2011

Testable Clock Distributions for 3d Integrated Circuits

Michael T. Buttrick
University of Massachusetts Amherst

TESTABLE CLOCK DISTRIBUTIONS FOR 3D INTEGRATED CIRCUITS

A Thesis Presented

by

MICHAEL T. BUTTRICK

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

May 2011

Department of Electrical and Computer Engineering

TESTABLE CLOCK DISTRIBUTIONS FOR 3D INTEGRATED CIRCUITS

A Thesis Presented

by

MICHAEL T. BUTTRICK

Approved as to style and content by:

Sandip Kundu, Chair

Israel Koren, Member

Russell Tessier, Member

C. V. Hollot, Department Head
Electrical and Computer Engineering

DEDICATION

"Bear in mind that the wonderful things you learn in your schools are the work of many generations. All this is put in your hands as your inheritance in order that you may receive it, honor it, add to it, and one day faithfully hand it on to your children. "

- Albert Einstein

To those whose knowledge I have inherited, and to those upon

whom it will be bequeathed.

ACKNOWLEDGMENTS

I would like to thank my thesis advisor Professor Sandip Kundu for his insight and guidance through the past years. Without him, this work would not have been possible. I would also like to thank Professors Israel Koren and Russell Tessier for their involvement in my thesis committee as well as in courses over the years.

Thanks to my fellow students for their knowledge and assistance. It was great to know I was surrounded by so many intelligent, resourceful, and kind people.

Finally, I would like to offer my most sincere thanks to my friends and family who have supported me throughout this process. I know I was not always forthcoming with information regarding my progress, but deep down I appreciated the fact that you kept asking. For those who always wanted to know more about my research, read what follows. . .

ABSTRACT

TESTABLE CLOCK DISTRIBUTIONS FOR 3D INTEGRATED CIRUITS

MAY 2011

MICHAEL T. BUTTRICK, B.S., UNIVERSITY OF MASSACHUSETTS AMHERST

M.S.E.C.E, UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Sandip Kundu

Recently, transistor scaling has allowed the size of a die to remain fairly constant even as the number of transistors present increased. However, as scaling slows, the size of a die threatens to increase. This increase would lead to longer global interconnects which are a major source of power dissipation and delay in an IC.

The 3D integration of dies promises to address these problems. By partitioning a design among two or more dies and stacking them vertically, the average interconnect length is greatly decreased and thus power is reduced. Also, since smaller dies will have a higher yield, 3D integration will reduce manufacturing costs. However, this increase in yield can only be seen if manufactured dies can be tested before they are bonded in a stack. If not, the overall yield for the die stack may actually be worse than that of the single, larger die.

One of the largest issues with prebond die testing is that, to save power, a single die may not have a complete clock distribution network until bonding. This thesis addresses the problem of prebond die testability by ensuring the clock distribution network on a single prebond die will operate with low skew during testing and at a reduced power consumption during operation as compared to a full clock network. A number of methods of driving disconnected clock networks with low skew are presented and evaluated. The development of a Delay Lock Loop is detailed and used to synchronize disconnected clock networks on a prebond die. This succeeds in providing a test clock network that operates with a skew that is almost exactly equal to the target postbond skew.

Additionally, a scheme to increase interdie bandwidth by multiplexing connections known as Through-Silicon Vias (TSVs) by the system clock is presented. This technique allows for great increase in the number of effective signal TSVs while imposing a negligible area overhead causing no performance degradation. These two solutions have the ability to advance 3D integration by making the organization more commercially viable.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS ..... **v**

ABSTRACT ..... **vi**

LIST OF TABLES ..... **xii**

LIST OF FIGURES ..... **xiii**

LIST OF ABBREVIATIONS ..... **xvi**

CHAPTER

1. INTRODUCTION **1**

1.1 Motivation 1

1.2 Contribution 2

1.3 Organization 2

2. THREE DIMENSIONAL INTEGRATED CIRCUITS **4**

2.1 Introduction 4

2.2 Benefits of 3D Integrated Circuits 4

2.2.1 Reduction of Global Interconnect Length 5

2.2.2 Increased Die Yield 6

2.2.3 Integration of Disparate Technologies 9

2.3 Partitioning a Design 10

2.3.1 Partitioning Granularity 11

2.3.2 Partitioning for Testability 13

2.4 Through Silicon Vias (TSVs) 13

2.4.1 Wafer Thinning 15

2.4.2 TSV Density 16

2.5 Wafer Bonding Approaches 18

2.5.1 Face-to-Face Bonding 18

2.5.2 Face-to-Back Bonding 19

2.5.3 Bonding Granularity (Wafer-to-Wafer, Die-to-Die, Die-to-Wafer) 20

2.6 Thermal Concerns for Stacked Dies 21

2.7 Costs of 3D Integration 22

3. CLOCK NETWORKS 23

3.1 2D Clock Distribution Networks 24

3.1.1 Symmetric Clock Trees 24

3.1.2 Geometric Load Matched Trees 26

3.1.3 Clock Meshes 27

3.1.4 Hybrid Clock Distributions 28

3.1.5 Example Clock Skews 29

3.2 3D Clock Distributions 32

3.2.1 Full 2D Clock Distributions on All Dies 32

3.2.2 Driver Layer with Sparse or No Local Networks 33

3.2.3 Larger Local Networks Disconnected Prebond 35

4. INTEGRATED CIRCUIT TESTING 37

4.1 Test Economics 38

4.2 Scan-Based Testing 40

4.3 Built-In Self Test (BIST) 42

4.4 3D Integrated Circuit Testing 43

4.4.1 3D Scan Chains 43

4.4.2 Testing 3D Clock Distributions 45

5. BUILDING A CLOCK SIMULATION MODEL 47

5.1 Choosing an Appropriate Design 47

5.2 Assumptions 48

5.2.1 Die Configuration 48

5.2.2 3D Clock Network 49

5.2.3 TSV Size and Density 50

5.3 Model Development 52

5.3.1 ASIC Design Flow with Commercial Tools 52

5.3.2 Design Parameter Extraction Scripting 54

5.3.3 Development of a Dynamic, Object-Oriented Model 56

6. TESTABLE 3D CLOCK DISTRIBUTION 59

6.1 Simple Routing for Testability 59

6.1.1 Routing from One Clock Probe 59

6.1.2 Routing from Four Clock Probes 61

6.2 DLL Based Clock Testing 62

6.2.1 DLL Design and Use 62

6.2.1.1 Operation Overview 63

6.2.1.2 Phase Detector 64

6.2.1.3 Charge Pump and Filter 66

6.2.1.4 Voltage Controlled Delay Line 67

6.2.1.5 Operating Characteristics 70

6.2.2 Necessary Provisions for Test (OPCG) 72

6.2.2.1 Test Clock Generation from Scan Clock 73

6.2.2.2 Scan Clock Generation from Test Clock 75

6.2.3 DLL Test Routing with Multiple Clock Probes 78

6.2.3.1 Arrival Time Aware Feedback Routing 79

6.2.3.2 Proximity-Based Feedback Routing 79

6.3 Daisy Chained Test Clock Networks 80

6.3.1 Daisy Chain Construction 81

6.3.1.1 Chain Length Impact 82

6.3.1.2 Algorithm for Chain Construction 82

6.3.2 A Fully-Connected Test Clock Network with Daisy Chained DLLs 84

6.4 Comparison of Results 87

7. INCREASING INTERDIE BANDWIDTH 89

7.1 Background and Related Work 89

7.2 Unidirectional Communication 90

7.3 Bidirectional Communication 92

7.4 Flip-flop Timing Violations 94

7.5 Experimental Results 94

7.5.1 Functionality and Performance 94

7.5.2 Overhead 96

8. CONCLUSIONS AND FUTURE WORK 99

8.1 Conclusions **... 99**

8.2 Future Work **.. 101**

BIBLIOGRAPHY 103

LIST OF TABLES

Table Page

1: Yields of individual dies and designs for different manufacturing processes 8

2: Benefits and Efforts of 3D Partitioning Granularities 12

3: Parasitic parameters for TSVs of various sizes.. 18

4: Summary of High-Performance Clock Distribution Networks... 30

5: Power Savings as a Function of TSV Size.. 51

6: Comparison of One-Probe and Four-Probe simple routing 62

7: Hardware and Power Comparison of Techniques 77

8: Comparison of Arrival Aware and Proximity Based Feedback Routing.................................. 79

LIST OF FIGURES

Figure Page

1: 2D die shown on right with maximum interconnect distance. 3D stack shown on left of the same total area with greatly decreased maximum distance. 5

2: Effect of die size on yield 7

3: Die micrograph and floorplan of a processor with integrated data and instruction cache 9

4: Partitioning granularity for coarse (a) to fine (d)[13] 11

5: A 3D stacked DRAM chip using WLP [24] 14

6: A 3D stack is shown with emphasis on Via-First TSVs and microbumps 15

7: SEM image of 10μm diameter TSVs showing a typical depth of less than 100μm [27]........... 16

8: Face-to-Face Fabrication Procedures [13] 19

9: Face-to-Back Fabrication Procedures [13] 19

10: Wafer-to-Wafer (a) and Die-to-Wafer (b) 3D Integration processes [30] 21

11: A three-level, fanout-of-three clock distribution network 25

12: H-tree (a), X-tree (b), and Tapered H-tree (c) symmetric click distributions 26

13: A geometric load matching clock distribution [32] 26

14: Block diagram of a mesh clock distribution 27

15: Top level view of clock mesh and driver routing 27

16: Mesh + Local Trees (MLT) hybrid clock distribution [35] 28

17: Tree + Local Meshes (TLM) hybrid clock distribution [35] 29

18: Percentage clock skew versus clock frequency of microprocessors in Table 4 [33]. 31

19: Replica Technology where a full H-Tree is present in each layer 33

20: Via Technology where a full distribution is present in a driver layer and clock sinks are fed directly through TSVs.. 34

21: Two layer stack with dedicated clock layer [29] 35

22: Prebond dies each local networks sharing one global network [38] 36

23: Flowchart detailing the costs of testing integrated circuits [40] 39

24: A representation of a scan chain used for structural testing 41

25: BIST architecture using a generic PRPG for test vector generation and a MISR for signature generation 42

26: Three methods of scan chain ordering in a 3D integrated Circuit [43]................................... 44

27: A testable 3D clock distribution using a redundant tree. The redundant tree is shown as a solid line while the remainder is dotted [37].. 46

28: Two die stack with buffers bounding TSVs to reduce interdie loading.................................. 50

29: Post-CTS view of the S1 Core TLU 53

30: Compression of Clock Tree Browser into simpler hierarchy information.............................. 55

31: A generic representation of the building of the clock model 56

32: Simple clock test routing with one clock probe and test-enabled gates.................................. 60

33: Simple clock test routing with four test probes and test enabled-gates 61

34: Block diagram of a Delay Lock Loop syncing the leaf of a clock tree with a reference signal.. 63

35: Linear Phase Detector producing overlapping UP and DN signals depending on input overlap 64

36: Phase detector operation with output shown as top waveform 65

37: Graphical representation of the charge pump and capacitive filter 66

38: One cell of a Shunt Capacitor Variable Delay Line.. 68

39: Effect of control voltage on delay of one variable delay cell 69

40: One cell of a Current-Starved Inverter Variable Delay Line 69

41: Delay and duty cycle versus control voltage for the current-starved inverter delay line......... 70

42: Waveform of DLL locking to reference signal 72

43: Block diagram of DLL with provisions to generate test clock from scan clock...................... 73

44: Launch and capture clocks generated from XOR of taps from fixed delay line 74

45: Block diagram of DLL with provisions to generate scan clock from test clock 75

46: Circuits to (a) divide clock by 5[4] and (b) 4 with 50% duty cycle 76

47: SPICE waveform of DLL syncing with test clock, then applying divided scan clock 77

48: Synchronizing regions of the die using one DLL for each probe point 78

49: Daisy chaining clock networks for low-skew operation during test 81

LIST OF ABBREVIATIONS

3D IC – 3-Dimensional Integrated Circuit

ALU – Arithmetic Logic Unit

ASIC – Application Specific Integrated Circuit

BIST – Built-In Self Test

CAD- Computer-Aided Design

CDR – Clock and Data Recovery

CMOS - Complementary Metal Oxide Semiconductor

CMP – Chemical and Mechanical Polishing

CTS – Clock Tree Synthesis

DET – Dual Edge Triggered

DFT – Design For Testability

DLL – Delay Lock Loop

DRAM – Dynamic Random Access Memory

DRIE – Deep Reactive Ion Etching

DSP – Digital Signal Processor

FDL – Fixed Delay Line

GaAs – Gallium Arsenide

KGD – Known Good Die

MISR – Multiple Input Shift Register

MSB – Most Significant Bit

OPCG – On-Product Clock Generation

PLL – Phase Lock Loop

PRPG – Pseudo-Random Pattern Generator

RTL – Register Transfer Level

SEM – Scanning Electron Microscope

SET – Single Edge Triggered

SPICE – Simulation Program with Integrated Circuit Emphasis

SUT – System Under Test

TDM – Time Division Mulitplexing

TLU - Trap Logic Unit

TSV – Through Silicon Via

TTM – Time To Market

VCDL – Voltage Controlled Delay Line

VCO – Voltage Controlled Oscillator

WLP – Wafer Level Packaging

CHAPTER 1

INTRODUCTION

1.1 Motivation

Since the first integrated circuit was made in 1958, there has been pressure to manufacture increasingly many transistors onto a die. In 1965, Gordon Moore of Intel predicted that the number of transistors that could be inexpensively manufactured on a die would double every two years. This became known as Moore's law and had been driving the development of integrated circuits for decades. With this increase in device density came increased functionality and performance. Until recently, transistor scaling has been effective in steadily increasing density and performance while decreasing power consumption. However, as the supply voltage approached the device threshold, it could no longer be scaled and power consumption became a real concern. Designs at the circuit and architectural levels have long aimed to find a balance between performance and power. This is the classic tradeoff designers are battling today.

A forerunner among the solutions that will increase performance while decreasing power is the 3D Integrated Circuit. The benefits of a 3D IC are a result of greatly decreased delay and power dissipation from long interconnect wires. A design is partitioned and manufactured on two or more dies which are then stacked vertically and bonded to make a 3D IC. Because of this stacking, the distance global signals must travel is greatly reduced when compared to a 2D design. This decrease in interconnect distance is seen as a reduction in delay which allows for a faster clock speed and higher performance as well as a reduction in wire capacitance which brings lower power dissipation. This three dimensional organization promises to allow current CMOS technologies to continue to provide us with increasing performance using reasonable power until a suitable alternative can be found. However, many challenges exist that must be solved before 3D integrated circuits will be commercially viable. Namely, it is essential that dies be tested prior to bonding to ensure that a faulty die is not bonded to a functional die causing a reduction in manufacturing yield. Also, while the manufacturing process for creating inter-die connections known as Through-Silicon Vias (TSVs) is in place, there is the opportunity for innovations at the circuit level that will ease some of the pressure on the push to create smaller and denser TSVs which will be less reliable.

1.2 Contribution

The contribution of this document will be the exploration of the 3D design space and issues related to prebond testing of dies. Solutions for creating a testable 3D clock distribution are be proposed, reviewed, and evaluated based on their effectiveness and overhead. Also presented is a means to increase the effective number of signal TSVs available in a 3D design without incurring unacceptable overhead. It is hoped that these solutions will be helpful in making 3D integrated circuits steps closer to being a commercially viable option.

1.3 Organization

This thesis document is organized to familiarize the reader with the issues surrounding the testing of prebond dies for 3D integrated circuits before detailing the work done in this area.

This will begin in Chapter 2 with a review of 3D ICs including their advantages, disadvantages, partitioning of designs, Through Silicon Vias (TSVs), wafer bonding approaches, and thermal considerations. Chapter 3 will detail typical clock distributions in 2D microprocessors and discuss advantages and disadvantages to different 3D clock distributions proposed by various sources. In Chapter 4 we will examine integrated circuit testing including test economics, scan-based testing, Built-In Self Test (BIST), and special issues that arise in testing 3D integrated circuits. Chapter 5 is dedicated to a description of how the clock simulation model was developed on which experiments will be performed and Chapter 6 describes how it is used to produce various organizations of testable 3D clock distributions. Also present in Chapter 6 is a detailed description of the development of a Delay Lock Loop (DLL) with provisions for On-Product Clock Generation so that it can be used for scan-based testing. Two implementations using DLLs to connect the clock network in prebond dies are presented and evaluated. Chapter 7 departs from prebond testing and focuses on improving the benefits of 3D integration by allowing for larger TSVs by multiplexing signals over these connections thus increasing interdie bandwidth. Finally, Chapter 8 summarizes and concludes this thesis and proposes further work that can be explored in this area.

CHAPTER 2

THREE DIMENSIONAL INTEGRATED CIRCUITS

2.1 Introduction

For the past several decades, transistors have been scaled to increase performance while reducing area and power consumption. As transistors scale, so too does the cross-sectional area of the interconnects that link them. While the number of transistors in a design is ever increasing, the size of the die on which a design is manufactured is limited by the lithographic process in which it is patterned. Because of this, most large microprocessor designs tend to have a similar die size. The result is that while transistors scale, the length of global interconnects in a design does not. As such, these interconnects are becoming an increasing source of power dissipation and are limiting design performance. This is one of the major motivations behind the vertical stacking of dies in an IC design.

2.2 Benefits of 3D Integrated Circuits

While decreasing global interconnect length is a major motivating factor pushing toward 3D integration in this world of demanding high performance while using less energy, there are other notable benefits of this organization that must be understood. While this survey is not exhaustive, it touches upon those characteristics that will make the design and manufacture of 3D ICs attractive to widespread adoption.

2.2.1 Reduction of Global Interconnect Length

The 3D integrated circuit offers a promising solution to the interconnect problem. In a typical 2D design of area A, the longest possible interconnect length can be shown to be

$$L_{max} = 2\sqrt{A} \tag{1}$$

When a design of the same area is partitioned onto N dies, each of a thickness T, the longest possible interconnect now becomes

$$L_{max} = 2\sqrt{A/N} + 2T \tag{2}$$

Generally the thickness of a die in a 3D stack is very low compared its *x* and *y* dimensions because of wafer thinning which will be discussed in Section 2.4. Thus, a 2D design will have a maximum interconnect length that is N^2 times longer than its 3D counterpart. This situation can be clearly seen in Fig. 1. While there will be very few metal lines that will approach the longest possible interconnect length, decreasing this length will lead to a decrease in the total global interconnect length for the design.

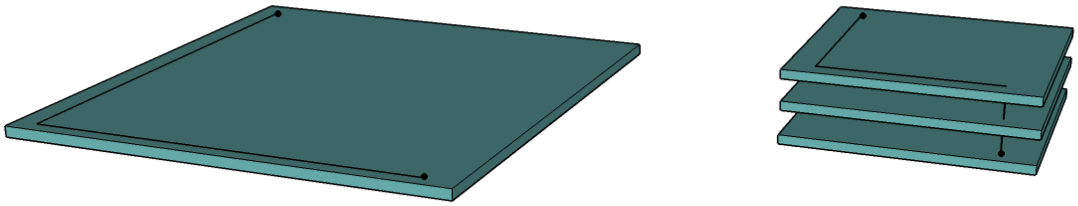


Fig. 1: 2D die shown on left with maximum interconnect distance. 3D stack shown on right of the same total area with greatly decreased maximum distance.

Since all metal lines have a capacitance and resistance per unit length, these values will increase as a metal line is made longer. To make matters worse, when technology scales, the metal lines are made thinner and their resistance per unit length increases. Together, resistance and capacitance in interconnects cause delay because charge must accumulate on the capacitance of the line in order to toggle the voltage of the net. The current used to charge this capacitance is limited by the resistance of the net. Additionally, a line that is toggled will dissipate power according to the formula

$$P = \frac{1}{2}CV^2 \tag{3}$$

where C is the capacitance of the net being toggled. Keep in mind this is for one charge or discharge of the net. For a net that exhibits periodic behavior, the amount will be multiplied by 2*f* where *f* is the frequency at which the net switches.

It is clear that decreasing the total interconnect length of a design will result in power savings and the potential for performance increases due to reduced delay. It should also be clear that 3D integration by stacking dies offers a promising solution toward this goal. However, 3D integration offers another very attractive benefit: increased yield.

2.2.2 Increased Die Yield

Manufacturing yield is defined as the percentage of dies that are printed on a wafer that pass tests designed to expose both permanent faults as well as timing failures in the manufactured design. There are many sources of these faults including wafer defects, lithographic non-uniformities, over- or under-etching, and many others. While most of the sources of failures mentioned are more likely to result in process variation and degraded performance, it is the wafer defects that are most likely to result in a non-functioning die. Luckily, the impact of wafer defects is greatly reduced by manufacturing a design in a 3D process.

Defects in a wafer can be due to impurities in a silicon ingot before wafers are cut, but can also be due to crystal sliplines and dislocations from heating or striations, dimples, and polishing scratches that are introduced by the manufacturing process [21]. Many efforts have been made to model these defects to more accurately predict yield. There are myriad factors that affect the distribution of these defects; so many in fact that empirical models often prove the most useful.

In a 3D integrated circuit, the design is partitioned among two or more dies. Because of this, each individual die will of course be smaller than the die in the original 2D design. This fact is a very powerful one in terms of design yield. The situation is rather simple and can be seen in Fig. 2. The two wafers shown are the same size and contain the same defects. Because each die in the wafer on the left is larger, a defect on the edge of one die will cause the whole die to be defective whereas this defect would only affect a fraction of the area on the wafer on the right where the dies are smaller.

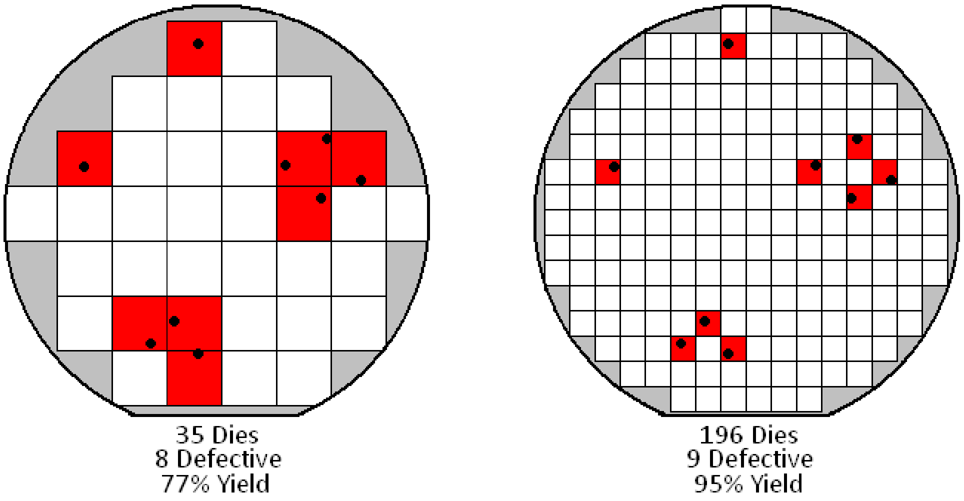


Fig. 2: Effect of die size on yield

To be more precise, we can use (4) to calculate the expected yield based on die size. This formula is derived empirically based on data from industrial fabrication sites.

$$die\ yield = \left(1 + \frac{defect\ desity \times die\ area}{\alpha}\right)^{-\alpha} \qquad (4)[19]$$

In (4), α is a parameter that depends on the complexity of the manufacturing process. A good estimate of the standard CMOS process is α=3. Since this number is roughly proportional to the number of masks in a design [19] and the basic die manufacturing steps are similar for both 2D and 3D designs, it is expected that this parameter will not be considerably different in a 3D process. Using this equation, we can see that if we have an original design of 1 cm^2 area with 0.6 defects/cm^2, the design will have a yield of 57%. Similarly, if this design is partitioned into three dies, and similar parameters are used, each die will have an expected yield of 82%. If the dies can be tested after they are manufactured and before they are bonded, the design yield will stay at approximately 82%. This is of course assuming 100% yield in the packaging process, which is not unrealistic as yield of 98.8-100% are reported in [20]. We can see that by manufacturing the design as a 3D IC instead of a 2D chip, we have increased yield by 25%. This is a powerful result that will lead to a lower manufacturing cost per die.

This increase in yield can only be seen, however, if all dies can be tested prior to bonding. This is an important stipulation and is central to this thesis. If the dies are not tested prior to bonding, a two good dies could be stacked with a bad die leading to a defective stack and the loss of all dies. Thus, without prebond testing the yield for the stack will be the product of the yields of each individual die. In our simple example where all dies have the same yield and three dies are used, assembling the die stack without testing the prebond dies, the yield drops to 55%. As you can see, this is yield is lower than if the design had been manufacture on one die in a 2D process. The comparison between these situations can be seen in Table 1.

What is clear is that manufacturing a die in a 3D process reduces the area of each die which in turn increases the manufacturing yield. Furthermore, dies must be tested prior to

Table 1: Yields of individual dies and designs for different manufacturing processes

	Die Yield (%)	Design Yield (%)
2D	57	57
3D Blind Bonding	82	55
3D with prebond testing	82	~82

bonding for this benefit to be seen, otherwise the design yield will be slightly lower than the 2D yield. Thus, prebond testing is essential to realizing the yield benefits of 3D ICs.

2.2.3 Integration of Disparate Technologies

Another exciting benefit that will be brought about by 3D integrated circuits is the ability to bond dies manufactured in different processes together. In many systems, there is a need to include subsystems that have been manufactured in different processes. For example, every personal computer currently produced contains a processor manufactured on one die that is mostly CMOS datapath, execution, and control logic. The remainder of the die is occupied by the cache; relatively dense storage that can be accessed quickly because of its physical proximity to the logic. This can be seen in Fig. 3. Because they can be manufactured in the same process, logic and cache share a die because of the of performance and cost benefits such an organization allows.

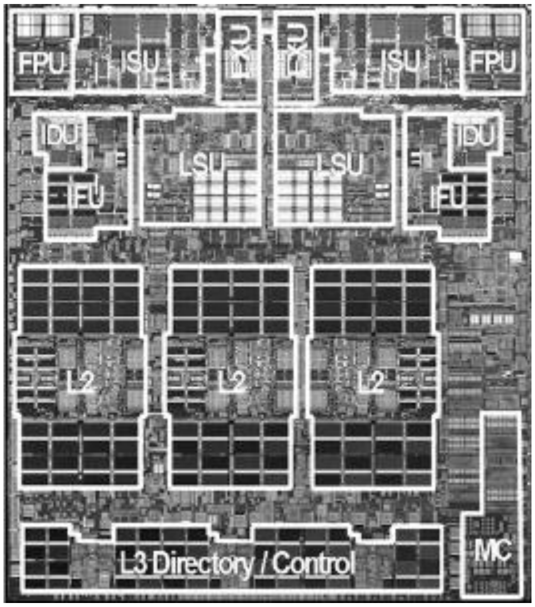


Fig. 3: Die micrograph and floorplan of a processor with integrated cache [18]

The same personal computer also contains main memory with is most often implemented using Dynamic Random Access Memory (DRAM). This type of memory is very dense as well as fast. Its density makes it less expensive to manufacture, but because its manufacturing process involves produced trench or stacked capacitors [6], DRAM cannot be manufactured on the same die as CMOS logic. Therefore, it resides near the processor, but connected through many centimeters of copper trace on a printed circuit board. This physical distance and wiring result in slower read and write times for the main memory.

This situation could be greatly improved if a die containing DRAM could be stacked in a 3D process directly on a processor. This would allow for a large on-chip cache close to the processor cores that would help reduce cache misses [13]. Because processor throughput is highly dependent on availability of data, the ability to integrate DRAM into the same die stack as the processor would lead to an appreciable performance increase.

The benefits of integrating disparate technologies do not end with logic and DRAM. Flash memory could also be included to allow for on-chip non-volatile storage. Additionally, high speed and microwave analog circuits fabricated in GaAs could be stacked on top of a Digital Signal Processing (DSP) die. This would allow for much smaller fully contained systems. In high production parts, this integration could lead to significant cost reduction in wireless applications such as cellular phones [47].

2.3 Partitioning a Design

When designing a 3D integrated circuit, one of the most important design steps is the partitioning between dies. The manner in which this is done will greatly affect the performance and power characteristics of the finished design. Partitioning must occur before placement, and placement will have a sizable impact on routing. Because routing reduction is responsible for many of the benefits of 3D ICs, these savings all rely on design partitioning.

2.3.1 Partitioning Granularity

When dealing with partitioning on a high level, we speak of partitioning granularity; that is, the relative size of the smallest blocks that will be determined to reside on what die. We will speak to partitioning granularities from coarse where partitioning is done using large blocks such as processor cores, to fine where gates or transistors are the blocks used in the partitioning process. Each granularity has its advantages and disadvantages. These can be seen in Fig. 4.

The coarsest partitioning granularity is performed on the core level. This is when one or more processor cores reside on one die with adjacent dies containing large caches or other processor cores. This allows for a greater ease of design since blocks and dies can be designed more or less independently and inter-die connections are low. However, this granularity does not offer much performance improvement over a 2D design.

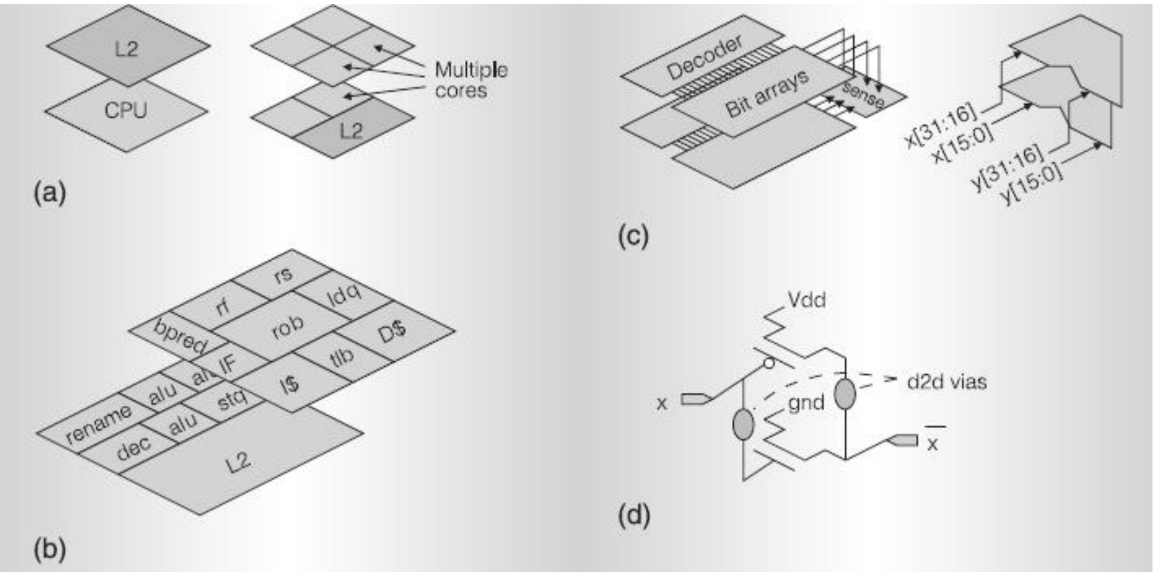


Fig. 4: Partitioning granularity for coarse (a) to fine (d)[13]

A finer granularity occurs when partitioning at the scale of functional units. Previously designed units such as instruction fetch units, re-order buffers, and ALUs would be partitioned among the dies. This would require re-floorplanning and retiming designs. However, this would allow for performance and power improvements without great redesign effort.

The finest partitioning granularity exists when logic gates or individual transistors are partitioned among dies. While this may offer large gains in power and performance for large, complex gates, these gains diminish as the inter-die connections become limited by TSV density. Also, design and redesign complexity becomes huge when dealing with granularities this fine. For this reason, this would only be used in circumstances where the performance improvements are worth the effort.

Table 2: Benefits and Efforts of 3D Partitioning Granularities

Granularity	Benefits	Effort	Examples
Core	Low: Power and performance of blocks unchanged	Low: Reuse of 2D design	Core-On-Core, Cache-On-Core
Function Unit	Medium: Reduced latency and power of routes	Medium: Re-floorplan and retime paths	ALU-On-ALU
Transistor Level	High: Greater decreased latency and power	Extreme: Almost no design reuse	NMOS/PMOS partitioning, domino logic

In practice, a partitioning that exposes the most benefits while minimizing costs will be one where the partitioning blocks may be of different granularities. For example, some high performance blocks may be partitioned finer for performance gains at the cost of a more difficult design whereas simpler, low performance units may comprise a larger block. A summary of the benefits and design effort for various granularities can be seen in Table 2.

2.3.2 Partitioning for Testability

While all partitioning granularities in the previous section are valid, a word must be added about design testability. As explained in Section 2.2.2, prebond testing of dies is essential to a commercial manufacturing process. However, this would be almost impossible at the finer granularities presented above. In a scan-based test methodology, flip-flops would be needed at every signal entering or leaving a die. Thus, partitioning at the gate or transistor level would require far too many such flip-flops.

It should also be noted that proper partitioning could lead to designs that would not only be easier to test, but allow for spare blocks on adjacent dies to replace faulty blocks. This manner of intelligent or fault-tolerant placement would allow for an even greater yield benefit from 3D integrated circuits.

2.4 Through Silicon Vias (TSVs)

In order for 3D integration to be able to deliver on all of its promises, an efficient means of making connections between dies is needed. A connection methodology currently being used in memory chips is called Wafer Level Packaging (WLP)[7]. It is often used to stack memory dies resulting in high capacity memory chips with low footprint. In WLP, 3D interconnections are made after all wafer processing has been completed and can be seen in Fig. 5.

While this method of 3D interconnects allows for dense memory chips and the ability to contain dies manufactured in different processed in one die, these coarse granularity interconnects do not fully realize the potential of 3D stacking. For this, more dense interconnects that can be made through the die are needed. This not only allows for finer granularity connections, but will also result in the wirelength savings discussed in Section 2.2.1.

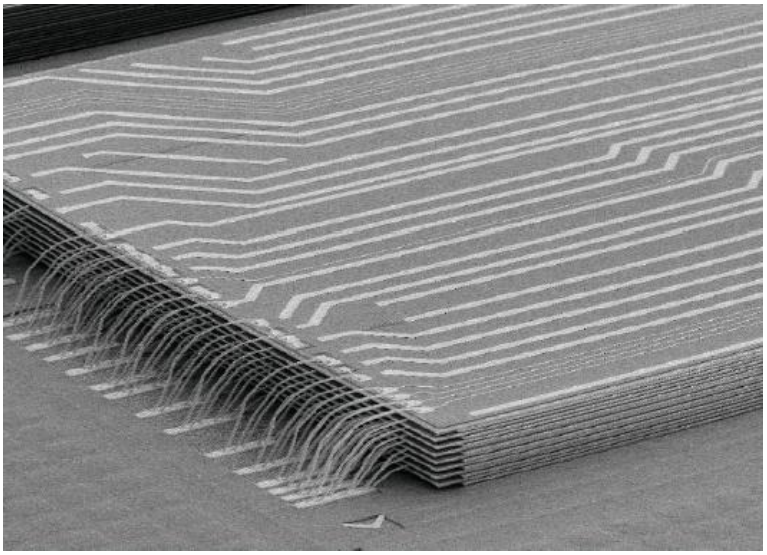


Fig. 5: A 3D stacked DRAM chip using WLP [7]

As you might expect, these inter-die connections will have to be somewhat unique in order to carry signals through the bulk of the silicon wafer. Most large scale production designs today are fabricated on 300mm (11.8 in) wafers that are approximately 775µm thick [15]. Typically, the active device layer extends only a few tens of microns into the silicon, leaving the remainder to provide mechanical strength and a path for heat generated at the devices to be removed. It is through this typically empty silicon that 3D vertical connections called Through-Silicon Vias (TSVs) must be made.

TSVs have been around for many years, but only recently have come into celebrity for their expected role in the future of integrated circuits. While there have been many suggested methods of manufacturing these connections, we will focus on what appears to be the most likely process. In a Via-First process, TSVs are created before the dies are bonded. These are then connected together by microbumps between the dies. TSV are made by creating a blind via (meaning the via does not go straight through the wafer at first) by Deep Reactive Ion Etching (DRIE) or laser drilling. After an insulating oxide layer is grown on the exposed silicon, either

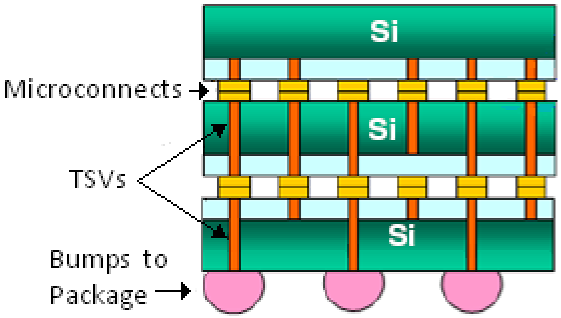


Fig. 6: A 3D stack is shown with emphasis on Via-First TSVs and microbumps

copper or tungsten is sputtered to fill the vias. Then, the wafer is bonded to a carrier wafer and is thinned by Chemical and Mechanical Polishing (CMP) to expose the TSVs. After the backside of the wafer has been metalized and microbumps added, the die is ready for bonding [17]. A depiction of a bonded stack is shown in Fig. 6.

2.4.1 Wafer Thinning

A TSV must pass through the bulk of the silicon to reach the active layer. If the length of a TSV is too large, manufacturing yield becomes a problem as sputtered metals may not reliably deposit in the etched via. Two solutions to this problem are to make the TSV wider or the length shorter. A wider TSV would have lower resistance, but occupy more of the active area on a die [8]. A more suitable solution is to decrease the length of the TSV. This is done by using CMP to thin the wafer to an acceptable thickness that will allow for higher TSV reliability.

As mentioned previously, today's wafers are approximately 775μm thick, and if the 400mm wafer is adopted as expected, this thickness will only increase. However, in order to manufacture TSVs that are reasonably reliable without occupying too great an area, their length must be less than 100μm [17]. This means wafers must be thinned considerably. However, wafers need thickness so they do not break during the manufacturing process. Special handling mechanisms and techniques must be adopted to handle these thinned wafers. As mentioned above, this will likely be achieved by temporarily bonding the wafer being fabricated to a stronger carrier wafer.

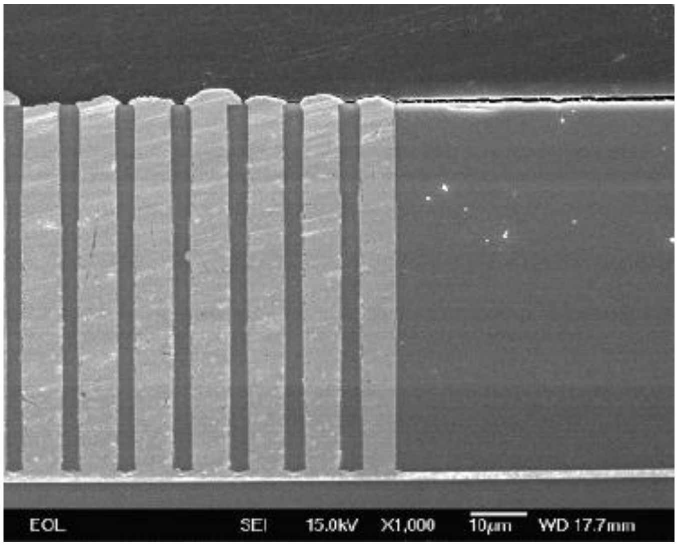


Fig. 7: SEM image of 10µm diameter TSVs showing a typical depth of less than 100µm [14]

2.4.2 TSV Density

The benefit seen from 3D integration is largely a function of TSV density. If only a limited number of TSVs are available to carry signals between dies, then designs must be partitioned at a coarse granularity [13]. However, if TSV density is high, functional units can really be optimized to fit in the smallest 3D volume, thus reducing wire length which will lead to performance gains and power savings. Ideally, we want TSV density to be incredibly high so that any possible connection between dies could be made at any point with limited routing from the source to the TSV. However, this ideal case is not feasible for many reasons.

Firstly, in order to have very high density, the width (or diameter) of the TSV must be very small. This causes problems because a small inter-die connection will have a higher resistance than desired. Moreover, as mentioned in Section 2.4.1, small diameter TSVs will have much lower yield and therefore will be unreliable. Second, when it is time to bond wafers together, smaller TSVs will require much higher precision to bond correctly than larger ones. Finally, because a TSV must pass through the silicon and into the active layer, very high density TSVs will have an impact on the layout of a design [8]. Because the smallest TSV would still be larger than many transistor features, these would crowd the design of even small units.

While TSV widths have been manufactured down to 1µm [13] and ITRS predicts sizes of 1-2µm through 2012 [15], TSV diameters of 10-20µm are common in today's designs [17]. This is because while those reported smaller sizes are possible, their aforementioned drawbacks make them infeasible. For these reasons, in our experiments we will most often assume a TSV width of 20µm.

When considering TSV density, it is also important to keep in mind the purposes for which they will be used. Just as in a 2D design, the distribution of global signals needs careful consideration. Because thorough distribution of power and ground is paramount, as much as 30 percent of TSV must be used for these nets [13]. Additionally, the system clock must be distributed between all dies with low skew. The percentage of TSVs used for clock is a much more flexible number, depending on the implementation of the clock networks on each individual die.

2.4.1 Parasitic Characteristics of TSVs

As with any interconnect, the parasitics introduced by a TSV must be taken into consideration. However, since they are constructed using either copper or tungsten (the same metals typically used in metal routing and standard metal layer vias, respectively), TSV paractitics are not much higher than standard routing. As shown in Table 3, resistance for a typical TSV is very small. Capacitance is slightly more of a concern because the size of a TSV is much larger than the width of a standard metal line. Thus, if we consider a 20µm TSV, we can expect that it will have a higher capacitance and lower resistance when compared to those below.

Table 3: Parasitic parameters for TSVs of various sizes

	Super-Via [33]	2^{nd} Generation [33]	Face-to-Face [33]	Field Model [42]
Size	4 μm x 4μm	1.2 μm x 1.2 μm	1.7 μm x 1.7 μm	4μm x 4μm
Capacitance	7 fF	2-3 fF	~0	2 fF
Resistance	<250mΩ	<350mΩ	Negligible	36mΩ

What is clear here is that TSVs do add parasitic capacitance and resistance to the signal nets which pass through them. This could be important when dealing with very high frequency signals transmitted over short distances. However, the parasitics of a longer net are large enough that a single pass through a TSV will not have an appreciable effect on the load that net imposes on its driver. For this reason, we generally ignore any load added by a TSV in simulations without sacrificing the validity of the results.

2.5 Wafer Bonding Approaches

In constructing a 3D die stack, there are two ways in which dies can be bonded. It is mostly a difference in manufacturing step that separates these methods. These two methods are referred to as Face-to-Face and Face-to-Back bonding. Here, the term "face" refers to the side of the die where metallization of the top metal layers occurs and "back" refers to the silicon substrate. After these methods are discussed, the options for bonding granularity are reviewed.

2.5.1 Face-to-Face Bonding

In this process, connections between dies are made at the top metal layer of each by bonding copper pads together with heat and pressure. Connections to packaging or other dies are made through TSVs on the backside of one of the dies. The steps are outlined in Fig. 8. Note in Step 4 how the wafer is thinned before TSVs are made as discussed in Section 2.4.1.

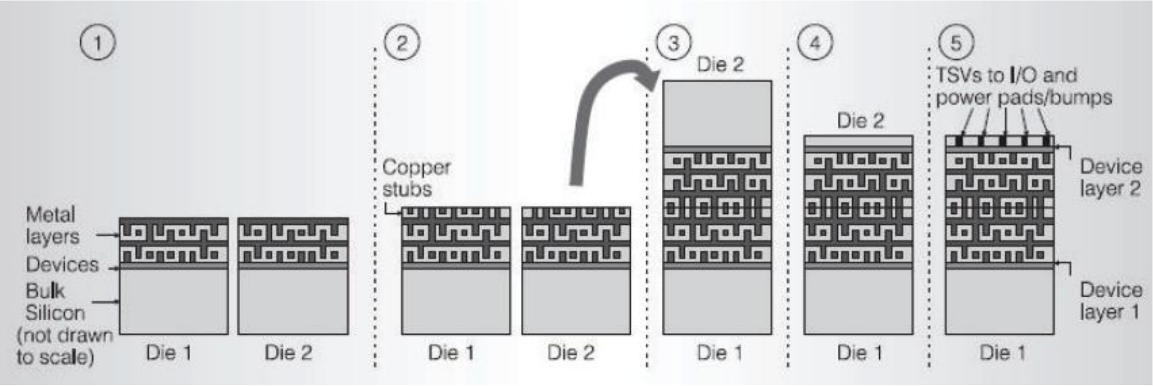


Fig. 8: Face-to-Face Fabrication Procedures [13]

2.5.2 Face-to-Back Bonding

In this type of bonding, copper bumps from the top level of one die are fused with copper pads connecting to TSVs on the backside of another. The process steps for Face-to-Back bonding can be seen in Fig. 9. Note the use of a carrier wafer (referred to as a handle wafer in the figure) for wafer thinning and construction of TSVs. It is important to realize that in Face-to-Back bonding, each wafer can be fully manufactured at the time of bonding. This eliminates dependency of manufacturing steps between the two dies.

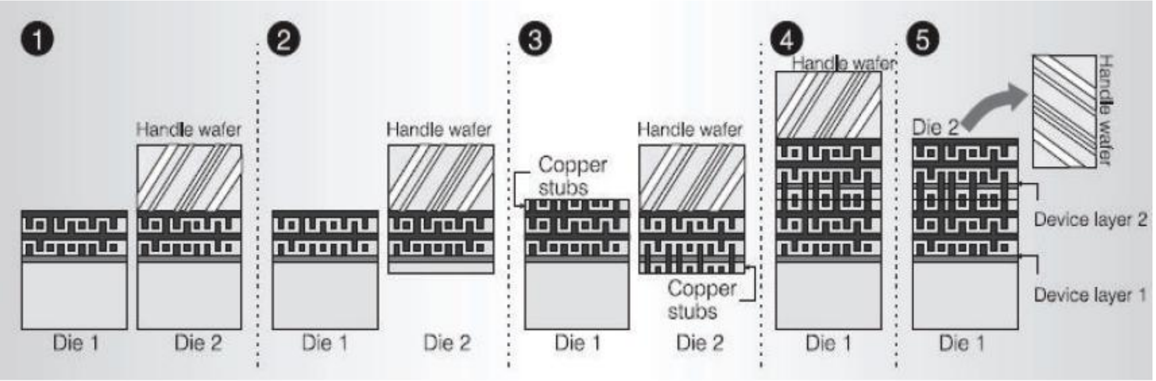


Fig. 9: Face-to-Back Fabrication Procedures [13]

While Face-to-Face bonding appears attractive for the ease of constructing inter-die connections in the top metal layer and relying on TSVs only for primary I/O and power, the benefits of the Face-to-Back organization appear to have made it the preferred method. As mentioned, in Face-to-Face bonding, there is manufacturing dependence between the two dies; they must be stacked before the second wafer can be thinned and TSVs made. This will reduce manufacturing efficiency and will also cause problems with prebond testing. Also, when more than two dies are bonded, one bond must necessarily be Face-to-Back. For these reasons, it makes the most sense to pick this as the preferred bonding approach and make all die bonds in this manner.

2.5.3 Bonding Granularity (Wafer-to-Wafer, Die-to-Die, Die-to-Wafer)

Now that we have considered the manner in which dies will be fabricated for bonding, we will now examine the higher-level methodology that is stacking granularity. Three possible bonding granularities are Wafer-to-Wafer, Die-to-Die, and Die-to-Wafer.

In Wafer-to-Wafer bonding, a complete wafer is placed on and bonded to another complete wafer. This has the benefit of being the fastest bonding technique, but suffers from serious drawbacks. First and foremost, this bonding granularity does not allow for the prebond testing of dies. To be more precise, the information gained by prebond testing of dies would be irrelevant because there would be no control over which dies are bonded together. A good die bonded to a faulty die produces a faulty stack. Additionally, because so many points would need to be aligned at once, TSV bonding yield would be low even with very precise alignment equipment. Thus, Wafer-to-Wafer bonding would suffer from very low yield [20].

Die-to-Die bonding is essentially the opposite of Wafer-to-Wafer bonding. All dies would be tested while a whole wafer before being cut into individual dies. After cutting, faulty dies are discarded while functional dies are sent to be bonded. Clearly, this allows for a Known Good Die (KGD) to be bonded to other KGDs. This has a great advantage in terms of yield over the Wafer-to-Wafer granularity. Also, with fewer points to align between dies, Die-to-Die bonding will see higher TSV yield. However, bonding time will be increased because now each die bond will have to occur individually instead of one whole wafer at a time.

A compromise between the two previous granularities is referred to as Die-to-Wafer bonding. In this step, both wafers whose dies are to be bonded are tested individually. One wafer is cut and the faulty dies are discarded. These dies are then bonded to the dies which were found to be KGDs on the other wafer. In this way, all dies can be tested and only good dies are bonded to other good dies. Also, the one whole wafer adds more stability which will help improve alignment. If all dies can be placed and aligned on the wafer and bonded in a single step as in [20], this will see the time savings of Wafer-to-Wafer bonding with yield similar to Die-to-Die bonding.

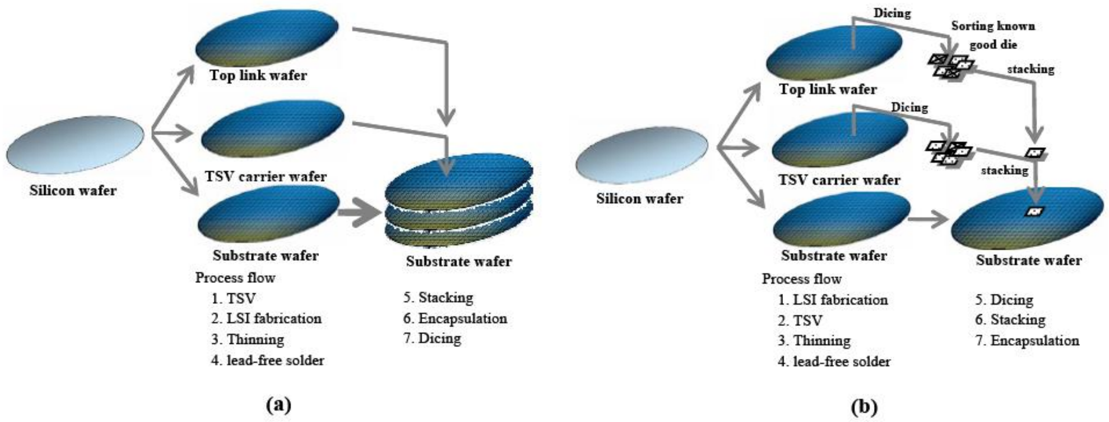


Fig. 10: Wafer-to-Wafer (a) and Die-to-Wafer (b) 3D Integration processes [20]

2.6 Thermal Concerns for Stacked Dies

In any high performance integrated circuit, heat dissipation is a major concern. Larger designs and faster switching frequencies have caused power dissipation to increase considerably. Heat is generally removed by means of a heat sink that is held in contact with the die substrate. However, when dies are stacked, only one substrate is available to remove the heat produced by all dies considering the other exposed face of the stack must remain exposed for I/O and global connections. Thus, naïvely stacking *N* dies would lead to an increase in power density by *N* times with the same area to remove that heat.

Luckily, the very nature of 3D integrated circuits makes the actual situation less grim than this naïve analysis. One of the promises of 3D ICs is an overall reduction in power because of reduced interconnects. Because of this reduction, the total power is expected to be reduced by 15-20 percent [13][44] which should help diminish power density concerns. Also, since TSVs are constructed from metals with high thermal as well as electrical conductivity, they help to decrease the thermal resistance between any die producing heat and the heat sink. TSVs can also be strategically placed for this purpose if needed and not necessarily always used exclusively for signals.

While thermal management is a major concern, many investigations including [13] and [29] have determined that the problem is manageable with proper floorplanning and power reduction techniques.

2.7 Costs of 3D Integration

While 3D integration does promise many potential benefits, it also must be acknowledged that it may carry some costs. TSV construction requires DRIE and CMP, both of which will increase manufacturing times. Also, the bonding of tested dies will require new equipment to align and fuse TSVs. Lastly, prebond testing of dies will add to the total test time for each unit produced. As discussed in Section 4.1, an increase in test time can cause a large increase in manufacturing cost. While these costs are acknowledged, the ability of 3D integration to offer lower power or higher performance design with increased die and design yields makes the organization attractive for future design starts.

CHAPTER 3

CLOCK NETWORKS

Almost all large designs today are synchronous circuits where operations and the transfer of data are controlled by a clock signal. In order for these circuits to function as expected, a clock signal must be distributed to every latch and flip-flop in a design meeting certain clock quality measures such as slew, skew, and jitter [3]. Slew is the rate of change of the clock signal on a transition. As long as the clock drivers are sized properly and not driving excessive loads, this measure only becomes an issue at high clock frequencies. Skew is defined as the difference in static arrival times of the clock signal as measured between two or more distribution points [3]. The skew between two points does not change during operation of the circuit and can only be altered during the design phase. Jitter is a cycle-to-cycle uncertainty in the arrival time of the clock signal caused by power supply variations, thermal noise, and capacitive crosstalk. Jitter cannot be determined or altered during the design phase. Thus, jitter must be added to cycle time budgets for critical paths. Since the design of the clock distribution network has the greatest effect on skew, this is the characteristic we will focus on the most closely.

There are many choices for implementing clock networks, and all trade off between skew and power. That is, the lowest skew networks often require the most metal to distribute the clock signal evenly and will consequently consume the most power. This is another arm of the power versus performance battle we see played out every day in electronic design. Various methods of distributing the system clock will be discussed below beginning with networks for standard 2D integrated circuits. Later, we will see how these are adapted to 3D ICs.

3.1 2D Clock Distribution Networks

The main goal of any clock distribution network is to ensure that the clock signal arrives at the data registers (also referred to as flip-flops, clock sinks, or leaf nodes) at the exact same time. If this can be achieved, then the clock network has zero skew. Of course, it is essentially impossible for there to be zero skew in a clock network, so the effort is placed on minimizing skew while keeping power usage reasonable. The main causes of skew are differences in the length of the clock routing connecting sinks as well as mismatches in the loads that these lines drive. Efforts to make suitable networks aim to minimize the difference in length of clock routing between sink (as in trees) or minimize the effect of a difference in load (as in meshes). We will also examine clock topologies that address both issues in tandem.

3.1.1 Symmetric Clock Trees

The most common method of clocking a large system is by implementing a tree structure to distribute the clock across the chip with minimal skew. In a general tree, there is a clock input that serves as the root for the design. From this root, signals fan out across the chip. After this signal is routed to a subsection of the die, it fans out yet again. In a symmetric clock tree, the number of fanouts at each clock layer will be the same throughout the chip and will occur at geometrically symmetric points on the die. Typically, a buffer is present at each fanout point to drive the subsequent lines. This allows for tighter control of skew and protects against signal degradation [10]. A simple example of a buffered tree can be seen in Fig. 11.

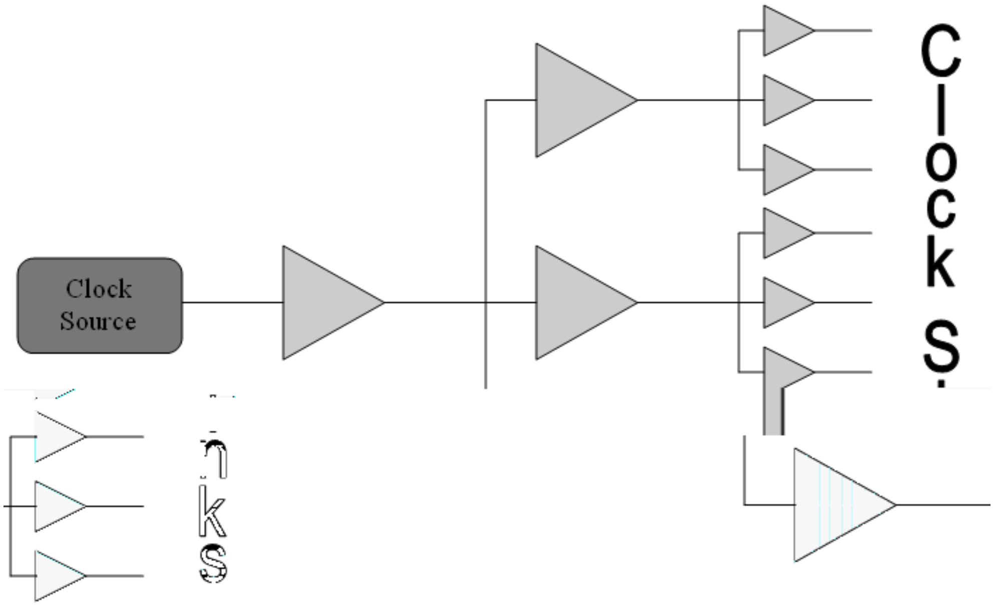


Fig. 11: A three-level, fanout-of-three clock distribution network

More specifically, symmetric trees are often given names based on the shapes the clock routes make. Two such examples are the H-Tree and the X-Tree as shown in Fig. 12a and Fig. 12b. In both of these tree types, the center point of the structure is fed by a buffered clock signal. This then fans out to the center points of the four quadrants of the chip where another buffer drives another, smaller structure. This continues for several levels of progressively smaller structures until a local buffer drives the clock sinks. Using this methodology with symmetrically sized buffers, each clock signal has traveled the same distance and experienced the same insertion delay. Thus, most trees aim to eliminate the difference between the routing lengths for each leaf node. However, skew will still be present because of load mismatch at the clock sinks and process variations in the metal lines and in the drivers [10].

In unbuffered tree clock distributions, the metal widths of each successive clock layer must be tapered to minimize reflections of a high speed clock signal at the branching points [10]. This can be seen in Fig. 12c.

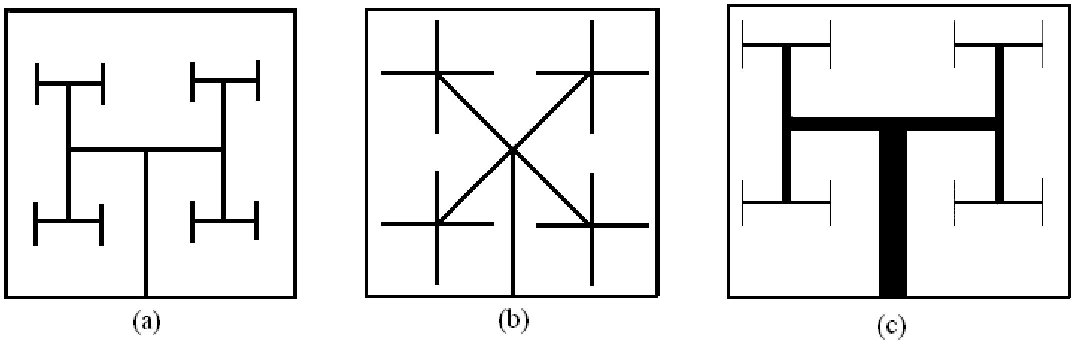


Fig. 12: H-tree (a), X-tree (b), and Tapered H-tree (c) unbuffered clock distributions

3.1.2 Geometric Load Matched Trees

Another approach to construction low skew clock distributions involves matching the geometry of the network to the clock loads present in the design. These networks are designed by a bottom-up algorithmic approach as opposed to a top-view global approach [10]. Typically, these are constructed using buffered binary trees. To minimize skew, two clock sinks are chosen and the point between the two is calculated where the load from the sink and the clock routing is equal on both sides. At this point, a buffer is placed and this point become an effective clock sink for the next highest clock level. In this way, a load-balanced tree is recursively built to minimize skew. This step-by-step process can be seen in Fig. 13. This method is popular in the automatic synthesis and layout of clock network which is typically seen in commercial place and route CAD tools for ASIC designs.

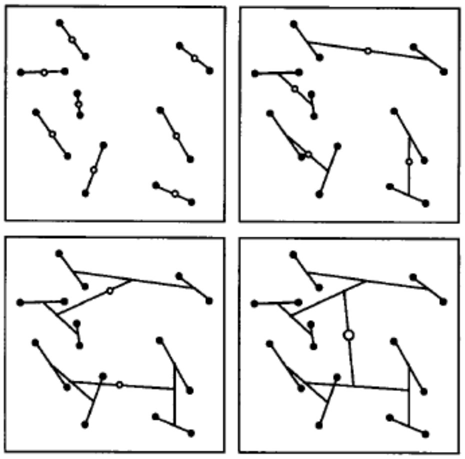


Fig. 13: A geometric load matching clock distribution [10]

3.1.3 Clock Meshes

When low skew in a design is paramount and power is less of a concern, clock meshes (also referred to as grids) are often employed. The benefits are obvious: a mesh can be thought of a tree where many redundant, skew reducing connections are made between the clock nets at the lowest level. Thus, the clock mesh is a signal net that spans the design supplying the clock. Because the mesh is one net, this is known as equipotential clocking; every point on the grid is at the same electrical potential. A block diagram of a mesh distribution can be seen in Fig. 14 while a more physical representation is found in Fig. 15.

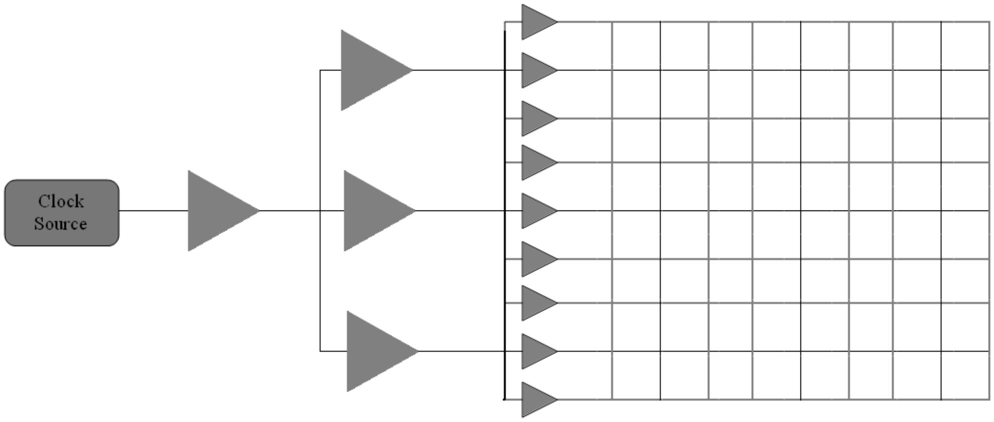


Fig. 14: Block diagram of a mesh clock distribution

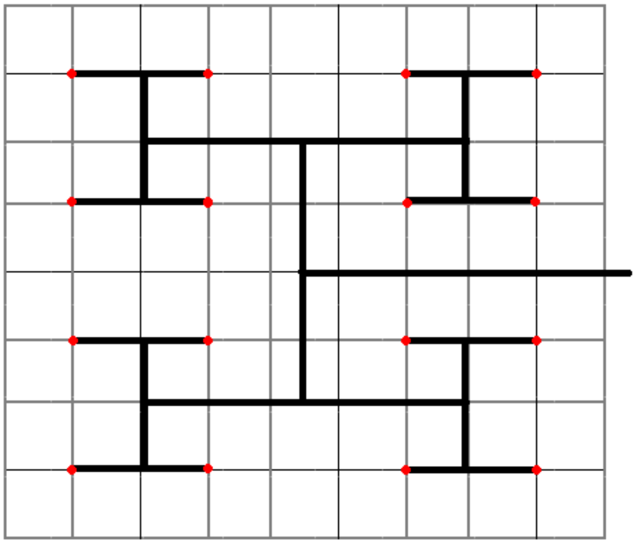


Fig. 15: Top level view of clock mesh and driver routing

As you can see in both Fig. 14 and Fig. 15, a structure that resembles a clock tree is necessary to distribute the clock signal to the mesh evenly. If the clock signal was injected into the mesh at one point, there would be significant clock skew even considering the much lower resistance of the mesh as compared to a single clock route. It is the redundant connections of the mesh as well at the small clock tree that make clock meshes low skew, but also add to their high power consumption. The amount of added routing is considerably high in a clock grid, and as such they are used only when low skew is more important than low power.

3.1.4 Hybrid Clock Distributions

As if often the case when two competing solutions to a problem each have strengths and weaknesses, it can be advantageous to develop a hybrid clock distribution network that seeks to exploit the strengths of both a mesh and a tree based distribution while avoiding the costs. These types of hybrid networks are referred to as Mesh + Local Trees (MLT) and Tree + Local Meshes (TLM) [4].

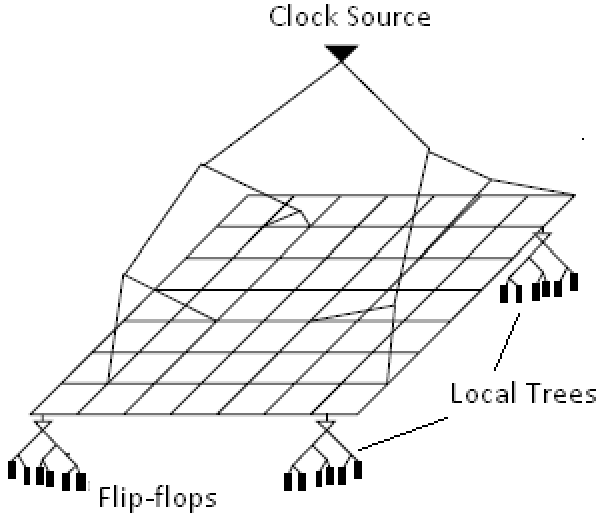


Fig. 16: Mesh + Local Trees (MLT) hybrid clock distribution [4]

The MLT hybrid architecture, as seen in Fig. 16, ensures low global skew across the chip. At the lower levels, the lower power tree is preferred here because the distance between points is not large enough to cause considerable skew.

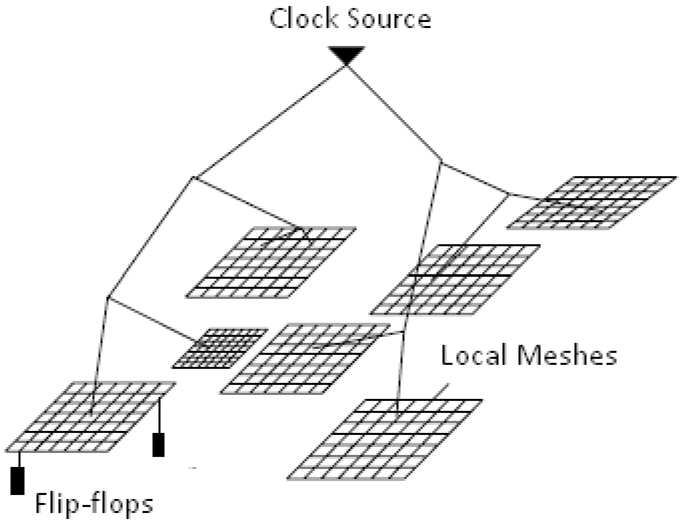


Fig. 17: Tree + Local Meshes (TLM) hybrid clock distribution [4]

In the TLM hybrid architecture which can be seen in Fig. 17, a global tree supplies a lower power global clock to local meshes. This ensures that data registers that communicate directly can do so with minimal skew. This allows for power saving in the global distribution because data signals that will travel larger distances may be pipelined anyway, so low global skew would not be as detrimental to performance.

3.1.5 Example Clock Skews

There are no hard and fast rules about how much skew a "good" clock distribution network contains. Allowable skew is a function of the slack available between two points in a timing path. If the clock period in a design is considerably longer than the propagation delay through the critical path, there is a large slack and a larger clock skew is acceptable. However, in high-performance designs that are clocked within tight tolerances, a large skew could lead to instability where the proper values are not captured by the target flip-flops. Additionally, skew is not always harmful. Because it is deterministic, the knowledge that the clock signal arrives later at a given flip-flop can be used to relax timing constraints on the paths that feeds it. This is called beneficial skew and can aid in timing closure.

So while there are no guidelines on how much skew a design can have, it is helpful to look at a sampling of designs to see typical skew numbers, as shown in Table 4. These have been taken from a number of sources representing a wide range of technologies, architectures and clock distribution networks. These were previously compiled and cited in [3].

Table 4: Summary of High-Performance Clock Distribution Networks

Microprocessor	f_{clk} (MHz)	t_x (10^6)	V_{dd} (V)	L (µm)	Area (mm^2)	Clock Distribution	Global Skew (ps)	% Skew
Alpha 21064	200	1.7	3.3	0.75	234	Binary Tree, Grid	293	5.86
Alpha 21164	300	9.3	3.3	0.50	299	Binary Tree, Grid	80	2.40
Alpha 21164	433	9.6	2.0	0.35	209	Binary Tree, Grid	90	3.90
Alpha 21264	600	15.2	2.2	0.35	318	X-Tree,H-Tree, RC tree, Grid	72	4.32
Alpha 21364	1200	152.0	1.5	0.18	397	X-Tree, H-Tree, Grid, Spine	90	10.8
Pentium II	300	7.5	2.8	0.35	203	-	120	3.6
Pentium II	450	7.5	1.4-2.2	0.25	131	Binary Tree, Spine	15	0.68
Pentium III	650	9.5	1.4-2.2	0.25	123	Binary Tree, Spine	15	0.98
IA-64	800	25.4	-	0.18	-	H-Tree, Binary Tree, Grid	28	2.24
Pentium III	1000	28.0	-	0.18	-	-	35-70	3.5-7
Pentium IV	2000	42.0	-	0.18	-	Binary Tree	16	3.2
S/390 G4	411	7.8	2.5	-	300	H-Tree	30	1.23
S/390 G5	609	25.0	1.9	0.25	215	H-Tree, Grid	12	0.73
S/390 G6	760	25.0	1.9	0.20	215	H-Tree, Grid	12	0.91
PowerPC	1150	19.0	1.9	-	-	H-Tree, Grid	15	1.72
Power4 1300	1300	174.0	-	-	-	H-Tree, Grid	25	3.25

f_{clk}: Clock Frequency **t_x: Transistor Count** **V_{dd}: Supply Voltage** **L: Channel Length**

It is clear from the table above that percentage skew varies greatly depending on the design. This can be seen further in Fig. 18, where percentage clock skew versus clock frequency is plotted. While the data points are spread greatly, a line is fit to the data to show the general trend: as processor clock frequency increases, global skew also tends to increase even as more advanced clock distribution schemes are used. This certainly leads to problems as we have discussed that higher clock frequencies make skew an even larger issue.

The data plotted in Fig. 18 shows us that while there is a large variation between the skews present in different microprocessor designs, values in the range of 3-6% are expected for clock frequencies between 250 and 2000 MHz.

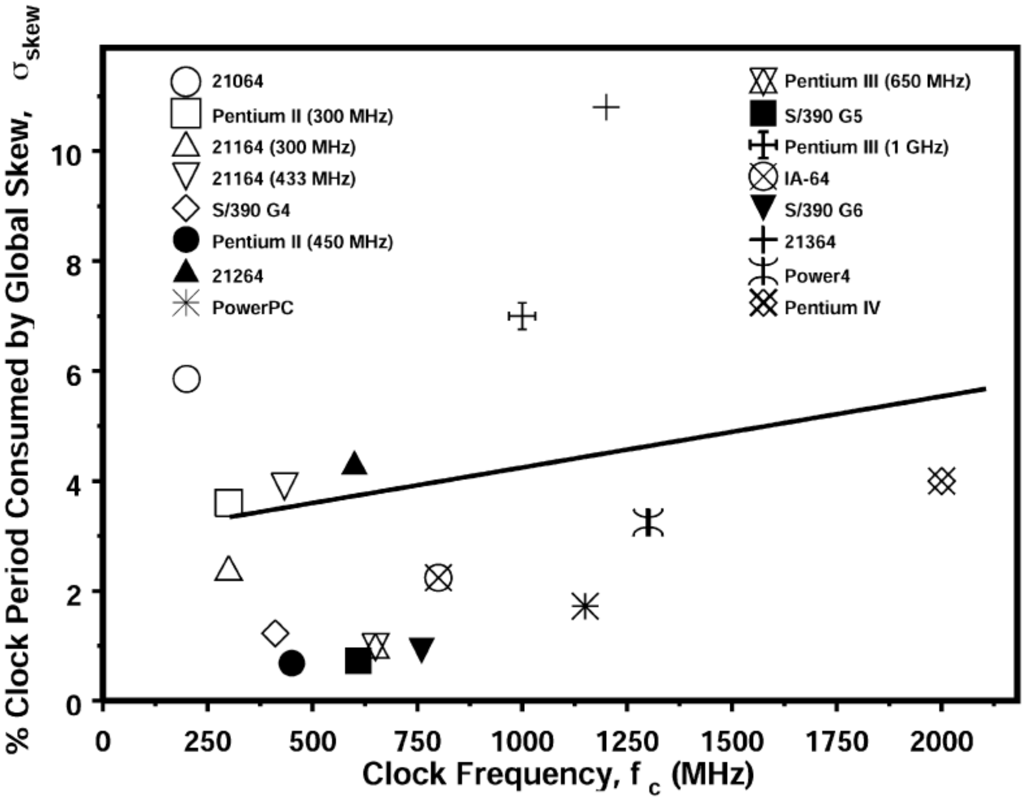


Fig. 18: Percentage clock skew versus clock frequency of microprocessors in Table 4 [3].

3.2 3D Clock Distributions

All of the issues that face clock distributions on 2D chips also apply to 3D clock distributions. For this reason, the development of a suitable 3D distribution is merely an effort to adapt well-known 2D distributions to the 3D problem. Many different topologies have been implemented, analyzed, and found to have varying strengths and weaknesses. From what has been seen in 2D clock distributions, it is unlikely that one topology or structure will dominate 3D clock designs. Rather, the appropriate structure will be used when its benefits outweigh its costs. We will examine, on a high level, a sampling of proposed 3D clock distributions and evaluate their respective effectiveness.

3.2.1 Full 2D Clock Distributions on All Dies

The simplest means of distributing a clock signal among multiple stacked dies with low skew is to construct a full 2D clock distribution on each die and feed the clock roots of each by a common TSV. This would eliminate inter-die dependence when calculating clock loads and synthesizing the clock tree. This is referred to in [24] as "Replica Technology" and can be seen in Fig. 19. Because of varying thermal profiles between the dies, while skew on each die may be low, the possibility exists for the skew of the die stack to be considerably higher. Also, it should be obvious that this method of replicating exactly the 2D clock distribution network on each chip will occupy considerable routing resources and, as such, will consume more power than a distribution network that takes more advantage of 3D vias.

Because this is just a replication of 2D clock distributions, there would be almost no changes that would be made to the clock synthesis flow. Also, testing methods that are currently in place could be used for prebond testing of dies.

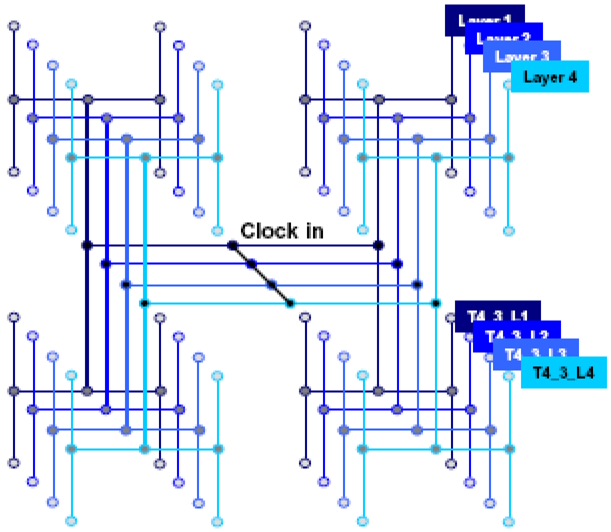


Fig. 19: Replica Technology where a full H-Tree is present in each layer

3.2.2 Driver Layer with Sparse or No Local Networks

The alternative to a full clock distribution of each die is one where a large, global network exists on one die and TSVs bring the clock signal down through adjoining dies to their sinks. This is referred to in [24] as “Via Technology” and can be seen in Fig. 20. Because the clock distribution is constructed globally, the skew of the die stack is ensured to be held in tighter control than in Replica Technology discussed above. The clock synthesis is performed using information about the locating and load of all clock sinks in the whole design. From the root, the clock is routed using TSVs and short routes to all sinks in the stack. This introduces much inter-die design dependence. Thus, clock synthesis can only be performed after all final placement and routing is finished.

The benefits are clear, however. It is shown in [24] that this method of 3D clocking consumes 2.35 times less power than the Replica Technology with considerably lower skew under temperature variations.

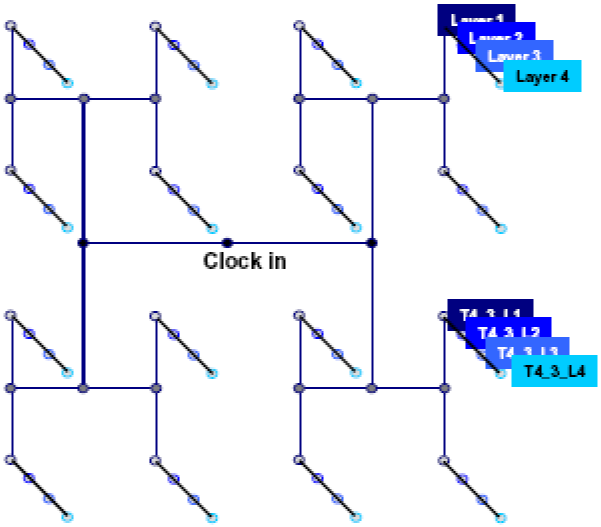


Fig. 20: Via Technology where a full distribution is present in a driver layer and clock sinks are fed directly through TSVs

A similar method of 3D clocking is presented in [42] wherein a dedicated clock layer is manufactured containing PLLs, clock drivers, and clock routing. From here, the clock is routed to leaf nodes through TSVs. This is sensible because it allows for the clock components to be manufactured in a more mature technology that would have higher stability and noise immunity than the latest, greatest, and smallest technology node. A representation of this can be seen in Fig. 21. The benefits in terms of power savings are not a drastic as in [24] at 10%, but this method does result in a 30% decrease in global skew over the 2D network [42].

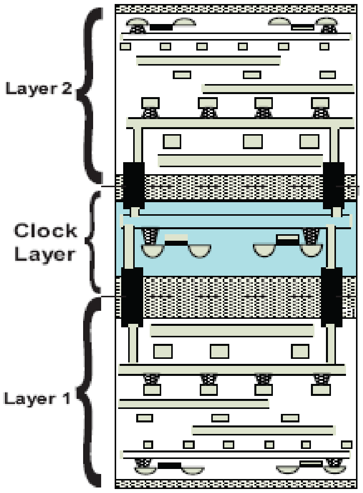


Fig. 21: Two layer stack with dedicated clock layer [42]

The largest issue with this type of 3D clock distribution is that it would make prebond testing of dies incredibly difficult. Before bonding, there are almost as many disconnected clock networks as there are clock sinks. Distributing a prebond test clock to these leaf nodes would incur an inacceptable amount of test routing overhead. While they can been shown to reduce power and skew in operation [42][24], the challenges they present to prebond testing make this kind of 3D distribution infeasible for commercial production.

3.2.3 Larger Local Networks Disconnected Prebond

As with many things we have examined previously, we will again see that the most feasible and advantageous 3D clock distribution is a compromise between the highly redundant distribution detailed in Section 3.2.1 and the sparse and untestable topology covered in Section 3.2.2. This is an organization where each die in a stack has at least their local or regional clock networks completely connected.

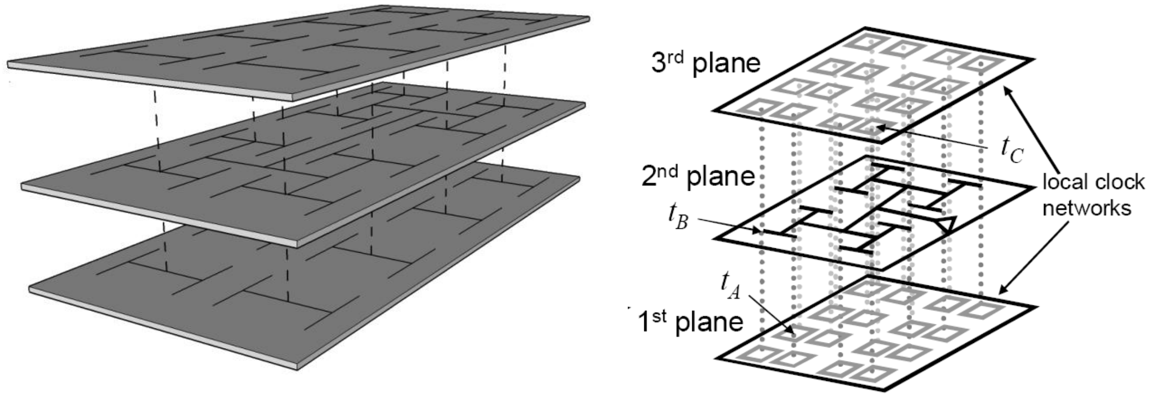


Fig. 22: Prebond dies each local networks sharing one global network [43]

It has been shown that eliminating redundant global clock wiring can reduce power compared to full distributions on each die [24][42][46], but networks that are too sparse on a prebond die cannot be tested which leads to poor yield and high manufacturing costs. The result is that each die must contain some local networks which can be tied together during testing so all sinks receive a clock signal with low global skew. This is an active area of research and one of the major components of this thesis. It should be noted that, while shown in Fig. 22, there is no need for one die to have a complete clock distribution network. The only criterion is that when the die stack is constructed, a full 3D distribution is present. Imagine a case where the clock generation circuitry, namely a PLL and supporting logic, is fabricated on one die in an older technology node that has more stability and noise immunity for the analog circuitry. In this same stack, the global clock drivers could be fabricated on another die in a more recent technology where the transistor drive strength is greater. In this arrangement, no one die would contain a complete distribution, but when the stack is bonded a complete 3D clock network is realized with a higher quality end product.

CHAPTER 4

INTEGRATED CIRCUIT TESTING

Before an integrated circuit can be sold, it must go through two very different but lexically confusing processes known as verification and testing. Verification occurs during the design process to ensure that that circuit to be manufactured will implement the desired functionality. Once this is complete, the chip is taped out and fabricated. Before packaging, the die must undergo testing to expose any issues that may have arisen during manufacturing that will cause the design to function incorrectly. Thus, if verification does not expose a design flaw that is present, every chip that is manufactured will not operate as desired. If no testing is performed, however, a product may or may function properly. While both verification and testing aim to expose functional problems with the design, the process and economics of each is quite different. A design only has to be verified once, assuming the design is correct. However, each die that is manufactured must be tested. The result of this is that test economics become an absolutely dominating factor in making a process commercially viable. It is for these reason that Design for Testability (DFT) has become such an important part of the design process in the last two decades.

In this chapter, we will examine past and current methods of integrated circuit testing and how these affect the testing of 3D integrated circuits. Before the methods can be investigated, however, it is important to examine the driving factors behind the DFT choices we make by discussing test economics.

4.1 Test Economics

A discussion of economics is invariably a discussion of money. Integrated circuit testing is no exception to the clichéd adage that time is money. The cost of testing for high volume production is staggering: it is estimated that some 25% of the total product cost is spent on testing [32][12]. Thus, it is essential that integrated circuit testing be as efficient as possible to minimize costs while ensuring that all possible manufacturing defects are exposed at test time.

The cost of testing relies on a large number of factors and almost all of these factors can be varied to achieve an optimized system. As mentioned above, test time has a large effect on the cost of testing; the faster units can be tested, the fewer testers are needed to maintain the same product throughput. However, if the actual testing of a chip takes 2 seconds and it takes one second to move on to the next chip, that tester is only running at a 66.7% utilization. When fixed costs are taken into consideration, it is essential that testers are running as close to 100% utilization as possible. This can be seen very neatly in Fig. 23 where it is shown is that the test time of a chip is only one factor associated with the cost of test. However, it can be one of the largest factors because of its impact on other costs such as the number of testers or the up-time of a test facility [1].

Another more indirect cost of testing is that of power dissipation. When testing thousands of fabricated designs, the power dissipated on chip will certainly have a measureable cost, but when compared to the power consumed by the tester or the test facility itself, this amount will be small. However, minimizing test power is essential to proper IC testing. When a chip is packaged and implemented into a system, a robust power delivery system must be in place for the circuit to function properly. Many pins must be supplied with clean, low-noise, low resistance and inductance traces with decoupling capacitors to help prevent against voltage spikes and drops. However, in a test environment, power must be supplied through small probes that make momentary mechanical contact with probe pads on the System Under Test (SUT).

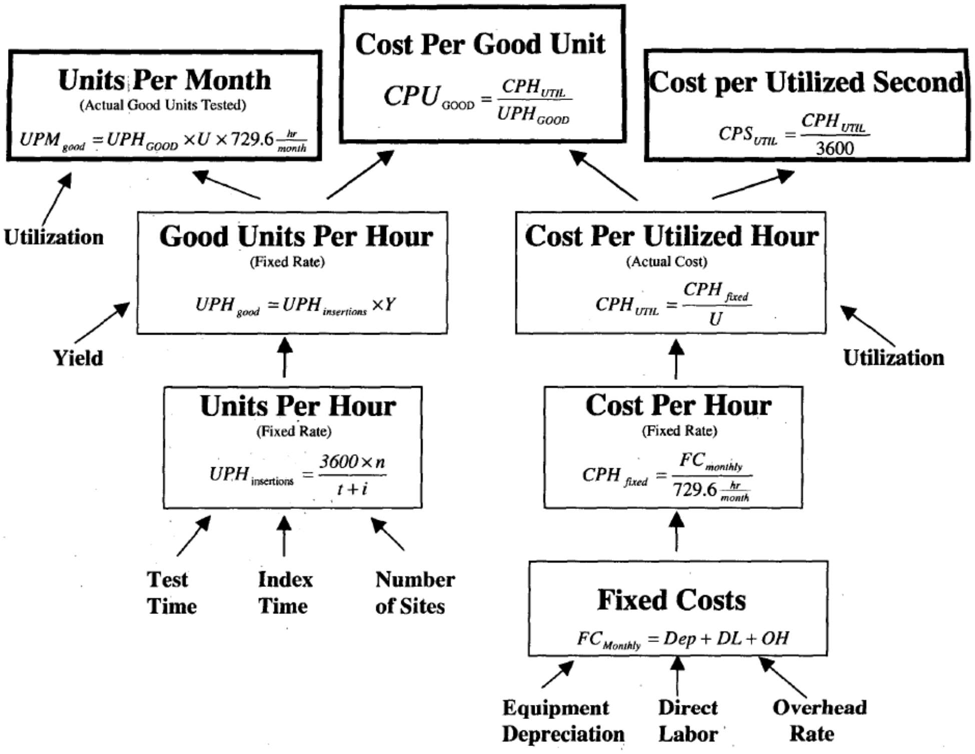

Units Per Month
(Actual Good Units Tested)
$UPM_{good} = UPH_{GOOD} \times U \times 729.6 \frac{hr}{month}$
Cost Per Good Unit
$CPU_{GOOD} = \frac{CPH_{UTIL}}{UPH_{GOOD}}$
Cost per Utilized Second
$CPS_{UTIL} = \frac{CPH_{UTIL}}{3600}$
Utilization
Good Units Per Hour
(Fixed Rate)
$UPH_{good} = UPH_{insertions} \times Y$
Cost Per Utilized Hour
(Actual Cost)
$CPH_{UTIL} = \frac{CPH_{fixed}}{U}$
Utilization
Yield
Units Per Hour
(Fixed Rate)
$UPH_{insertions} = \frac{3600 \times n}{t + i}$
Cost Per Hour
(Fixed Rate)
$CPH_{fixed} = \frac{FC_{monthly}}{729.6 \frac{hr}{month}}$
Test Time
Index Time
Number of Sites
Fixed Costs
$FC_{Monthly} = Dep + DL + OH$
Equipment Depreciation
Direct Labor
Overhead Rate

Fig. 23: Flowchart detailing the costs of testing integrated circuits [1]

Because of contact resistance between the probes and the landing pads, these electrical contacts cannot supply excessive power to the SUT. This is one reason why power is limited during test.

Another reason is that dies cannot be properly heat sinked until they are packaged. This leads to an increase in on-die temperature during testing. Because more logic is active during test than in typical operation [33], even a short test time will still result in significant temperature increases.

While there are certainly many factors that influence DFT, the two key players are power and test time minimization. Because these can be controlled within a design, for the purposes of this work, we will focus on these factors chiefly and disregard the influence on test infrastructure and utilization when discussing test costs.

4.2 Scan-Based Testing

For many years, most of the testing on integrated circuits was conducted using functional-based tests [30]. In this type of testing, typical patterns are applied and the response is collected and analyzed. If the expected result is produced, the system has passed the functional test and is considered to be fault free. While this may seem like the most sensible means to test a system – with real patterns performing typical operations – many challenges face functional testing in a commercial environment. For one, it can take months to develop tests that will adequately and efficiently test a system. This test development time is almost always an unacceptable increase in Time To Market (TTM). In addition, in order to perform at-speed tests with full-pinout, very complex and expensive testers are required.

An alternative to functional-based testing is structural testing. Instead of supplying real inputs to a system, a set of test vectors is developed that will have maximal coverage of faults based on a given fault model. Thus, these patterns may not replicate inputs that a system would see in operation, but will expose any manufacturing defects present in a system. A very fast and effective way to implement structural testing is referred to as scan-based testing.

Scan testing is used to place known values in flip-flops so precise test vectors can be run through the system. Without the ability to set the initial conditions of the test, there would be no way to determine the expected response. Thus, flip-flops are equipped with logic that will allow a value to be scanned in and out when a scan enable signal is high. In this way, flip-flops are strung together in a chain that allows a long test vector to be scanned into the chain to set their initial states. Then, the scan enable signal is deasserted and two clock pulses are sent to the system. The first is referred to as a launch pulse which effectively applies the scanned-in patterns to the system, and the second is a capture pulse that latches the resulting data back into the flip-flops. This result is then scanned out the chain while other patterns are scanned in for the next test vector. A representation a scan chain implementation can be seen in Fig. 24.

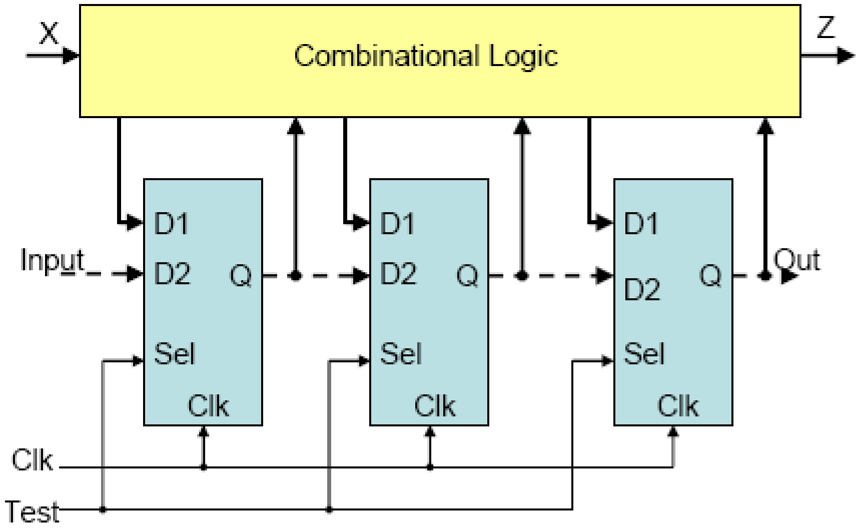


Fig. 24: A representation of a scan chain used for structural testing [45]

Because scanning in patterns causes more logical transitions than during typical use, the speed at which the inputs are scanned in must be considerably slower than the system clock to stay within a test power envelope [33]. While the scan clock speed depends on the system, power envelope, and design team, it is generally in the 100-400 MHz range. Because the patterns must be scanned for as many clock cycles as there are flip-flops in a scan chain, the length of each chain is determined by optimizing test time versus number of probes available. Obviously fewer, longer scan chains would occupy more time scanning in and out patterns but have the advantage of fewer necessary probes.

Scan-based testing allows for structural testing at-speed (assuming the launch and capture pulses are at the functional clock speed), a simplification of testers because they do not have to directly probe buses, and better test coverage because it can detect more defects in fewer patterns than functional testing. Because of these reasons, scan testing is very popular in the testing of ASIC and large custom designs.

4.3 Built-In Self Test (BIST)

In an effort to make IC testing more efficient and less expensive, Built-In Self Test (BIST) is gaining ground as a promising means of reducing tester complexity and speeding up test time. Originally, BIST was developed to perform in-field testing of logic, but its use in post-manufacturing test has been increasing [12].

Most BIST systems are based on an underlying scan methodology, with test patterns being generated on chip. These are typically pseudorandom patterns that are scanned into the flip-flops and applied to the design in an at-speed test. The result is then scanned out and the response is compressed. After many responses are collected, a signature remains which can be scanned out or compared internally to verify proper circuit functionality. Generating the test vectors and compressing the responses on chip allows for less complex testers that can oversee the test process on many SUTs at once. A representation of a generic scan-based BIST architecture can be seen in Fig. 25.

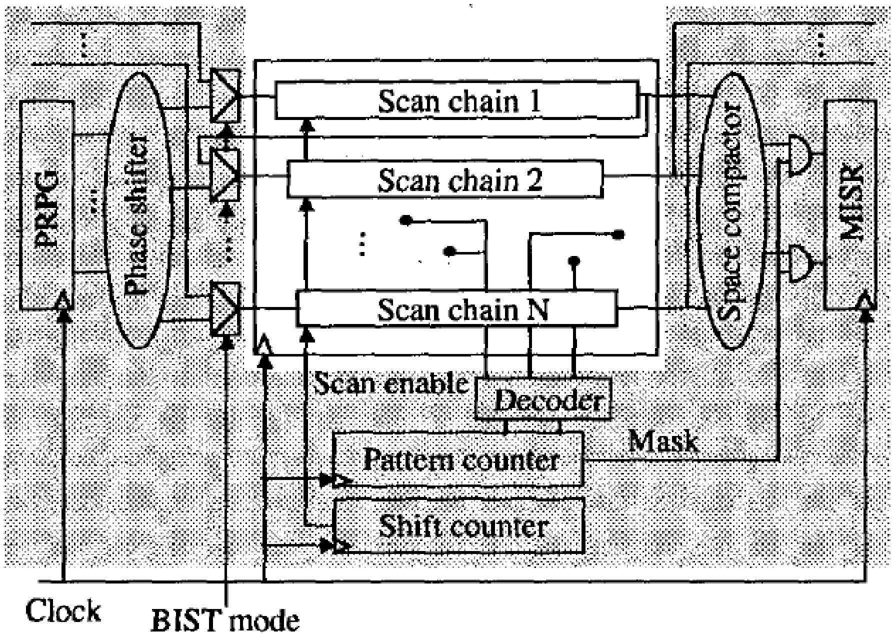


Fig. 25: BIST architecture using a generic PRPG for test vector generation and a MISR for signature generation [12]

While it appears that BIST will be increasing implemented in large commercial circuits, because it is based on the typical scan test methodology, we will not be concerned with a BIST implementation. It is shown here merely for informational and contextual purposes.

4.4 3D Integrated Circuit Testing

Testing 3D ICs poses some interesting challenges resulting from the fact that each die does not contain a complete system. For example, in a somewhat fine partitioning granularity, one-half of an ALU may reside on one die while the rest is located on an adjacent die. While we discussed earlier that such a partitioning granularity is unlikely for reasons we will see soon, it illustrates the unique challenges of testing an incomplete die. Certainly, if the design is not complete at the time of testing, functional testing is not a valid option. If structural testing is used, the incompleteness of the prebond system will not be an issue. From the previous sections, it is clear that a scan-based test methodology should be used here.

4.4.1 3D Scan Chains

Some discussion is necessary to address the potential differences between a scan chain in a typical 2D design and that in a 3D integrated circuit. In order for dies to be tested prebond, there must be a scan enabled flip-flop at every signal TSV. This is essential to be able to ensure a known value for all inputs and to be able to capture and analyze all results. Because of this, we say they are “flop-bound.” This is why in Section 2.3.2 we discussed that very fine partitioning granularities would be unrealistic because of testing concerns. In such a partitioning, the number of interdie connections would be so large that more transistors would be used implementing the TSV flip-flops than the logic they interface with. An ideally-partitioned design would place as many already in place flip-flops at die boundaries to minimize the number of extra TSV flip-flops required. So, the first difference between a 2D design and a 3D design in terms of scan chains is the number of flip-flops required.

Early works in 3D scan chains proposed methods for building chains based on current methods to minimize scan routing [45]. They propose three methods to connect scan flip-flops between dies. The results of these methods can be seen in Fig. 26. In the leftmost scan chain ordering, dies are treated individually and their scan chains are connected with a single TSV. In the middle ordering in the figure, all scan cells are mapped to a 2D space where scan chain ordering is determined. The third method takes into account TSV length and number. This is their optimized approach and suggestion for scan chain implementation in 3D ICs.

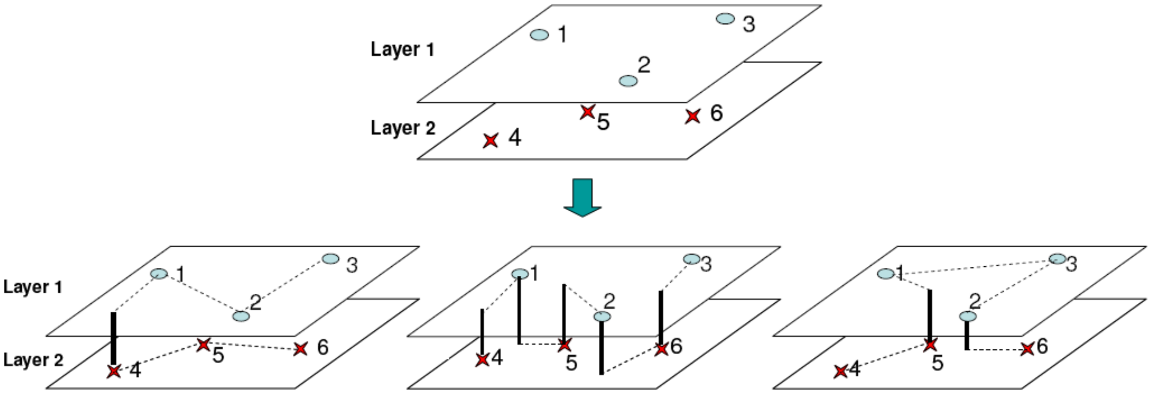


Fig. 26: Three methods of scan chain ordering in a 3D integrated Circuit [45]

However, their analysis is only valid for postbond testing. That is, in the second two scan ordering methods, the scan chains will not be complete until the dies are bonded. While it will still be necessary to test the die stack, it is prebond testing that will allow for commercial adoption of 3D ICs. Thus, it seems clear that out of these three scan ordering methods, the only acceptable one is the first. Scan chains would be constructed independently on each die, then either tied together or just broken out to a probe point for postbond testing. Thus, it seems that very few changes would need to be made to the 2D scan chain design in order for scan to be implemented in a 3D design.

4.4.2 Testing 3D Clock Distributions

We have examined in Chapter 3 the differences between a 2D clock network and a 3D distribution. We will now go further into detail about the challenges of testing a prebond die with a 3D clock distribution.

As with a finished design, during testing the clock signal must reach all clock sinks with low skew – this is essential for at-speed testing. However, this can be a problem if a die is made with a low power clock network that is only complete after it is bonded with other dies. The easiest solution would be to supply a clock signal to each clock TSV from the tester, thus connecting all clock regions. However, the size of a TSV is too small to probe using conventional probe cards [11]. In order to probe such a small contact, the probe must have a very small diameter. However, very small probes suffer from low mechanical strength and do not survive many landings. Constantly replacing delicate probe cards would not be acceptable. Thus, a method for distributing the clock to all disconnected local clock networks with low skew during testing using only a small number of probes is necessary. In a typical design, only one clock signal is supplied to the die. However, there is no reason this could not be increased if it were advantageous.

At this time, there has been only one published work that addresses the problem of incomplete clock distributions on a prebond 3D die. In [46], a redundant global tree is fabricated on each die that would otherwise have had an incomplete clock network (as seen in Fig. 27). This redundant tree is only connected to the rest of the clock distribution during testing. The authors report good results, but they are based on a few unrealistic assumptions. First, they assume that TSV density will be very high, where under the worst circumstances one clock TSV supplies approximately 12 clock sinks. Realistically, this number will be much closer to a few hundred. Also, the power savings they report from their sample clock distributions are based on very small benchmarks where the top two or three layers of clock distribution account for a considerable percentage of the total clock routing length. From their results, it is clear that power savings diminish as design size increases, so it would be interesting to see how their solution fares on larger designs.

In the following sections, we will work towards a testable 3D clock distribution suited for large designs that does not rely on duplicating the global clock distribution on each prebond die.

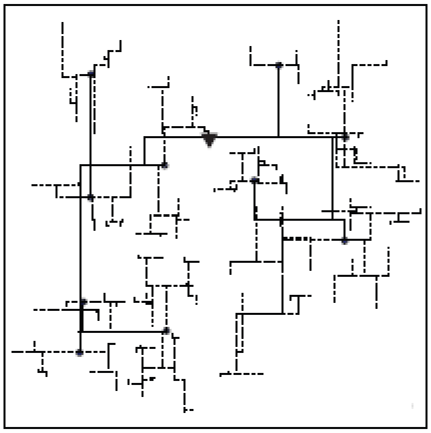


Fig. 27: A testable 3D clock distribution using a redundant tree. The redundant tree is shown as a solid line while the remainder is dotted [46]

CHAPTER 5

BUILDING A CLOCK SIMULATION MODEL

The purpose of this chapter is to detail the development of a simulation model of a 3D clock distribution network. The goal is to produce a model of a realistic clock network upon which experiments can be carried out to determine the effect of various design decisions. Instead of carrying out analyses on small benchmark circuits, it was decided to take a large unit of an available open-source microprocessor and develop our model around that. The method in which the clock model was constructed from the Verilog description of the processor follows.

5.1 Choosing an Appropriate Design

The techniques used to evaluate the methods of prebond testing of dies will be carried out in the manner of a case study. As a result, we will be focusing on making one specific circuit testable and applying the results of that analysis to the general case. The reason for this is in order to test one method across many types and sizes of circuits, many different tests must be run. However, in order for this to be feasible, the size (in terms of gate or transistor count) of these test cases must be small in order to have reasonable runtimes. As you will see in the next few sections, there are numerous steps required to produce a clock model. Unfortunately, one can only extrapolate the results generated by a sampling of small circuits to a larger one. As we have seen in [46], skew and power gains diminish as design size increases. For this reason, we need to do our analysis on the largest design that still has reasonable runtimes.

In choosing a design to study, we looked for an available open-source microprocessor. One was found in the OpenSPARC T1 64-bit microprocessor released by Sun Microsystems under the GNU General Public License [27]. Because this is a very large design, SimplyRISC released the S1 Core; the RTL of one of the eight cores in the T1 processor. It was determined that this would make an acceptable candidate for study. The Verilog was synthesized using Synopsys Design Compiler in the 45nm node using Nangate's Open Cell Library developed from North Carolina State University's FreePDK [26]. Synthesis was run on the S1 Core for seven days before it was terminated by the user. At this point, it was determined that a smaller design partition must be chosen in order for processing times to be reasonable. The RTL for the S1 Core was partitioned into five blocks of varying sized. One block of a medium size that was synthesized before the process was terminated was the Trap Logic Unit (TLU) that consists of trap logic and program counters, and is responsible for handling traps generated internally and external interrupts. The TLU of the S1 Core contains 14,607 clock sinks and 937,965 transistors and as such is much bigger than available benchmarks. While not on the scale of a large microprocessor, a design this size allows us to produce meaningful results in a reasonable amount of time.

5.2 Assumptions

In order to develop a clock model for a 3D design, some initial decisions are made that shape the model based on assumptions on the way in which a 3D stack is constructed. Before delving into the specifics of the model development, we will outline and justify the assumptions we have made.

5.2.1 Die Configuration

Because the focus of this thesis is to enable testing on a single prebond die, the analysis and clock modeling is confined to a single die. A model of a clock distribution network on a single prebond die that is not fully connected is first produced. We then add the means to connect these disconnected networks and compare the results to the same design if it had a full tree distribution. This is a fair comparison because a full tree distribution on each die is the easiest and most basic 3D clock distribution scheme because it matches the standard 2D scheme exactly. Thus, we compare the in-operation power savings and skew during testing to the reference case of a full clock distribution. Because we are using a partition of a design for our experiments, we can assume that the TLU of the S1 Core has been given its own die for partitioning purposes.

5.2.2 3D Clock Network

Because all clock sinks must be connected during testing, we have ruled out a 3D clock distribution with one main global network on a die that feeds clock sinks on other dies directly as in Section 3.2.2. A full clock distribution would have to be implemented on each die and enabled only during testing to make this type of 3D clock distribution testable before bonding. This would be a waste of routing resources and would increase coupling capacitance during operation, lowering performance. Thus, we consider the case more similar to the clock network presented in Section 3.2.3 where the clock sinks on each die are at least connected by local networks. This ensures that test routing needs to connect a much smaller set of points to allow for prebond testing. The size and number of these local networks is something that can be optimized for power and performance. It is also a function of TSV size and density, which will be discussed in Section 5.2.3.

Furthermore, we will assume that clock synthesis is done on the entire design with knowledge of the 3D partitioning and placement. That is, when all dies are bonded in a stack, the skew across the whole stack is low. From a prebond testability perspective, we focus solely on ensuring the skew is low during testing. In this way, we can take the position that a 3D clock distribution had been completed and we are handed a die with an incomplete clock network. The task we are charged with is to connect these networks with low skew during testing without appreciably affecting performance during operation.

Also, we assume that clock TSVs feeding another die are driven very close to the TSV and that a clock driver is located very close to the TSV on the receiving side as seen in Fig. 28. This minimizes the clock load between dies.

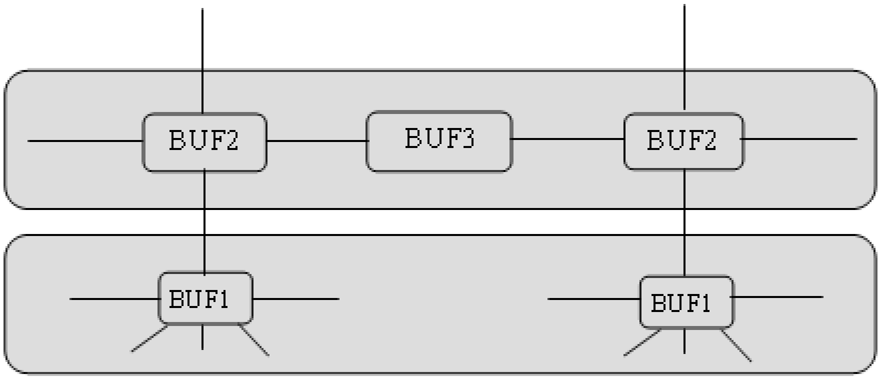


Fig. 28: Two die stack with buffers bounding TSVs to reduce interdie loading

5.2.3 TSV Size and Density

The density of TSVs has a great impact on the benefits we can get from 3D integrated circuits. Because size influences pitch and pitch begets density, we will mostly refer to TSV size when discussing interdie connections. While a TSV may be either round or square in cross section, we will refer to them by their dimensions as if they were square. This is easiest because where wires are routed orthogonally a round or a square TSV effectively take up the same space on the die, with the added benefit that a square TSV has a greater cross sectional area.

As TSV size affects the number of signals that can be used to route the clock signal among dies, it will greatly affect not only the performance of data communications, but also the structure of the 3D clock network. In order to take the most advantage of 3D integration, we will use as many clock TSVs as are available. We have limited this amount to 10% of the total number of TSVs. This is because approximately 30% will be necessary for power [13], and we want to ensure there are plenty remaining for data signals.

Using a complete tree clock distribution, we perform a cut at a given clock level to produce a lower power 3D distribution on one die. The places where the cut is performed will be driven by a clock driver from an adjacent die. In this way, we can examine the effect of TSV size on power saving. Because our clock network was driven by inverters to improve duty cycle, if the network was cut at an even level, an inverter would be necessary at each clock input to have the proper clock polarity. Because the number of nodes at a given level of a tree grows exponentially, TSVs must be shrunk considerably before we will be able to perform a cut on the clock tree at a lower level. Because of this, we see diminishing returns in power savings based on TSV size in Table 5.

Table 5: Power Savings as a Function of TSV Size

TSV Size (µm x µm)	# Clock TSVs	Power (mW)	% Savings
20X20	21	95.6	2.4
10X10	87	95.3	2.6
5x5	359	92.7	4.3

The table above is for a design with a size of 610µm x 595µm. As you can see, the number of available clock TSVs increases quickly as the size is decreased. However, the number of drivers at a given node also increases quickly. Each successive TSVs size shown in Table 5 represents one fewer level of clock network on the 3D die. As alluded to before, extra clock inverters were needed for the 10x10 case which accounts for the small change in power savings between that and the 20x20 size. The two to compare are the 20x20µm TSVs to the 5x5µm TSVS. The area of the TSV decreases by 16 while the power savings is not even doubled. This is because it is really only the largest global clock routes that dissipate a disproportionate amount of power. After they are eliminated, the contribution of the thousands of local routes really dominates and removing a dozen or so regional clock routes does not have much of an effect. Also, small TSVs pose huge challenges in terms of bonding alignment. For these reasons, we have determined that at least from a clocking perspective, the ideal TSV size is 20x20µm. While this does lower the number of interdie data connections, we will address this in Chapter 7 with a scheme to increase interdie bandwidth without increasing TSV count.

The choice of 20x20μm TSVs dictates the number of clock TSVs and thus the number of disconnected networks the 3D prebond die will contain. While according to Table 5 only 21 TSVs are available for clock, the clock model we have developed has 24 clock drivers at the 5th level of the clock tree. Therefore, this is a suitable place to make our 3D cut. Thus, the prebond die will have 24 disconnected clock networks that will be driven from an adjacent chip when bonded.

5.3 Model Development

This section will detail the development of the clock distribution model. The goal is to develop a chain of steps that will produce a SPICE file representing a clock distribution network based on the output from commercial place, route, and clock synthesis tools along with designer input about the structure of the 3D network with added circuitry to enable prebond testing.

5.3.1 ASIC Design Flow with Commercial Tools

Beginning with an RTL description of the TLU, Synopsys Design Compiler was used to synthesize the design into the 45nm technology node using Nangate's Open Cell Library. Once mapped, the new Verilog file was moved to Cadence SOC Encounter where the remainder of the work was done.

Following a typical design flow, the synthesized Verilog was given to the tool along with physical and timing libraries for the technology. After the design was floorplanned and the physical dimensions of the chip determined, the routing of power and ground nets was completed. Next, standard cell placement was carried out to arrange the cells in a manner that would minimize routing between the cells. Next, Clock Tree Synthesis (CTS) was conducted on the design. This added clock drivers to the design based on certain parameters given to the tool. For example, in automatic CTS, the user can specify the insertion delay, maximum allowed skew, number of clock levels, and other optional specifications and the tool will build a suitable clock tree. Also, in manual CTS the user can specify which types of buffers to use, the number of tree levels, and the number of buffers in each tree. The tool then builds a clock tree based on this input. In this step, clock buffers are placed in the design and a trial clock route is conducted. In this case, we ran manual CTS on the TLU using inverters for clock drivers to improve duty cycle. If the cell utilization is too high in floorplanninng and placement, these buffers will not be able to be inserted at ideal locations. If this does not cause the CTS to fail because the buffers cannot be placed, it will certainly lead to a higher-skew clock tree. A view of the CTS route can be seen in Fig. 29. Notice the clock signal entering the design from the left-hand side in green and the large global clock routes that can be seen more prevalently. The length of the clock route is roughly proportional to the level of the route.

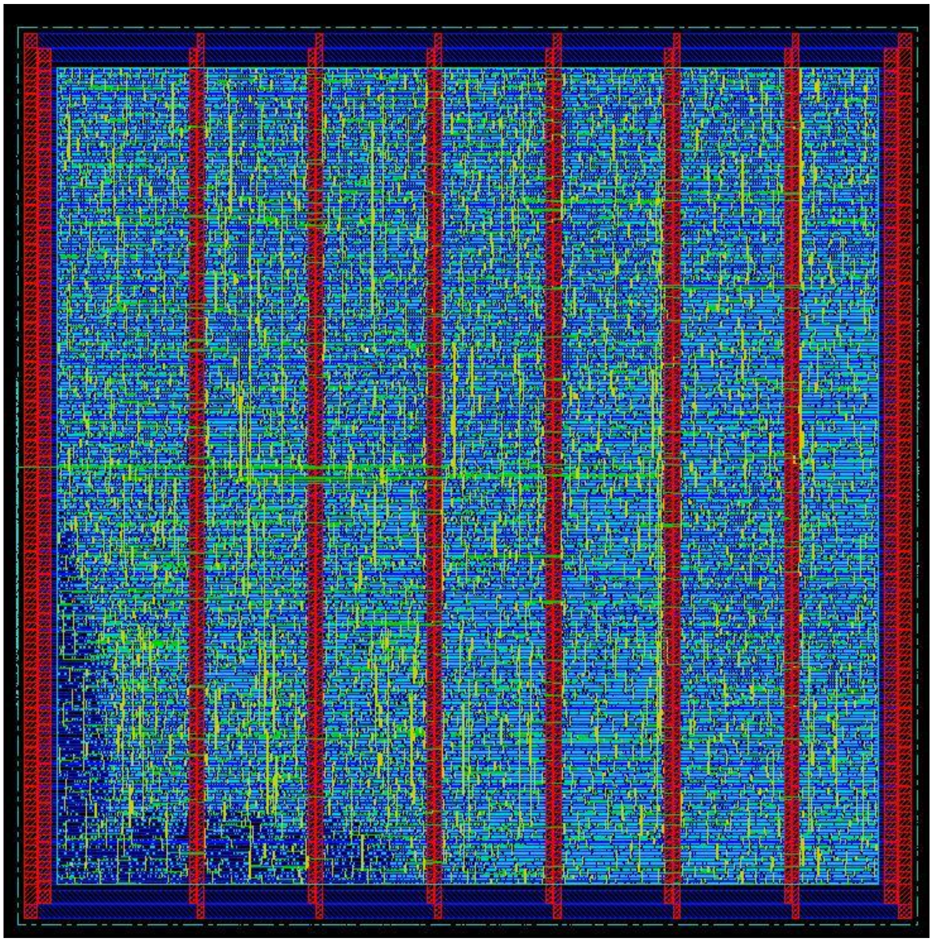


Fig. 29: Post-CTS view of the S1 Core TLU

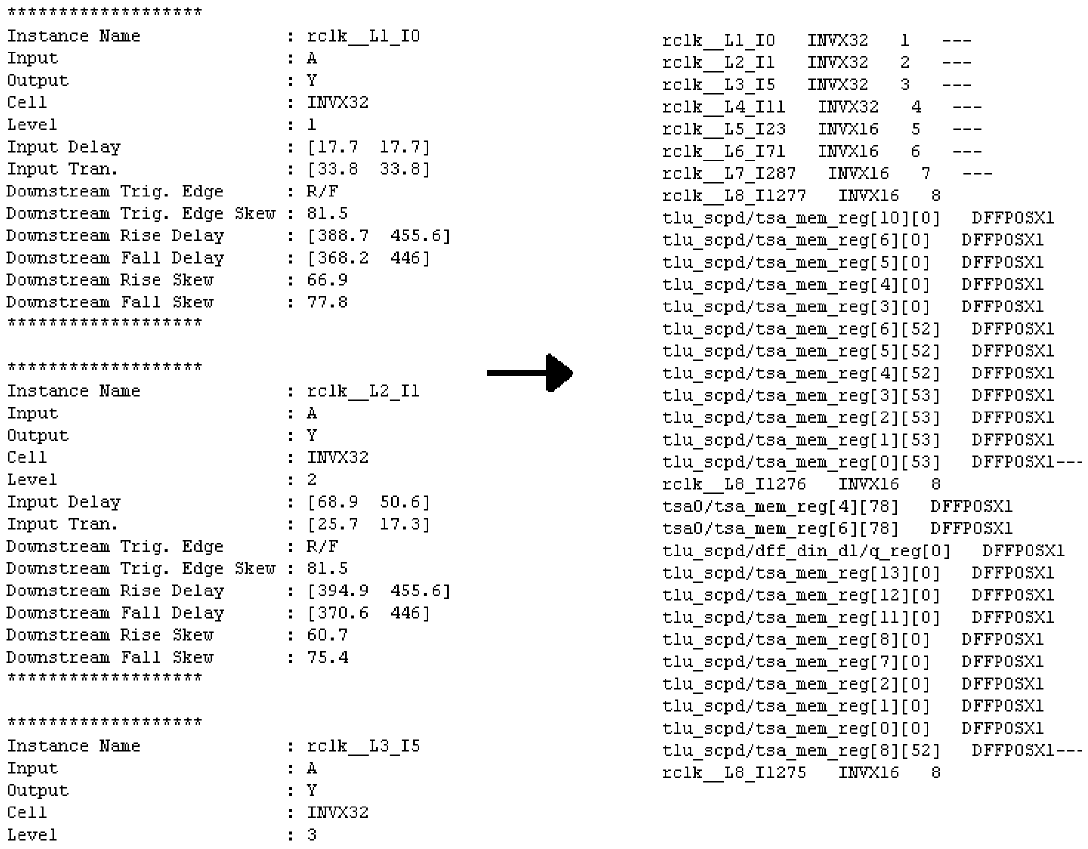
After CTS is conducted, a full route is conducted to make all the necessary connections between standard cells. The time this takes depends greatly on the timing constraints, the effort the user chooses, and the size of the design. In this case, CTS took about 3 hours and final routing about 5-7. After these steps are taken, a timing analysis is performed for timing closure.

5.3.2 Design Parameter Extraction Scripting

Once the steps outlined in the previous section are complete, all the necessary information to build the clock tree simulation model is in place. Encounter contains a tool that will extract a SPICE model for the clock tree. However, not only does this require a noise characterization of the technology library that can only be generated by a proprietary tool, but when the model was generated for the TLU of the S1 Core, it contained over 440,000 transistors and caused the machine on which the simulation was running to run out of memory. This is because the SPICE file included the models of all the flip-flops as well as the clock drivers. It was clear that a simpler model must be generated for simulation to be feasible.

Luckily, Encounter is capable of providing enough information that the clock network can be reconstructed in a simpler fashion. Upon completion of CTS, a clock browser file is generated that contains all the necessary information about the hierarchy of the drivers in the design as well as what type of standard cell they have been mapped to. With the judicious use of Perl scripting, this file can be greatly condensed to just the necessary information as seen in Fig. 30. The output of this script is a simple text representation of the structure of the clock network.

After timing analysis is run, capacitance and resistance information for all nets can be extracted from the design. Two separate lists are created that each contain a net name and its capacitive load or resistance, depending on the output file. A very simple script filters out the unnecessary nets and leaves the clock nets. After this is completed, we are left with three simple text files that allow for the SPICE clock model to be constructed.



```
*******************
Instance Name             : rclk__L1_I0
Input                     : A
Output                    : Y
Cell                      : INVX32
Level                     : 1
Input Delay               : [17.7  17.7]
Input Tran.               : [33.8  33.8]
Downstream Trig. Edge     : R/F
Downstream Trig. Edge Skew : 81.5
Downstream Rise Delay     : [388.7  455.6]
Downstream Fall Delay     : [368.2  446]
Downstream Rise Skew      : 66.9
Downstream Fall Skew      : 77.8
*******************

*******************
Instance Name             : rclk__L2_I1
Input                     : A
Output                    : Y
Cell                      : INVX32
Level                     : 2
Input Delay               : [68.9  50.6]
Input Tran.               : [25.7  17.3]
Downstream Trig. Edge     : R/F
Downstream Trig. Edge Skew : 81.5
Downstream Rise Delay     : [394.9  455.6]
Downstream Fall Delay     : [370.6  446]
Downstream Rise Skew      : 60.7
Downstream Fall Skew      : 75.4
*******************

*******************
Instance Name             : rclk__L3_I5
Input                     : A
Output                    : Y
Cell                      : INVX32
Level                     : 3
```

```
rclk__L1_I0   INVX32   1   ---
rclk__L2_I1   INVX32   2   ---
rclk__L3_I5   INVX32   3   ---
rclk__L4_I11   INVX32   4   ---
rclk__L5_I23   INVX16   5   ---
rclk__L6_I71   INVX16   6   ---
rclk__L7_I287   INVX16   7   ---
rclk__L8_I1277   INVX16   8
tlu_scpd/tsa_mem_reg[10][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[6][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[5][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[4][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[3][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[6][52]   DFFPOSX1
tlu_scpd/tsa_mem_reg[5][52]   DFFPOSX1
tlu_scpd/tsa_mem_reg[4][52]   DFFPOSX1
tlu_scpd/tsa_mem_reg[3][53]   DFFPOSX1
tlu_scpd/tsa_mem_reg[2][53]   DFFPOSX1
tlu_scpd/tsa_mem_reg[1][53]   DFFPOSX1
tlu_scpd/tsa_mem_reg[0][53]   DFFPOSX1---
rclk__L8_I1276   INVX16   8
tsa0/tsa_mem_reg[4][78]   DFFPOSX1
tsa0/tsa_mem_reg[6][78]   DFFPOSX1
tlu_scpd/dff_din_d1/q_reg[0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[13][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[12][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[11][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[8][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[7][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[2][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[1][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[0][0]   DFFPOSX1
tlu_scpd/tsa_mem_reg[8][52]   DFFPOSX1---
rclk__L8_I1275   INVX16   8
```

Fig. 30: Compression of Clock Tree Browser into simpler hierarchy information

A script is used again to generate the SPICE file. This relies on the way the drivers and signal nets are named by Encounter. For example, a clock driver with the designation rclk_L4_I23 is the 24th instance of a clock driver on the 4th level of the clock network that is referred to as “rclk.” This driver drives the net named rclk_L4_N23 where the ‘N’ stands for ‘net.’ Thus, we can build a lumped resistance and capacitance model of the clock network by reading directly down the simple clock hierarchy file. By keeping track of what driver was last encountered for each level, we know that the net that feeds a driver is the last net seen of the level below that driver in question. The generic situation can be seen in Fig. 31.

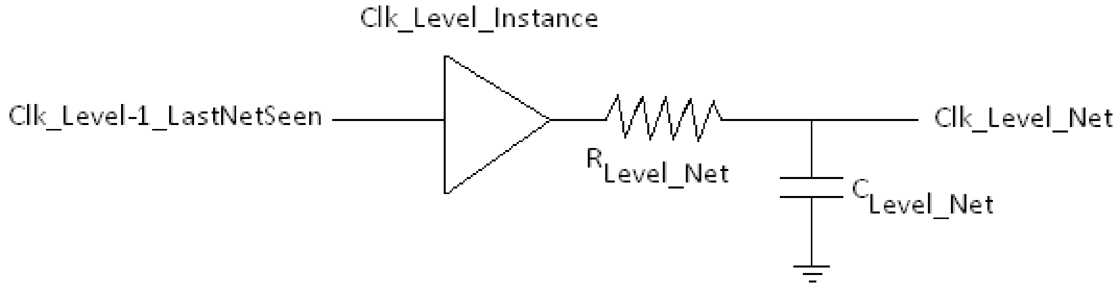


Fig. 31: A generic representation of the building of the clock model

The script will build the clock network in this way while writing the SPICE file. Upon reaching a leaf node, two important things must be done. First of all, we need to characterize each flip-flop as a load capacitance. This will help to keep the model size to a minimum by not including the SPICE model for all flip-flops just to get accurate timing numbers for skew. Based on pin capacitance tables for the technology, a load capacitance is added to each leaf. The value of added capacitance varies depending on the type of flip-flop from 1.7pF – 8.2 pF in the 45nm library. Secondly, when a leaf is reached, the name of the node must be added to an array of node names. Once all processing is complete, this array is traversed and measure statements are added to the simulation file to determine the arrival time of the clock at these leaf nodes. Finally, the model is run in a SPICE simulator. The output of the session is saved to a log file which is parsed for the arrival information and a calculation is performed to determine the global skew of the clock network.

5.3.3 Development of a Dynamic, Object-Oriented Model

While using scripts was acceptable for building the initial SPICE file, in order to be able to make structural changes to our clock model, it is necessary to make an object oriented model. Thus, we will be able to move the structure of our clock tree to a tree data structure. This will make the construction of algorithms to alter the 3D clock network much easier and faster. Instead of starting again with the files generated by Encounter, a simple method is constructed to read in the saved capture of the HSPICE session when the original clock tree was run. This log file contains the original SPICE file as well as the results of the measure statements used to determine the arrival times at the leaf nodes. We can also use the .DEF file that can be extracted from Encounter to provide the physical placement of each driver. From these two files we can have all the necessary information about a clock driver, including:

- Driver Name
- Cell Type
- Parent Driver
- Child Driver
- Position
- Arrival Time
- Capacitance of Load
- Resistance of Load

Building the clock tree in a data structure actually starts with reading the placement file and instantiating a new driver object any time a clock driver is encountered. The driver name, type, and position can be found in the .DEF placement file. Once this is finished, another method reads the output file from the previously run SPICE simulation to obtain clock arrival times and the resistive and capacitive loads that each buffer drives. In the final parsing method, the clock tree hierarchy text file is read to populate the "Parent" field of each driver. As in the building of the initial SPICE file using the scripts, a note is made of the most recently visited driver of each level. In this way, a driver's parent driver is the last one encountered in the previous level. Lastly, since all the data has been read into the program and the "Child" field is still left blank, the tree must be traversed for each driver to determine its children. For a given driver, its children drivers will be those that have the driver in question listed as their parent. In fact, the implementation of the tree data structure is such that it can also be accessed by level, meaning that this step is sped up considerably because when searching for children, only the next lowest level must be searched. After this is complete, all the necessary information we need about our tree has been gathered and filled in. Now it will be possible to add methods to this program that will be able to alter the clock network for our experimentations.

Before this, however, another method must be written to print the SPICE netlist to a text file. This way, when changes are made to the tree structure, a new simulation file can be written quickly. However, this method ends up being the most intricate because while algorithms to simulate special cases are run elsewhere, the information that is put in the SPICE file is parsed and printed in this method. Therefore, it is rife with switches and flags to indicate how the file should be written. When a new type of test routing is added or another method is employed, this method must be altered so that the SPICE file is written accurately.

Because of a mismatch between the timing models used by Encounter and the lumped load model developed in this work, the clock network produced by the tool did not always simulate to have acceptably low clock skew in these simulations. Because of this, it was necessary to run CTS in manual mode and specify clock tree parameters that would yield good results when run through the Perl scripts. Once a reasonably low skew solution was produced by the tools, a method was written to retime the network by looking at the arrival times of the clock signal at various loaf nodes and changing the upstream driver types to lower skew. Once this was completed, we were left with a clock model for a 2D design of the TLU of the S1 Core. Considering the place, route, and CTS was performed in a commercial tool made for ASIC designs, the clock speed is limited to approximately 500MHz. At this speed, the global skew of our 2D model is 4.08% of the total clock period which is approximately 82ps.

CHAPTER 6

TESTABLE 3D CLOCK DISTRIBUTION

This chapter will focus on the work done to make a partial clock distribution on a prebond 3D die testable by connecting the disconnected clock regions with routing only utilized during testing. While the aim is to minimize both power and skew, a solution with lower skew will be preferred over one with lower power as long as the added power is not unnecessarily much. Another goal is to minimize the added wire length of the testing solution. This is mainly to avoid creating an entire redundant tree during testing. As you will see, however, a redundant tree is constructed in Section 6.1.1 as a point of reference. The first section below will deal with a simpler case of routing the clock signal to the root drivers of each disconnected tree from one or more clock probes. The second section employs the use of Delay Lock Loops (DLLs) to synchronize the disconnected networks with low skew.

6.1 Simple Routing for Testability

The simplest approach to connect all the subtrees on a prebond die is to create a global clock distribution that is only active during testing. This tree is a redundant tree and thus requires significant routing resources and power. However, the structure and routing are straightforward.

6.1.1 Routing from One Clock Probe

To make the test process as similar as possible to that of a standard 2D design, we aim to make the prebond clock distribution similar as well. This is done by supplying the design with one clock probe from testing. From this one probe, all disconnected clock networks are driven. A simple rout is conducted from the probe to the root of each disconnected local clock network. This is done as an approximation of the final route. For simulation purposes, the length of the route is determined by the Manhattan distance between the two points. While no physical routing of wires is performed, this approximation will capture the effect of the distance between the source and the destination of the route and will give us an accurate representation of delay in the simulation.

Because we do not want a larger tree connected to our clock network during testing, a gate must be placed on this route close to the root of each disconnected clock tree. This way, during operation the clock drivers that drive each local network will not also have to drive the test clock routing. Luckily, because nothing is connected to the clock TSVs during prebond testing, the clock signals need not be multiplexed – the test clock signal can merely be disconnected when the wafers have been bonded. This is fortunate because in this configuration the clock signal in operation does not need to pass through any additional gates, eliminating a potential source of skew and power dissipation. A representation of this simple routing with one clock probe and gates to disconnect it during operation can be seen in Fig. 32. In the figure, the large square in the center represents a clock probe point and the small squares in the middle of the local trees represent TSVs.

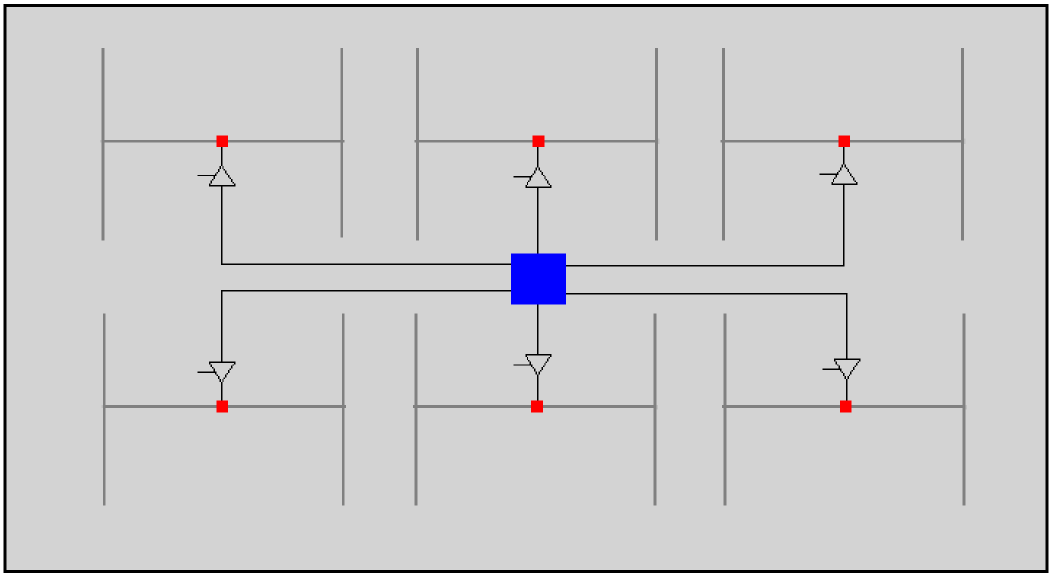


Fig. 32: Simple clock test routing with one clock probe and test-enabled gates

6.1.2 Routing from Four Clock Probes

The extension from one clock probe points to four is a straightforward one that allows for the disconnected networks to receive a test clock signal from a closer clock source than in the case with one probe. This will be helpful in decreasing skew because the difference between the shortest and longest route will be less than in the previous case.

However, while it is tempting to increase the number of clock probes to the number of disconnected networks, this will not be a feasible solution as the number of probes on a probe card is limited and are best used for control signals and scan chain connections. Additionally, the physical dimensions of the TSVs prohibit direct probing[11]. To keep the probes symmetrically spaced, it is natural to increase the number from one to four. However, the next logical number would be nine, which would be too many probes to dedicate to clock.

The four-probe routing is easily imagined, but can also be seen in Fig. 33. Keep in mind that the number of disconnected networks will be larger than the six shown, so the differences in distance will be more varied than shown in these figures.

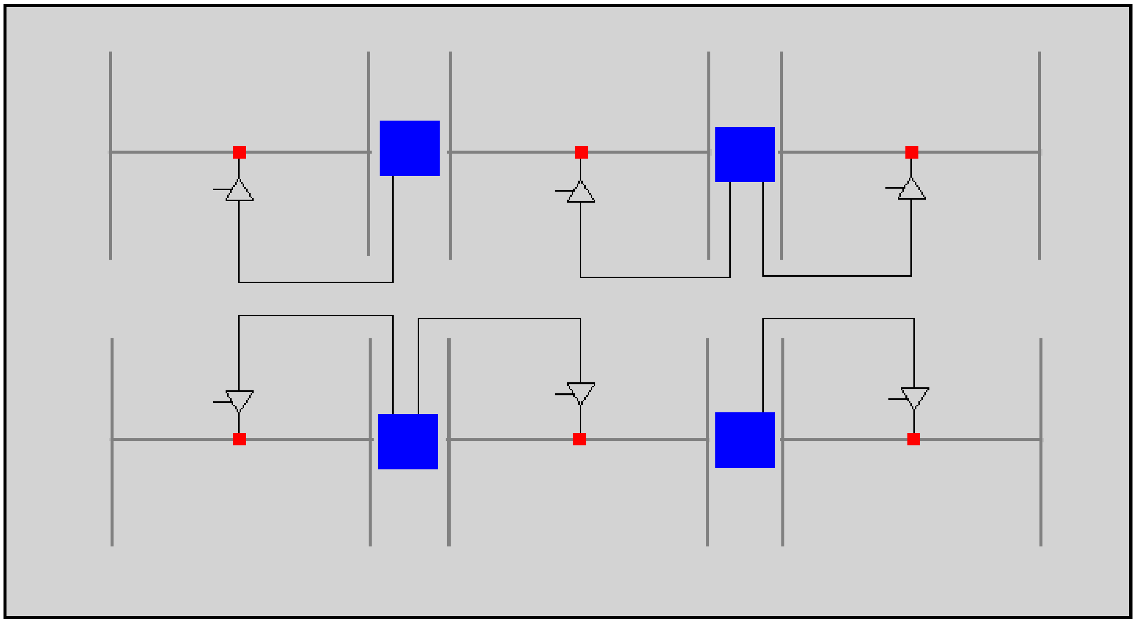


Fig. 33: Simple clock test routing with four test probes and test enabled-gates

In order to perform test routing, it is determined which quadrant of the design a given disconnected clock tree root resides in. Once this determination is made, the route is performed from that local network's TSV to the probe that resides in its quadrant. Because of this (as visible in Fig. 33), not all probes will drive the same number of local networks. However, because of the geometry of the design placement, it is unlikely that these will be very unbalanced.

The results for simple routing from one clock probe and four clock probes are shown in Table 6.

Table 6: Comparison of One-Probe and Four-Probe simple routing

	Clock Skew (pS)	%	Power (mW)
One Probe	186	9.3	96.7
Four Probes	152	7.6	96.4

6.2 DLL Based Clock Testing

To reduce the skew between clock sinks, we employ the use of DLLs to ensure that the clock signal arrives at all leaf nodes with as similar phase as possible. We examine ways in which this can be accomplished, but first it is important to understand the way this circuit is constructed and how it works.

6.2.1 DLL Design and Use

Delay Lock Loops are a more recent innovation than their close relative, the Phase Lock Loop (PLL). Both are control circuits that use feedback in order to synchronize an output clock signal with some reference input clock. While a PLL uses a Voltage Controlled Oscillator (VCO) to generate a clock signal in phase with a reference clock, a DLL uses a Voltage Controlled Delay Line (VCDL) to delay the reference clock signal the appropriate amount until the feedback received from the target location is in phase with the input reference. Because a DLL eliminates the need for an oscillator, it is easier to design, less sensitive to voltage and temperature fluctuations, and more stable than a PLL. A DLL does also does not suffer from jitter accumulation, as does a PLL [31].

6.2.1.1 Operation Overview

A DLL is a mixed-signal control circuit that changes the control voltage on a VCDL until the signal at the output of the VCDL is in phase with the input reference signal. Unlike a PLL, the output of the VCDL is not a clock generated by an oscillator, but merely the reference clock itself with a phase shift. The block diagram of a typical DLL can be seen in Fig. 34.

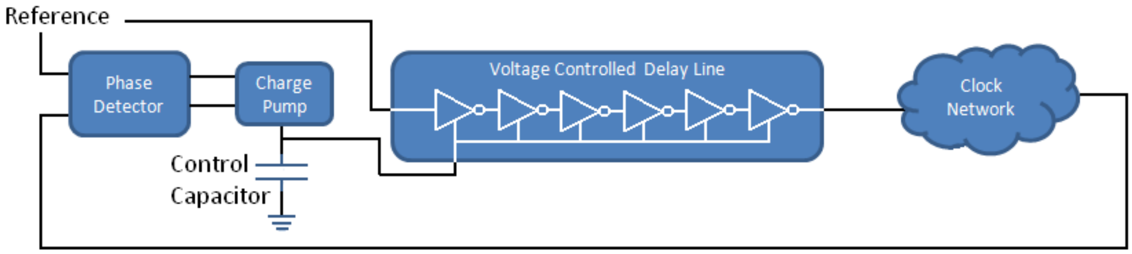


Fig. 34: Block diagram of a Delay Lock Loop syncing the leaf of a clock tree with a reference signal

The feedback line is taken from the point at which the clock transition should be in sync with the reference clock. In the case of distributed clocking (where the goal is to operate separate clock networks with low skew) the feedback is taken from one of the data flip-flops that is to be synchronized with the reference. The phases of the feedback and the reference clock are compared using a phase detector circuit. The output of this is two signals; one that indicates when the feedback clock leads the reference clock, and one that indicates when the feedback clock lags the reference clock. These two signals are connected to a charge pump that, through some filtering, adjusts the control voltage to correct for the phase difference between the two clocks. This control voltage is then directly connected to the VCDL. The control voltage will affect how long it takes for the reference clock to propagate through the VCDL. Tuning this control to the proper voltage will cause the feedback clock to be in phase with the reference clock. The relationship between the control voltage and the delay is dependent on the type of VCDL that is employed.

6.2.1.2 Phase Detector

The duty of a phase detector is to generate a signal that is proportional to the difference between the phases of two input signals. The simplest phase detector is an XOR gate. When the inputs are not the same value, the output of the XOR gate will be a logic 1. When the inputs take the same value, the output will be a logic 0. Thus, inputs that are perfectly in sync will show a constant 0 on the output of this simple phase detector. The average value of the output voltage will indicate how out of phase the two signals are. Unfortunately, it is not possible to determine from using an XOR gate as a phase detector if increasing or decreasing the delay in the VCDL will cause the input signals to align. Because of this, a more advanced phase detector is required.

There are many phase detector implementations, some using standard logic gates and others being designed on the transistor level. The phase detector shown in Fig. 35 is a linear phase detector from [39] and was chosen because of its low transistor count and the use of UP and DN outputs that can be used to control the charge pump.

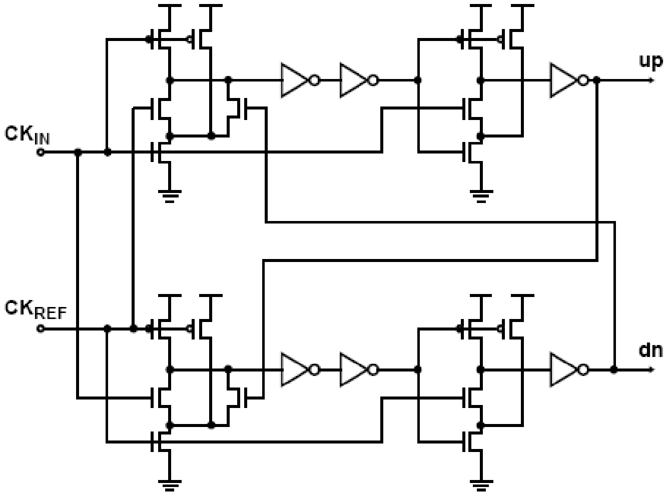


Fig. 35: Linear Phase Detector producing overlapping UP and DN signals depending on input overlap

While analyzing the design of this detector is time consuming, its functionality is easy to understand. The UP and DN outputs provide pulses that are proportional to the degree to which the two input signals are out of phase. When the feedback clock lags the reference clock, the pulse on UP is longer than the pulse on DN, indicating that the control voltage should rise. When the feedback clock leads the reference clock, the pulse on DN is longer than the pulse on UP, decreasing the control voltage. This is assuming of course that a higher control voltage will lead to shorter delay in the VCDL. While this is not always the case and depends on the construction of the VCDL, the names of UP and DN can be reversed to ensure that the labeling makes sense. Regardless, it is the relative widths of the pulses that indicate the phase shift between the two inputs and direct the charge pump to change the control voltage.

The output of the phase detector based on the overlap of the input and reference clocks can be seen in Fig. 36. When the two inputs are in phase (this occurs at approximately 4ns), the widths of the two output pulses are identical. When there is a phase difference between the two input signals, the widths of the UP and DN pulses are not equal. This will cause the charge pump to change the control voltage to account for the phase difference.

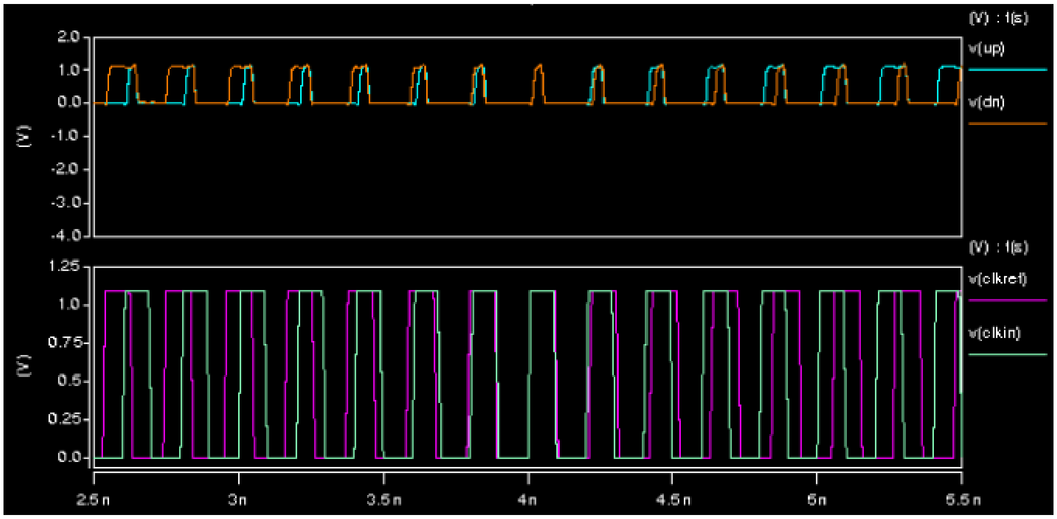


Fig. 36: Phase detector operation with output shown as top waveform

6.2.1.3 Charge Pump and Filter

The charge pump is a very simple circuit that is responsible for increasing or decreasing the control voltage based on the pulse widths on its UP and DN inputs. The filter smoothes out the voltage transition and ensures that if the UP and DN pulses have equal width, the control voltage will remain unaffected. In the case of this DLL, the filter is merely a large capacitor. A more complex filter would make the charges in control voltage less dramatic, but would also increase the lock time of the DLL; an important performance metric.

The charge pump consists of two current sources and two switches. A pulse on UP causes charge to be pumped onto the capacitor, thus increasing the control voltage while a pulse on DN causes charge to be pumped out of the capacitor, causing the control voltage to decrease. If UP and DN are high at the same time, the current will merely flow to ground. Because of this, it is only the difference between the two pulse widths that determines the change in control voltage and ensures that if the two pulses have the same width, the control voltage does not change.

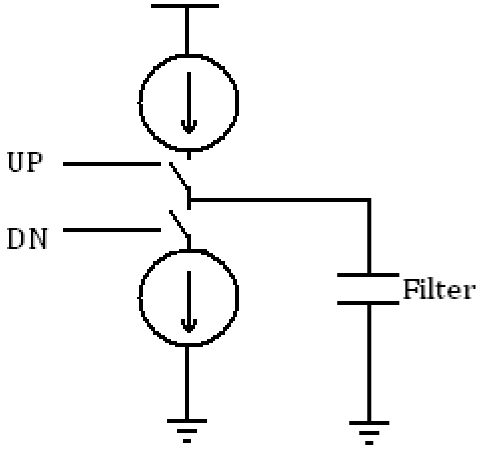


Fig. 37: Graphical representation of the charge pump and capacitive filter

It is also important to ensure that the transistors in the charge pump are sized properly such that a pulse of a given width removes that same amount of charge from the capacitor if the pulse is applied to the DN input as the amount of charge that would be added if the current is applied to the UP input. If this is not done carefully and the charge pump is not balanced, there will be a subtle oscillation in the control voltage and the DLL will never fully lock.

6.2.1.4 Voltage Controlled Delay Line

There are two common types of VCDLs that are used in DLLs: the shunt capacitor delay line and the current-starved inverter delay line. Both employ a string of inverters that are used to provide delay. The difference between the two is the manner in which the control voltage is used to change the delay through the line. In order to be able to sync the feedback clock to the reference clock under all possible propagation delays of the clock network, the VCDL must be have a delay range of at least one full clock period. That is, the difference of the delay when the control voltage is at its highest possible value and when the control voltage is at its lowest possible value (ideally well above threshold) should be one period. The construction of the VCDL determines whether a higher control voltage results in a larger or smaller delay.

The first type of VCDL we will discuss is the Shunt Capacitor Delay Line. One cell of this type can be seen in Fig. 38. It consists of an inverter whose output is loaded by an RC circuit that consists of two transistors. The capacitance is the gate capacitance of a transistor whose drain, source, and body are tied to ground. The resistance is an NMOS transistor whose gate is connected to the control, thus acting as a variable resistor. This device controls how much current flows from the output node of the inverter into and out of the capacitor. The higher the control voltage, the lower the resistance, and thus the higher the current that flows into and out of the capacitor. In this way, a high control voltage loads each intermediate node more, which increases the delay through a string of these cells which make up the VCDL. Because this type of

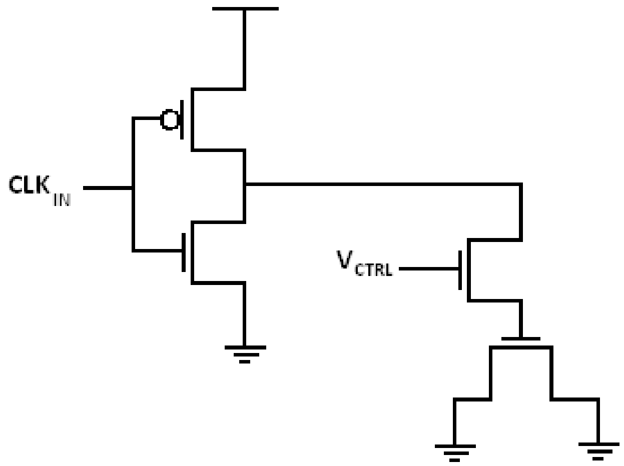


Fig. 38: One cell of a Shunt Capacitor Variable Delay Line

VCDL is basically just an inverter chain that has a variable load, it has high noise immunity, and is very linear. However, in order to get a wide range of delays, the shunt capacitor has to be very large to load down the inverter output node when the control voltage is high. This need for a large capacitor in every cell translates to a high area overhead. Add to this the fact that this large capacitor is charged and discharged on every cycle, using a large amount of power.

Instead of loading down each inverter output node, the current-starved inverter delay line provides a variable delay by allowing the control voltage to adjust the amount of current the inverter can use to switch. As seen in Fig. 39, a voltage-dependant current source is used to control how much current is used for pulling up the output node and an NMOS transistor is used to control the pull down current. The control voltage determines how much current can flow through these elements. Contrary to the shunt capacitor delay line, a higher control voltage decreases the delay through a current-starved inverter delay line. This type of delay line has a larger delay range than the previous type, and thus will require fewer cells to achieve the full period delay range. Also, because large transistors are not needed to load the lines, this solution will occupy less area. The current-starved inverter delay line was implemented in this work

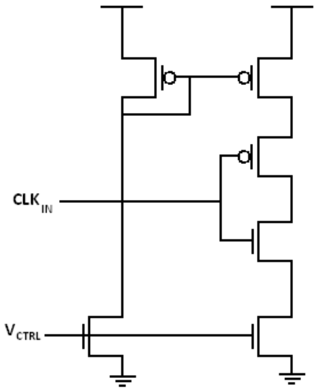


Fig. 39: One cell of a Current-Starved Inverter Variable Delay Line

mainly because of its larger delay range and lower potential area. The delay range of one cell of this type is shown in Fig. 40.

However, one major weakness of the current-starved inverter delay line is its nonlinearity. As the control voltages decreases to the point where it begins to flirt with the threshold voltage, the delay of the VCDL increases greatly. For reasons of signal integrity, a decision must be made about how close to the threshold voltage the delay line will be required to go. Additionally, low control voltages can cause a variation in the duty cycle of the output signal. The difference in threshold voltage between the NMOS and PMOS transistors is to blame for the duty cycle suffering when the control voltage is very low.

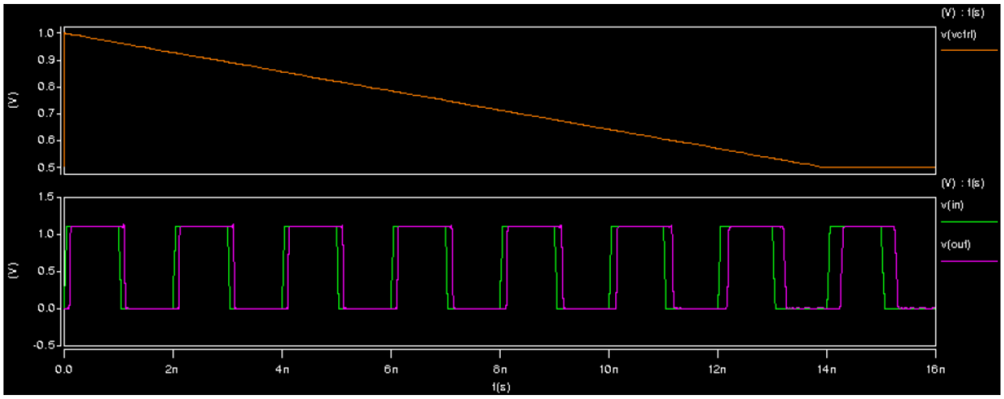


Fig. 40: Effect of control voltage on delay of one variable delay cell

As seen in Fig. 41, the output duty cycle is dependent on the control voltage. Careful consideration and sizing went into designing the cells that generated this plot. As you can see, the plot of control voltage vs. duty cycle is not monotonic, so it is difficult to maximize delay range without the duty cycle suffering. In practice, the best solution is to add extra delay cells to ensure that the control voltage stays well away from the threshold voltage.

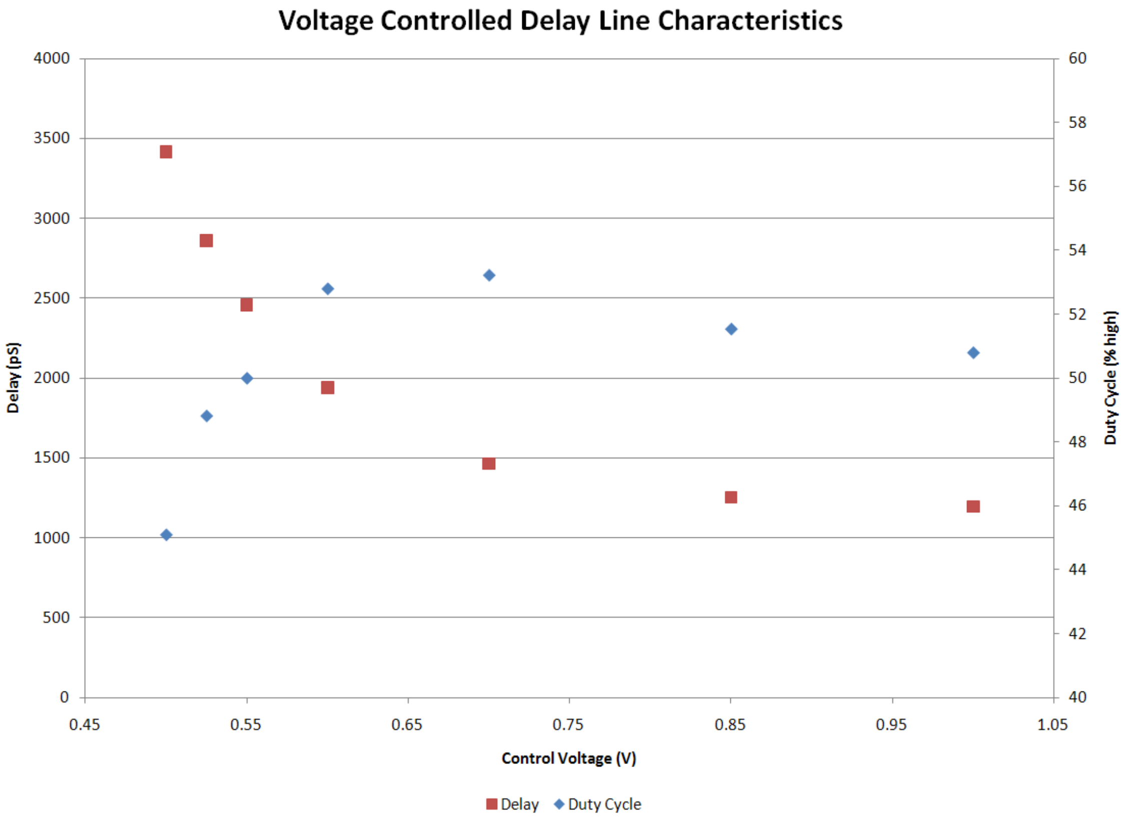


Fig. 41: Delay and duty cycle versus control voltage for the current-starved inverter delay line

6.2.1.5 Operating Characteristics

Since our simulation model of the 3D clock network is designed to run at 500MHz, the target operating frequency of the DLL will be the same. Thus, the difference in delay between the highest control voltage and the lowest allowable control voltage must be at least one clock period, which is 2ns in this case. The highest control voltage will be V_{DD}, which in the 45nm technology node is typically 1V. The lowest allowable voltage is a design choice. As you can see in Fig. 41, the largest change in delay comes when the control voltage is low. Because we desire a large delay range, it would seem advantageous to set our lowest allowable control voltage as low as possible. However, at very low voltages, the duty cycle really suffers. Thus, we have chosen to keep our minimum allowable voltage to be 0.52 V. This will keep the duty cycle from going too far in either direction. Based on simulations, it is determined that 14 variable delay cells are required to produce the necessary delay range. This will result in a range of 2220ps, which is an ample margin to keep the control voltage from getting too close to the minimum voltage.

When in operation, the DLL will roughly lock its output to the reference signal within 11 clock cycles to within 2% maximum phase error. After 14 cycles, the control voltage has settled down and the maximum phase error drops to 0.5%. After this point, there is some jitter as the DLL tries to match the reference signal exactly, but it stays within this error. The DLL consists of approximately 350 transistors, most of these residing in the variable delay line. At 500MHz, the circuit consumes less than 1 mW of power, again mostly in the VCDL. Because the transistors in the VCDL are being cut off at lower control voltages, they will consume much more power than standard logic transistors. From this it is understandable that the variable delay line dissipates a disproportionate amount of power when compared to the rest of the circuit. It is thus preferred to keep the size of the VCDL as small as possible. A waveform of the pertinent signals of a locking DLL can be seen in Fig. 42.

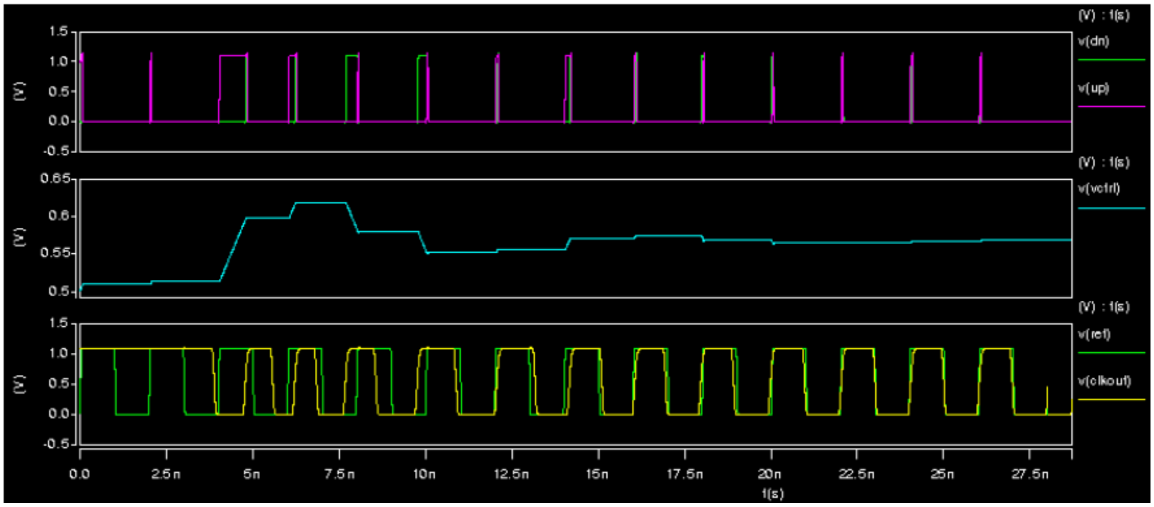


Fig. 42: Waveform of DLL locking to reference signal

6.2.2 Necessary Provisions for Test (OPCG)

While DLLs will be used during testing to ensure that the at-speed clock is distributed with low skew, additional circuitry must be added to make the solution suitable for scan-based testing. In this test methodology (as detailed in Section 4.2), it is necessary to scan in test patterns at a slow clock frequency (usually around 100MHz) then apply two clock pulses (a launch and a capture pulse) at the intended operating frequency of the design. The captured result is then scanned out at the scan frequency as a new pattern is scanned in. The result is then compared to the expected result and a determination is made as to whether the design performed as expected.

Because it takes a DLL several cycles to lock on to a reference clock frequency, it will not be possible to start and stop the DLL during scan testing. Futhermore, a DLL can only be efficiently designed to operate at a narrow range of frequencies. This is because, as discussed in Section 6.2.1.4, the range of delay in the variable delay line must be at least one clock period. Thus, in theory a DLL could run at a higher frequency than it is designed for, but this would mean unnecessary hardware and would require the control voltage to be more delicately tuned. Also, if the clock frequency is changed abruptly, as in scan-based testing, the DLL would not lock for several cycles. This is unacceptable and thus provisions to make the DLL suitable for test are necessary.

These provisions are referred to as On-Product Clock Generation (OPCG). Typically, OPCG refers to the use of a PLL to multiply a slow incoming clock signal to a faster one that is at the operational frequency the system requires. This type of OPCG is utilized in [1][23][36]. However, because of our choice to use DLLs for their advantages over PLLs (See Section 6.2.1), our treatment of OPCG with a DLL for use in scan testing is unique. As we will see, the nature of the DLL leads us to some unexpected conclusions. We will begin with the situation where the scan clock frequency is supplied by the tester and a test clock in generated on-chip.

6.2.2.1 Test Clock Generation from Scan Clock

The means to generate a test clock from a scan clock is shown in Fig. 43. While scanning in and out patterns, the DLL is operating at the scan frequency as supplied by the tester, Scan Enable is asserted, Operation Enable is deasserted, and the scan clock is fed to the clock network. When the pattern has been scanned in and an at-speed clock is to be applied (which we will refer to as the test clock), Scan Enable and Operation Enable are deasserted, and the clock network is fed a signal that comes from the Exclusive-Or of four taps from a Fixed Delay Line (FDL). These taps are chosen so that the delay between each tap is one-half the period of the desired test clock frequency. When these taps are fed through an XOR tree, the output will be a launch and capture pulse at the test clock frequency as seen in Fig. 44.

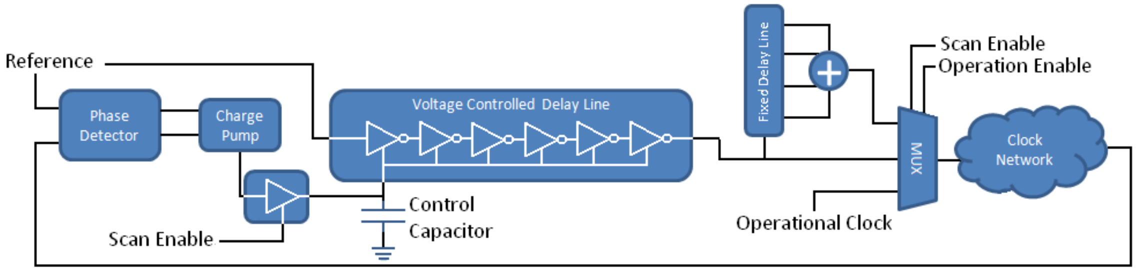


Fig. 43: Block diagram of DLL with provisions to generate test clock from scan clock

Also note the buffer located between the charge pump and the control voltage capacitor in Fig. 43. This can either be an AND gate or a transmission gate that will ensure that the control voltage will not change when the test clock is being applied. Because the signal that is sent to the clock network will not correspond to the reference clock, this gate is needed to keep the control voltage from changing while the test clock is being applied, which would cause the DLL to unlock.

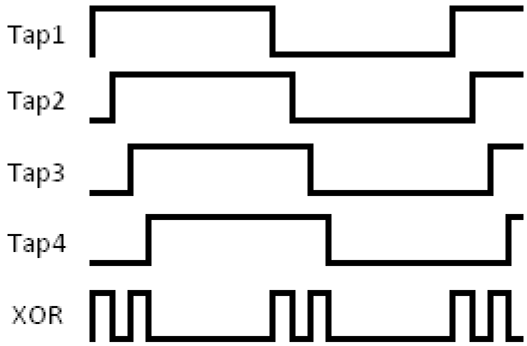


Fig. 44: Launch and capture clocks generated from XOR of taps from fixed delay line

While we see that this is a valid approach for using a DLL with some addition circuitry for OPCG, it turns out that this solution has some undesirable features that make it not ideal for our target application which requires low on-chip power dissipation. In previous works examined, it made sense to generate a faster clock on chip because they used a PLL. To generate a faster oscillation in a PLL, less delay and thus fewer cells are needed in the Voltage Controlled Oscillator (VCO). However, in requiring our DLL to run at a low frequency, namely that of the scan clock, a larger delay range is needed in the VCDL. This is because in order for the DLL to be able to lock for all phase shifts possible in the clock network, the delay range must be equal to one full period of the reference clock. If the DLL is required to run at 125 MHz (representing a typical scan clock) the range of delay for the VCDL must be at least 8ns. This corresponds to a very large number of variable delay cells. Even though lower switching frequencies generally correspond to lower power, because of the hardware overhead needed to run at such a low frequency, this solution will consume more power than running at a faster frequency. Add to that the power used by the FDL, and it becomes clear that when using a DLL for OPCG, generating the test clock from the scan clock may not be the ideal solution. With that in mind, we must now explore another option.

6.2.2.2 Scan Clock Generation from Test Clock

In previous works ([1][23][36]), it was attractive to supply the chip with a slow clock used for scan-based testing and generating a faster test clock on chip using a PLL. It was often stated that this would allow for simpler test equipment. However, when using a DLL to synchronize a clock signal between disconnected clock networks in the case of a prebond 3D IC, supplying a die with a scan clock and generating a test clock leads to high on-chip power dissipation. For this reason, we will explore the situation where the die is supplied with a clock signal running at the test frequency (which is also the intended operational frequency) and generating the scan clock on-chip.

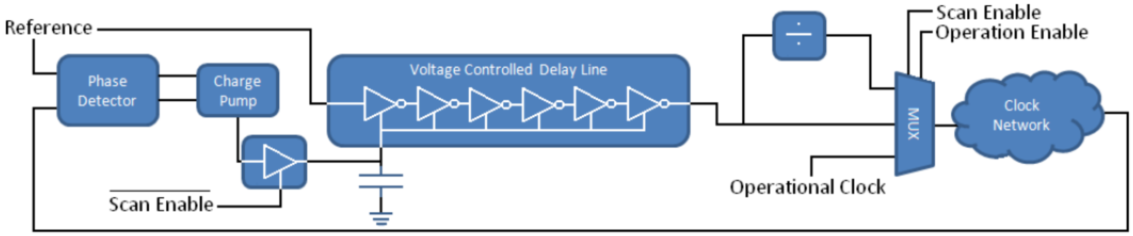


Fig. 45: Block diagram of DLL with provisions to generate scan clock from test clock

While this means running the DLL at higher frequencies, significant hardware savings make this a lower power solution. As mentioned previously, the delay range of the VCDL must be equal to one full period of the intended operating frequency of the DLL in order for it to be able to lock for all possible delays. Running the DLL at four times the frequency roughly means four times fewer delay cells in the delay line. Also, the circuitry needed to divide the output of the DLL to generate the scan clock can be rather straightforward. If the scan frequency is one-quarter the test frequency, just two flip-flops are needed to divide the clock. Similarly, a division of 8 would require three flip-flops. Slightly more complicated is the case where one would want to divide by a number that is not a power of two, but these circuits are well known and incur less overhead than needed to implement the fixed delay line shown in Fig. 43. A few simple dividers are shown in Fig. 46. To divide a clock by 10, for example, the divide by 5 circuit could be followed by a single flip-flop to divide again by 2.

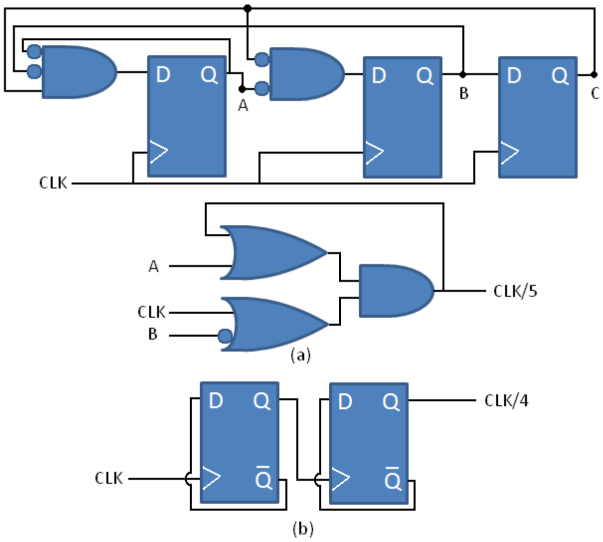


Fig. 46: Circuits to (a) divide clock by 5[22] and (b) 4 with 50% duty cycle

The full visual of this solution can be seen in Fig. 45. While it may not appear on the surface to be much smaller than the implementation in Fig. 43, the VCDL can be much shorter because of the higher frequency and the dividing circuitry requiring many fewer transistors than the FDL in Section 6.2.2.1. This reduced circuitry results in lower power consumption despite the higher frequency.

Note that in this case, after the DLL is locked, the charge pump will be disconnected from the control voltage capacitor during the scan operation. When the Scan Enable signal is lowered and the launch and capture pulses are applied to the logic, the DLL will adjust the control voltage if it has drifted. Because of the size of the filtering capacitor and the time between opportunities to sync, there will be little very little change in the control voltage. The operation of the DLL with OPCG generating the scan clock from the test clock can be seen in Fig. 47.

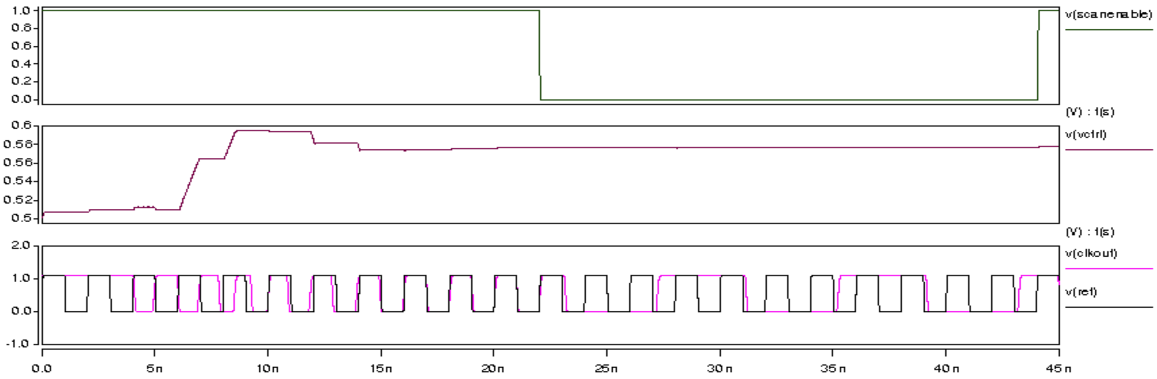


Fig. 47: SPICE waveform of DLL syncing with test clock, then applying divided scan clock

The results of these two ways to handle OPCG for scan-based testing can be seen in Table 7. Case 1 refers to the test clock being generated from the scan clock, and Case 2 is when the scan clock is divided from the test clock. The DLL operating frequencies are shown, as well as the relative sizes of the two implementations and their power usage. It is clear that that dividing the test clock to generate the scan clock makes much more sense in terms of the area and power of the solution. Also, because of the higher operating frequency, the initial lock will occur much faster in this case.

Table 7: Hardware and Power Comparison of Techniques

	DLL Frequency (MHz)	VCDL Cells	Transistor Count	Power (mW)
Case 1	125	51	2086	2.56
Case 2	500	13	458	0.82

6.2.3 DLL Test Routing with Multiple Clock Probes

Only after provisions have been added the DLL for test can they be used to synchronize disconnected clock regions on a prebond 3D IC. There are a couple ways this can be done and we will look first at the method where more than one clock probe is available to the die during testing. The goal of this method is to synchronize different regions of the die with the same master clock signal.

As in Section 6.1.2, we will use four clock probes for this experiment. As explained in that section, four probes allows for a benefit in skew management by symmetric probe placement without being of an inordinate number. Because there are four probes providing reference clocks to four places on the die, we will utilize four DLLs to ensure that the quadrants of the die are synchronized with each other. Unfortunately, because there are more disconnected networks than DLLs available, the feedback to the DLL will have to be a representative leaf node in order to minimize skew. The method of determining this representative node is the subject of some experimenting and will be detailed. A visualization of this method for using multiple clock probes and DLLs to synchronize regions of the die can be seen in Fig. 48. In this figure, each side of the die is synchronized because both DLLs are locked to the reference clock which is supplied from the tester to the probe points.

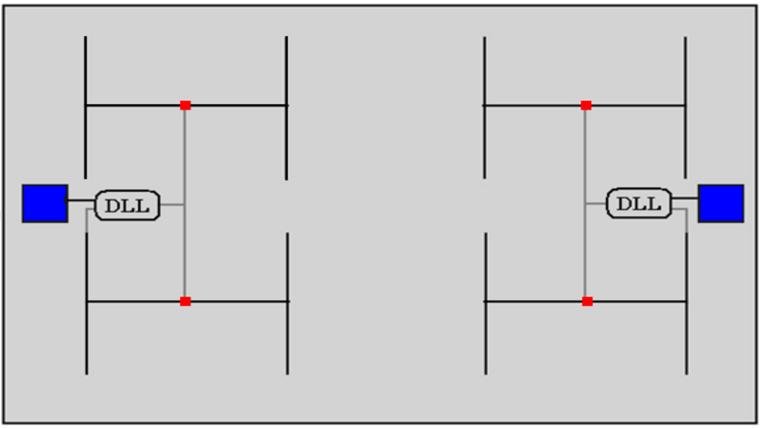


Fig. 48: Synchronizing regions of the die using one DLL for each probe point

6.2.3.1 Arrival Time Aware Feedback Routing

Initially, the leaf node chosen to provide feedback was the one whose clock arrival time fell the closest to the average of the earliest and latest arrival times. In other words, the target arrival time minimized the difference between the arrival times of all leaf nodes. Once this target was calculated, the leaf node whose arrival time was closest was chosen as the feedback to the DLL. A load model for the routing from the leaf node to the DLL input is based on extracted characteristic resistance and capacitance per unit length from this design. This arrival time aware approach did not yield the desired skew because the routing from the feedback leaf to the DLL added delay. Thus, an alternative method of choosing the feedback node was required.

6.2.3.2 Proximity-Based Feedback Routing

Because it was the length of the feedback routing that was adding delay to the feedback signal and thus causing the DLLs to lock out of phase, the feedback leaf node was chosen based on the proximity to the DLL it was feeding. Thus, the closest clock sink to the DLL input was chosen as the feedback node. This provided improved results over the Arrival Aware feedback as well as the simple routing case with four probes detailed in Section 6.1.2. The results for both the arrival time aware and the proximity-based feedback routing with DLLs are shown in Table 8.

Table 8: Comparison of Arrival Aware and Proximity Based Feedback Routing

	Clock Skew (pS)	%	Power (mW)
Arrival Aware	188	9.4	101.5
Proximity-Based	138	6.9	101.2

6.3 Daisy Chained Test Clock Networks

Because the best results from the DLL test routing in Section 6.2.3 show a skew of approximately 6.9% and in order to fairly say the solution is suitable to at-speed test we need the skew to be as close as possible to the 2D case of 4.08%, a better solution is called for. Also, the previous solution required four clock probes from the tester and the use of redundant trees (although because they were distributed they are not as large as the redundant trees in Section 6.1.1).

A means of synchronizing all disconnected clock regions with a master test clock may be accomplished by implementing the structure shown in Fig. 49. In this methodology, only one probe is required to supply the die with a clock signal. A DLL is used at the closest local disconnected clock network to synchronize the leaf nodes with the master test clock. Each subsequent disconnected network will contain its own DLL which will synchronize its leaf cells with the nearest active clock signal. In this way, all leaf nodes in the die will be synchronized with the master clock. Because of delay in routing, the active clock signal will be chosen based solely on proximity to the DLL for the not-yet-connect network. The delay in the routing from the DLL output to the root of the local clock tree is not as issue as the DLL will synchronize the feedback and the reference regardless of the existing delay in output.

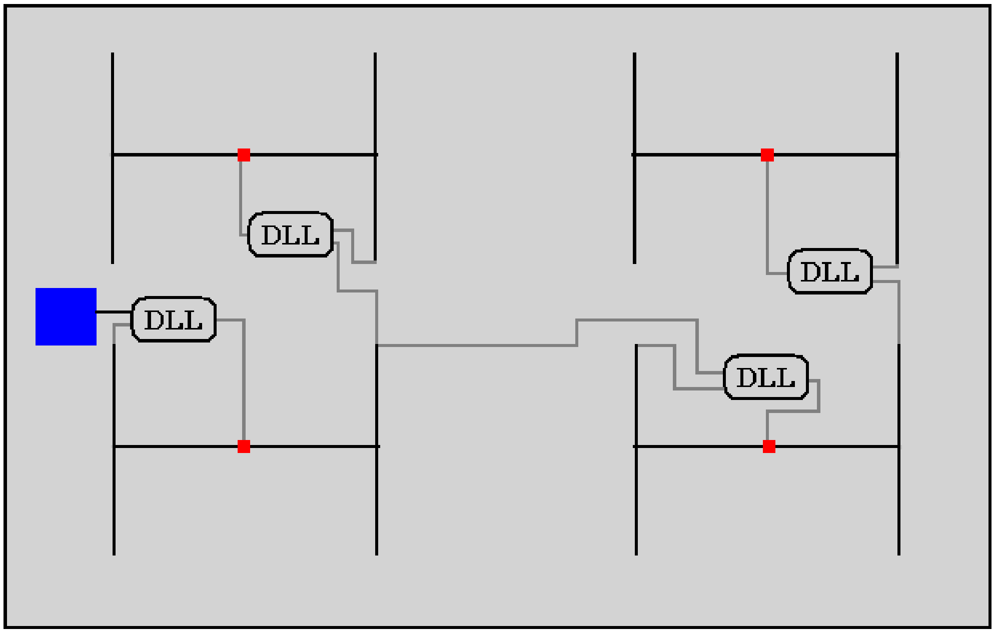


Fig. 49: Daisy chaining clock networks for low-skew operation during test

6.3.1 Daisy Chain Construction

We begin by assuming the probe for the clock signal is located at the center of die. The clock sinks in the design are grouped by which disconnected clock network they belong to. This allows us to focus on and connect one local network and aids in speeding up the construction algorithm. Two lists of clock sinks are kept: those that are still disconnected and do not carry a live clock signal, and those that have already been connected using a DLL to synchronize with the reference clock.

Initially, all clock sinks are contained in the list of not-live sinks and the live sink list is empty. The first connection is made between the clock probe and the closest clock sink to the probe. These will be reference and the feedback node for the DLL. The output of the DLL is then routed to the driver for that local network and the clock sinks are removed from the list of not-live sinks and added to the list of live sinks. From this point on, each not-live network uses the closest live clock sink as its reference node. As each network is connected, the sinks are moved from the not-live list to the live list until all clock sinks are live.

6.3.1.1 Chain Length Impact

Assuming delay in the feedback routing is kept to a minimum by using a proximity-based approach, much of the skew during testing will be from phase error of the DLLs. As stated in Section 6.2.1.5, each DLL has a maximum phase error of 0.5%. Because our 3D die has 24 disconnected networks, the total maximum error is 12% of the clock cycle. However, it is extremely unlikely that this will occur because the error is based on over- and under-shooting the control voltage. It is much more likely that the effects of this phase error will average out in the end.

However, this does bring up the effect of the length of the daisy chain on the global skew of the solution. The larger the number of networks in the daisy chain, the higher the possibility of increased skew due to the phase error from the locking of the DLLs. Thus, we will make an attempt to limit the length of the daisy chain by constructing multiple chains. This is done simply by connecting the local networks closest the clock probe first and working outwards, always picking the next closest not-live local network to the probe. This makes several shorter chains form emanating from the probe and decreasing the impact of accumulated phase error. Also, if we find that somehow a long chain is still formed, a cap can be put on the maximum chain length so that no reference is ever taken from a live-network that is more than a predetermined depth in a chain. For the TLU, a longest chain of eight occurred without a cap. To keep skew to a minimum, a cap was set at a length of six networks in a chain.

6.3.1.2 Algorithm for Chain Construction

Because of the complexity of the algorithm that is responsible for daisy chaining the clock networks for testing, it must be organized sensibly to avoid unnecessarily long runtimes. This is accomplished mainly by taking some initial setup time to store the information we will be accessing frequently in suitable data structures. In this way, we are decreasing runtime at the cost of increased memory usage. Because the number of clock sinks is only in the thousands, this will be an acceptable tradeoff. The result of this is that we can quickly search live sinks, not-live sinks and clock sinks by local network. The algorithm for daisy chaining is seen in Algorithm 1.

Algorithm 1: Daisy Chain Construction

Input: Lists of clock sinks with locations and local network number and β, the maximum daisy chain length

Output: A reference node and a feedback node for each disconnected network for DLL daisy chaining

```
for numClockSinks, do
  closestSinkToProbe = closest not-live sink
  thisNetwork = closestSinkToProbe(networkNum)
    for all sinks in thisNetwork $P_i$, do
      closestLive = closest live sink
        if dist(closestLive, $P_i$) < smallestDist and closestLive(depth) < $\beta$ then
          reference = closestLive
          feedback = $P_i$
        end if
    end for
  add reference to route list
  add feedback to route list
  mark tree as live
end for
```

As mentioned previously, we avoid long daisy chains by always choosing the closest not-live network to the clock probe to work with. Once the closest not-live network is determined, we need to find the pair of clock sinks, one in the network we are working on and one in a live (already connected) network, whose separation is the smallest. By choosing the reference sink and the feedback sink with the smallest separation we can minimize the delay introduced by the routing to the phase detector. At this point, we also check to make sure the live sink being considered is not at the end of a long chain whose length is equal to the maximum allowed length, β. If the depth is less than β, the sinks are chosen and added to a route list that will be used in a later method that produces the SPICE file.

6.3.2 A Fully-Connected Test Clock Network with Daisy Chained DLLs

The TLU contains 24 disconnected clock networks when the top five levels of global clock network are removed. This cut is determined by the number of TSVs available for the clock signal. Thus, 24 DLLs are needed to synchronize the disconnected networks. The algorithm that connects these networks in a daisy chain takes approximately 30 seconds to run and produces a SPICE file of the test network. A visual representation of the disconnected networks that have been synchronized by chaining using DLLs can be seen in Fig. 50. It should be noted that for clarity, the lowest level drivers are shown in this figure as showing all clock sinks would be impractical.

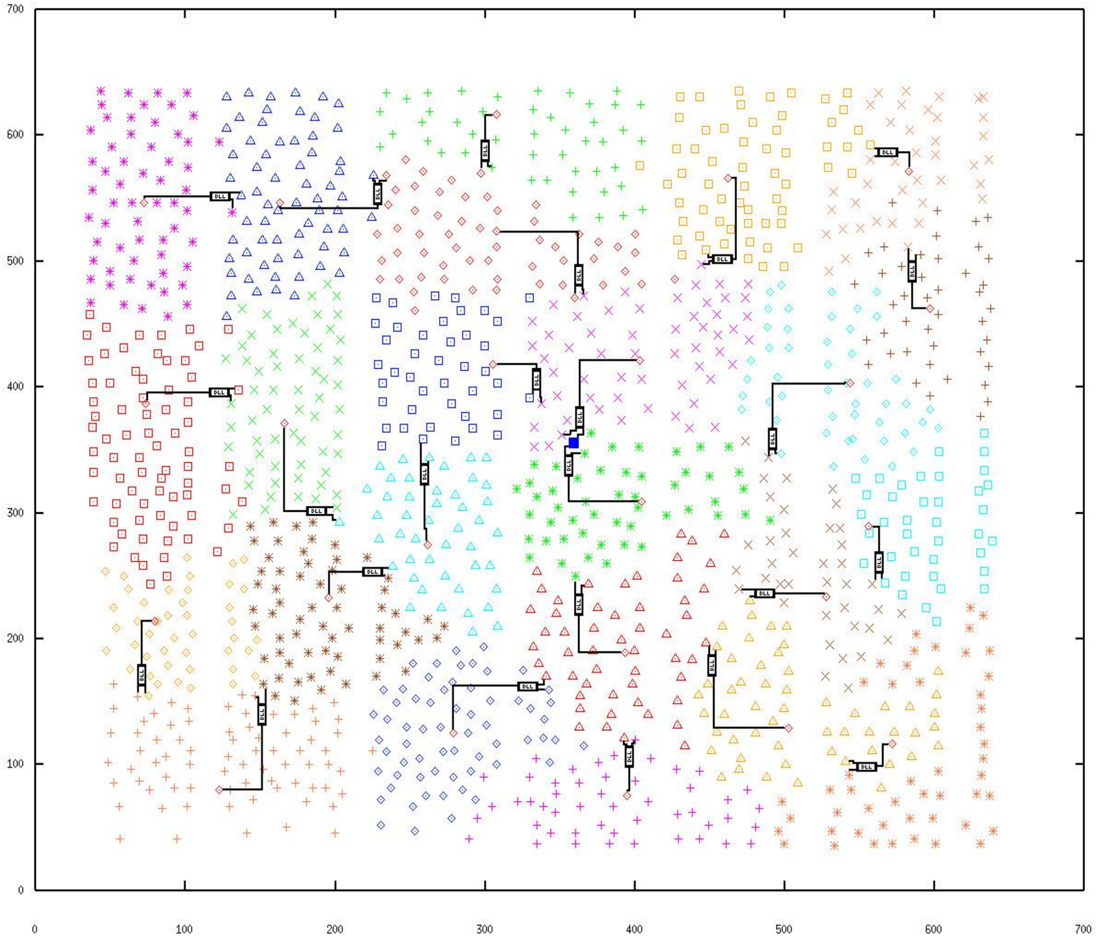


Fig. 50: Previously disconnected networks synchronized for test using DLLs

In the center of the design, two local networks use the probe directly as their reference. From then on, each network takes its reference from the closest live sink to minimize delay from routing. As you can see, there is often very little routing from the reference and feedback to the DLL. The majority of routing comes from the output of the DLL to the driver for that local network. Luckily, the DLL adjusts for this delay so the length of the route does not affect the quality of the solution. In Fig. 51, you can more clearly see the local networks and how they are connected for test. The lines originate at the reference clock sink and terminate at the driver for the disconnected network. The numbers indicate the depth in the chain of each network. As you can see, the length of the longest chain is six.

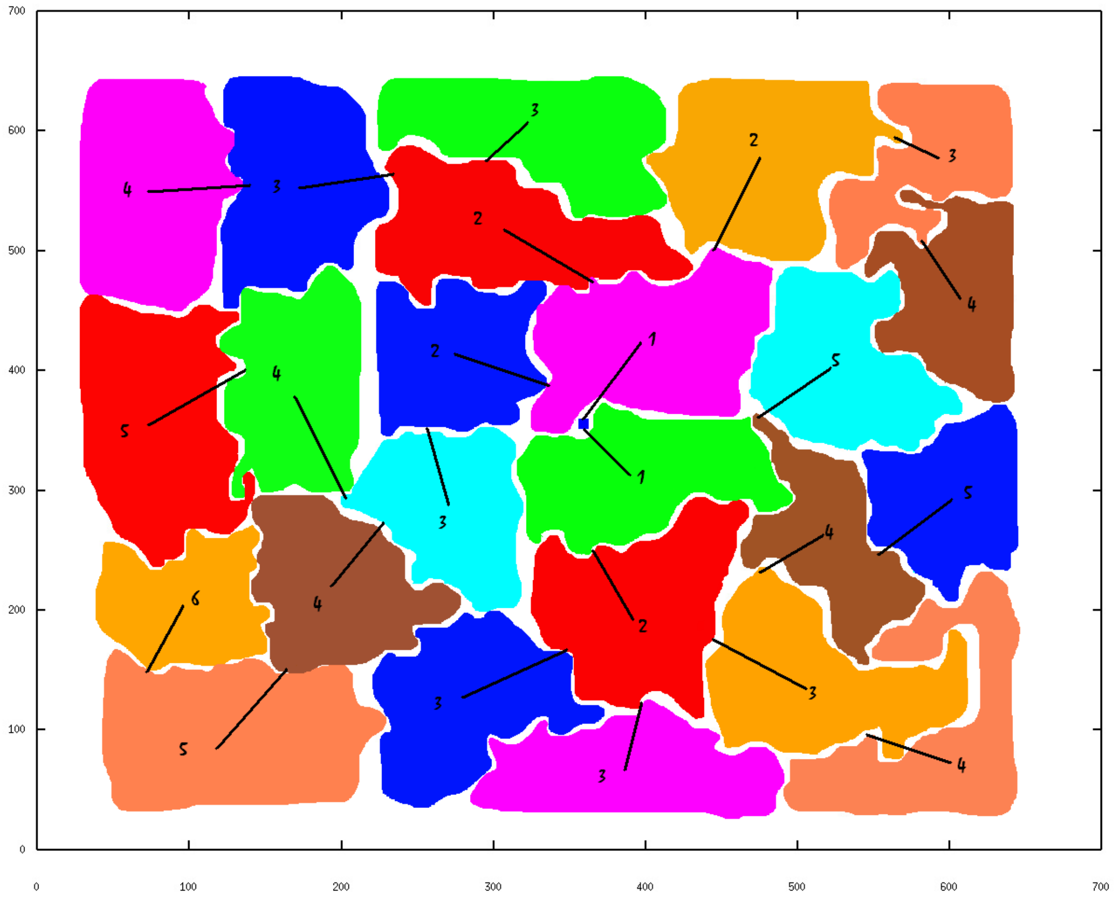


Fig. 51: A higher level view of the daisy chained clock networks

When testing begins, it takes time for the clock signal to propagate through the chains and stabilize. It is important to remember that many of these DLLs are not operating with a golden reference. Instead, the reference signal is changing phase as it attempts to lock to its reference, which may also be changing. In our simulations, all 24 DLLs locked within 80ns, which is actually quite good considering a single DLL is expected to lock in approximately 20ns. When locked, the DLLs were able to drive the clock sinks with a global skew of less than 4.09% of the clock period. This shows it is possible to use DLLs to daisy chain disconnected clock networks for test at or near speed considering an operational skew of 4.08% in our initial 2D tests. While the skew achieved using this solution is much better than the previous solutions examined, it does use considerably more power during test. With 24 DLLs running, the power consumption for the total clock network is approximately 120mW, up from 96mW for the 3D network postbond. While this is certainly an increase in test power, we will see in the following section that it is an acceptable cost for the quality of the solution provided.

6.4 Comparison of Results

In this section, we will examine the various methods of connecting disconnected clock networks in a prebond 3D integrated circuit. These methods include: simple routing from a single probe, simple routing from four probes, synchronization of die regions using DLLs, and daisy chained DLLs. We will review the skew achieved, power dissipation (during test), and routing overhead for each method. A discussion of the benefits of each implementation will follow.

Table 9: Comparison of methods to drive disconnected clock regions in a prebond die

Type	Skew (%)	Power (mW)	% Increase Test Power†	Additional Transistors	% Increase Transistors*	Added Wirelength (μm)
No Testing	4.08	95.6	-	-	-	-
1 Probe	9.3	96.7	0.17-0.23	48	0.005	7147
4 Probes	7.6	96.4	0.13-0.17	48	0.005	3631
DLL Regions	6.9	101.2	0.88-1.17	1832	0.20	3246
DLL Chain	4.09	120.9	3.97-5.29	10992	1.17	1335

† Assuming clock consumes 15-20% total power *TLU contains 936,965 transistors

It is clear from the table above that the simple routing for one and four clock probes offer the lowest power solution, but also the worst skew. These solutions employ unbuffered, redundant trees that, while consuming little power because of the lack of additional clock drivers, introduce appreciable skew since the trees are not balanced. Based on the target skew for our clock network, these solutions will not provide acceptably low skew.

When four DLLs are used to synchronize die regions, we can reduce the global skew to a number lower than the simple routing, but still not low enough to perform at-speed test. This solution also does not provide for considerably lower use of routing resources over the simple routing case.

Using DLLs to daisy chain disconnected clock networks is finally successful at providing a low-skew solution that will allow for at speed test of a prebond die with an incomplete clock network. Of course, this comes at the price of thousands of additional transistors in order to implement the 24 DLLs. However, when compared to the size of the full design, this overhead is not significant. Since this logic will be dormant during operation, the DLL blocks can be spread out and used as filler cells to aid in manufacturability and thermal uniformity. We have said that the DLL must be placed close to the reference and feedback clock sinks, but really it is only the phase detector that must abide by this constraint. Since this block contains only 30 transistors, it should not be a problem to locate it where necessary. The rest of the DLL can be fairly spread out; because the it is the nature of the DLL to lock for all possible phases, any delay introduced by routing within the DLL itself will be corrected for.

We have mentioned several times that keeping test power within an envelope is essential. This is certainly the case, and has motivated a smart design of the DLL to use as little power as possible. However, given that this scheme provides a suitable method of prebond testing, the 4-5% increase in test power is acceptable. It should be noted that during operation when the provisions for prebond testing are not active, none of these solutions presented above have any appreciable effect on the skew or power of the design. This, of course, is essential to this problem. In short, daisy chaining disconnected clock networks using DLLs is a viable solution to the problem of testing prebond dies of a 3D IC with disconnected clock networks without requiring an unacceptable level of overhead during testing or having any affect on the design during operation.

CHAPTER 7

INCREASING INTERDIE BANDWIDTH

Presented in this chapter is a techique to increase interdie bandwidth by multiplexing signals over TSVs. As we have examined previously, smaller TSVs allow for more interdie connections which increases the benefits that can be gained from 3D stacking. However, these small TSVs suffer from reliability concerns arising from proper bonding alignment. Thus, it seems that if we require high yield connections, we must employ large TSVs and limit power and performance gains. However, by using one TSV for more than one signal, high reliability TSVs can be implemented while still allowing for high interdie bandwidth.

7.1 Background and Related Work

Multiplexing has been used extensively for decades to allow two or more things to share an expensive resource. In digital circuits, Time Division Multiplexing (TDM) is used to send multiple signals over a shared communication channel. As we have seen, a TSV connecting two dies in a 3D stack is considered to be an expensive resource and a 3D design would benefit from a scheme to allow for this one physical channel to carry multiple signals.

There has been little published work dealing with increasing the signal carrying capacity of TSVs for 3D designs. However, presented in [35] is a method of constructing a serial vertical interconnect specifically for use in 3D ICs. In the paper, a serial transmitter and receiver are designed and each is placed on adjacent dies. A ring oscillator is present in both transmitter and receiver to clock the serial data. A ring counter and a shift register are also employed in each unit. While a quality solution is provided, the transmission frequency of the design tops out at approximately 2GHz. This means that a high level of serialization will lead to performance penalties in high-speed designs. For this reason, the author recommends a 2:1 or 4:1 serialization. It is clear for these low levels of serialization a solution that imposes less overhead is required.

In [40] and [25], a clocking method employing dual-edge triggered (DET) flip-flops to allow for the halving of the clock frequency in a design to save power is examined. Using flip-flops that trigger on both the rising and falling edges of the clock allow for greater utilization of the clock signal. These works act as a stepping stone to the method presented here to better utilize the nature of the clock signal to increase the number of signals that can be sent through a TSV.

7.2 Unidirectional Communication

Fig. 52 shows the block diagram of the multiplexing and capturing circuitry for the unidirectional case. In one of the dies which will become part of a 3D stack resides circuitry producing two signals which are destined for an adjacent die in the stack. This circuitry is represented by the clouds producing S0 and S1 in the figure. Without any special consideration to improving the usability of links between the dies, two TSVs would be required to carry these signals to their destinations. However, if properly multiplexed, these signals can share one TSV. The way this is accomplished is as follows: both signals are fed into the inputs of a 2:1 multiplexer whose select input is tied to the system clock. When the clock is low, the signal from Source 0 (S0) will be selected and connected to the TSV. When the clock transitions to high, the signal from Source 1 (S1) will likewise be fed to the TSV. This succeeds in getting both signals from one die to the next in the span of one clock cycle.

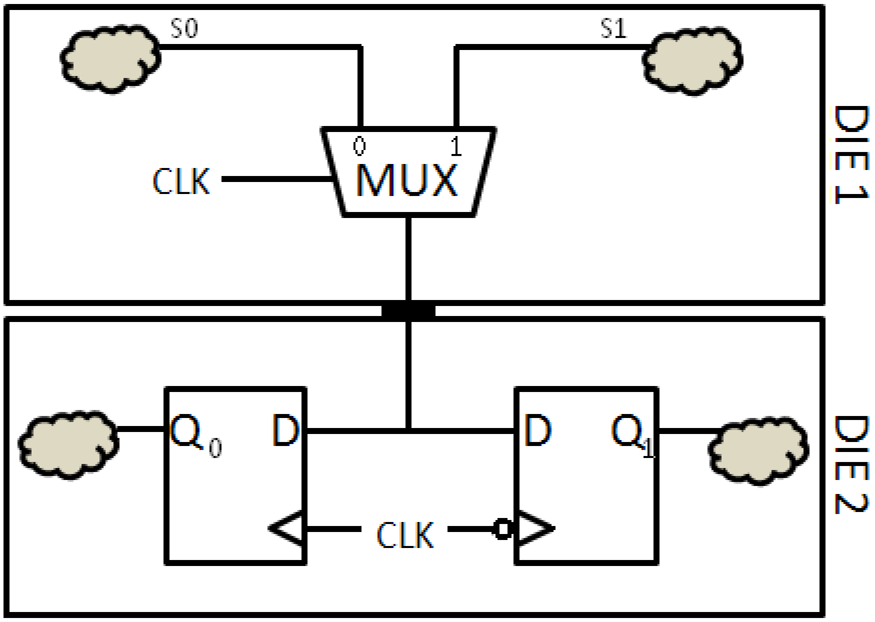


Fig. 52: Block diagram of unidirectional TSV multiplexing with corresponding capturing flip-flops

In order to recover the data, we need to be able to latch data on both the rising and falling edges of the clock. Following the work presented in [25] and [40], originally we examined using a DET flip-flop to capture the data, then demultiplex the signals to their destinations. However, it became clear that a better solution would be to employ a pair of flip-flops; one that triggers on a rising clock edge and one that triggers on the falling clock edge. This way, the outputs of the flip-flops would hold the appropriate value and no demultiplexing would be necessary. There are several advantages to this method. First, if a DET flip-flop was used to capture the signals before they were multiplexed out, it is likely that the signals would need to be latched again at their destination points. This would mean a signal would be unnecessarily latched once during its journey. Additionally, since most implementations of DET flip-flops are merely two SET flip-flops, one with an inverted clock line, whose outputs are multiplexed by the clock, using two SET flip-flops with opposite polarity effectively saves two multiplexers per instance. This elimination of redundancy is a logical optimization to perform.

In order to capture the correct values in the proper flip-flop, the signal sent on a given clock polarity must be destined for a flip-flop that is triggered on the opposite polarity transition. That is, a signal that is placed on the TSV when the clock is high (positive) must be captured by a flip-flop that triggers when the clock transitions from the high to low state (negative trigger). In Fig. 52, the signal generated on the left-hand side of Die 1 is captured by the flip-flop on the left-hand side of Die 2. The same can also be clearly seen in the timing diagram in Fig. 53. Corresponding to Fig. 52, when the clock signal is high, Signal 1 (S1) is selected and placed on the TSV. This signal is then captured on Q1 by the negative edge-triggered flip-flop when the clock goes low.

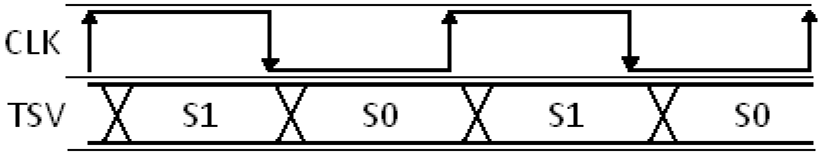


Fig. 53: Timing diagram of TSV multiplexing

7.3 Bidirectional Communication

Examined in the previous section was the case in which two signals originating in one die were destined for the other. Now we will look at the situation where two signals originate on separate adjacent dies and must be communicated to the next. This situation is partly just a reorganization of the previous unidirectional case and can be seen in Fig. 54. While transmission gates are used in the implementation, tri-state buffers are shown here for simplicity of the figure.

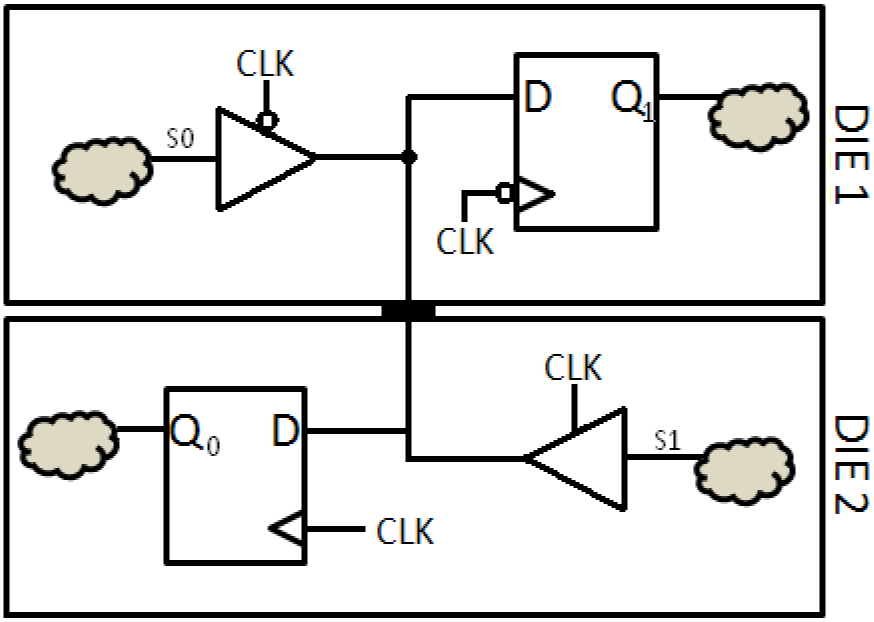


Fig. 54: Multiplexed bidirectional communications over a TSV.

Instead of both receiving flip-flops being on the same die, the two are now on separate dies at the signal destinations. Unlike the previous case, however, we cannot rely on multiplexers to place the appropriate signal on the TSV.

Because the two signals will be originating from separate locations, the gate that selects the signal must have a high-impedance state so as to allow the other signal access to the TSV. Thus, we can either add an enable signal to the multiplexers, or replace them with a tri-state buffer or a transmission gate. In an effort to minimize the transistor count to keep the overhead from this solution as low as possible, a simple transmission gate has been used to provide the senders access to the TSV. In Fig. 54, two types of tri-state buffers are shown (instead of transmission gates for simplicity of the figure) that are active on opposite polarities of the enable (in this case, the clock) signal. This could also be accomplished with two identical such gates, one enabled by an inverted clock. This, of course, imposes a slight overhead over the solution shown.

Since the unilateral configuration used one multiplexer, bus (or TSV) contention was not an issue. However, since in the bidirectional configuration the drivers are timed by different clock lines, it is important that clock skew between these points is low to avoid contention. If such a situation occurs, the overlap will be fleeting and will only result in a small amount of wasted power. The timing diagram for this bidirectional case will be identical to the unidirectional case.

7.4 Flip-flop Timing Violations

As visible in the timing diagram in Fig. 53, the transmitted signal will be available to be latched well ahead of the capturing clock pulse. However, if skew exists between the clock signals on the two dies, a hold time violation may occur where the signal that is present on the TSV changes before the clock transition occurs on the capturing flip-flop. This would not be an issue if there was considerable delay in the data line, but since TSVs have characteristically low paracitics [5], this delay will only be appreciable at very high clock speeds. Care should be taken that hold time violations do not occur when timing the design using this technique. If these violations do occur, they can be mitigated by inserting a delay stage in line with the TSV, imposing a small overhead.

7.5 Experimental Results

In this section we will examine the functionality, performance characteristics, power, and area overhead of this implementation

7.5.1 Functionality and Performance

Because the only difference in hardware between the unidirectional and bidirectional cases is one multiplexer (6 transistors, assuming a transmission gate multiplexer) in former case is replaced by two transmission gates (4 transistors each, including clock inverting), the difference in overhead in terms of transistor count and power between the two will be negligible.

Since the timing and simulation waveforms will be identical between the two cases, we will only examine the results for the unidirectional case.

As seen in Fig. 55, the proposed TSV multiplexing method succeeds in doubling the number of signals that can be sent over one TSV. You can clearly see the two source signals on the originating die, the voltage on the TSV, and the two signals captured on the other die. The simulation shown here is for a clock frequency of 2GHz. The values present at the originating logic are represented by the signals S0 and S1. These are signals in time and will be represented as nibbles with the MSB as the first value present. S0 takes the value 1011 while S1 is 1101. These can clearly be seen as the second and third waveform, respectively. The next line is the voltage on the TSV. The last two waveforms show the data 1011 has reached Q0 and 1101 is present at Q1. Also visible in Fig. 55 is the fact that the latency from when the signal is available and when it shows up as the output of the flip-flop is not the same for both signals. This must be taken into account during timing closure. While it was mentioned in Section 7.4 that a delay element may need to be inserted in the data line to avoid a hold time violation, in our experiments clock skew was not serious enough to warrant this.

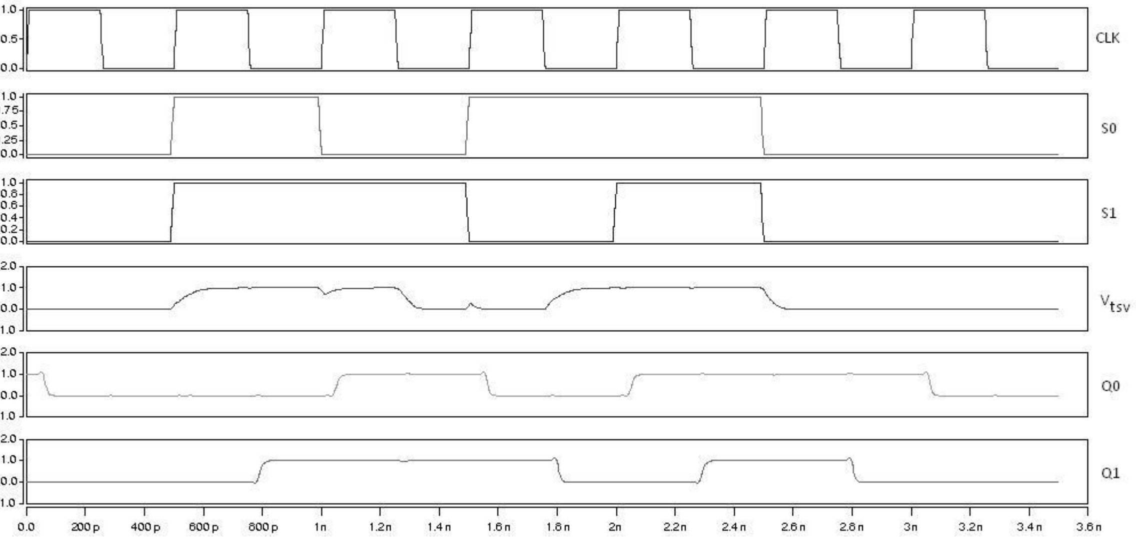


Fig. 55: Waveform of TSV multiplexing from HSPICE simulation

Implemented in the 45nm technology node, this circuit can reliably operate at clock frequencies up to 6GHz, allowing for the possibility of simple, high-speed die-to-die interconnects. Because flip-flops can only be triggered on clock edges, and there are only two polarities of clock transitions, this method can only be used to send two signals on a TSV, no more. A more complex circuit operating at a speed multiple times higher than the system clock would be needed to increase the number of signals that can be sent in one cycle.

While this method ensures that two signals are sent in one clock cycle, it should be noted that if the TSV is not flop-bound, there may be some ramifications on performance on a signal that is captured by the negative edge-triggered flip-flop. Because this signal is placed on the TSV when the clock is high, it must be available at the multiplexer a half-cycle before the other signal. However, as we have discussed it is almost essential for signal TSV to be at cycle boundaries for testing, so in practice this will not be of concern.

7.5.2 Overhead

This minimalist design is intended to impose as low of an overhead in terms of power usage and transistor count as possible. The amount of power this circuitry uses is, of course, highly dependent on clock speed. For this reason, we will focus on the area overhead as measured by the number of additional transistors needed. In order to calculate overhead, we must only include those elements that would not already be present in the design. In the case of unidirectional communication, the overhead is merely a multiplexer whereas in the bidirectional case, it is two transmission gates. The elements included in the simulation were a multiplexer (or transmission gates in the bidirectional case), a TSV modeled as a series resistance and two shunt capacitances in a π model, and the capturing flip-flops. Since the flip-flops would already be present in the design (needed certainly for scan-based testing), and the TSV is already present, the only overhead is the logic that connects the appropriate signal to the TSV based on the phase of the clock.

To examine the overhead imposed by this solution, we will examine the POWER5 microprocessor from IBM. The design occupies 389 mm^2 , consists of 276M transistors, and operates at 1.5GHz [18]. We partition the design into three dies, each having an area of approximately 130μm^2 and containing 92M transistors. Assuming a TSV pitch of 20μm [41], each die will contain 324,167 TSVs. Furthermore assuming that 30% of these will be used for power [13] and ground, and 10% used for clock, that leaves 194,500 TSVs available for signals on each die. Using the unilateral case where the overhead imposed by multiplexing the TSVs is six transistors, we can calculate the percentage overhead as a function of the fraction of TSVs that are multiplexed. This can be seen in Table 10.

Table 10: Hardware overhead as a function of TSVs Multiplexed

% MUXed	# MUXed	Effective TSVs	Additional Transistors	% Increase
0	0	194500	0	0
10	19450	213950	116700	0.1268
20	38900	233400	233400	0.2537
30	58350	252850	350100	0.3805
40	77800	272300	466800	0.5074
50	97250	291750	583500	0.6342
60	116700	311200	700200	0.7611
70	136150	330650	816900	0.8879
80	155600	350100	933600	1.0148
90	175050	369550	1050300	1.1416
100	194500	389000	1167000	1.2685

What we can see from these numbers is that if every signal TSV on a die were multiplexed, the number of signals that could be sent between dies would be doubled with only a 1.27% increase in the number of transistors used. In fact, the situation is actually better than this. The 1.27% overhead accounts for the multiplexers present on one die, while no overhead is incurred on the die that is receiving the signals. In a three die stack, the overhead for the whole stack works out to be approximately 0.847% if every TSV is multiplexed. This is a powerful result. While in reality it is unlikely that every TSV would need to be multiplexed, we can see here that the overhead that this would result in would be entirely acceptable. When compared to how expensive in terms of manufacturing increasing the TSV density is, this small overhead will almost certainly be worthwhile.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

The work presented in this thesis is directed at advancing 3D integration to a point where it will be more commercially feasible. First addressed is the issue of testing a prebond die in a 3D design that does not have a complete clock network. In order to save power, the clock distribution is spread among the dies and is only complete after bonding. However, in order to allow for suitable yields, dies must be tested prior to bonding to ensure that only Known-Good Dies (KGDs) are bonded together. Because of the small size of inter-die TSV connections, they cannot be probed directly. As a result, a test clock network that connects all previously disconnected clock networks must be constructed that is fed by only a small number of probes.

Initially, we examine two cases in which the clock signal is routed directly from clock probes to the drivers of the disconnected networks. This is done using one central clock probe for one test, and four clock probes situated at the midpoint of four quadrants of the die. These both provide full clock connectivity with low test power, but incur large routing overhead and unacceptably high skew.

To combat the skew introduced to the clock network by this routing, we designed and employed Delay Lock Loops (DLLs) to synchronize the four clock regions. This helped in lowering the skew, but not significantly. In order to individually synchronize all of the disconnected clock networks, each network requires a DLL and a clock reference with which to lock. The issue with this scenario is the distribution of that reference clock. This problem is overcome by connecting the clock networks in a daisy chain where each network is synchronized using a DLL with another clock network nearby. At the beginning of this chain is a network that is locked with a golden reference clock supplied by a test probe. In this way, all clock networks have their sinks aligned with the clock probe. This technique is successful in providing a test clock distribution with suitably low skew for at-speed test of the prebond die. While it consumes more power than the other solutions, the increase is reasonable and only seen during test. Additionally, this technique also imposes the lowest amount of routing overhead.

Also investigated is a scheme to increase interdie bandwidth by increasing the signal carrying capacity of TSVs. TSVs are what allow 3D integration to offer the benefits it does. However, because of their small size, manufacturing concerns limit the size and density of TSVs in a design. This leads to them becoming a precious resource that must be used to their full potential.

Bandwidth is increased by multiplexing two signals over one TSV. In order for this scheme to not impose a performance penalty, the signals must be multiplexed at a speed faster than the system clock. If this was not the case, one unlucky signal would be delayed one clock cycle before it was transmitted and received. Normally, the logic needed to control this transfer is complicated and can add considerable overhead. However, our solution used the phases of the system clock to control the multiplexing of the signals over the TSV. Thus, two signals are sent over one TSV in once clock cycle. On the receiving side, flip-flops whose capture polarities are matched to receive the intended signal are employed. Because TSVs must be flop-bound to allow for scan testing, the only additional overhead imposed by this solution is one multiplexer at each TSV being multiplexed. This low overhead allows for this to be implemented liberally where needed to increase bandwidth. Also presented is case where the signals originate and are destined for opposite dies.

These contributions hold potential to be used in industry to increase the commercial viability of 3D integrated circuits which promise to offer lower power designs that can be manufactured with higher yield, performance, and will allow for the integration of disparate technologies in a single die stack.

8.2 Future Work

While the work for this thesis may be completed, the goal of improving the commercial outlook for 3D integrated circuits most certainly is not. There are still many issues that must be tackled before a 3D design will undergo a large production run. Clearly, the area that needs the most improvement is the tools sector. There are very few available 3D partition, place, and route tools in existence. This is mainly because of a lack of agreement on certain aspects of 3D design including TSV size, clock distribution, and partitioning granularities to name just a few. These are things that will only be agreed upon in time; until then, tools must remain flexible to allow for these variations.

More specifically related to the work in this thesis, work can be done to mitigate the overhead imposed by the daisy chaining of clock networks for testing. While the skew is acceptably low and the power dissipation acceptable, the addition of 24 DLLs that remain unused during testing could be addressed. Specifically, using one or some of those DLLs to serve other functions during operation would help to reclaim this otherwise unused area. For example, they could be used to perform Clock and Data Recovery (CDR) while being used in a high-speed data link, or used to perform clock gating of certain areas to decrease power consumption without adding skew by the addition of a clock gate.

Lastly, inter-die bandwidth could be increased further by multiplexing more signals over the TSV during one clock period. This could be in the form of a 4:1 multiplexer powered by a clock signal that is four times faster than the system clock or a true high-speed link that takes advantage of the low paracitics of TSVs by sending large data packets between dies. These high speed links could be used as links between a main memory made of DRAM residing on an adjacent die and a CPU core, or anywhere else that high inter-die bandwidth would be required but not supported by the existing TSV density. This improvement would allow for the performance of 3D ICs to increase greatly while keep power consumptions low.

BIBLIOGRAPHY

[1] A.C. Evans, “Applications of Semiconductor Test Economics, and Multisite Testing to Lower Cost of Test,” *International Test Conference,* 1999.

[2] A. Uzzaman, B. Li, B. Keller, and T. Snethen, "Using Programmable On-Product Clock Generation (OPCG) for Delay Test," in Asian Test Symposium, 2007.

[3] A.V. Mule, E.N. Glytsis, T.K. Gaylord, and J.D. Meindl, “Electrical and Optical Clock Distributions Networks for Gigascale Microprocessors,” *IEEE Transactions of VLSI Systems, Vol. 10, No. 5,* pp. 582-594, October 2002.

[4] C. Yeh, G. Wilke, H. Chen, et al., “Clock Distribution Architectures: A comparative Study,” *ISQED ’06: Proceedings of the 7th Internations Symposium on Quality Electronic Design,* pp. 85-91, 2006.

[5] D. Henry et al, “Development and Characterization of High Performance TSVs for 3D Applications,” *11th Electronics Packaging Technology Conference,* December 2009.

[6] D. James, “Recent Innovations in DRAM Manufacturing,”*Advanced Semiconductor Manufacturing Conference,* pp. 264-269, July 2010.

[7] D. Lee, “3D chip stacking technology for memory device,” *Design of 3D-Chipstacks, ISSCC* (2007).

[8] D.H. Kim, D.H. Kim, K.Athikulwongse, and S.K. Lim, “A Study of Through-Silicon-Via Impact on the 3D Stacked IC Layout,” *International Conference on Computer-Aided Design,* November 2009.

[9] D.L. Lewis and H.S. Lee, “Testing Circuit-Partitioned 3D IC Designs,” *IEEE Computer Society Annual Symposium on VLSI,* May 2009.

[10] E.G. Friedman, “Clock Distribution Networks in Synchronous Digital Integrated Circuits,” *Proceedings of the IEEE,* Vol. 89, No. 5, pp.665-692, May 2001.

[11] E. J. Marinissen and Y. Zorian, "Testing 3D Chips Containing Through-Silicon Vias," in *International test Conference,* 2009.

[12] G. Hetherington, T. Fryars, N. Tamarapalli, et al., “Logic BIST for Large Industrial Designs: Real Issues and Case Studies,” *International Test Conference,* 1999.

[13] G. Loh, Y. Xie, B. Black, “Processor Design in 3D Die-Stacking Technologies,” *IEEE Micro,* Vol. 27, Issue 3, June 2007.

[14] H.H. Chang, Y.C. Shih, Z.C. Hsiao et al., “3D Stacked Chip Technology using Bottom-up Electroplated TSVs,” *Electronic Components and Technology Conference*, pp. 1177-1184, May 2009.

[15] International Technology Roadmap for Semiconductors (ITRS),http://www.itrs.net/ (2009)

[16] J. Cong, A.B. Kahng, G. Robins, “Matching-Based Methods for High Performance Clock Routing,” *IEEE Transactions of Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 8, August 1993.

[17] J.H. Lau, “TSV Manufacturing Yield and Hidden Costs for 3D IC Integration,” *Electronic Components and Technology Conference*, pp. 1031-1042, June 2010.

[18] J. Clabes et al., “Design and Implementation of the POWER5^{TM} Microprocessor,” *Design Automation Conference*, June 2004.

[19] J. Rabaey, A. Chandrakasan, B. Nikolic, Digital Integrated Circuits – A Design Perspective. Prentice Hall, NJ, 2003.

[20] K. Sukuma, P.S. Andry, C.K. Tsang, et al., “Characterization of Stacked Die using Die-to-Wafer Integration for High Yield and Throughput,” *Electronic Components and Technology Conference*, pp. 18-23, May 2008.

[21] L. Dou, M.P. Broderick, “A New Technique for Automated Wafer Inspection and Classification of Particles and Crystalline Defects,” in *Advanced Semiconductor Manufacturing Conference and Workshop*, pp. 180-184, September 1997.

[22] M. Arora, "Clock Dividers Made Easy," Design Flow & Reuse, *ST Microelectronics Ltd, SNUG* Boston 2002.

[23] M. Beck et al., "Logic Design for On-Chip Test Clock Generation- Implementation Details and Impact on Delay Test Quality," in *DATE*, 2005.

[24] M. Mondal, A.J. Ricketts, S. Kirolos, et al., “Thermally Robust Clock Schemes for 3D Integrated Circuits,” *Design Automation and Test in Europe*, 2007.

[25] N. Nedovic and V.G. Oklobdzija, “Dual-Edge Triggered Storage Elements and Clocking Strategy for Low-Power Systems,” in *IEEE Transactions on VLSI Systems*, vol 13, no 5, May 2005.

[26] Nangate 45nm Open Cell Library, CDNLive! EMEA, April 2008, *http://www.cdnusers.org/desktopmodules/ntforums/viewer.aspx?portalid=0&moduleid=467&attachid=402*

[27] OpenSPARC T1 Processor, *http://www.opensparc.net/opensparc-t1/index.html*

[28] P. Hofstee, N. Aoki, D. Boerstler, et al. “A 1 GHz Single-Issue 64b PowerPC Processor,” *International Solid State Circuits Conference*, pp. 90-93, February 2000.

[29] P. Leduca, F. de Crecy, M. Fayolle, et al., “Challenges for 3D Integration: Bonding Quality and Thermal Management,” *International Interconnect Technology Conference*, pp.210-

220, June 2007.

[30] P. Nigh, “Scan-Based Testing: The Only Practical Solution for Testing ASIC/Consumer Products,” *International Test Conference*, 2002.

[31] R. Farjad-Rad et al., "A Low-Power Multiplying DLL for Low-Jitter Multigigahertz Clock Generation in Highly Integrated Digital Chips," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 12, pp. 1804-1812, December 2002.

[32] R.J. Curry, “Evaluation of Economics of Testing in a Manufacturing Environment,” *AUTOTESTCON Proceedings*, pp. 481-489, December 2002.

[33] R. Sankaralingam, N.A. Touba, “Controlling Peak Power During Scan Testing,” *Proceedings of the 20th IEEE VLSI Test Symposium*, 2002.

[34] S. Gupta, M. Hilbert, S. Hong, R. Patti, "Techniques for Producing 3D ICs with High-Density Interconnect," *Available from Tezzaron Semiconductor* (2005).

[35] S. Pasricha, “Exploring Serial Vertical Interconnects for 3D ICs,” *Design Automation Conference*, San Francisco, CA, 2009.

[36] S. Pei, H. Li, and X. Li., "An On-Chip Clock Generation Scheme for Faster-than-at-Speed Delay Testing," in *DATE*, 2010.

[37] S. Ravi, "Power-aware Test: Challenges and Solutions," in *International Test Conference*, 2007.

[38] S. Rusu and G. Singer, “The First IA-64 Microprocessor,” *IEEE Journal of Solid State Circuits*, vol. 35, no. 11, pp. 1359-1544, November, 2000.

[39] S. Sidiropoulos, D. Liu, J. Kim, et al., “Adaptive Bandwidth DLLs and PLLs using Regulated Supply CMOS Buffers,” *IEEE Symposium on VLSI Circuits Digest of Technical Papers*, 2000.

[40] S.E. Esmaeili, G.E.R. Cowan, and A.J. Al-Khalili, “Power Reduction in Energy Recovery and Square-Wave Clock Distribution Networks Operating at Half Frequency with Dual-Edge Triggered Flip-Flops,” *Joint 6th International IEEE Northeast Workshop on Circuits and Systems*, June 2008.

[41] *The International Technology Roadmap for Semiconductors: Interconnect*, 2009

[42] V. Arunachalam, “Clock Distribution in a 3D Microprocessor,” Master’s Thesis, University of Massachusetts Amherst, 2008.

[43] V.F. Pavlidid, I. Savidis, E.G. Friedman, “Clock Distributions for 3-D Integrated Circuits,” *IEEE Custom Integrated Circuits Conference*, 2008.

[44] W.R. Davis et al, “Demystifying 3D ICs: The Pros and Cons of Going Vertical,” in *IEEE*

Design and Test of Computers, December 2005.

[45] X. Wu, P. Falkenstern, Y. Xie, “Scan Chain Design for Three-Dimensional Integrated Circuits (3D ICs),” *International Conference on Computer Aided Design*, October 2007.

[46] X. Zhao, D.L. Lewis, H.H.S.Lee, and S.K. Lim, “Pre-bond Testable Low-Power Clock Tree Design for 3D Stacked ICs,” *ICCAD*, pp. 184-190, November 2009.

[47] Y. Guillou and A.M. Dutron, “3D IC Products Using TSV for Mobile Phone Applications: An Industrial Perspective,” *Microelectronics and Packaging Conference*, June 2009.