaling Algorithm

The Chirp Scaling Algorithm [58] is very similar to the Range Doppler Algorithm, except that it does not use a time-domain interpolator to do range migration correction. Instead it uses a scaling principle in which frequency modulation is applied to a chirp encoded signal, hence the name of the algorithm. This allows range cell migration to be done as phase multiplies in the frequency domain instead of interpolation in the time-domain. Doing range cell migration correction in this fashion allows azimuth frequency dependent secondary range compression. This ability to do frequency dependent secondary range compression gives this algorithm an advantage for high squint angles and wide apertures. There is one complication with this algorithm. This complication is that there is a limit on the maximum frequency shift that can be applied by modulation in the frequency domain without distorting the signal’s center frequency and bandwidth. Getting around this limit requires applying range cell migration correction in two steps. These two steps are known as bulk and differential range cell migration.
Figure 2.5: Flowchart for the different variants of the Range Doppler Algorithm
Figure 2.6: Flowchart for the Chirp Scaling Algorithm
The steps of the algorithm are shown in Figure 2.6.

The Omega-K Algorithm

Unlike the other algorithms, the Omega-K [5] algorithm is derived from seismic processing algorithms. The algorithm is designed to handle of the range dependence of range-azimuth coupling and handle wide apertures and high squint angles. To do this the algorithm uses a special operation in the two-dimensional frequency domain. The operation has two key steps. The first step is known as the reference function multiply. This step involves calculating a reference function based on the middle of the swath and then applying it on the input data. This correctly focuses the targets in the middle of the swath. The second step of the operation is known as the Stolt interpolation. This step focuses the targets left over of the targets by using interpolation in the range frequency domain. Although, this algorithm is very versatile in terms of aperture widths and squint angles, it assumes that the radar velocity is constant. This assumption has two major consequences. First, the algorithm is unsuitable for airplane use as airplanes do not move at constant speeds. Second, the algorithm’s accuracy is severely limited for targets that are not close to the middle of the swath. The algorithm comes in two versions, the accurate and the approximate version.

The SPECAN Algorithm

The SPECAN algorithm is geared towards faster yet less accurate SAR processing, which makes it an ideal candidate for real-time processing. However, it cannot achieve the high resolution of the other three algorithms. The SPECAN algorithm is relatively similar to the Range Doppler algorithm, except in the manner that it does azimuth compression. The SPECAN algorithm does azimuth compression by using a de-ramping operation followed by a Fast Fourier Transform. This method of azimuth compression severely limits the obtainable resolution, which is why range cell correction is done in the azimuth dimension rather than in the range dimension. With the exception of resolution, the SPECAN algorithm has three limitations. First, since
Figure 2.7: Flowchart for the two versions of the Omega-K Algorithm
the radar data is never in the true azimuth frequency domain, only limited range cell migration correction can be performed. Second, the SPECAN algorithm does not give accurate phase results. If one is only interested in the final image this not a problem. However for interferometric SAR where phase is important this can be a problem. To mitigate the problem the phase information can be adjusted using a phase compensation technique. Third, since the SPECAN algorithm is designed for linear FM signals, if the signal is not linear the output quality will be degraded. The SPECAN algorithm consists of the following steps:

Figure 2.8: Flowchart for the SPECAN Algorithm
2.1.3 Summary

From the algorithms listed above, it can be seen that there are various different processing algorithms for processing SAR data. The most important point to gather from this discussion is that all the algorithms are built from the same basic building blocks and these algorithms are forward and inverse FFTs, and function multiplies.

2.2 Computational And Memory Requirement Analysis

All the algorithms we showed in Section 2.1.2 are built up from these same basic building blocks. In this section we outline the computational requirements of the building blocks and then use them to determine the computational and memory requirements for an implementation of the Range Doppler Algorithm.

2.2.1 Basic Operations

This section analysis the number of floating point operations that are required to complete the basic processing operations including, the Fast Fourier Transform, the FIR filter, real by complex multiplication and complex multiplication.

Complex Multiplication

A complex multiplication can be done using three multiplies, two additions and one subtraction as seen by \((a + bj)(c + dj) = a(c - d) + adj + cbj\) for total of six arithmetic operations. Real by complex multiplication can be done using two arithmetic operations, one multiply for the real component and another for the imaginary.

Fast Fourier Transforms

In a Radix-2 Cooley-Tukey FFT, one complex multiply and one negation is required to implement the butterfly structure. The number of butterfly structures required for an N point FFT
is shown in Equation 2.9.

\[
\text{Number of butterfly structures required} = \frac{N}{2} \log_2(N) \tag{2.9}
\]

Since we do one complex multiply and one negation for each butterfly structure we have to do seven arithmetic operations per butterfly structure, which means an N point FFT will require the following arithmetic operations.

\[
\text{Arithmetic operations} = 7 \frac{N}{2} \log_2(N) \tag{2.10}
\]

**Inverse Fast Fourier Transforms**

An inverse FFT can be accomplished using a regular FFT with two minor modifications. The first minor modifications is that after performing the FFT as normal, every element is scaled by multiplying with \( \frac{1}{N} \). The second modification is that after the scaling the sign is flipped on the imaginary component. Equation 2.11 shows number of operations required for the IFFT.

\[
\text{Arithmetic operations} = 7 \frac{N}{2} \log_2(N) + N \tag{2.11}
\]

**FIR Filter**

For one sample, a real FIR filter requires one multiplication for every coefficient, along with one addition for every coefficient except the first one. The number of operations required for FIR filtering a L long sequence using K coefficients is shown in Equation 2.12:

\[
\text{Arithmetic operations} = LK(K - 1) \tag{2.12}
\]

**Summary**

Table 2.1 shows the operations required for every operation.
Table 2.1: Process Summary

<table>
<thead>
<tr>
<th>Process</th>
<th>Number of Ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex Multiplies</td>
<td>6</td>
</tr>
<tr>
<td>Real by Complex Multiplies</td>
<td>2</td>
</tr>
<tr>
<td>N-Point Fast Fourier Transforms</td>
<td>( \frac{7N}{2} \log_2(N) )</td>
</tr>
<tr>
<td>N-Point Inverse Fast Fourier Transforms</td>
<td>( \frac{7N}{2} \log_2(N) + N )</td>
</tr>
<tr>
<td>K-Coefficient FIR filter(L long seq.)</td>
<td>( LK(K - 1) )</td>
</tr>
</tbody>
</table>

2.2.2 Range Doppler Algorithm

In order to understand the computational and memory requirements for SAR image processing, we have to choose a specific algorithm to analyze and implement. We have two main criteria for choosing the algorithm. First, the algorithm must have both test data and a reference implementation readily available. Second, the algorithm must be a good representative of SAR algorithms in general. The algorithm we choose is the Range Doppler Algorithm used in the RASSP project [23], which meets both criteria. It meets the first criteria by having quality test data from the M143 mission along with source code. Second, it is very representative of SAR algorithms because it is a variant of the Range Doppler Algorithm that is the core of many SAR processing algorithms. The algorithm is identical to the version of Range Doppler described in Section 2.1.2 with the exception that de-ramping is done as a part of the sampling process and as a result it is not included in our requirement analysis. The algorithm essentially consists of three major steps, converting the received signal to baseband, applying range compression and applying azimuth compression. They are explained in the following paragraphs.

The sequences from the even and odd data are modulated by \((-1)^n\) to convert the received signal to baseband. These sequences are then filtered by an eight coefficient low-pass FIR filter. Since the IQ demodulation is essentially just a simple FIR filter applied separately to the imaginary and real component it is referred to as FIR Filtering from now on.

In this particular algorithm, range compression is achieved simply by taking the Fast Fourier
Transform along the range gates. Range compression is referred to as Range FFT in the following sections. In order to complete the FFT it is necessary to reshuffle the data. The reshuffle step is known as range reorder step.

Azimuth compression is slightly more complicated than range compression. Azimuth compression is accomplished by convolving each azimuth row by the approximate response of a point scatterer located at the range gate of the row to be processed. Different kernels are used for different ranges. Furthermore, for efficiency the convolution is done in the frequency domain by doing a FFT followed by frequency domain multiplication and then an IFFT. In the following sections the three distinct steps required for azimuth compression are referred to as Azimuth FFT, Azimuth or Complex Multiply and Azimuth IFFT. As with the Range FFT, reshuffling is also necessary for the Azimuth FFT and the Azimuth IFFT, the reshuffle steps for those two operations are known as azimuth reorder. In order to implement the IFFT, an additional scaling step is required; this step is referred to as azimuth scaling. A sample image from the M143 mission processed using steps described above is shown in Figure 2.9.

Figure 2.9: M143 data sample output
Computational Requirements

Now that we have defined the number of arithmetic operations needed to complete each of the basic operations and explained briefly the algorithm used, we can compile the total number of floating point operations (flops) required for the image formation of one frame. Again, the numbers used in this analysis come from the Range Doppler Algorithm (see Section 2.1.2) used in the RASSP [23] project. The algorithm from the original project operates on grid that is 1024 bins wide in the azimuth dimension and 2048 bins wide in the range dimension and has a resolution of 30 cm. Table 2.2 shows the number of operations required to implement each process.

Table 2.2: Flops/frame required for base RASSP algorithm

<table>
<thead>
<tr>
<th>Process</th>
<th>Type</th>
<th>Operations</th>
<th>Repeat</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>IQ Demodulation</td>
<td>8 Coef. Real FIR Filter</td>
<td>229,376</td>
<td>512</td>
<td>117,440,512</td>
</tr>
<tr>
<td>Range FFT</td>
<td>2048 Point FFT</td>
<td>78,848</td>
<td>512</td>
<td>40,370,176</td>
</tr>
<tr>
<td>Azimuth FFT</td>
<td>1024 Point FFT</td>
<td>35,840</td>
<td>2048</td>
<td>73,400,320</td>
</tr>
<tr>
<td>Azimuth Multiply</td>
<td>Complex Multiply</td>
<td>6,144</td>
<td>2048</td>
<td>12,582,912</td>
</tr>
<tr>
<td>Azimuth IFFT</td>
<td>1024 Point IFFT</td>
<td>41,984</td>
<td>2048</td>
<td>85,983,232</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td>329,777,152</td>
</tr>
</tbody>
</table>

As stated earlier the original RASSP algorithm operates on a grid of 1024 by 2048 with resolution of 30cm. We can extrapolate this to higher resolutions and/or grid sizes. The results are shown in Table 2.3.

Memory Transaction Requirements

Just having the hardware resources to complete the computation is not sufficient; there must also be enough memory and memory bandwidth to process the algorithm. The memory required to store the processing grid, the twiddle factors and filter kernels is shown in Table 2.4, Table 2.5
Table 2.3: Gigaflips/frame required for higher resolutions of the RASSP Algorithm

<table>
<thead>
<tr>
<th>Resolution</th>
<th>30 cm</th>
<th>15 cm</th>
<th>8 cm</th>
<th>4 cm</th>
<th>2 cm</th>
<th>1 cm</th>
<th>0.5 cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range Bins</td>
<td>2048</td>
<td>4096</td>
<td>8192</td>
<td>16384</td>
<td>32768</td>
<td>65536</td>
<td>131072</td>
</tr>
<tr>
<td>Azimuth Bins</td>
<td>1024</td>
<td>2048</td>
<td>4096</td>
<td>8192</td>
<td>16384</td>
<td>32768</td>
<td>65536</td>
</tr>
<tr>
<td>IQ Demodulation</td>
<td>0.117</td>
<td>0.470</td>
<td>1.879</td>
<td>7.516</td>
<td>30.065</td>
<td>120.259</td>
<td>481.036</td>
</tr>
<tr>
<td>Range FFT</td>
<td>0.040</td>
<td>0.176</td>
<td>0.763</td>
<td>3.288</td>
<td>14.093</td>
<td>60.130</td>
<td>255.551</td>
</tr>
<tr>
<td>Azimuth FFT</td>
<td>0.073</td>
<td>0.323</td>
<td>1.409</td>
<td>6.107</td>
<td>26.307</td>
<td>112.743</td>
<td>481.036</td>
</tr>
<tr>
<td>Azimuth Multiply</td>
<td>0.013</td>
<td>0.050</td>
<td>0.201</td>
<td>0.805</td>
<td>3.221</td>
<td>12.885</td>
<td>51.540</td>
</tr>
<tr>
<td>Azimuth IFFT</td>
<td>0.086</td>
<td>0.373</td>
<td>1.611</td>
<td>6.912</td>
<td>29.528</td>
<td>125.628</td>
<td>532.576</td>
</tr>
<tr>
<td>Total</td>
<td>0.330</td>
<td>1.393</td>
<td>5.864</td>
<td>24.629</td>
<td>103.213</td>
<td>431.644</td>
<td>1,801.739</td>
</tr>
</tbody>
</table>

Table 2.4: Memory size requirements for main processing grid

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Size</th>
<th>Grid Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 cm</td>
<td>256 Mb</td>
<td>2048 x 1024</td>
</tr>
<tr>
<td>15 cm</td>
<td>1 Gb</td>
<td>4096 x 2048</td>
</tr>
<tr>
<td>8 cm</td>
<td>4 Gb</td>
<td>8192 x 4096</td>
</tr>
<tr>
<td>4 cm</td>
<td>16 Gb</td>
<td>16384 x 8192</td>
</tr>
<tr>
<td>2 cm</td>
<td>64 Gb</td>
<td>32768 x 16384</td>
</tr>
<tr>
<td>1 cm</td>
<td>256 Gb</td>
<td>65536 x 32768</td>
</tr>
<tr>
<td>0.5 cm</td>
<td>1 Tb</td>
<td>131072 x 65536</td>
</tr>
</tbody>
</table>

Furthermore, there must also be enough memory bandwidth available to the processing memory to retrieve and store all the data operands in the time allotted. In the Range Doppler Algorithm under consideration the following ten memory transactions are required:

1. FIR → Memory
2. FIR ← Memory
Table 2.5: Memory size requirements for twiddle factors

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Size</th>
<th>Max FFT Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 cm</td>
<td>16 Kb</td>
<td>2048</td>
</tr>
<tr>
<td>15 cm</td>
<td>32 Kb</td>
<td>4096</td>
</tr>
<tr>
<td>8 cm</td>
<td>64 Kb</td>
<td>8192</td>
</tr>
<tr>
<td>4 cm</td>
<td>128 Kb</td>
<td>16384</td>
</tr>
<tr>
<td>2 cm</td>
<td>256 Kb</td>
<td>32768</td>
</tr>
<tr>
<td>1 cm</td>
<td>512 Kb</td>
<td>65536</td>
</tr>
<tr>
<td>0.5 cm</td>
<td>1 Mb</td>
<td>131072</td>
</tr>
</tbody>
</table>

Table 2.6: Memory size requirements for azimuth compression kernels

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Size</th>
<th>Kernel Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 cm</td>
<td>256 Kb</td>
<td>32</td>
</tr>
<tr>
<td>15 cm</td>
<td>1 Mb</td>
<td>64</td>
</tr>
<tr>
<td>8 cm</td>
<td>4 Mb</td>
<td>128</td>
</tr>
<tr>
<td>4 cm</td>
<td>16 Mb</td>
<td>256</td>
</tr>
<tr>
<td>2 cm</td>
<td>64 Mb</td>
<td>512</td>
</tr>
<tr>
<td>1 cm</td>
<td>256 Mb</td>
<td>1024</td>
</tr>
<tr>
<td>0.5 cm</td>
<td>1 Gb</td>
<td>2048</td>
</tr>
</tbody>
</table>
3. Range FFT → Memory

4. Range FFT ← Memory

5. Azimuth FFT → Memory

6. Azimuth FFT ← Memory

7. Complex Multiply → Memory

8. Complex Multiply ← Memory

9. Azimuth IFFT → Memory

10. Azimuth IFFT ← Memory

In order to analyze the memory bandwidth required we must first determine the number of memory transactions required to process one SAR image frame, then determine the time scale in which the transactions need to occur in. Table 2.7 and Table 2.8 show the memory transactions required on a memory resolution basis.

Table 2.7: Memory transactions for larger resolutions

<table>
<thead>
<tr>
<th>Resolution</th>
<th>30 cm</th>
<th>15 cm</th>
<th>8 cm</th>
<th>4 cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIR → Memory</td>
<td>1,048,576</td>
<td>4,194,304</td>
<td>16,777,216</td>
<td>67,108,864</td>
</tr>
<tr>
<td>Range Reorder</td>
<td>1,015,808</td>
<td>4,128,768</td>
<td>16,515,072</td>
<td>66,584,576</td>
</tr>
<tr>
<td>Range FFT</td>
<td>11,534,336</td>
<td>50,331,648</td>
<td>218,103,808</td>
<td>939,524,096</td>
</tr>
<tr>
<td>Azimuth Reorder 1</td>
<td>2,031,616</td>
<td>8,126,464</td>
<td>33,030,144</td>
<td>132,120,576</td>
</tr>
<tr>
<td>Azimuth FFT</td>
<td>20,971,520</td>
<td>92,274,688</td>
<td>402,653,184</td>
<td>1,744,830,464</td>
</tr>
<tr>
<td>Complex Multiply</td>
<td>2,097,152</td>
<td>8,388,608</td>
<td>33,554,432</td>
<td>134,217,728</td>
</tr>
<tr>
<td>Azimuth Reorder 2</td>
<td>2,031,616</td>
<td>8,126,464</td>
<td>33,030,144</td>
<td>132,120,576</td>
</tr>
<tr>
<td>Azimuth IFFT</td>
<td>20,971,520</td>
<td>92,274,688</td>
<td>402,653,184</td>
<td>1,744,830,464</td>
</tr>
<tr>
<td>Azimuth IFFT Scale</td>
<td>2,097,152</td>
<td>8,388,608</td>
<td>33,554,432</td>
<td>134,217,728</td>
</tr>
<tr>
<td>Total</td>
<td>63,799,296</td>
<td>276,234,240</td>
<td>1,189,871,616</td>
<td>5,095,555,072</td>
</tr>
</tbody>
</table>
Table 2.8: Memory transactions for smaller resolutions

<table>
<thead>
<tr>
<th>Resolution</th>
<th>2 cm</th>
<th>1 cm</th>
<th>0.5 cm</th>
<th>Apx Pct</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIR</td>
<td>268,435,456</td>
<td>1,073,741,824</td>
<td>4,294,967,296</td>
<td>1.37%</td>
</tr>
<tr>
<td>Range Reorder</td>
<td>266,338,304</td>
<td>1,069,547,520</td>
<td>4,278,190,080</td>
<td>1.34%</td>
</tr>
<tr>
<td>Range FFT</td>
<td>4,026,531,840</td>
<td>17,179,869,184</td>
<td>73,014,444,032</td>
<td>18.38%</td>
</tr>
<tr>
<td>Azimuth Reorder 1</td>
<td>532,676,608</td>
<td>2,130,706,432</td>
<td>8,556,380,160</td>
<td>2.69%</td>
</tr>
<tr>
<td>Azimuth FFT</td>
<td>7,516,192,768</td>
<td>32,212,254,720</td>
<td>137,438,953,472</td>
<td>34.02%</td>
</tr>
<tr>
<td>Complex Multiply</td>
<td>536,870,912</td>
<td>2,147,483,648</td>
<td>8,589,934,592</td>
<td>2.74%</td>
</tr>
<tr>
<td>Azimuth Reorder 2</td>
<td>532,676,608</td>
<td>2,130,706,432</td>
<td>8,556,380,160</td>
<td>2.69%</td>
</tr>
<tr>
<td>Azimuth IFFT</td>
<td>7,516,192,768</td>
<td>32,212,254,720</td>
<td>137,438,953,472</td>
<td>34.02%</td>
</tr>
<tr>
<td>Azimuth IFFT Scale</td>
<td>536,870,912</td>
<td>2,147,483,648</td>
<td>8,589,934,592</td>
<td>2.74%</td>
</tr>
<tr>
<td>Total</td>
<td>21,732,786,176</td>
<td>92,304,048,128</td>
<td>390,758,137,856</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Real Time Requirements

Now that we have determined the number of arithmetic operations required to process one SAR frame, we can directly calculate the required processing rate to process the SAR data in real-time. The processing rate is directly tied to the pulse repetition frequency, which is based on the aircraft speed. For aerial application such as the M143 the PRF can vary from 200 to 556 Hz. This corresponds to an aircraft ground speed of 45.7 m/s to 127.1 m/s respectively. A PRF of 556 Hz corresponds to one range-line every $\frac{1}{556}$ seconds = 1.799 milliseconds and since one frame contains 512 range lines, one frame needs to be processed in $\frac{512}{556}$ seconds = 0.9209 seconds.

Now since we have the number of operations, memory transactions and the time required to complete them in, we can calculate the computational requirements in Giga Flops/s and the memory bandwidth requirements in Gigabytes/s.
Figure 2.10: The computational requirements for the various resolutions in Giga Flops/s

Figure 2.11: The memory bandwidth requirements for the various resolutions in Giga Bytes/s
Chapter 3

Memory Circuits

As we have shown in Section 2.2, the SAR application require a significant amount of memory bandwidth. Although, there are several types of memory circuits available to meet this bandwidth requirement, the overview only focuses on the major ones. Many of these memory circuits are manufactured in a design process that is not compatible with the processes used to manufacture digital logic circuits. As a result these memory circuits cannot be accessed on-chip resulting in latency and power penalties. Three-Dimensional integration can mitigate these penalties by allowing close to on-chip accesses to circuits manufactured in different technologies. This chapter describes three different memory circuits and explains how they are effected by 3D integration.

3.1 SRAM

Static random access memory (SRAM) is a type of memory circuit that uses bistable latching to store a given bit. The basic SRAM cell usually consists of six transistors and is shown in Figure 3.1. Four of these transistors form a pair of cross-coupled inverters that store the actual bit. The other two transistors control reading and writing through the wordline by connecting the cell to the bitline at the appropriate time. The length of the word and bitline is very important because it has a large impact on the delay, power, and area of the circuit. To
mitigate the impact of word and bitline length an SRAM array is usually divided into sub-arrays because it effectively shortens the bit lines to reduce power and delay, allowing all memory operations to occur in a single cycle. Apart from the basic six transistor cells there are several variants that use additional transistors to increase stability or provide additional read and write ports. A great advantage of SRAM is that it can be manufactured in the same technology process as digital logic circuits. However, since SRAM cells require at least six transistors they require more area than most other types of memory circuits and are more susceptible process variation. Additional information on SRAM can be found in the following references[74, 1].

![SRAM Circuit Diagram](image)

Figure 3.1: Basic six transistor SRAM.

### 3.2 DRAM

Dynamic random access memory (DRAM) is a type of memory circuit that stores data as charge on a capacitor. Since capacitors leak over time, the charge must be refreshed periodically. To remedy this additional circuitry must be added, which increases the area and power consumption
of the circuit. The basic DRAM cell is known as a “1T1C” cell and consists of a capacitor along with a control transistor. This basic cell is shown in Figure 3.2.

![Figure 3.2: 1T1C DRAM cell.](image)

These basic cells are organized into arrays that are known as banks, which operate independently. In a given bank the transistors in every row are connected by wordlines and the transistors in every column are connected by bitlines. The bitlines are connected to a sense amplifier and the wordlines are connected to the address decoding circuitry. A full DRAM memory system will contain several banks as shown in Figure 3.3. Reads in “1T1C” DRAM are destructive, which means that after every read the original value must be rewritten before it can be accessed again. DRAM reads occur in the following manner:

1. The sense amplifier is turned off and the bit lines are precharged to a voltage midway between ground and the power supply voltage.
2. The precharge circuit is turned off.

3. The selected row’s word line is driven high. This connects one storage capacitor to the bitline. Charge is shared between the selected storage cell and the given bit line.

4. The sense amplifier is switched on, sensing the voltage difference on an entire row and storing it inside its row buffer. At this point, the column can selectively be read from the sense amplifier.

![Internal DRAM organization](image)

**Figure 3.3:** Internal DRAM organization.

The data stored in a DRAM cell must be refreshed periodically to prevent information loss because the storage capacitor leaks over time. This is done by a memory controller that keeps track of how long information has been stored on a given cell and orchestrates the refreshing of the data when necessary. How and when the memory controller executes the refresh operation is known as the refresh policy. The two most commonly used refresh policies are “close page” and “open page”. When the open page refresh policy is utilized the memory contents are cached in
the sense amplifier’s row buffer until another row is activated. This caching allows subsequent accesses to the cached row at a lower latency than accesses to rows that are not cached. A disadvantage of the open page policy is that keeping the row buffer open consumes additional power. When a close page refresh policy is used, a row is refreshed immediately after a column read has occurred and the row buffer powered down immediately. This has the advantage of allowing the DRAM bank to start precharging in anticipation of the next memory read, potentially eliminating the need to wait for the precharge operation for that access. Depending on the memory access pattern, different refresh policy are favorable. For memory access patterns that frequently access the same row the open page policy is preferable. For memory access patterns that are random in nature the close page policy is preferable. Figure 3.4 shows the command flows with the typical SDRAM latency in cycles for each command.

Designing a storage capacitor that has good capacitance and low leakage in a small footprint is quite challenging and requires utilizing a manufacturing process that is quite different than the process used to manufacture digital logic circuits. A DRAM process requires transistors to have a high threshold voltage to minimize leakage in the bit cell, whereas logic process requires a lower threshold voltage for faster transistor switching. As a result, manufacturing DRAM in the same process as digital logic is not optimal. This can be done at considerable density and power consumption cost and is known as embedded logic DRAM[45].

3.3 MRAM

Magnetoresistive random access memory (MRAM) is a type of memory circuit that stores data using a magnetic tunnel junction (MTJ) as the storage element[27, 78]. The basic MRAM cell is known as a “1T1J” cell and is shown in Figure 3.5. The cell consists of two ferromagnetic plates separated by a thin insulating layer. The first plate has a permanent magnet with its magnetic direction fixed to a specific polarity, which is known as the pinned layer. The second plate is known as the free layer and its magnetic field or spin valve can be changed between
Figure 3.4: Open page versus close page policies.
different polarities using a variety of different methods. One of these methods works in a fashion similar to core memory. The method involves passing current through a pair of write lines that are at right angles to each other and located above and below the cell, inducing a magnetic field in the free layer. This method requires a significant amount of current to work properly. A newer method that does not require as much current is called spin torque transfer (STT). STT changes the direction of the free layer by directly passing spin-polarized currents through the junction. Reading the spin value of the MTJ is significantly simpler than writing. Because of the magnetic tunnel effect, the spin value changes the electrical resistance of the MTJ. As a result reading the spin value can be accomplished by simply measuring the resistance. This is accomplished by applying a small voltage difference between the bitline and the source line and measuring the resulting current using a sense amplifier. MRAM technology is non-volatile in the sense that it does not require any refreshing and retains the stored value when the the power turned off. MRAM is similar to DRAM in the sense that it requires changes to the typical digital manufacturing process to be fabricated. This makes it difficult to manufacture MRAM in the same process as digital logic.

3.4 Summary and 3D Implications

It is important to note that the respective memory circuits have different properties that fill different roles in the memory system. A summary of these different properties is shown in Table 3.1. However, all the circuits except SRAM and eDRAM need to be manufactured in a process that is different then a normal digital logic process. This means the system logic and the memory circuit will be on different dies and as result they must communicate through traces on a printed circuit board. These traces are very lossy and result in a significant power and latency overhead. This overhead can be avoided using 3D integration which is explained in the following chapter.
Figure 3.5: 1T1J MRAM cell.

Table 3.1: Memory Technology Summary

<table>
<thead>
<tr>
<th>Memory</th>
<th>Logic Compatible</th>
<th>Refresh</th>
<th>Volatile</th>
<th>Speed</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Fastest</td>
<td>Low</td>
</tr>
<tr>
<td>DRAM</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Slowest</td>
<td>Highest</td>
</tr>
<tr>
<td>eDRAM</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Slow</td>
<td>High</td>
</tr>
<tr>
<td>MRAM</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Fast</td>
<td>High</td>
</tr>
</tbody>
</table>
Chapter 4

3D Integration

As alluded to in the previous chapter, 3D integration can provide several benefits, especially in regards to memory circuits. The main benefits of 3D integration are as follows. First, 3D integration can benefit the critical wires that connect memory and logic by allowing memory circuits to be stacked on top of logic. When logic and memory are integrated in this fashion the power consumption is drastically cut because of a reduction in the parasitics of the interconnect between the logic and the memory. Second, 3D integration allows dies that are manufactured in different technologies to be closely integrated. This can reduce power consumption as logic can be integrated with memory that is manufactured in processes optimized for memory circuits. Such optimizations include low leakage processes, DRAM trench capacitors or adaptions that enable more exotic memories such as MRAM\cite{64}. Third, 3D stacking can be used to 3D integrate logic blocks. This reduces the wiring in the logic block leading to a reduction in interconnect parasitics. This reduction in interconnect parasitics increases performance and reduces the power consumption of the logic. The amount of power savings depends on two components. First, how much of wire length reduction can be expected and second how a given percentage of wire length reduction translates to in terms of reducing the overall power consumption of the system. Section 4.3.1 and Section 4.3.2 addresses these components respectively by analyzing them in the context of four designs.
In order to reap the benefits of 3D integration described above it is necessary to complete 3D stacking, fabrication, integration and processing successfully. Fabricating 3D integrated circuits is not trivial and requires more technological capability in the foundry and assembly house than for 2D circuits. A summary of the technologies required for 3D fabrication is given in Section 4.2, along with examples of 3D integration technologies from academia and industry. 3D integration also adds five key challenges on the design side that must be overcome. These challenges are: power delivery, thermal density, design for test, clock tree design and floorplanning and are detailed in Section 4.1.

4.1 Overview of 3D Integration Technologies

Three-dimensional integration[18] is a broad term that encompasses several methods of integrating devices vertically. These methods can be broadly categorized into three categories: 3D stacking with TSVs, 3D packaging and monolithic 3D fabrication. If a circuit is fabricated sequentially layer by layer from one base layer then the 3D integration approach is a monolithic 3D approach. The other two approaches involve stacking and assembling multiple circuits that have been fabricated independently. What differentiates the two is whether the integrated dies are connected with a through silicon via (TSV), which makes the approach a 3D Stacking with TSV approach or whether the die are just interconnected by wire bonding on the periphery of the die, which is known a 3D packaging. Finally, 3D stacking approaches that include TSVs can be differentiated depending on whether the stacking is die-to-die, die-to-wafer or wafer-to-wafer. Figure 4.1 shows the different 3D integration approaches.

4.1.1 Monolithic 3D Approaches

What differentiates monolithic 3D circuits from other circuits is that they are fabricated from one base layer. Monolithic 3D processing techniques have two limitations. First, they require a high processing temperature and second they produce transistors that are inferior to the ones
created in a regular 2D process ones. As a result of these two limitations monolithic 3D approaches are currently not as commercially viable as the other 3D integration approaches. As such, we just give a brief overview of the three major monolithic 3D processing approaches. For a more comprehensive overview of the different monolithic approaches see Souri et al.[62]. Essentially, there are three main monolithic approaches: Beam Recrystallization, Silicon Epitaxial Growth and Solid Phase Crystallization. They are described below.

**Beam Recrystallization**

Beam Recrystallization [35] involves depositing polysilicon on top of an existing substrate. The polysilicon is then re-crystallized using an intense laser or a electron beam. A thin film transistor is then fabricated on top of the polysilicon. There are two downsides to this approach. First, melting the polysilicon requires a high temperature. Second, the process causes uneven grain sizes, which degrades the performance of each transistor.

**Silicon Epitaxial Growth**

Silicon Epitaxial Growth [52] involves etching a hole in a passivated wafer and epitaxially growing a single silicon crystal seeded from an open window in the dielectric. This approach
has the potential to create higher quality devices than Beam Recrystallization because the second layer of silicon has more even grain sizes. This approach has two major disadvantages. First, like Beam Recrystallization, the process requires a high process temperature. Second it is difficult to use this approach over metallization layers. High density SRAM has been successfully built using this approach[33]. The density in that SRAM was achieved by eliminating the cell areas consumed by n-wells completely and replacing the the cross-coupling wires with 3D vias.

**Solid Phase Crystallization**

Solid Phase Crystallization [37] involves randomly crystallizing an amorphous film of silicon to form a film of polysilicon. On top of this film the second device layer is created. The performance of devices made in this fashion can be greatly improved by removing the silicon grain boundaries. This can be done in one of two ways, either by patterned seeding of germanium, or by using Metal Induced Lateral Crystallization. Although the quality of the devices made in this fashion is quite good, they are still inferior to the quality of single silicon crystal devices. The main advantage of Solid Phase Crystallization is that it has been shown to work at lower process temperatures. Furthermore, stacked SRAMs and EEPROM [6] have been fabricated using this approach.

### 4.1.2 3D Stacking with TSVs

3D stacking with TSVs is characterized by two features that set it apart from the other two 3D integration approaches. Unlike 3D packaging, 3D stacking with TSVs includes as the name implies TSVs and unlike monolithic 3D approaches, 3D stacking with TSVs is realized by assembling multiple tiers that have been fabricated independently. The system can be stacked in three different ways. The first way is known as wafer-to-wafer stacking and involves stacking both wafers undiced. This stacking gives the best yield of the three and costs the least. The second way the system can be stacked is die-to-wafer, where a diced tier is stacked on a undiced wafer. The final stacking arrangement is die-to-die stacking, where both tiers have been diced.
Die-to-die stacking allows the use of known-good-die testing to improve yield.

Regardless of which tiers have been diced, stacking can only occur in three different orientations: face-to-face, face-to-back or back-to-back. In this terminology face refers to the side by the top most metal layer and back refers to the substrate side. The 3D interconnect between the two stacked tiers is different depending on which orientation is used. For face-to-face the connections between tiers are through microbumps. The interconnect parasitics of microbumps is similar to a regular via, which is much less than the parasitics of a through-silicon via. Furthermore, unlike a TSV the microbumps never block any metal routing layers. For the other two orientations the connections occur through a TSV. For a back-to-back orientation the TSV has to go through two substrates resulting in a longer TSV with worse interconnect parasitics than a face-to-back TSV. For this reason back-to-back is generally avoided if possible. Figure 4.2 shows the three orientations.

![Figure 4.2: The three different stacking orientations, with the interconnect and substrate shown.](image)

There are four key enabling technologies that make building 3D die stacked systems with TSVs possible. These are alignment, bonding, wafer thinning and the ability to successfully
fabricate TSVs. They are discussed in the following sections.

Alignment

Alignment is crucial to 3D integration as it influences the density and yield of 3D interconnects. The way alignment works is that each of the two tiers that are to be aligned have two keys that are used as reference points. The keys are optically aligned under microscopes. Typically either direct or indirect alignment is used. If one of the substrates is transparent to visible or infrared light, direct alignment can be used. In direct alignment a pair of microscopes is used to simultaneously image both tiers. The tiers are then shifted and rotated until both keys precisely align with each in the microscopes. If neither tiers are transparent to visible or infrared light, direct alignment has to be used. In indirect alignment the first tier is aligned to a reference point and then lifted up. The second tier is brought in an aligned to the to the same reference point. Indirect alignment is not as accurate as direct alignment.

Bonding

Bonding is very important to 3D integration because if the bond does not hold, the circuit will not functioning correctly. There are several types of bonding technologies that can be used to bond tiers together. The are four types of bonding that are most commonly used are: direct oxide bonding, metallic thermo compression bonding, adhesive bonding using a glue and finally soldering.

Direct oxide bonding involves bonding together two tiers using an oxide on the surface of the tiers, which is usually silicon dioxide. The main advantages of direct oxide bonding is that can be done at a very low temperature and it is very easy to integrate into semiconductor processing as deposited oxides are typically used in most integrated circuit process technologies including silicon on insulator. The disadvantage of direct oxide bonding is that it requires good chemical-mechanical planarization and sophisticated cleaning of the tier beforehand.

Metallic thermo compression bonding is done with either copper or gold. Gold bonds
more easily than copper and does so at a lower pressure and temperature. Copper, however, is cheaper, has a higher bond strength and is more easily incorporated into semiconductor processing. The main advantages of metallic thermo compression bonding is that it provides both mechanical attachment and electrical connectivity in the same step and produces no out-gasing during bonding. A disadvantage of metallic thermo compression bonding is it requires a higher temperature and more pressure to bond than adhesive polymer bonding. Additionally, due to the high bonding temperature, if the temperatures of the two bonding wafers are not the same, thermal expansion of the metals can cause alignment errors. Copper thermo compression bonding is used by Tezzaron[54].

The most successful adhesive bonding technology used in 3D integration is the polymer adhesive bonding technology developed at the Rensselaer Polytechnic Institute[39]. There are four main advantages of this bonding technology over the other bonding technologies. Since this bonding uses a glue it is relatively insensitive to the topology of the bonding surface and can join practically any wafer materials together. Furthermore, the bonding features at a low temperature and is compatible with standard CMOS processing.

Soldering is a very well known technology commonly used in the assembly of printed circuit boards. It can also been used for 3D integration[36]. Like metallic thermo compression bonding, it provides both mechanical attachment and electrical connectivity in the same step.

**Wafer Thinning**

Building TSVs with a high enough aspect ratio to go through an unthinned silicon wafer is infeasible. As a result of this, successful thinning of the wafer is a necessity. Thinning of the wafer is typically accomplished in the following manner. The back of the wafer is course ground using a wheel with large diamond grains. This is followed by a fine grind using a wheel with smaller diamonds. Finally, chemical mechanical polishing (CMP) is done to reduce stress. Once a wafer has been thinned it is very hard to handle without damaging it. For this reason they are often mounted to handle wafers if more processing is required.
TSV Fabrication

Since TSVs are the main enabling technology of 3D integration, fabricating them successfully is very important. TSVs can be divided into different categories depending on when they are fabricated in respect to the regular metal wires. TSVs that are created before the first regular metal layer is created are known as front-end-of-line (FEOL) TSVs. Since FEOL TSVs are manufactured before metallization they are not made out of metal instead they are made out of highly doped polysilicon. Since polysilicon is more resistive than metal, FEOL TSVs will be at a disadvantage in terms parasitic resistance when compared to metal TSVs. FEOL TSVs are utilized in the 3D integration processes of Zycube[38]. TSVs that are fabricated with the metallization are known as back-end-of-line (BEOL) TSVs. BEOL TSVs are made out of metal, which is typically copper or tungsten. BEOL TSVs are used in 3D integration technology of MIT Lincoln Laboratory[4] and Tezzaron[54]. It is also possible to fabricate TSVs after all the processing steps have taken place. The TSVs made in this fashion are known as post-BEOL TSVs. A big advantage of post-BEOL TSVs is that it allows a die that has been manufactured in processes that does not have the capability to manufacture TSVs to be 3D integrated. There are two main disadvantages of post-BEOL TSVs. First, post-BEOL TSVs require a large exclusion zone where the TSV will be fabricated, which uses up area and limits the 3D interconnect density. Second, creating post-BEOL TSVs is more complicated than creating regular FEOL and BEOL TSVs.

Regardless of what material the TSV is made from, the silicon must be selectively removed to form a trench for the TSV. The trench must be accurately built with an appropriate side-wall angle. Two methods are most commonly used to create the TSV trench. The first method is known as deep reactive ion etching (DRIE) and is used in Tezzaron’s 3D integration process[54]. DRIE involves using alternating phases of sulfur hexafluoride to etch the trench and octafluorocyclobutane to passivate the side-walls of the trench to insure a vertical etch. The other method used to create TSV trenches involves using UV laser drilling. This approach is used in MIT Lincoln Laboratory’s 3D fabrication process[4]. DRIE and laser drilling have their re-
spective advantages and disadvantages. A significant advantage of laser drilling is that it does not require additional masks, which reduces cost. However, laser drilling has three disadvantages when compared to DRIE. First, laser drilling can damage active areas close to the trench because of the heat caused by the laser. Second, the drilling process creates debris which can cause problems for other fabrication steps. Third, the side-walls of laser drilled trenches are not as smooth as DRIE trenches.

4.1.3 3D Packaging

3D packaging is different than the other two 3D integration techniques because it does not contain any through silicon vias. Instead the connections in 3D packaging are typically made on the periphery of the integrated dies using wire bonding. As a result 3D packaging can not reduce wire length to the same extent as the other two 3D integration techniques. However, 3D packaging still has benefits including: integrating dies that are manufactured in different technology processes, reducing footprint and exploiting “known good die” to improve overall yield. Additionally, 3D integration allows embedding passive components such as decoupling capacitors, which is not possible with either of the other two processing technologies. An example of an advanced 3D packaging process known as 3D MINT is described in the following section.

Irvine Sensors 3D MINT

The Irvine Sensors 3D MINT process [63] is a 3D packaging process. As such it does not have through silicon vias. The process provides three major benefits. First, it allows embedded passives such as decoupling capacitors, which is not possible in monolithic or 3D die stacked TSV approaches. Second, it allows integration of devices manufactured in heterogeneous processes. Third, it allows dies to be tested before integration. This means that the “known good die” phenomenon can be exploited to improve yield. 3D MINT uses conventional CMOS processing to do wiring as opposed to wire bonding that is used in most packaging processes. The process
works in the following manner[68].

First, holes are cut into silicon or alumina substrates. Embedded elements, which include active dies, passives, and copper plugs, are then placed in the holes. Epoxy resin is used to hold the embedded elements in place in the substrate. Once the embedded elements have been inserted into the substrate, a conventional integrated circuit process that provides up to six layers of metal is used to interconnect the pads, the passives and the copper plugs. The substrate layers are then interspersed between copper thermal management layers and stacked vertically. The thermal management layers are very important to draw the heat away from the active devices to the outside. The resulting stack is then packaged in traditional manner, in this case using a ball grid array. Figure 4.3 shows a cross sectional drawing of an example 3D MINT package with four active dies, and four thermal management layers.

\[ \text{Figure 4.3: Cross sectional view of Irvine Sensors’ 3D Mint Process} \]

\section*{4.2 Challenges for 3D}

In order to reap the benefits of 3D integration there are five key design challenges that must be overcome. The first design challenge is power delivery. This is a challenge because in a 3D integrated circuit there is a larger supply current flowing through the package power delivery pins, along with a longer power delivery path than in a comparable 2D system. The second
challenge is increased heat density because in 3D integrated circuit the active devices are closer to each other and have less capacity to remove heat. The third design challenge is that 3D clock tree distribution is more difficult to accomplish than in 2D because the most commonly used methodologies and design tools are geared towards 2D designs. In addition, and process variation between the different tiers makes it harder to keep skew, jitter and power consumption down. The fourth challenge is design for test. This is harder to do in 3D because TSVs provide another point of failure and post fabrication repairs such as Focused Ion Beam are more difficult to perform in 3D. Finally, the last design challenge is that floorplanning is drastically different in 3D than in 2D and all the four aforementioned issues must all be taken into account during 3D floorplanning.

4.2.1 Power Delivery

The effect of on-chip power delivery on signal integrity is a major challenge in three-dimensional integrated circuits (3DICs) when compared to conventional two-dimensional (2D) designs. The three main reasons for this is that 3D designs have larger supply currents flowing through the package power pins, longer power delivery paths, and additional resistance on the power delivery network contributed by Through Silicon Vias (TSVs) compared 2D chips[30, 28]. This can cause all the same issues that are seen in traditional 2D designs including: thermal reliability issues, signal quality, jitter, timing simultaneous switching noise, and incorrect functionality in extreme cases. Just as in 2D designs, timing delay effects of the power delivery networks (PDNs) originate from fluctuations in the rail voltage. These fluctuations appear as a result of high demands for instantaneous current. As the rail voltage fluctuates, the delay through each logic stage increases or decreases, which can lead to timing violations and incorrect behavior. Approaches typically used in 2D to address power supply noise do, however, work in 3D. These approaches include widening power delivery wires, optimizing the power grid topology, and placing decoupling (bypass) capacitors close to where they are needed[61]. Furthermore, approaches unique to 3DICs have been proposed, including multistory power delivery [30],
decoupling layers sandwiched between tiers[67], and on-chip DC-DC converters[60].

4.2.2 Thermal Density

The second 3D specific issue is thermal density[57]. In 3D heat dissipating active devices are now stacked directly above and below each other, which means that a 3D chip will have a higher heat density than a comparable 2D chip. It is also harder to remove heat from a 3D chip because only the first tier is next to the heat sink. As you move further away from the first tier, heat removal becomes successfully more difficult. The side effect of increased thermal density is that a 3D chip is likely to see a larger thermal gradient than a 2D chip would. There are some solutions that can be used to mitigate these thermal problems. The first solution is to use thermally aware 3D floorplanning and placement to keep thermal density down as demonstrated by Goplen et al [20]. The second solution is to use packaging that has a lower thermal resistance. The downside to this is that it increases the cost of the system drastically. A third solution is to use dielectrics that have good thermal conductance, as they can help greatly with heat removal. Finally, increasing the footprint of the power and ground networks will help remove heat as the metals in the power and ground network have a good thermal conductance that will pull heat out of the system.

4.2.3 Clock Tree Design

Designing clock trees in 3D is more difficult than in 2D because the most commonly used methodologies and design tools are geared towards 2D. Furthermore, process variation between different tiers makes it harder to minimize skew, jitter and power consumption in a 3D clock tree. There are five major approaches that can be used to design a 3D clock tree. The first approach is to build a conventional clock tree using an H or X-Tree structures in each individual tier connected at the root of the tree. This approach was used by Davis et al. in their Winograd FFT[14]. The second approach is to build the clock tree using an H or X-Tree structure that mainly resides in one tier and feeds the clock sinks in other tiers directly by using 3D vias. The
second approach requires an order of magnitude more 3D vias, than the first because a 3D via is required for every leaf node. As a result the second approach is harder to test. An experimental comparison of the two approaches was performed by Pavlidis et al. [56]. In the third approach different tiers can be treated as different clock domains and standard clock domain crossing techniques can be used such as passing the signal through a series of synchronizing flip-flop structures. The drawback to this approach is that there is significant design and area overhead. Furthermore, this approach will usually add at least two clock cycles of latency to any signal that crosses between the different domains. The upside to this approach is that several major EDA vendors supply functional verification tools that support clock domain crossing. For the right design that has very structured dataflow between tiers this may be a good design approach. The fourth approach is to use a network-on-chip or asynchronous signaling. Using a network-on-chip in 3D has been demonstrated experimentally by Mineo et al.[49]. This approach suits SoC designs well. In the fifth approach all the flip-flops are placed on one tier and all the logic is placed on either the same tier or a different tier. This results in the clock tree residing completely on one tier. The upside to this approach is that it is well equipped to deal with process variations between tiers. The downside of this approach is that it may require a significant number of 3D vias to connect logic and registers residing on different tiers.

4.2.4 Design for Test

Design for test is more difficult to do in 3D than in 2D and can be more costly[41]. There are four main reasons for this. First, 3D processing provides additional failure mechanisms due to added 3D processing steps, such as: wafer thinning, alignment, bonding and building TSVs. These mechanisms need to be understood and modeled correctly to be successfully overcome. Second, thinned wafers have a harder time dealing with the weight of the probes on the probe card. This makes wafer probing with a high number of probes very challenging. Third, post fabrication repairs such as Focused Ion Beam are hard or impossible to do with 3D integrated structures. Fourth, effective yield is quite different in 3D because it depends on
stacking method used. If die-to-die stacking is used, the dies can be tested before integration, which allows “known-good-die” testing. Known-good-die testing can be exploited to improve the yield of a 3D integrated system above a 2D one with a similar circuit area. The reason for this is that one defect will only ruin one tier of the 3D system rather then ruining the whole circuit for a 2D system. Exploiting known-good-die for wafer-to-wafer stacking is not possible because individual good dies can not be paired together. However since wafer-to-wafer stacking typically has better yield than die-to-die, it is complicated to determine which stacking method will give a better yield for a given system.

Several solutions that address 3D design for test issues have been proposed. First, adding self testing circuitry that tests the integrity of 3D vias and then scans out the results in a fashion similar to flip-flop scan chains[72]. Second, drastically increasing the use of built in self test and adding redundant structures, especially TSVs. Third, for certain architectures such as chip multiprocessors that have repeated structures, designing the architecture so that all or some tiers are identical greatly simplifying the manufacturing and increasing yield. Fourth, utilizing 3D scan chains[76].

4.2.5 3D Floorplanning And Placement

Of all the 3D specific issues, 3D floorplanning and placement is one of the better studied ones. Furthermore, floorplanning is unique to the other four issues because it has to take them into account. 3D floorplanning and placement can be either fine or coarse grain. In fine grain 3D placement the modules that need to be placed are divided between one or more tiers. Several fine grain 3D placement tools have been demonstrated that take the thermal effects into consideration[24, 13, 21, 10]. One disadvantage of fine grain 3D integration is that it does add complications to testing, as dies can no longer be tested before integration. The use of fine grain 3D placement in full custom designs is typically limited to circuits that are interconnect dominated such as memories and image sensors. In coarse grain placement individual modules are divided between the different tiers. A common arrangement that is often used is memory-
on-logic. In memory-on-logic one or more tiers of memory are integrated with one or more tiers of logic, which is done because the interconnect from the logic to the memory is very critical and tends to use a significant amount of power.

### 4.3 Wire Length Reduction

As stated above, 3D integration has the potential to reduce power consumption significantly. The power reduction depends on two components. First, how much of wire length reduction can be expected and second how a given percentage of wire length reduction translates to in terms of decreasing the overall power consumption of the system. Section 4.3.1 and Section 4.3.2 addresses these components respectively.

#### 4.3.1 3D Wire Length Reduction

This section surveys the literature on the wire length reduction achieved by 3D integration. Hentschke et al. present a quadratic placement algorithm for 3D and report a 7% reduction in wire length with 2 tiers and 32% reduction with 5 tiers when 3D placing the ISPD 2004 benchmarks using an average of 13046 TSVs for 3 tiers[24]. Davis et al. show a 14.3% percent reduction in wire length on a 24 bit fixed point Radix-2/4/8 FFT engine using 837 TSVs[14]. Deng et al. show a 20% total wire length reduction for the ami33 benchmark and a 30% total wire length reduction for ami49 using their 3D floorplanning technique[15]. Zhou et al. report a 63.0% reduction in total wire length (from 182.42 m down to 67.42 m ) for a 3 tier version of their layout of a 3D LDPC decoder that uses 10631 TSVs, when the architecture is compared to an equivalent 2D counterpart[79]. The reason the LDPC circuit shows such drastic wire length reduction is that low density parity checkers typically interconnect dominated. Xie et al. show a wire length reduction of 32% when moving various benchmark circuits to 3D[77]. Banerjee et al. calculate, using closed-form expressions, that the wire length reduction of using 2 tiers should be around 29% [3]. The wire length reduction for all the studies is summarized
Table 4.1: Average Wire Length Reduction.

<table>
<thead>
<tr>
<th>Author</th>
<th>Approach</th>
<th>Reduction</th>
<th>Tiers</th>
<th>TSVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hentschke et al. 2006</td>
<td>3D Placement</td>
<td>7-32%</td>
<td>2-5</td>
<td>13046</td>
</tr>
<tr>
<td>Davis et al. 2005</td>
<td>FFT</td>
<td>14.3%</td>
<td>3</td>
<td>837</td>
</tr>
<tr>
<td>Deng and Maly 2001</td>
<td>2.5D Placement</td>
<td>20-30%</td>
<td>3-5</td>
<td>-</td>
</tr>
<tr>
<td>Zhou et al. 2007</td>
<td>LDPC Decoder</td>
<td>63.0%</td>
<td>3</td>
<td>10631</td>
</tr>
<tr>
<td>Xie et al. 2006</td>
<td>Benchmarks</td>
<td>32%</td>
<td>3</td>
<td>-</td>
</tr>
<tr>
<td>Das et al. 2003</td>
<td>Benchmarks</td>
<td>28-51%</td>
<td>2-5</td>
<td>-</td>
</tr>
<tr>
<td>Banerjee et al. 2001</td>
<td>Calculation</td>
<td>29%</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>This Work</td>
<td>FFT</td>
<td>56.9%</td>
<td>3</td>
<td>8280</td>
</tr>
</tbody>
</table>

in Table 4.1.

The conclusion that can be drawn from this literature survey is that for an average circuit it is reasonable to expect around 30% wire length reduction.

4.3.2 3D Power Reduction

In the previous section we have explored how much 3D integration can be expected to reduce wire lengths. In this section we explore how much of an overall power reduction can be expected given a certain decrease in wire length. The methodology used is as follows, three designs were synthesized to the highest possible operating frequency using Synopsys Design Compiler. The three designs were an AES cryptography core (AES), floating point processing element (PE) and a parallel merge multiple-input and multiple-output wireless decoder[50] (MIMO). The designs were synthesized at two technology nodes, a 180 nm node using a commercial standard cell library with over 300 cells and a predictive 45 nm node using OSU’s standard cell library. The parasitics of the design were then extracted and read into PrimeTime along with the design. In PrimeTime the effect of reducing the average wire length by a given percent was simulated for all the designs and the power usage and delay was characterized. Figure 4.4 shows the power reduction for a given average wire length reduction and the improvement in the critical path.
Figure 4.4: Power and critical period reduction given a certain wire length reduction.
Chapter 5

Case Study 1: Using 3D for Memory Partitioning

This chapter presents the first case study. In this case study we explore the benefits, in terms of power consumption of re-architecting systems to explicitly to use memory-on-logic 3D integration. To do this exploration we implement an FFT processor in MIT Lincoln Lab’s 3D FDSOI 1.5 V process[65]. The FFT processor that is implemented can compute 32-bit floating FFTs and IFFTs that are 1024-point (60 cm resolution) or smaller in size. Overall, the power consumption in the FFT processor is reduced by using two techniques. These two techniques are 3D integration and a novel memory division scheme[70, 71]. The two techniques complement each other because the memory division scheme trades interconnect complexity for an improvement in the energy required for each memory access, whereas the 3D integration lessens the impact of the more complex interconnect. The memory division scheme serves two purposes. First, dividing up the memory reduces power consumption because smaller memories use less energy per memory access because there is not as much capacitance on the bitline. Second, dividing up the memories improves memory bandwidth in two ways. First, the divided memories can be accessed simultaneously. Second, the smaller divided memories have a smaller bitline capacitance allowing the memories to operate at a faster rate providing more memory bandwidth.
There are, however, two drawbacks to dividing up the processing memory. The first drawback is that a small memory will require more area as each divided memory will require its own set of peripheral logic (write driver and sense amp) leading to an overall increase in area. The second drawback is that dividing the memory up leads to an overall increase in interconnect wiring between the memory and the logic. The effect of this is mitigated by using 3D integration. The tradeoffs that come into result when the memory is divided up are shown in Figure 5.1.

![Figure 5.1: The memory division tradeoffs with the memory division scheme on the right.](image)

In addition to the two techniques that are used to reduce power consumption, this chapter explains the tool flow required to realize the FFT processor with off-the-shelf commercial 2D tools and explores the impact of 3D integration on the thermal profile of the system. Overall, the chapter is organized in the following manner. Sections 5.1, 5.2 and 5.3 discuss the FFT the memory division scheme, the overall system and the 3D implementation of the system respectively. Section 5.4 contains a thermal analysis of the 3D implementation that was done in conjunction with Samson Melamed. Section 5.5 contains the results of the case study which includes an evaluation of the memory division, a comparison between 2D and 3D implementations of the design and comparison to other FFT architectures. Section 5.6 setup used to test the fabricated chip. Finally, Section 5.7 concludes the chapter.
5.1 Memory Division Scheme

As we have explained in the previous section splitting the memory into several smaller memories is beneficial as smaller memories are faster and consume less power. Furthermore, since each memory subgroup can be accessed simultaneously, the system can perform a greater number of reads and writes per cycle. The memory division scheme is based on an in-place radix-2 Cooley-Tukey FFT[11]. There are advantages and disadvantages of using radix-2 over larger and split radixes. One of the main advantage of using radix-2 FFT is that any power of two transform that is smaller than the maximum size permitted can be performed. This is a significant advantage for applications that need to perform FFTs of varying sizes, such as image processors. This flexibility comes with a price. When compared to split-radix FFT optimized for a specific FFT size the radix-2 will need to perform more arithmetic operations than split-radix FFT. Cooley-Tukey FFTs require each data element to be swapped with the data element located at the address that corresponds to the reverse of its binary digits. This is known as decimation and can be performed in either the time or the frequency domain. The difference between the two is that in decimation in time the swapping occurs before the FFT computations but in decimation in frequency the swapping occurs after the FFT computations. In our 3D implementation decimation is performed in the time domain.

The algorithm requires $\log_2(N)$ stages where $N$ is the size of the FFT. In each of these stages the data element stored at location $x$ is multiplied with the data element stored at location $x + 2^{stage}$. This results in the data in these two locations being data dependent on each other. Based on this dependency we can represent the FFT as a hypercube. In such a hypercube each node corresponds to a data location and each vertex represents a data dependency. As we stated earlier the main goal is to split up the memories. Reaching this goal is equivalent to partitioning the graph of the hypercube into overlapping partitions with two constraints. The first constraint is that every edge has to be contained in at least one partition. The second constraint is that all the partitions have to contain the same number of vertices to ensure that reads and writes are distributed evenly between all the memories. Once a partitioning scheme
that meets these requirements has been generated it can be translated to an architecture in the following manner. In such a partitioning scheme every partition corresponds to a processing element that implements a radix-2 butterfly. Additionally, every unique intersection between sets corresponds to a memory connected to all the processing element sets that come together at that given intersection. We illustrate this with a simple case of a 4-point FFT. This case is particularly important as it is the base case that higher order partitions can be derived from. In the case of the 4-point FFT, there is only one way to partition it so that it meets our criteria. This is shown in Figure 5.2.

![Diagram of a 4-point FFT and basic partitioning.](image)

Figure 5.2: Four point FFT on the left along with the basic partitioning on the right.

Looking closely at this partition we see there are four partitions, which correspond to the following sets: $P_0 = \{00, 01\}$, $P_1 = \{00, 10\}$, $P_2 = \{01, 11\}$ and $P_3 = \{10, 11\}$. Furthermore, for this partitioning scheme there are four unique intersections: $P_0 \cap P_1$, $P_0 \cap P_2$, $P_1 \cap P_3$ and $P_2 \cap P_3$. Using our definition that every partition corresponds to a processing element and every unique intersection corresponds to an individual memory, the architecture for this basic
A partitioning scheme can be realized in the manner shown in Figure 5.3.

![Diagram of Memories and Processing Elements](image)

**Figure 5.3**: Architecture for basic partitioning.

In order to illustrate how the partitioning approach scales, we have to first introduce a specific node labeling scheme. We label each node with a unique bit-string so that the labels of all nodes that are connected by a vertex differ by one bit. This is equivalent to a hamming distance of 1 and matches the mathematical definition of a hypercube. Based on this notation, each partitioning scheme can be described as a series of expressions that match the nodes assigned to that particular partition. In such an expression, “0” and “1” match a 0 or 1, respectively, whereas a “*” matches both, in the given position of the bit-string. In addition to a bit-string, a partitioning scheme can also be expressed in a form that is similar to a Karnaugh Map. In a Karnaugh Map the hamming distance between all vertically and horizontally adjacent squares is 1. This makes it very easy to visually see that the first constraint is met because every vertical and horizontal adjacency in the map will be included in a partition. We use both notations to describe the basic partitioning scheme used for Figure 5.2 and 5.3, they are shown in Table 5.1 and Figure 5.4 for the bit-string and the Karnaugh Map respectively.
Table 5.1: Basic partitioning scheme for 4-point FFT.

<table>
<thead>
<tr>
<th>Partition</th>
<th>Bit 0</th>
<th>Bit 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>P1</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>P3</td>
<td>1</td>
<td>*</td>
</tr>
</tbody>
</table>

Figure 5.4: Karnaugh Map for partitioning scheme for 4-point FFT.
The basic partitioning scheme can be extended in two ways. First, any partitioning scheme can be scaled up to a larger power of 2 by simply appending a "*" to the end of every expression for that partition. Second, a new partitioning scheme that has twice the number of partitions can be derived from another partitioning scheme in the following manner: A copy of every expression is created. For every partition that has an expression that starts with "*" a "*0" and "*1" is appended to the original and the copy respectively. For each expression that starts with either “1” or “0”, a “0*” or “1*” is respectively appended to the original and the copy. An extension of the basic partitioning scheme from four to eight partitions is shown in Table 5.2 in bit-string format and in Figure 5.5 in Karnaugh Map format. Additionally, a diagram showing this partitioning scheme applied to a 64 point FFT is shown in Figure 5.6.

Table 5.2: Eight partition partitioning scheme.

<table>
<thead>
<tr>
<th>Partition</th>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>0</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>P1</td>
<td>0</td>
<td>*</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>P2</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>*</td>
<td>0</td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>P4</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>P5</td>
<td>*</td>
<td>1</td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>P6</td>
<td>1</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>P7</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>*</td>
</tr>
</tbody>
</table>

After performing the hypercube split, there is an additional split that can be performed based on the parity of the memory addresses. The additional split takes advantage of the fact that hypercubes are bipartite. This can be demonstrated by a simple coloring scheme that splits the nodes into different groups based on whether there is an even or odd number of 1’s in the bit-string. The bipartite nature of the graph allows the memory to be further subdivided.
Figure 5.5: Karnaugh Map for eight partitions.
Figure 5.6: 64 point FFT hypercube with the partitioning shown.
into even and odd parity groups. This is especially convenient as every butterfly operation in a radix-2 FFT requires one operand to be from an even parity memory location and the other from an odd parity location.

It is important to understand the effect of extending the number of partitions on the relationship between the memories and processing elements after both the hypercube and even/odd splits have been performed. The number of memories after both splits is equal to \(((\text{Number of Partitions})^2)/2\). Each processing element is connected to \((\text{Number of Partitions})/2\) memories and each memory is always connected to two processing elements. Table 5.3 shows this relationship for several partitioning schemes. From this table is it important to note that because the number of memories increases exponentially with the number of partitions it rarely makes sense to extend the number partitions beyond sixteen unless extremely large FFTs are involved.

Table 5.3: Relationship between partition scaling, memories and processing elements.

<table>
<thead>
<tr>
<th>Number of Partitions</th>
<th>Processing Elements</th>
<th>Number of Memories</th>
<th>Memories Connected to Each PE</th>
<th>PEs Connected to Each Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>32</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>128</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>512</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>2048</td>
<td>32</td>
<td>2</td>
</tr>
<tr>
<td>128</td>
<td>128</td>
<td>8192</td>
<td>64</td>
<td>2</td>
</tr>
<tr>
<td>256</td>
<td>256</td>
<td>32768</td>
<td>128</td>
<td>2</td>
</tr>
</tbody>
</table>

Finally, it is important to note that the partitioning approaches that have been detailed so far are not the only possible ones. There are other partitioning approaches that have different properties in terms of PE to memory ratio and connectivity that are not extended from the approach in Figure 5.2. For example for a 16-point FFT 32 way partitioning is possible that is
drastically different than the eight way partitioning shown earlier. This approach is shown in Figure 5.7 and Table 5.4 and the relationships between the memory and processing elements in Table 5.5.

Figure 5.7: Basic Partitioning shown for N=16 hypercube
Table 5.4: The alternate thirty-two partition partitioning scheme.

<table>
<thead>
<tr>
<th>Partition</th>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P1</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>*</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P3</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>*</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P5</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P6</td>
<td>1</td>
<td>*</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P7</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P8</td>
<td>0</td>
<td>*</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P9</td>
<td>*</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P10</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P11</td>
<td>*</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P12</td>
<td>0</td>
<td>*</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P13</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P14</td>
<td>1</td>
<td>*</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P15</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>P16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>P17</td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>P18</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>P19</td>
<td>0</td>
<td>0</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>P20</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>P21</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>P22</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>P23</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>P24</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>P25</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>P26</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>P27</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>P28</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>P29</td>
<td>1</td>
<td>0</td>
<td>*</td>
<td>1</td>
</tr>
<tr>
<td>P30</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>*</td>
</tr>
<tr>
<td>P31</td>
<td>1</td>
<td>0</td>
<td>*</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.5: Relationship between partition scaling, memories and processing elements for the 32 partition partitioning approach.

<table>
<thead>
<tr>
<th>Number of Partitions</th>
<th>Processing Elements</th>
<th>Number of Memories</th>
<th>Memories Connected to Each PE</th>
<th>PEs Connected to Each Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32</td>
<td>32</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

5.2 System Overview

To build the overall system we use the eight-way partitioning scheme detailed in Table 5.2. The overall system is shown in Figure 5.8 and consists of four different components: eight processing elements, one controller, thirty two SRAMs, and eight ROMs. The processing elements are the core of the system, implementing the FFT butterfly with four floating point multipliers and six addition/subtraction units. The internal structure of the processing element is shown in Figure 5.9. The controller orchestrates the overall operation of the system by setting the addresses and read enables of the memories. The controller only requires three signals to communicate with a processing element. The SRAMs implement the main processing memory using 8-transistor dual ported SRAMs. The ROMs store the FFT twiddle factors and are implemented as single ported NOR type ROMs[19].

5.3 Tool Flow and 3D Implementation

A low power FFT processor that utilizes the memory division scheme detailed in Table 5.2 of Section 5.1 was implemented in MIT Lincoln Lab’s 3D process. Before the implementation is explained, it is important to understand the manufacturing and 3D integration process. The MIT Lincoln Lab’s manufacturing process is a three tier, 180 nm wafer scale 3D integration process[4]. The process uses a 1.5 V low power fully depleted silicon-on-insulator CMOS technology with one layer of polysilicon and three metal layers per tier. Additionally, there is a
Figure 5.8: The divided memory architecture.

Figure 5.9: The block diagram of the processing element.
back-metal layer between the top two tiers, and a metal layer on top of the entire stack. The bottom tier is referred to as A, the middle tier B and the top tier C. Tier A is closest to the heat sink. Tier C is the only tier which has off-chip inputs and outputs. Tiers B and C face down, while tier A faces up. Figure 5.10 shows a cross-section of the process with the silicon-thru vias. In this process the dimensions a single thru-silicon via is $2.5 \times 2.5 \mu m$ and each TSV can be placed on $3.9 \mu m$ pitch.

The design is a mix between standard cell and full custom design. The processing elements and controller are built using standard cells, while the memories (SRAMs and ROMs) are implemented using full custom design. One of the benefits of using full-custom memories over an off-the-shelf memory generator, is that it allows the thru-silicon vias to be implemented as part of the memory. This simplifies the design flow as the thru-silicon vias get placed with the memory. Not all the TSVs in the system connect memory to logic. In the system there are also 24 logic-to-logic TSVs, which must placed in the final assembly stage in a predetermined location.

The design flow, which is shown in Figure 5.11 mainly consists of the followings steps: 3D floorplanning, partitioning and selecting the locations for the memories. In the 3D floorplanning phase, the main objective is to get the memories as close as possible to the processing elements that use them. We define PE0, PE1, PE2 and PE3 to be the lower numbered processing elements and PE4, PE5, PE6 and PE7 to be the upper numbered processing elements. It can be seen from Figure 5.8 that every memory is connected to one lower numbered PE and one upper numbered PE. To exploit this connectivity we partition the system so that the controller and the memories are on the middle tier (tier B). The upper numbered PEs and their respective twiddle factor ROMs are placed on tier A and the lower numbered PEs along with their ROMs placed on tier C. This partitioning scheme is relatively well balanced as only $1.310 mm^2$ of area is left unused on tier B. This corresponds to approximately $5.59\%$ of the total 3D area. In addition, the partitioning scheme guarantees that a memory is never more than one tier away from the processing elements that are connected to it. This means the memory is also on the
Figure 5.10: A side view of the MIT Lincoln Labs’ process with the silicon-thru vias and tier orientation shown.

same tier as the controller it is connected to through the memory’s address lines. On the middle tier we have thirty-two memories and one controller to place. To accomplish this, we use an $11 \times 3$ grid. We place the controller in the center location of the grid in the middle tier. For the remaining memories we use a Python constraints package to generate an optimal memory placement by minimizing the distance from a given memory to the two processing elements that use it. The resulting floorplan is shown in Figure 5.13. There are a total of 8280 thru-silicon vias in the system. A total of 4128 vias connect the logic on tier A to the memories on tier B, another 4128 connect the logic on tier C to the memories on tier B. The remaining 24 TSVs connect the controller to the processing elements.

The next step in the design flow is synthesis, which uses a standard cell library based on the IIT-SoC library from the Illinois Institute of Technology. In this step, each tier is synthesized separately in Synopsys Design Compiler. After synthesis, we perform static timing analysis and
add additional pipeline stages to the processing elements until adding another pipeline stage does not result in any overall speed increase. The optimal pre-place and route pipeline depth for the system was discovered to be five stages for this manufacturing process and standard cells, yielding a maximum operation frequency of 196 MHz (without parasitics).

After synthesis, we perform place and route. This stage deviates the most from a conventional 2D flow. In order to successfully complete place and route, the global information about the placement of the memories and the thru-silicon vias is required. Using standard string and file manipulation functions built into the TCL interpreter in Encounter, the thru-via and pin locations can easily be extracted by parsing the DEF files (which can easily generated from Virtuoso) of the custom memories designs. Using the information from the DEF file, routing and placement is blocked over the areas of the memories and inter-tier silicon-thru via location. Normal placement is then performed, followed by clock tree synthesis. Due to the fact that the process only allows three metal layers per tier, the clock tree is not routed before regular
routing, as is common for processes with a large number of metal layers. Instead, the clock tree is routed along with other signals. This causes slightly more clock skew than would have occurred if a greater number of metal layers had been available. After clock tree synthesis, the “preassignPin” command is then used to place virtual input/output pins directly on top of the thru-vias on the edge of the memories. Encounter then performs routing as normal, connecting the standard cells, clock tree and virtual pins (effectively performing 3D routing). After place and route, the design along with the information on interconnect parasitics is imported into PrimeTime and post-place and route timing analysis is performed. In this step it is important to make sure that each tier has no setup or hold violations. It is also important to make sure that signals that travel between tiers also have no setup or hold violations. This step is greatly simplified due to the fact there are very few logic-to-logic vias (24) and the remaining signals are either data pins to the SRAMs or address pins to the twiddle factor ROMs.

Finally, all of the tiers are imported separately into Virtuoso. In Virtuoso the three tiers and the full-custom memories are combined. For the 24 signals that connect the controller to processing elements, the through-silicon vias are placed by hand. These TSVs were placed by hand because for such a small number it was quicker to place them by hand rather than write a script to automate the placement. Placing of these TSVs could easily be scripted using Skill code in Virtuoso as the location of all the through-silicon vias are known. Furthermore, scripting this process is necessary for other 3D designs that contain a greater number of logic to logic through-silicon vias. Next, the power and ground rings of the three tiers are combined into 3D meshes by placing thru-silicon vias all along the perimeter. Due to the fact that Encounter routed over the power and ground rings in a some areas, thru-silicon vias could not be placed in those areas. A total of 4554 power and ground vias fit between tiers A and B and 4800 vias between tier B and C. The final step in the design flow was to place the input and output pads and perform a final DRC and LVS. Figure 5.12 shows the three tiers stacked, along with the thru-silicon vias.
Figure 5.12: The 3D SAR FFT processor with thru silicon vias drawn in.

Figure 5.13: The 3D floorplan.
5.4 Thermal Analysis

As circuits move to smaller technology nodes, the power density increases, which leads to an increase in the junction temperature of transistors. Additionally, smaller technology nodes use porous low-k dielectrics in an effort to overcome signal integrity issues. These dielectrics are designed to decrease the parasitic capacitance between wires, but have a negative impact on the thermal performance of the circuit. This results in a longer thermal path, which makes it more challenging to remove heat from the active devices. This potentially increases thermal density and creates a larger thermal gradient in the chip. Increased heat density is undesirable because it reduces carrier mobility and the threshold voltage of the transistors on the chip. Furthermore, higher thermal density increases sub-threshold leakage and increases the rate of failure of wiring due to electromigration. High thermal gradients have also been known to cause unexpected logical failure [7].

In 3D integrated circuits heat density is even more of a problem than in 2D circuits because heat dissipating devices are now stacked directly on top of each other, leading to a higher heat density than in a comparable 2D chip. 3D integration also moves the majority of active devices further away from the heatsink. This also results in a degraded thermal path that can potentially increase both the temperature and thermal gradient in the chip.

5.4.1 Reducing Thermal Bottlenecks

There are four major approaches that can be taken to reduce thermal bottlenecks. The first approach is to use packaging with a lower thermal resistance or a larger heat sink on the package. Although this approach works well, it significantly increases the production cost of the system. The second approach is to use dielectrics that have a better thermal conductance. However, this can cause signal integrity problems. The third approach is to take thermal considerations into account in the floorplanning stage[42]. This can be done by placing the blocks that generate the most heat on the tier closest to the heatsink or in the vicinity of blocks that generate less heat. The fourth approach is to use thermal vias[22]. Thermal vias lower the effective-thermal
resistance of the chip, which helps to remove heat. However, as thermal vias block routing in all metal layers they must be placed in areas where they are the most effective while minimally impacting the routability of the area.

5.4.2 Thermal Simulation

To assess the thermal effects of 3D integration for our implementation we perform a thermal analysis of the system using FireBolt from Gradient Design Automation in conjunction with custom scripts to extract the power values\cite{46, 47}. The thermal analysis is accomplished in the following manner. First, we gather the information on the power consumption of the system. This is done differently for the custom designed memories and the standard cell parts of the design. For the custom designed memories (SRAMs and ROMs) we obtain the power consumption values from SPICE simulations. These results are then averaged over the entire memory. The process is not as straight forward for the standard cell parts of the design. For the standard cell parts of the design we obtain the power consumption information in the following manner. First, the switching activity information is recorded from a Verilog simulation into a SAIF file. The interconnect parasitics of the place and routed design are then extracted to a SPEF file using Encounter. The SAIF and SPEF files are then brought into PrimeTime, which calculates the power consumption of each standard cell. The cell power consumption value is then evenly divided over each transistor in the standard cell. Having gathered power consumption information for both parts of the chip, we annotate the layout to include the power dissipated in the given region. The layout that we annotate is the final one that used for fabrication. It contains through-silicon-vias, the substrate, interconnect wire metals, vias and the metal fill layers for every metal.

FireBolt is then run using the annotated layout as an input to simulate the temperature of all three tiers of the implementation. Additionally, an alternative version of the design that flips the memory tier with the bottom tier is also simulated. To complete the thermal simulation FireBolt uses an element size between 7000 nm to 80 nm, which it adjusts dynamically until the
simulation resolves temperatures to 0.1°C. The simulation took 90 hours and required slightly less than 64 GB of memory to run. It is important note that the temperature a given transistor is operating at affects its power consumption, which in turn effects its temperature. For digital logic circuits this change in temperature is minimal unless the majority of the power in the circuit is consumed due to leakage. As our design is implemented in a 180 nm SOI process using transistors with a relatively high threshold voltage effectively a small portion of the total power is consumed by leakage and as result it is acceptable to ignore the temperature on the power consumption of a given transistor. The simulation features are summarized in Table 5.6.

Table 5.6: Thermal simulation results.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Element Size (nm)</td>
<td>80</td>
</tr>
<tr>
<td>Maximum Element Size (nm)</td>
<td>7000</td>
</tr>
<tr>
<td>Approximate Runtime (hours)</td>
<td>90</td>
</tr>
<tr>
<td>Approximate Memory Usage (GB)</td>
<td>&lt;64</td>
</tr>
<tr>
<td>Maximum Temperature Rise (°C)</td>
<td>24.7341</td>
</tr>
</tbody>
</table>

The thermal simulation assumes that the handle silicon of tier A is attached to an ideal heat sink, and uses 27°C for the boundary condition. The bottom tier is unique from a manufacturing perspective because unlike the other two tiers the handle silicon has not been thinned. In addition to being directly connected to the heat sink, the handle silicon has a high thermal conductivity (as compared to the interlayer dielectric), and serves as a heat spreader. As a result the bottom tier has a significantly better thermal conduction path than the upper tiers, which can be seen in the results. The temperature profiles of the active regions for all three tiers are shown in Figure 5.14 for the regular and alternative design.
Figure 5.14: The temperature profile for all three tiers.
5.4.3 Thermal Discussion

In the regular design, the top and bottom tiers are almost identical yet their thermal profile is significantly different. This is due to the bottom tier’s close proximity to the heatsink. A similar effect can be observed in the alternative design where moving the memory tier closer to the heatsink cools the memory tier significantly. This illustrates a very important point: instead of balancing the power consumption between the tiers, it is more important to place the big power consumers on the tier next to the heat sink. Looking at the thermal profiles of all the tiers in the regular design, several of the biggest hot spots occur on tier B in the four rows between the SRAM memories. Upon further inspection it was discovered that the hot spots are caused by clusters of clock buffers. Figure 5.16 shows a closeup of the clock buffer hot spots on tier B.

In terms of the four general approaches to reducing thermal bottle necks, the two that are the most applicable to this design are inserting thermal vias and thermally aware floorplanning. With the exception of the 9354 power and ground TSVs that were inserted on the periphery of the design no additional thermal vias were inserted into the design because the design was already severely routing congested as a result of the memory division scheme. The power and ground vias do not affect the routing congestion because they are located on the periphery of the die. In terms of floorplanning, moving the memory tier closer to the heatsink helps to reduce hotspots. Although this reduction can be seen in Figure 5.14 a better way of visualizing it is by looking at a histogram of the junction temperatures of the all transistors in both designs, which is shown in Figure 5.15.

5.5 Results

We present the results in three parts in the following sections. In Section 5.5.1 we present the memory power advantage of the memory division by itself. Section 5.5.2 compares the
Figure 5.15: A histogram of junction temperatures for the regular and the alternative designs.

Figure 5.16: A closeup of hot spots caused by the clock buffers on the middle tier (tier B).
3D implementation of the FFT engine to a 2D equivalent implemented in the same process technology. It is hard to compare to the FFT processor to other architectures because the processor uses floating-point arithmetic and is built in an experimental 3D process using standard cell library that only has a small number of standard cells. In order to get a fair comparison, we re-implement the architecture using fixed point arithmetic in a 180 nm commercial process with a large standard cell set. Section 5.5.3 contains this comparison. Finally, Section 5.5.4 summarizes the results.

5.5.1 Memory Division Power Savings

In this section we analyze the energy benefits of the 32-way memory division on the implementation of the 1024 floating-point FFT architecture. Since we do not have memory models for the 3D process, we perform this analysis by using Cacti 4.1[75] at the 180 nm technology node, for both the divided and the undivided memory. By dividing the processing memory up, we can reduce the energy per read by 60.8% (from 68.205 to 26.718 pJ) and the energy per write by 57.6% (from 14.48 to 6.142 pJ). Furthermore, we can increase the bandwidth by 854.9% (from 13.4 to 128.4 GBps). Unfortunately, the number of memory pins connecting the memory to the logic increases by 1414.7% (from 150 to 2272) along with the distance they have to span. In addition, the total area required for the processing memory increases by 67.6%, which may seem like a big increase but is only a 16.8% increase in the context of the whole system. An overview of the differences between the undivided and divided designs are shown in Table 5.7.

5.5.2 3D Improvements Over 2D

In order to assess the energy benefits of the 3D circuit over its 2D counterpart, we implement the design in 2D. Additionally, to ensure a fair comparison between the two circuits, the design is not resynthesized, instead the same synthesis result is used. For the comparison we use an identical floorplan. This floorplan is essentially the floorplan of tier B expanded with the
Table 5.7: The design metrics of the undivided and the divided memory arrangements.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Undivided</th>
<th>Divided</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth ((GBps))</td>
<td>13.4</td>
<td>128.4</td>
<td>+854.9%</td>
</tr>
<tr>
<td>Energy Per Write ((pJ))</td>
<td>14.48</td>
<td>6.142</td>
<td>-57.6%</td>
</tr>
<tr>
<td>Energy Per Read ((pJ))</td>
<td>68.205</td>
<td>26.718</td>
<td>-60.8%</td>
</tr>
<tr>
<td>Average Access ((pJ))</td>
<td>41.3425</td>
<td>16.43</td>
<td>-59.2%</td>
</tr>
<tr>
<td>Memory Pins (#)</td>
<td>150</td>
<td>2272</td>
<td>+1414.7%</td>
</tr>
<tr>
<td>Memory Area ((mm^2))</td>
<td>1.616</td>
<td>4.990</td>
<td>+67.6%</td>
</tr>
<tr>
<td>Total Area ((mm^2))</td>
<td>23.400</td>
<td>3.373</td>
<td>+16.8%</td>
</tr>
</tbody>
</table>

Figure 5.17: The 2D floorplan for comparison.
ROMs placed in similar locations to the 3D version, shown in Figure 5.17. Due to increased congestion, the 2D design does not route successfully with the same area as its 3D counterpart (4.8 × 4.8 mm). To remedy this, the area used for place and route is grown until the design routes without any design rule violations. Compared to the 3D version, the total area expands significantly from 3 × 2.6 × 3 mm for the 3D circuit to 5.6 × 5.6 mm for its 2D counterpart. This is 25.3% increase in total area. To compare core placement area, we exclude the power and ground rings from the total area (0.1 mm on every side) and the comparison becomes 3 × 2.8 × 2.4 mm versus 5.4 × 5.4 mm. The area discrepancy between the total area and the core area illustrates an interesting point. Given the same total area and same power and ground ring width, a 3D design will devote more area to the power and ground rings. The next metric examined was total and average interconnect length. We extract the information directly from Encounter for all the nets, combining the information from all the different tiers. As expected, the average wire length decreased drastically from 836.0 µm down to 392.9 µm, which is a 53.0% decrease. Similarly, the total wire length decreased from 19.107 m to 8.238 m, which is 53% decrease. A histogram of the wire length distribution is shown in Figure 5.18.

![Histogram of wire lengths of the SAR FFT processor for both the 2D and 3D versions](image)

Figure 5.18: Histogram of wire lengths of the SAR FFT processor for both the 2D and 3D versions (bin size = 250µm).
In order to gather the speed and power metrics of the design, we have to extract the parasitics and characterize the switching activity of the design. This is done using Encounter and the results are exported to a SPEF file. The switching activity is generated by simulating a testbench in Mentor Graphics Modelsim and exporting the resulting activity of the test bench to a SAIF file. Both files were then read into Synopsys PrimeTime. In PrimeTime the fastest clock period that did not cause any setup violations was determined. Overall, the 3D design simulated correctly at 79.4 MHz (12.6ns), whereas the 2D design simulated correctly at 63.7 MHz (15.7ns). This is a 24.6% increase in maximum operating frequency and a 19.7% improvement in clock speed. As the maximum operating frequency may seem a bit slow for the given technology node, it is important to keep two points in mind. First, as stated earlier the implementation process is an experimental 3D process that is not as well characterized as a commercial one. Second, the standard cell library is relatively small and does not have the larger combined adder cells that many commercial libraries do have. For the 3D design the power dissipation is determined for both the maximum operating frequency of the 2D and the 3D design, while for the 2D design the power dissipation is only determined at the maximum 2D operating frequency. At an operating frequency of 79.4 MHz the 3D design dissipates 409.2 mW. Operating at 63.7 MHz the 3D design dissipates 324.9 mW and the 2D design dissipates 340.0 mW. This is a 4.4% improvement. Using the power numbers of both circuits operating at maximum frequency, we compute the energy (excluding memory accesses) required per 1024-point FFT. The energy required for completing the FFT using 3D integration is 3.366 $\mu$J as opposed to 3.552 $\mu$J without 3D integration. This is a 5.2% improvement over the 2D version. The results are summarized in Table 5.8.

5.5.3 Comparison to Other FFTs

Two issues make it difficult to compare the 3D implementation to the state of art. First, the design is implemented in an experimental 3D process that is not directly comparable to a mature commercial process with a large set of carefully characterized standard cells. Second,
Table 5.8: Improvements from using 3D integration in the implementation of the FFT.

<table>
<thead>
<tr>
<th>Metric</th>
<th>2D</th>
<th>3D</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Area (mm$^2$)</td>
<td>31.36</td>
<td>23.40</td>
<td>-25.3%</td>
</tr>
<tr>
<td>Mean Net Length ($\mu$m)</td>
<td>836.0</td>
<td>392.9</td>
<td>-53.0%</td>
</tr>
<tr>
<td>Total Wire Length (m)</td>
<td>19.107</td>
<td>8.238</td>
<td>-56.9%</td>
</tr>
<tr>
<td>Max Speed (MHz)</td>
<td>63.7</td>
<td>79.4</td>
<td>+24.6%</td>
</tr>
<tr>
<td>Critical Path (ns)</td>
<td>15.7</td>
<td>12.6</td>
<td>-19.7%</td>
</tr>
<tr>
<td>Logic Power (mW)</td>
<td>340.0</td>
<td>324.9</td>
<td>-4.4%</td>
</tr>
<tr>
<td>FFT Logic Energy ($\mu$J)</td>
<td>3.552</td>
<td>3.366</td>
<td>-5.2%</td>
</tr>
</tbody>
</table>

The implementation is designed to be used in synthetic aperture radar imaging processor, which requires the use of floating-point arithmetic with a wide word size and the ability to perform FFTs of varying power of two sizes. This contrasts with most implementations in the literature which target specific FFT sizes and typically use fixed-point arithmetic with a smaller word size. Regardless, it is important to estimate how the divided memory architecture compares to other architectures in the literature to illustrate the strengths and weaknesses of the scheme. In order to do this we modify the processing elements to use fixed-point arithmetic, a word-length of 11 and re-synthesize them in a commercial 180 nm process. With one flip-flop of pipelining in the butterfly, the fastest clock speed the design can operate at is 1.91 ns, corresponding to 534 MHz. As a 8192-point radix-2 FFT requires 6656 cycles to complete ($N/(2(PEs)) \cdot (\log_2(N))$) At this operating speed it takes 12.71 $\mu$s to complete the 8192-point transform. At this speed the logic power required to realize the 8192 FFT is 7.91 $\mu$J. Furthermore, using Cacti 4.1[75] at the 180 nm technology node, we estimate that completing the 106496 ($N \cdot (\log_2(N))$ memory accesses required for the 8192 FFT would consume 3.00$\mu$J of energy. An estimation of the divided memory performance is compared to the results measured for a 8192 point DVB-T FFT processor[43] is shown in Table 5.9.
Table 5.9: Comparison between FFTs

<table>
<thead>
<tr>
<th>Metric</th>
<th>Sung et al. 2010</th>
<th>Lin et al. 2004</th>
<th>This Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word-length</td>
<td>16</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>Radix</td>
<td>2/8</td>
<td>2/4/8</td>
<td>2</td>
</tr>
<tr>
<td>Process (nm)</td>
<td>180</td>
<td>180</td>
<td>180</td>
</tr>
<tr>
<td>Voltage (V)</td>
<td>1.8</td>
<td>1.8</td>
<td>1.8</td>
</tr>
<tr>
<td>Clock Frequency (MHz)</td>
<td>200</td>
<td>20</td>
<td>534</td>
</tr>
<tr>
<td>8K FFT Run Time (µs)</td>
<td>395</td>
<td>717</td>
<td>12.71</td>
</tr>
<tr>
<td>Energy per 8K FFT (µJ)</td>
<td>46.2</td>
<td>18.1</td>
<td>10.91</td>
</tr>
</tbody>
</table>

5.5.4 Summary

After assessing the energy per access reduction from the memory division and the logic power reduction due to 3D integration separately, it is important to note the overall effects of both optimizations on the complete system. The 59.2% reduction in energy per access translates to a 22.8% (from 5.476 µJ to 4.227 µm) reduction in system power and the 5.2% reduction in logic energy translates to a 3.4% (from 5.662 µJ to 5.476 µm) reduction in system power. Finally, for a system that has neither optimization the overall energy per 1024-point FFT is reduced by 25.3% (from 5.662 µm down to 4.227 µm). Table 5.10 contains a summary of these reductions. We have shown that the memory division compares favorably to other FFTs in literature in terms of energy consumption (10.91 versus 18.1 µJ). In addition, we have shown that we achieve a significant reduction in total wire length 56.9%. This indicates that the divided memory FFT architecture is a good candidate for 3D integration.

5.6 Test Setup and Measurement

This section describes the test and measurement setup used to test the FFT processor chip fabricated using MIT Lincoln Lab’s 3D integration technology, which is shown in Figure 5.19. The chip has two inputs and six outputs, which are as follows:
Table 5.10: Energy per 1024 point FFT for comparison between undivided memory, divided memory, 2D and 3D versions.

<table>
<thead>
<tr>
<th>Energy</th>
<th>Undivided</th>
<th>Divided</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D Logic Energy (µJ)</td>
<td>3.552</td>
<td>3.552</td>
</tr>
<tr>
<td>2D Memory Energy (µJ)</td>
<td>2.110</td>
<td>0.861</td>
</tr>
<tr>
<td>Total</td>
<td>5.662</td>
<td>4.413</td>
</tr>
<tr>
<td>3D Logic Energy (µJ)</td>
<td>3.366</td>
<td>3.366</td>
</tr>
<tr>
<td>3D Memory Energy (µJ)</td>
<td>2.110</td>
<td>0.861</td>
</tr>
<tr>
<td>Total</td>
<td>5.476</td>
<td>4.227</td>
</tr>
</tbody>
</table>

→ clockoffchip The clock input to the chip.

← twidbitout This signal outputs one bit of the last twiddle factor that was read by PE0.

← srambitout This signal outputs one bit of the last data that was read from memory by PE0.

← resultbitout This signal outputs one bit of the last PE0 calculation result.

→ resetfromoffchip This signal is the reset for the controller’s finite state machine.

← pe0realsign This signal specifies the sign of the first input to PE0, which is generated by the controller’s finite state machine.

← pe0loweriseven This signal specifies the control bit that determines, which input multiplexer are used by PE0. This signal is also generated by the controller’s finite state machine.

← pe0fliprealandimag This signal specifies whether the PE0 should flip the real and imaginary components of the twiddle factor and is generated by the controller’s state machine.

In order to speed up the testing procedure a simple sign-of-life test was first conducted to see if the chip worked well enough to warrant building a customized PCB. The objective
of the sign-of-life test was to test for basic functionality of the main controller’s finite state machine. To complete this sign-of-life test we utilized a pre-existing board designed and built by Ravi Jenkal and Samson Melamed. This PCB board was originally used to test a 90 nm sphere decoder design[31] and is shown in Figure 5.20. The central component of this board is a 40-pin dual in-line package (DIP) socket that allows the chip that is under test to be easily swapped in and out.

Before the chip can be put in the socket it needs to be packaged into a 40-pin DIP. This was accomplished by gluing the chip to an empty package and then using a Westbond 7476E wedge-wedge wire bonder to bond the chip pads to the pads on the package. The bonding diagram that was used is shown in Figure 5.21.

After wire-bonding the clock and the reset input signals were connected to a Tektronix HFS 9009 digital pattern generator, the output of signal was connected to a Tektronix TDS684B oscilloscope and the power and ground to a PS2520Gs programmable power supply, as shown in Figures 5.22 and 5.23. After the power supply was connected the chip starting drawing 139 mA
Figure 5.20: Sphere Decoder test board used for sign-of-life testing.

Figure 5.21: The bonding diagram used for wire-bonding the chip to the package.
Using the pattern generator the clock was set to toggle at 0.1 MHz and the reset signal was asserted for one cycle. The expected output from the “pe0LowerIsEven” was a clock like signal that toggled every cycle as the state of the finite state machine changed. Unfortunately, this was not the case. The “pe0LowerIsEven” signal became active once after the reset but did not toggle again. This means that the chip showed signs-of-life but did not function correctly. Although, we can not assert the cause for chip working incorrectly with certainty, we believe
the cause may be due to plasma induced gate oxide damage or the antenna effect or other yield issues. The way this occurs is that during fabrication metal wires can act like antennas and collect charge. The accumulated charge is then violently discharged through the gate oxide destroying it. Typically antenna rules are supplied by the foundry. However, since the MIT process is an experimental process, antenna effects have not been characterized for the process and as such are not enforced by the DRC deck.

5.7 Conclusion

We have demonstrated a novel memory division scheme that lowers the energy required for a 1024 point FFT from 5.476 $\mu$J down to 4.227 $\mu$J. This memory division scheme compares favorably to other approaches in the literature in terms of energy consumption (10.91 versus 18.1 $\mu$J per 8192 point FFT). The resulting divided memory architecture is a good candidate for 3D integration as it sees a 56.9% reduction in wire-length.

The thermal profile of the 3D implementation illustrates that there is a significant difference between the thermal profile of the tier next to the heat sink and the other tiers (even when the layouts of the tiers are identical), which emphasizes the importance of taking heat dissipation into account in 3D floorplanning.

The use of memory division and 3D integration complement each other because the memory division scheme trades interconnect complexity for an improvement in energy per memory access, whereas 3D integration mitigates the impact of the increased interconnect complexity. Given these advantages, we believe 3D integration has the potential to address interconnect problems for architectures such as the divided memory architecture presented.
Chapter 6

Case Study 2: 3D Memory and Logic-on-Logic

This chapter presents the second case study, which explores the benefits derived from 3D integrated memory and logic-on-logic 3D integration. To accomplish this we build a SAR processor that uses Tesselron’s 3D stacking technology and leverages: 3D DRAM-on-logic, 3D logic-on-logic and a reconfigurable processing element. This processor implements the Range Doppler Algorithm[32] described in Chapter 2 at a imaging resolution of 15 cm. At this resolution the processing grid consists of 4096 by 2048 64-bit complex floating-point numbers (32 imaginary and 32 real bits). The grid must be fully processed in slightly less than a second. Additionally, the CAD aspects required to realize the processor and do a 3D place and route for logic-on-logic 3D integration are explored.

The chapter is organized in the following manner. Section 6.1 describes the architecture and operation of the SAR processor. Section 6.2 describes the 3D manufacturing process. Section 6.3 looks at how the EDA tool flow is used to implement the processor and Section 6.4 explains the 3D power delivery approach that was used. Finally, Section 6.5 presents the results and Section 6.6 concludes the chapter.
6.1 Architecture And Operation

This section describes the architecture and operation of the system. The first half of the section describes the six parts of the system and the second half describes the operation of the system and how the radar processing is accomplished. Overall, the architecture of the system consists of six different parts listed below and shown in Figure 6.1.

- 3D Integrated DRAM System Memory
- Memory Controller
- 2 Register Files (Operand and Result)
- Instruction Decoder
- 8 Twiddle Factor Read Only Memories (ROMS)
- 8 Processing Elements (PEs)

6.1.1 Main Memory DRAM

The main memory of the system is a DRAM that stores the processing grid of the system along with the filter kernels required to complete the radar processing. The memory is built out of three 3D integrated tiers. Two of these tiers consist of 1 Gbit DRAM bit cells, which combine to give the system a total of 2 Gbits of memory. The final layer consists of control circuitry, sense amplifier and error correction circuitry. The error correction circuitry transparently performs error correction to improve robustness and preserve the integrity of the stored data. Overall, the DRAM has eight independent 128-bit wide ports. Each port consists 8192 rows per layer with 128 columns in each row. The memory can be operated in bursts of either four or eight words. If all eight ports are given the same commands they can be utilized as one wide memory with a word size of 1024 bits. In this configuration a total of 16 data points can be stored per memory word, as each pixel occupies 64 bits.
6.1.2 Memory Controller

The memory controller is the interface between the register files and the DRAM. The controller takes care of opening and closing DRAM rows and making sure that each memory bank is refreshed often enough to avoid data loss. The memory controller is relatively simple in the sense that it does not perform any data caching or queuing of memory commands. The reason for this is that the data access patterns for large FFTs typically do benefit much from being cached and the extra cache consumes power and area.

6.1.3 Register Files

The system consists of two separate register files, the operand and result register files. Each of these register files consists of two sets of sixteen 64-bits wide registers, which are referred to as “R” and “P” registers respectively. The register files act as intermediates between the DRAM and the PEs. The operand register file reads from the DRAM and stores the result until it is needed as an input for the PE to use. Conversely, the result register file stores...
the computation result from the PE and writes it to the DRAM at the appropriate time. Conventionally, register files are implemented in a similar fashion to a multiported SRAM that includes peripheral circuitry like sense amplifiers and address decoders (See Section 3.1 for more details). This is not the case for these register files as the access to them is not random. Instead each PE only reads and writes to specific registers. Knowing which PE reads and writes to which register allows optimizing the implementation to build a smaller register file then would be required for random access. So instead of the conventional register file implementation, we store the data using flip-flops and then use multiplexers to connect the given flip-flop to the correct PE. This implementation approach has the advantage of allowing extra functionality, such as shifting to be implemented in the register file. A block diagram of the operand and result register files is shown in Figure 6.2 and Figure 6.3 respectively.

Figure 6.2: The operand register file.
6.1.4 Instruction Decoder

The system is programmed using several different instructions. The instruction decoder interprets these instructions and depending on the instruction sends the correct control signals to the other components of the system. The instruction set is not a general one. Instead it is tailored specifically towards the radar processing algorithm by providing a specific instructions for the DSP operation and removing unneeded general instructions such as branching. This specialization provides improved performance and energy efficiency over a general purpose instruction set. Overall, the instruction set has eight instructions, two loads, two stores, two immediate instructions, one shift instruction and one DSP instruction. Each instruction is encoded using 42 bits and the encoding for the instructions is shown in Figure 6.4.
Figure 6.4: Instruction set implemented in the instruction decoder.
6.1.5 Twiddle Factor ROMs

Computing an N-point FFT requires using $N/2$ FFT twiddle factors of the same precision. Before the twiddle factors can be used they must be stored somewhere. This can be done by either storing them in the main DRAM or in a dedicated ROM. The different approaches have different advantages. The advantage of storing the twiddle factors in the main DRAM is that this does not use up any additional chip area. The disadvantages is that the twiddle factors must be loaded into the memory at startup time and then retrieved from main memory every time a calculation requires them. Storing the twiddle factors in memory reduces the memory bandwidth available for loading and storing regular operands. As one of the main design constraints for the system is memory bandwidth, the twiddle factors are stored in a ROM to conserve the bandwidth. The ROM is generated using Artisan’s ROM compiler, which gives three options for the ROM that depend on the multiplexer width. Each of the options provides a different aspect ratios and total area, which are shown in Table 6.1.

<table>
<thead>
<tr>
<th>Choices</th>
<th>Width ($mm$)</th>
<th>Length ($mm$)</th>
<th>Area ($mm^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 Wide Multiplexer</td>
<td>0.470</td>
<td>0.270</td>
<td>0.127</td>
</tr>
<tr>
<td>16 Wide Multiplexer</td>
<td>0.732</td>
<td>0.185</td>
<td>0.136</td>
</tr>
<tr>
<td>32 Wide Multiplexer</td>
<td>1.329</td>
<td>0.145</td>
<td>0.194</td>
</tr>
</tbody>
</table>

Interestingly, the design choice is different then it would have been in a 2D implementation. In a 2D implementation the ROM with an 8 wide multiplexer would have been chosen because it requires the least area. However, since both the length and width of the 8 wide option exceed 250 micron it is impossible to use the ROM in Tezzaron’s 3D design process because the process has a requirement that no TSV can spaced further than 250 microns from another TSV (see Section 6.2). This spacing is impossible to achieve for the 8 wide option as the TSVs
cannot go through the ROM circuitry and as result the system is implemented using the 16 wide multiplexer option. The layout of the ROM is shown in Figure 6.5. Additionally, to minimize the storage of the FFT twiddle factors in ROM two optimizations are used. First, trigonometric properties are used to reduce the number of twiddle factors stored from \(\frac{N}{2}\) to \(\frac{N}{8} + 1\). This effectively reduces the number of twiddle factors stored from 2048 down to 513, which is enough for a 4096 point FFT.

![Figure 6.5: The layout of the Twiddle Factor ROM.](image)

6.1.6 Processing Elements

The processing elements (PEs) are the core of the system consisting of 149,936 transistors each. All the computation required by the radar processing algorithm is completed by the eight PEs in the system. In order to achieve a high energy efficiency in terms of mWatts per GFlop, no pipelining is used. This eliminates the power that would be consumed by flip-flops. However, the downside to having very little pipelining is that it reduces the maximum operating frequency as the critical path is longer. The PEs complete the computation using a few variations of four distinct DSP operations: FFT, IFFT, complex multiplication and FIR filtering. These DSP operations utilize the four multipliers, three adders and three subtraction units in each PE to complete the given operation in a different manner depending on the operation. Due
to the fact that in the algorithm not all the operations need to be done at the same time or in parallel it is possible to use resource sharing to reduce the area of each PE. We take advantage of this by sharing the same multipliers and adders to implement the different steps of the algorithms inside the PE. We do this by adding multiplexers to control the data flow, as shown in Figure 6.6. The signals that direct the multiplexers are set by the instruction decoder and allow the reconfiguration of each PE in different ways to implement each step of the SAR algorithm. Overall, the PE can be set to seven different states, which are shown in Table 6.2. The data-flow for three of these states is explained in the following sections and shown in Figure 6.7, 6.8 and 6.9 respectively.

Figure 6.6: PE with reconfiguration multiplexers shown.

**FFT**

In this configuration each PE operates independently to implement a basic radix-2 butterfly. The configuration can execute any power-of-two FFT up to 4096 points (12 stages) as it is
Table 6.2: Controls for various operations

<table>
<thead>
<tr>
<th>Function</th>
<th>C2</th>
<th>C1</th>
<th>C0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFT Lower Is Even</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FFT Lower Is Even Twiddle Factors Flipped</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FFT Lower Is Odd Twiddle 1 is Real</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FFT Lower Is Odd Twiddle Factors Flipped</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Complex Multiplication</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Complex Multiplication Imag. Negated</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FIR Filtering Operation</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>No Operation</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 6.7: The reconfigurable PE configured for FIR the FFT butterfly.

Figure 6.8: The reconfigurable PE configured for FIR complex multiplication.
limited by the number of twiddle factors stored in the Twiddle Factor ROM. As the radix-2 butterflies are the building block of the radix-2 Cooley-Tukey FFT algorithm[11], the full FFT calculation can be done by completing each radix-2 operation in a given PE during a given time slot. An example of such scheduling for 64-point FFT is shown Table 6.3.

One important aspect to note about the radix-2 Cooley-Tukey algorithm is that the distance between the two memory addresses that go into a given PE is always $2^{stage}$. This means that the registers that are used as input to the radix-2 butterfly change as the stage increases. Table 6.4 shows which registers are used for what stage. Furthermore, after stage 3 the two memory locations are so far apart that they are no longer in the same memory word as the effective memory word size is 1024 (divided into sixteen 32-bit operands). This means an extra load must be executed at the beginning of every stage before any FFT calculation can be done. This is is shown as an “X” in the table. However, after this load has occurred the FFT can come complete eight radix-2 butterflies per cycle just like in stages 0-3.
Table 6.3: Input addresses for the eight PEs in a 64 point FFT.

<table>
<thead>
<tr>
<th>Stage</th>
<th>PE0</th>
<th>PE1</th>
<th>PE2</th>
<th>PE3</th>
<th>PE4</th>
<th>PE5</th>
<th>PE6</th>
<th>PE7</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>0</td>
<td>1</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>11</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>17</td>
<td>25</td>
<td>24</td>
<td>26</td>
<td>27</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>33</td>
<td>41</td>
<td>40</td>
<td>42</td>
<td>43</td>
<td>35</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>49</td>
<td>57</td>
<td>56</td>
<td>58</td>
<td>59</td>
<td>51</td>
<td>50</td>
</tr>
<tr>
<td>S1</td>
<td>3</td>
<td>7</td>
<td>10</td>
<td>14</td>
<td>9</td>
<td>11</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>23</td>
<td>26</td>
<td>30</td>
<td>25</td>
<td>27</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>39</td>
<td>42</td>
<td>46</td>
<td>41</td>
<td>43</td>
<td>32</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>55</td>
<td>58</td>
<td>62</td>
<td>57</td>
<td>59</td>
<td>48</td>
<td>50</td>
</tr>
<tr>
<td>S2</td>
<td>3</td>
<td>7</td>
<td>10</td>
<td>14</td>
<td>9</td>
<td>11</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>23</td>
<td>26</td>
<td>30</td>
<td>25</td>
<td>27</td>
<td>16</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>39</td>
<td>42</td>
<td>46</td>
<td>41</td>
<td>43</td>
<td>32</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>55</td>
<td>58</td>
<td>62</td>
<td>57</td>
<td>59</td>
<td>48</td>
<td>50</td>
</tr>
<tr>
<td>S3</td>
<td>0</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>24</td>
<td>25</td>
<td>17</td>
<td>26</td>
<td>18</td>
<td>19</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>40</td>
<td>41</td>
<td>33</td>
<td>42</td>
<td>34</td>
<td>35</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>56</td>
<td>57</td>
<td>49</td>
<td>58</td>
<td>50</td>
<td>51</td>
<td>59</td>
</tr>
<tr>
<td>S4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>16</td>
<td>17</td>
<td>1</td>
<td>18</td>
<td>2</td>
<td>3</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>8</td>
<td>9</td>
<td>25</td>
<td>10</td>
<td>26</td>
<td>27</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>48</td>
<td>49</td>
<td>33</td>
<td>50</td>
<td>34</td>
<td>35</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>56</td>
<td>40</td>
<td>41</td>
<td>57</td>
<td>42</td>
<td>58</td>
<td>59</td>
<td>43</td>
</tr>
<tr>
<td>S4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>32</td>
<td>33</td>
<td>1</td>
<td>34</td>
<td>2</td>
<td>3</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>8</td>
<td>9</td>
<td>41</td>
<td>10</td>
<td>42</td>
<td>43</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>48</td>
<td>49</td>
<td>17</td>
<td>50</td>
<td>18</td>
<td>19</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>56</td>
<td>24</td>
<td>25</td>
<td>57</td>
<td>26</td>
<td>58</td>
<td>59</td>
<td>27</td>
</tr>
</tbody>
</table>
Complex Multiplication

In this data flow configuration, which is shown in Figure 6.8 every PE operates independently to compute a single complex multiplication. This configuration always multiplies a value from a “P” register with one from an “R” register. Due to the flip-flop and multiplexer design of the register file (see Section 6.1.3) not every combination is possible and certain PEs must be used to multiply certain combinations. The system only allows the registers shown in Table 6.5 to be multiplied together using the PE listed in the table. Additionally, to support IFFT computation, the complex multiplication configuration supports doing a complex multiplication followed by flipping the sign bit on the imaginary component in one cycle.

IFFT

The IFFT is accomplished by first executing a regular FFT, followed by complex multiplication by \( \frac{1}{n} \) that flips the sign on the imaginary component. Just like the FFT and the complex multiplication configurations, every PE operates individually. As the system is designed with IFFTs in mind, each PE can do a complex multiplication and flip imaginary sign in one cycle.
Table 6.5: Which PE can multiply which P and R register for complex multiplication.

<table>
<thead>
<tr>
<th></th>
<th>0R</th>
<th>1R</th>
<th>2R</th>
<th>3R</th>
<th>4R</th>
<th>5R</th>
<th>6R</th>
<th>7R</th>
<th>8R</th>
<th>9R</th>
<th>10R</th>
<th>11R</th>
<th>12R</th>
<th>13R</th>
<th>14R</th>
<th>15R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0P</td>
<td>PE0</td>
<td>PE0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1P</td>
<td>PE1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2P</td>
<td></td>
<td>PE2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE2</td>
<td>PE2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3P</td>
<td></td>
<td>PE3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE4</td>
<td>PE4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE5</td>
<td>PE5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE6</td>
<td>PE6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE7</td>
<td>PE7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8P</td>
<td></td>
<td>PE0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE0</td>
<td>PE0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE1</td>
<td>PE1</td>
</tr>
<tr>
<td>10P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE2</td>
<td>PE2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11P</td>
<td></td>
<td>PE3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE4</td>
</tr>
<tr>
<td>13P</td>
<td></td>
<td>PE5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14P</td>
<td></td>
<td>PE6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PE7</td>
<td>PE7</td>
</tr>
</tbody>
</table>

FIR Filtering

Unlike all the other PE configurations, all the PEs work together for FIR filtering. In the FIR configuration each PE implements a fourth order FIR filter section. All the PEs are then chained together to realize a 32nd order FIR filter. This chain connection can be seen in Figure 6.1.

6.2 Manufacturing Process

This system is built in Tezzaron’s 3D integration process[55, 53, 66]. The Tezzaron 3D stack-up consists of the following tiers. First, two of the tiers are fabricated in Chartered Semiconductor’s 130 nm low power logic process (these are referred to as the top and bottom logic tiers respectively). The next two tiers contain the DRAM bit cells, which are fabricated in a DRAM process. The final tier contains the address decoders and sense amplifiers for the DRAM, also fabricated in Chartered Semiconductor’s 130 nm low power logic process. The advantage of
manufacturing the DRAM in two different processes is that it allows building the bit cell in a process that maximizes the capacitance and density of the storage capacitor and the peripheral logic in a different process that maximizes the speed of the sense amplifiers and address decoders simultaneously. This is not possible without 3D integration.

In the process, 3D assembly is accomplished in the following manner. First, the two logic tiers are wafer-stacked face-to-face using Tezzarons Copper-to-Copper thermocompression process. Because the tiers are stacked face-to-face, they need to mirrored along the y-axis before being stacked. The resulting wafer stack is then diced, flipped over and die-to-wafer bonded to a 3D wafer stack containing the three DRAM tiers to complete the five tier stack up. In the stack, tier-to-tier connectivity differs between tiers. A side view of the stack is shown in Figure 6.10. The two logic dies that are bonded face-to-face connect using 4.4 \( \mu m \) by 4.4 \( \mu m \) copper micro-bumps that are on a fixed via grid that has a pitch of 5.0 \( \mu m \). The interconnect parasitics of the copper micro-bumps is similar to a regular via, which is less than a through-silicon via. Furthermore, unlike a TSV the micro-bumps do not block routing in any metal layer. The connectivity between tiers other than the two logic tiers is accomplished using through-silicon vias known as a “Super-Contact” [55], which takes up 1.2 by 1.2 \( \mu m \) of area. To ensure uniform stress across the chip “Super-Contacts” must be placed no further 250 microns apart from each other. This requirement can cause problems for certain applications.

### 6.3 Tool Flow

All the tool flow can be conceptually divided into three parts, the 3D placement of the PE, 3D LVS and the rest of the flow. The first part is important because achieving a quality 3D placement is the key to getting a good reduction in energy consumption and improvement in performance. The 3D LVS must be explained separately, as 3D LVS is more complicated than regular 2D LVS in Tezzaron’s 3D process. Finally, the overall 3D flow must be explained as it differs from a normal 2D flow in a few key areas and is tricky to execute when using 2D
CAD tools. Overall, this section is divided into three subsections that covers each of the three parts. Section 6.3.1 covers the 3D placement and routing of the PE. Section 6.3.2 covers the 3D LVS and Section 6.3.3 details the rest of the tool flow that occurs after the 3D PE has been generated and describes it in detail emphasizing the aspects that differ from a 2D tool flow.

### 6.3.1 3D Placement

Several 3D placement tools[24, 13, 21, 10, 9], and approaches that involve memory-on-logic[69] or approaches that convert 2D placements to 3D placements [8] have have been proposed. In this design we use 2D tools to realize the 3D system. Since the interconnect parasitics draw a significant amount of power in the PEs it makes sense to try to reduce them using 3D integration. To realize this we execute the following tool flow, which is shown in Figure 6.11. First, the PEs are synthesized using Synopsys Design Compiler. After this a hypergraph representation of the synthesized netlist is generated. Hmetis[34] is then used to partition the hypergraph into two balanced halves that have a minimum number of edges between them. In the partitioning, the area balance of the two partitions is favored over minimizing the number of edges between

![Figure 6.10: A side view of the 3D manufacturing process.](image-url)
the two halves. This ensures a maximum footprint reduction. Furthermore, all cells that use the clock are forced to be in the bottom partition. This is done to simplify the generation of the clock tree by using a 2D clock tree generated by conventional 2D clock generation tools (Cadence Encounter Cts). After the PE has been partitioned, it is placed in 3D in the following manner. First, all the input and output pins are removed from the bottom partition, and the result is then 2D placed using Encounter. Removing the input and output pins prevents the tool from incorrectly assigning the pins as regular I/O pins, resulting in a placement that is not constrained by incorrectly assigned pins. This placement is then exported out of Encounter and into a custom program that assigns each inter-tier signal to a through-tier vias on the via grid.

The custom program that assigns through-tier vias is based on Lee's algorithm[40]. The algorithm works in the following manner. First, a grid is generated that has the same number of squares as the number of available through-tier vias for the placed circuit area. Each inter-tier signal is then placed in the grid square closest to its placement location, which results in some grid squares having multiple inter-tier signals. A shifting operation is then performed on every grid square that has more than one inter-tier signal, starting with the grid squares that have the highest number of inter-tier signals and proceeding downwards. The shifting operation works as follows: the shortest path from the grid square to a free grid square is found using Lee’s algorithm and the content of every grid square along that path is moved one towards the free square thus reducing the number of inter-tier signals in the target square by one. This shifting operation is performed until no square has more than one connection. The algorithm takes less than a second to assign the 362 inter-tier signals required by the PE on to a 79 by 79 via grid. The algorithm is shown below:
Figure 6.11: The design flow for 3D placement.
**Input:** Location of cells that connect to 3D vias

**Output:** The 3D via assignment

```plaintext
AssignEveryInterTierSignalToNearestGridSquare();

foreach Grid Square i j do
  if 3D vias assigned to i j > 1 then
    while 3D vias assigned to i j > 1 do
      k = ShortestPathToFreeGridSquare();
      foreach 3D Via on path k do
        Shift3DViaAlongPath();
      end
    end
  end
end
```

In Encounter the inter-tier signals for the bottom tier are now set using the “preassignPin” command to the location assigned by the custom program and cell placement is performed again this time followed by routing. Finally the top partition is brought into Encounter, where the inter-tier signal pins are set in a similar manner using the “preassignPin” and cell placement is performed on the tier. 2D clock tree synthesis is then performed followed by routing. Finally, the design along with information on the interconnect parasitics is imported into PrimeTime and post-place and route timing and power analysis is performed, the results of which are detailed in Section 6.5.

### 6.3.2 3D LVS

3D LVS is trickier to execute than 2D LVS for Tezzaron’s 3D process for two reasons. First, the foundry only supplies 2D LVS decks. Second, the foundry process design kit can only support up to 256 design layers. This number is not enough to accommodate all layers in both logic tiers. To get around these limitation some scripting was required. First, a script was developed
that generated a 3D LVS deck from a 2D deck. The script works by making a copy of every layer for each tier and assigning a unique layer number to every layer on the top tier by adding a specific offset to it. The script was written in a generic manner so that it is reusable and can be applied to any Calibre LVS deck. A second script was written that merges the two gds files from the top and bottom tiers and adds the same offset to the top tier. Finally, LVS was run using Calibre on the merged gds file with the generated 3D LVS rules to insure that the layout does indeed match the schematic netlist.

6.3.3 Overall Tool Flow

Given that we have generated, placed and routed a 3D integrated PE, we can use it to realize the overall system. First, in order to understand the overall tool flow it is important to realize that the only 3D integrated logic in the system is in the PE. The rest of the system is all in the bottom tier. This is illustrated by Figure 6.12.

Before any floorplanning is done, it is important to take into account the dummy TSV requirement of the Tezzaron process. This requirement states that any TSV can not be any further than 250 microns (ideally 100 microns) from another TSV. Additionally, dummy TSVs can only be placed in a location where they line-up with a specific back-metal pattern on the adjacent tier. In order to comply with these requirement, we do the following. First, we reserve several small squares that are 100 microns apart for dummy TSVs. On these squares we block placement and put a special dummy TSV standard cell. This cell contains an enlarged metal 1
area that is connected to ground with no doping underneath it. Due to the fact that the TSV has to line up with the back-metal pattern we cannot put the TSV directly in the dummy cell, because the two grids would not align as they have different grid sizes. The enlarged metal 1 area is designed so that it is large enough to be guaranteed to cover any possible overlap of the two grids. This guarantees that a TSV can be placed to meet both requirements regardless of how the two patterns fall on each other. The actual dummy TSV is placed using a script. This script takes the location of the dummy cells and back-metal pattern as input to generate a gds file that includes the dummy TSVs in the right location to satisfy both requirements. This gds file is then included in the final gds file. A picture of the dummy standard cell, the back-metal pattern and the TSV is shown in Figure 6.13.

Figure 6.13: The dummy TSV standard cell, with TSV and back-metal pattern.

The next step in the tool flow is floorplanning. In the floorplan we place the PEs on the outside edges of the chip. This is done to get a maximum continuous placement area in the
center for the rest of the system. Next to each PE we place its associated twiddle factor ROM. For both the PE and the ROM alignment is critical. The PE’s must be aligned correctly so that they fall on the 3D microbump via grid that they were originally placed and routed on. The twiddle factor ROMs must be placed in way that does not violate the TSV distance restriction stated in the previous paragraph. Again, this restriction states that no TSVs can be any further than 250 microns from another TSV. Since the ROM is generated using a memory compiler that has no concept of dummy TSVs there is no space on the ROM to place any dummy TSVs and as a result we take two actions to ensure that the restriction is satisfied. First, the ROM is designed so that at least one of its dimensions is less than 250 microns. Second, each ROM is placed centered between two rows of dummy TSV cells.

After we have placed the ROMs, we place both the north and south memory connectors. The memory connectors are very simple circuits that only contain the TSVs to connect to the memory and protection diodes to guard against harmful ESD events. The memory connectors must be placed at the correct location so that the TSVs can successfully connect to the adjacent tier. The north memory connector is shown in Figure 6.14.

![Figure 6.14: The north memory connector.](image)

The off-chip IO circuits are placed next. Before we explain those circuits in detail, it is important to explain how the IO works in the system. In the system all the IO on the chip goes to the first layer of the memory stack via a TSV. There it connects to a back-metal trace.
that routes the signal to a bond pad on the periphery of the first layer of the memory stack. In this sense, the off-chip IO circuits on the logic tier are not pads in the traditional sense because they do not include bond pads. But like pads these circuits include the same circuitry typically found inside a pad, which include: ESD protection, an output driver, a input buffer and so forth. This minor difference necessitates additional work because the foundry supplied libraries can not be used as they are normally in the 2D case. Instead they must be modified to account for the difference. Luckily, the width of one of the pad libraries is exactly twice that of the pitch of the off-chip back-metal pattern. This allows the foundry pads to be adopted to fit with the back-metal pattern quite easily. As the pad library from foundry contains over 350 pads is too time consuming to modify each pad by hand. Instead, the adaptation is done using a script that the replaces the bond pad with a TSV connector in the gds file. A close up of an IO circuit that has been adapted to use a TSV instead of a bond pad is shown in Figure 6.15.

![Figure 6.15: An IO circuit that has been adapted to use a TSV instead of a bond pad.](image-url)
Finally, the rest of the system in the bottom logic tier is placed using Encounter, resulting in the floorplan shown in Figure 6.16. Routing is then performed to complete all the required wiring between the PEs, ROMs, memory connectors, and off-chip IO circuits. Last design step for the bottom tier is to export it to gds. A separate placement instance for the top tier is then initiated. In this placement instance the top half of all the PE’s are placed with an orientation that mirrors the one on the bottom tier on the y-axis at a location that also mirrors the bottom half of the y-axis. This is done to satisfy the requirement that the top tier is to be mirrored on the y-axis in respect to the bottom tier before face-to-face stacking is employed. The top tier layout is then exported to gds, completing the overall tool flow. In summary, most of the steps in the tool flow for 3D designs involve extra work that is adds up, with the exception of 3D placement, which is fundamentally different than regular 2D placement and is very important for quality results.

6.4 Power Delivery

As pointed out in Section 4.2.1 power delivery is more challenging in 3D than it is in 2D. In this section we describe how 3D power delivery to a processing element is accomplished. Before this can be done it is important to estimate the maximum currents that can be delivered through a given via or metal trace for the process. This is shown in Table 6.6.

Table 6.6: Estimated maximum currents for vias and metals in mA per micrometer

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Via 5</th>
<th>Via 1-4</th>
<th>Metal 1</th>
<th>Metal 2-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>85°C</td>
<td>3.13</td>
<td>0.87</td>
<td>6.28</td>
<td>8.47</td>
</tr>
<tr>
<td>110°C</td>
<td>0.713</td>
<td>0.199</td>
<td>1.43</td>
<td>1.93</td>
</tr>
<tr>
<td>125°C</td>
<td>0.451</td>
<td>0.126</td>
<td>0.91</td>
<td>1.22</td>
</tr>
</tbody>
</table>
Figure 6.16: The floorplan of the system.
Now, given that we know the power consumption of the 3D PE is 5.692 mW with interconnect parasitics (by simulation) and that the power supply voltages is 1.5 V, we can calculate the required power supply current required for the PE.

\[ P = I \times V \]  
\[ 5.692mW = I \times 1.5V \]
\[ I = 3.747mA \]

In order to ensure adequate power delivery to the PE two conditions must be met. First, the PE power ring must be wide enough to support both the vertical connections to the adjacent tier and the current needed by the PE. Second, sufficient power must be delivered to the 3D connected tier above. Third, sufficient vertical strapping to the standard cell power rails must be provided. To ensure that the first condition is met, we consider the pitch of the 3D microbumps along with the required width based on the current supported by the first metal layer at the worst case temperature.

\[ \text{Current Density} = \frac{I}{(\text{Metal Width})} \]
\[ 0.91 \frac{mA}{\mu m} = \frac{3.747mA}{\text{Metal Width}} \]
\[ \text{Metal Width} = 4.11\mu m \]

From the parameters and equations we derive the minimum required width for the power ring to be 4.11 \( \mu m \). Additionally, the pitch required for a 3D microbump is 6.0 \( \mu m \). In order
to satisfy both these requirements we use a metal width of 7.0 \mu m.

To insure that the second condition is met we consider the total vertical current capacity. The total vertical current capacity is limited by the via, which can take 0.126 \mu A. Now given that each side of the the PE is 400 microns and the pitch of the microbumps is 6.0 microns, a total of 66 microbumps will fit per side, which adds up to a grand total of 264 microbumps for all four sides. The 264 vias can deliver a total vertical current of 33.0 mA. This greatly exceeds the required power draw of 3.747 mA.

Finally to insure the third condition is met we need to calculate the amount horizontal strapping that is needed. We do this by first calculating the maximum current that can be supported by all horizontal metal in the first metal layer.

\[ I_{M1Max} = \text{Rail Width} \times \text{Number of cell rows} \times 2 \times \text{Current Density} \]  \hspace{1cm} (6.7)

\[ I_{M1Max} = 0.5 \times 105 \times 2 \times 0.91 \]  \hspace{1cm} (6.8)

\[ I_{M1Max} = 95.5 mA \]  \hspace{1cm} (6.9)

From the calculation we determine that the maximum current supported by all horizontal metal in the first metal layer is 95.5 mA. Since this number greatly exceeds the total current draw of 3.747 mA that is required, we can supply all the current we need from the horizontal rails and do not need any strapping.

6.5 Results

In this section we evaluate the power reduction obtained from using 3D integration. As the power reduction due to 3D integration comes from two different areas, a reduction in main memory power and a reduction in power consumed by the parasitics of the interconnect of the
6.5.1 Memory Results

To compare the memory access of the 3D memory to a similar 2D off-chip memory solution we compare the 3D memory to 16 of Micron MT48LC2M32B2P-5 64 Mbit SDRAM parts, as they have a similar word width and column access latency. Sixteen of these parts give the same total word width of 1024 and overall size of 2 Gbit, although the micron part only operates to an operating frequency of 200 MHz where as the 3D memory works up to 1 GHz. According to Micron’s System Power Calculator[48], the power consumption of the each Micron chip is 644.0 mW which sums up to total of 10.304 W for all 16. This compares to a total of 3.1 W for the 3D memory and as such is a 70% reduction in memory power. This is a significant improvement especially since it does not consider the additional I/O power overhead required on the logic side and the fact that the Micron memories only run up to 200 MHz, where as the 3D integrated memory works up to 1 GHz.

6.5.2 Thermal Profile

3D integrated circuits will have a higher heat density than 2D circuits because active devices are now stacked on top of each other, which increases heat density and makes it so that only one of the tiers in the stack is in direct contact with the heat sink. Having a higher operating temperature has two major effects. First, if the temperature becomes too high the circuit will not function reliably. Second, as the temperatures increases more power will be consumed due leakage and as a result, a 3D circuit will see an increase in power consumption. In order to assess this increase in leakage power and make sure that it does not negate the 3D power advantages, it is important to analyze the leakage power with respect to temperature for our 3D design. To get an estimate of the thermal profile of the system, we use HotSpot[29] to model the temperature of the five tier stack-up. The result generated by HotSpot is shown in Figure 6.17 and in the result the highest temperate is $49.64^\circ C$ ($322.79^\circ K$).
Figure 6.17: HotSpot simulation results.
Additionally, to explore the effect of leakage on power consumption for Tezzaron’s process, we use HSPICE to simulate the leakage power for a drive-strength-one standard cell inverter with respect to temperature. The results of the simulation are normalized to the leakage power of the inverter at 30°C and shown in Figure 6.18.

![Figure 6.18: Normalized leakage power versus temperature for drive-strength-one standard cell inverter in Chartered Semiconductor’s 130 nm low power logic process.](image)

### 6.5.3 Logic-on-Logic Results

To quantify the improvements of using logic-on-logic 3D integration in the PE, we compare a single PE that is placed and routed using the method described in Section 6.3 to one that is placed and routed in a normal 2D fashion using the same design tools. We compare the speed and power using the following approach. First, we gather the switching activity of the circuit...
by simulating a Verilog description of the PE with a test bench in Mentor Graphics Modelsim
Verilog simulator and exporting the switching activity to a SAIF file. We then extract the
interconnect parasitics of both designs to a SPEF file using Cadence Encounter. We read the
SAIF, SPEF and Verilog files into Synopsys PrimeTime to assess the maximum clock period
and the power consumption at 25°C. As leakage power increases with temperature, we scale the
leakage power of the 3D PE from 0.511µW at 25°C to 1.022 at 50°C, which is the maximum
temperature reported by the HotSpot thermal simulation. The overall power consumption for
both the 2D and 3D case is shown in Table 6.8 and the other metrics in Table 6.7.

Table 6.7: Summary of comparison between the 2D and 3D PE.

<table>
<thead>
<tr>
<th>Metric</th>
<th>2D</th>
<th>3D</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE Footprint (mm²)</td>
<td>0.3058</td>
<td>0.1552</td>
<td>-49.2%</td>
</tr>
<tr>
<td>Total Wire Length (mm)</td>
<td>588.0</td>
<td>487.3</td>
<td>-17.1%</td>
</tr>
<tr>
<td>Max Frequency (MHz)</td>
<td>31.61</td>
<td>33.84</td>
<td>+7.1%</td>
</tr>
<tr>
<td>Max Performance (MFlops)</td>
<td>316.1</td>
<td>338.4</td>
<td>+7.1%</td>
</tr>
<tr>
<td>Efficiency (mW/GFlop)</td>
<td>18.9</td>
<td>18.0</td>
<td>-5.0%</td>
</tr>
</tbody>
</table>

Table 6.8: Summary of power comparison @ 31.61MHz between the 2D and 3D PE.

<table>
<thead>
<tr>
<th>Power</th>
<th>2D</th>
<th>3D</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Power (mW)</td>
<td>4.181</td>
<td>4.176</td>
<td>-0.1%</td>
</tr>
<tr>
<td>Parasitic Power (mW)</td>
<td>1.794</td>
<td>1.516</td>
<td>-15.5%</td>
</tr>
<tr>
<td>Leakage Power (mW)</td>
<td>0.000511</td>
<td>0.001022</td>
<td>+100.0%</td>
</tr>
<tr>
<td>Total Logic Power (mW)</td>
<td>5.975</td>
<td>5.692</td>
<td>-4.8%</td>
</tr>
</tbody>
</table>

From these results we can see that the 3D PE reduces the required footprint by 49.2%,
requiring an area of 0.1552 mm² instead of area of 0.3058 mm². Furthermore, the total wiring
in the PE is reduced from 588.0 mm down to 487.3 mm, which is a 17.1% reduction. Figure 6.19 shows the distribution of the wire lengths for both the 2D and the 3D PEs and from this figure it can be seen that the 3D PE has a larger portion of smaller nets than the 2D PE. Overall, the wire-length reduction allows the 3D PE to operate 7.1% faster than its 2D equivalent, which in turn allows the PE to process 7.1% more MFlops at 5.0% better efficiency. Furthermore, this wire-length reduction reduces the power consumed by the parasitics of the wiring by 15.5%, leading to 4.8% reduction in the overall logic power when running at 31.61 MHz. This increase dwarfs the additional 511 nW consumed by leakage power that stems from running at a higher temperature in the 3D case. However, at smaller technology nodes and in a process that has a low threshold voltage, the increase in temperature resulting from 3D integration is an issue that must be faced.

![Figure 6.19: Histogram showing the wire-length distribution for both 2D and 3D PEs.](image-url)
6.5.4 Comparisons To Other Works

Finally, to put the work into perspective we compare the efficiency and the performance of the overall system including all eight PEs to three other works that also utilize single precision floating-point calculations. This comparison is shown in Table 6.9. From the table we can see that the system is more power efficient than the other three works with 3D integration helping to increase the gap further. However as the other systems achieve a higher maximum GFlops it is likely that they could trade performance for power efficiency to some degree to approach the power efficiency of this work, but the bottom line is that 3D logic-on-logic integration helps increase power efficiency of the system.

Table 6.9: Comparisons to other works.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficiency (mW/GFlop)</td>
<td>18.9</td>
<td>18.0</td>
<td>194</td>
<td>43.75</td>
<td>89.28</td>
</tr>
<tr>
<td>Performance (GFlops)</td>
<td>2.529</td>
<td>2.707</td>
<td>6.2</td>
<td>32.0</td>
<td>2.8</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>47.8</td>
<td>48.736</td>
<td>1200</td>
<td>1400</td>
<td>250</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>31.61</td>
<td>33.84</td>
<td>3100</td>
<td>5600</td>
<td>400</td>
</tr>
<tr>
<td>Process (nm)</td>
<td>130</td>
<td>130, 3D</td>
<td>90</td>
<td>90</td>
<td>130</td>
</tr>
<tr>
<td>Result</td>
<td>Simulated</td>
<td>Simulated</td>
<td>Measured</td>
<td>Measured</td>
<td>Measured</td>
</tr>
</tbody>
</table>

6.6 Conclusion

We have presented a SAR processor that uses reconfiguration and 3D integration to meet the size and power efficiency demanded by SAR applications in aircraft and satellites. By reconfiguring the data path of the PE we can reduce the number of arithmetic units required in every PE by 14 (from 24 down to 10, a 58.3% reduction), reducing the total area required. By using 3D integration in the memory we can reduce the power consumption of the memory by 70%, in
addition to saving the energy required for off-chip communications. By using 3D logic-on-logic integration we demonstrate a PE that can operate 7.1% faster in simulation, using less power and requiring a 49.2% smaller footprint than a 2D PE. We have proposed an algorithm for TSV assignment based on Lee’s algorithm and shown how the algorithm when used in conjunction with 2D CAD tools can be used to realize the 3D design. In summary, we have presented a 3D integrated reconfigurable SAR processor design that has a power efficiency of 18.0 mW/GFlop and meets the computational, size and power constraints of the SAR application.
Chapter 7

Conclusions and Future Work

This chapter presents our conclusions, future work and gives a summary of the four main contributions of this dissertation.

7.1 Summary of Contributions

In this dissertation we have demonstrated two case studies that show how 3D integration can be used to improve the performance and power consumption of synthetic aperture radar circuits. In the first case study we have shown how a novel memory division scheme can be used in conjunction with 3D integration to reduce the energy required for a 1024 point FFT from $5.476 \, \mu J$ down to $4.227 \, \mu J$. In this case study, using the memory division achieves an energy reduction of 59.2% per memory access and using 3D integration lowers the power consumption of the logic by 5.2%. In essence the two techniques are synergetic because the memory division scheme trades memory bandwidth for an increase in routing congestion. This increase in routing congestion is mitigated by 3D integration. In the second case study we demonstrated how 3D integration was used in two different ways to realize a full SAR processor. First, 3D integration was used in the main memory to build a 3D memory that consumes 70% less power than a comparable 2D memory. Second, logic-on-logic 3D integration was used in the processing elements (PEs) to allow them to operate 7.1% faster and decrease the interconnect power
consumption by 15.5%. Additionally, in the second case study we explored how 2D CAD tools could be used to implement logic-on-logic 3D integration in the PEs and proposed a new algorithm for TSV assignment based on Lee’s algorithm.

### 7.2 Future Work

In terms of future work based on this dissertation, we believe it would interesting to explore alternate memory division schemes to the one used in the first case study. These memory schemes would not be based on the basic partitioning in Figure 5.2 but instead would be based on the partitioning in Figure 5.7 or possibly other partitioning schemes. These alternate memory partitioning schemes could provide more bandwidth or have less routing congestion when compared to the scheme used in Chapter 5. Additionally, it would be interesting to explore if the routing demands caused by the hypercube memory division scheme could be met through other process features such as a very high number of metal layers. Another area for future work that was not really addressed in the case studies is design for test. In order for 3D integration to be successful, design for test issues for 3DIC must be improved. As such, it would interesting to redesign the systems in both cases studies from ground up with a focus on design for test. In terms of future work for 3DIC search community as a whole. We think that designing new 3D EDA tools and methodologies, working to improve the reliability and cost effectiveness of 3D integration and TSVs, and exploring better ways to do 3D design for test should be a priority.
REFERENCES


132
