Abstract

SCHOENFLIESS, KORY MICHAEL. Performance Analysis of System-on-Chip
Applications of Three-dimensional Integrated Circuits (Advisor: Dr. W. Rhett Davis)

In the research community, three-dimensional integrated circuit (3DIC) technology has garnered attention for its potential use as a solution to the scaling gap between MOSFET device characteristics and interconnects. The purpose of this work is to examine the performance advantages offered by 3DICs. A 3D microprocessor-based test case has been designed using an automated 3DIC design flow developed by the researchers of North Carolina State University. The test case is based on an open architecture that is exemplary of future complex System-on-Chip (SoC) designs. Specialized partitioning and floorplanning procedures were integrated into the design flow to realize the performance gains of vertical interconnect structures called 3D vias. For the post-design characterization of the 3DIC, temperature dependent models that describe circuit performance over temperature variations were developed. Together with a thermal model of the 3DIC, the performance scaling with temperature was used to predict the degree of degradation of the delay and power dissipation of the 3D test case. Using realistic microprocessor workloads, it was shown that the temperatures of the 3DIC thermal model are convergent upon a final value. The increase in delay and power dissipation from the thermal analysis was found to be negligibly small when compared to the performance improvements of the 3DIC. Timing analysis of the 3D design and its 2D version revealed a critical path delay reduction of 26.59% when opting for a 3D implementation. In addition, the 3D design offered power dissipation savings of an average of 3% while running at a proportionately higher clock frequency.

PERFORMANCE ANALYSIS OF SYSTEM-ON-CHIP APPLICATIONS OF THREE-DIMENSIONAL INTEGRATED CIRCUITS

by

KORY MICHAEL SCHOENFLIESS

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

In

COMPUTER ENGINEERING

Raleigh, NC

2005

Approved by:

_______________________________ _______________________________

Dr. Paul Franzon Dr. Douglas Barlage

____

Dr. W. Rhett Davis
Chair of Advisory Committee

To my loving grandparents, Frank and Helen King, who sadly passed away during my time in graduate school.

Biography

Kory Michael Schoenfliess was born on August 25, 1981 in Rosedale, a suburb of Baltimore, Maryland. Kory has spent most of his life in Maryland, and only recently moved away from the State to pursue graduate studies. He is the son of Lothar and Sharon Schoenfliess, who both still reside in Rosedale. Kory's interest in computer technology was spurred by his attendance of Eastern Technical High School of Essex, Maryland, where he concentrated in engineering. In 1999, Kory was accepted to the University of Maryland, Baltimore County and given an academic scholarship for his exemplary achievement in High School. Shortly before leaving UMBC, Kory was blessed to meet his fiancé, Valerie Beach. Kory graduated Summa Cum Laude in December 2003. By January of the next year, Kory was attending North Carolina State University in pursuance of his Masters in Computer Engineering.

Kory's work experience includes a research internship at the University of Maryland, College Park where he mostly functioned as a test engineer for fabricated ASICs in the ECE department. The summers and winters of 2004 and 2005 were spent at the Columbia, Maryland business unit of Scientific Applications International Corporation (SAIC), where he was the hardware engineer on a project involving the design and testing of FPGA embedded systems. Currently, Kory has plans to obtain his doctorate degree at a later date.

Acknowledgements

First and foremost, I would like to thank my parents for giving me the opportunity and support to succeed in higher education institutions. They are responsible for keeping me grounded in life’s most important values. At every critical milestone in my life, my parents have been there for direction and kind words.

I would also like to acknowledge the loving support of my fiancé, whose strength and hopefulness during my time in North Carolina was remarkable. Without her dedication, I never could have ventured as far as I did.

A great deal of credit should be given to my advisor, Dr. W. Rhett Davis. Dr. Davis initially brought me to NC State and has seen me through my entire Masters degree. He never was at a loss for ideas to surmount any problems I had with in my research, and his enthusiasm in doing so is commendable. I would also like to thank Dr. Paul Franzon and Dr. Doug Barlage for serving so graciously on my Advisory Committee.

My “colleagues” in graduate school also deserve notice for their help in classes, research, and maintaining my sanity during the busiest times of each semester. In no particular order, I would like to thank the following people: Chris Mineo, Samson Melamed, Hao Hua, Ambarish Sule, Brandon Conover, Charles Suh, and anyone else I am forgetting.

Table of Contents

List of Tables

List of Figures

1 Introduction

The International Technology Roadmap for Semiconductors (ITRS) predicts the rapid increase in total interconnect length in the coming technology generations. In just three years, the amount of active wiring that will be necessary for traditional integrated circuits is expected to nearly double. Compounding the design challenges of engineers even more is the outlook for interconnect RC delays. In the same timeframe, the delays for local, intermediate, and global wiring are expected to increase by no less than 25% [1]. These trends can be seen in Figure 1-1 and Figure 1-2.

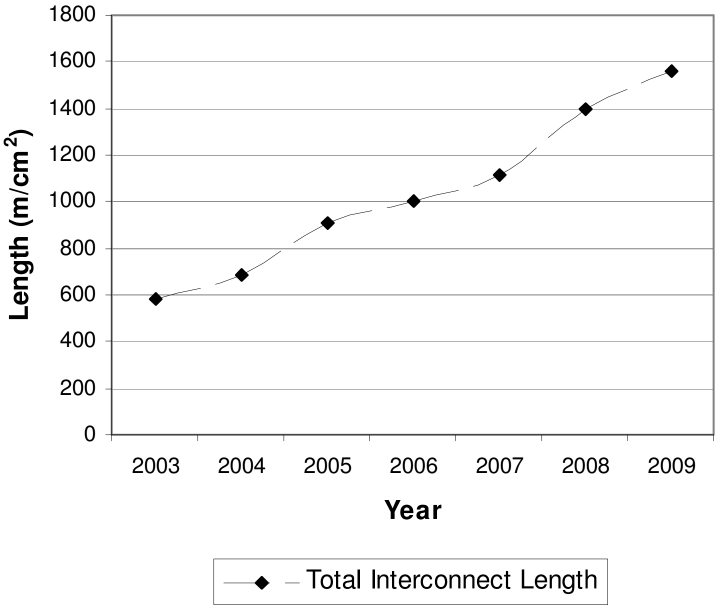


Figure 1-1: ITRS Data for the Total Interconnect Length of an IC

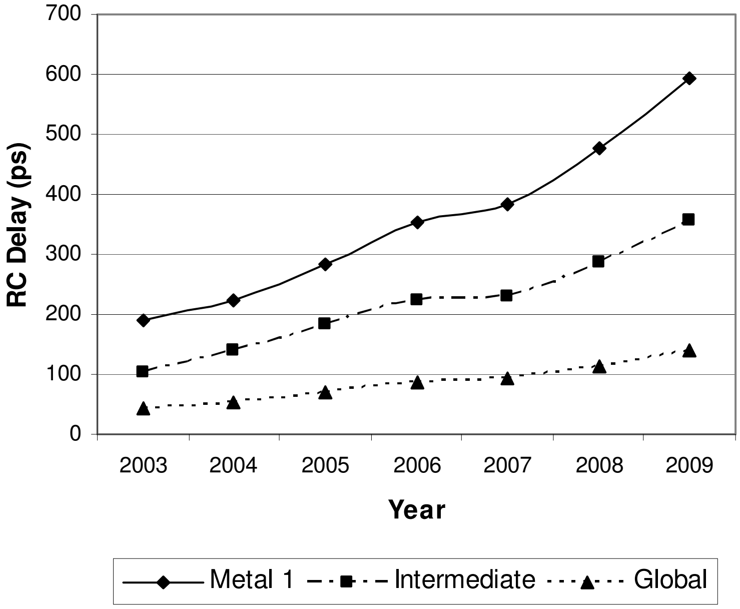


Figure 1-2: ITRS Data for Interconnect RC Delay for three types of interconnect

Although the nominal gate delays will continue to decrease with each successive generation for quite some time, the same cannot be said about the performance of the metallization used to provide connectivity between these gates. The apparent scaling gap between MOSFET device and interconnect performance has caused researchers to gain an interest in three-dimensional integration as a possible solution to the foreseeable limitations of standard two-dimensional circuit design. Three-dimensional integrated circuits (3DICs) alleviate the delay concerns of interconnects by offering the vertical axis as a possible routing direction. With the increasing popularity of System-on-Chip (SoC) architectural solutions, 3DICs can meet the density and functionality demands of these complex systems by "stacking" multiple tiers vertically and realizing inter-tier connectivity by means of short, vertical interconnects. It is the hope of the interested research community that 3DICs will help reduce the amount of delay and power that is "wasted" in wires.

In this work, a realistic 3D design test case based on the OpenRISC microprocessor is presented in order to qualify the relative advantages of 3DICs. Specifically, the design of an IC based on the OpenRISC Reference Platform System-on-Chip (ORPSOC) is detailed [2]. The test case is created by means of the 3DIC design flow of the Methodologies for User-Friendly System-on-a-chip Experimentation (MUSE) research group at North Carolina State University. This design flow is specific to MIT Lincoln Lab's fully-depleted silicon-on-insulator (FDSOI) 3D process [3]. Moreover, the design flow is unique in that it takes full advantage of the maturity of industry-standard 2D Electronic Design Automation (EDA) tools and simply adapts them to the 3D environment. It will be shown that, through intelligent user partitioning between the tiers and floor planning within each tier of a 3DIC, a 3D microprocessor-based design can operate at a higher clock frequency than feasible for a two-dimensional IC. The consideration of the performance gains from porting the design to 3D will also include the complexity issues of integrating multiple fabricated tiers in one package. These issues are largely a function of the thermal model of a 3DIC, which introduces the well-known poor heat removal ability of a multi-tiered chip [4]. Skeptics have always pointed to the concerns of the higher average chip temperatures of 3DICs, and this work will do the same.

In general, the interplay between a circuit's delay, power dissipation, and operating temperature has a degrading effect on its performance. A chip that is operating at a higher temperature will tend to slow down and dissipate even more power, which, in turn, will force a positive feedback loop between the power dissipation and temperature of an IC [5]. This work presents two models that can be used to predict the temperature dependence of power dissipation and combination path delay in order to capture the amount of performance degradation due to temperature in 3DICs. In doing so, this work presents an illustrative example of the aforementioned temperature-power positive feedback loop using the ORPSOC test case. It is shown that this iterative loop does converge upon a final average tier temperature for the 3D ORPSOC design and that the temperature is not high enough to counteract the speedup from using vertical interconnects.

The organization of this work is as follows. First, the derivation and verification of the two temperature-dependent circuit parameter models is reported. Next, the tool flows for estimating power dissipation and path delays of a 3D design are discussed. These will later be used to assess the performance improvement when moving from 2D to 3D and include consideration of heat generation in the 3DIC. Following the flow development, a brief overview of the FDSOI 3D process is described. Subsequently, an outline of the ORPSOC architecture used for the test case is presented. In particular, the discussion includes the merits of the ORPSOC as a 3D case study and how the ORPSOC was modified for experimentation. Afterwards, the 3DIC design flow used for the physical design of the test case is detailed. Once it is understood how to can characterize a 3DIC for delay and power per tier, the results from the timing and power analyses of the 2D and 3D designs can be used to gauge the effectiveness of 3D integration for the test case. To conclude, a summary of this work is presented in addition to ideas for areas of future work that could help reduce 3D design complexity.

2 Temperature Dependent Models for use in 3DICs

The thermal performance of 3DICs is a hot topic in the research community and one that still remains a mystery. A critical issue in the migration of interconnect-centric designs to the 3D environment is the non-uniform temperature profile across the tiers of a 3DIC. Since every tier beyond the bottom-most is handicapped by having heat-sources (active transistors) far away from the heat-removal fixtures, the average temperature of a tier increases as more tiers are added to the 3D stack and hence, the reliability of the IC worsens. One can no longer assume a single temperature from which to project the power and speed of an integrated system once it is transitioned to 3D. Two temperature dependent models have been developed and verified in this work to aid in the understanding of 3D circuit behavior. The phenomena of temperature dependence on power dissipation and combinational path delay are presented in this section. Acknowledging the temperature non-idealities, the former is used in the estimation of the increase in power dissipation and the latter is utilized to determine the degree to which a 3DIC slows down. Both of these models were verified against the actual results given in SPICE simulations using BSIM SOI models [6]. Good agreement is shown between these temperature dependent models and the complex transistor model of BSIM.

2.1 Power Dissipation Temperature Dependence

Recently, power dissipation has emerged as a limiting factor on the performance of integrated circuits. Circuit designers can no longer simply target speed as the ultimate measure of a chip’s performance. The motivation for this is not only in the prevalence of portable electronics that demand low-power operation but also in the continual scaling of the feature size of MOSFET devices. A great deal of interest in the power consumption of traditional 2D ICs has brought about an onslaught of low-power design methodologies as well as methods for the integration of power dissipation considerations into design flows. Especially central to the design of 3DICs, any accurate consideration of power dissipation must account for its dependence on the operating temperature of the dissipating device. With each tier of a 3DIC operating at a different average temperature, the initial characterization of power dissipation in the design flow could prove to be too optimistic.

2.1.1 Model Derivation

The three major sources of CMOS power dissipation are encapsulated in equation 2.1, where $P_{dynamic}$ is the contribution to total power dissipation from dynamic sources, P_{dp} is the power dissipated from direct-path currents, and P_{static} is the portion of power dissipation that is static [7].

$$P_{Total} = P_{dynamic} + P_{dp} + P_{static} \quad (2.1)$$

For the purpose of this work, direct-path power is ignored as it is generally a very small ratio of the total dynamic power. Moreover, direct-path power is a function of the peak current and node transition time. Both of these factors have opposite reactions to temperature change. Indeed, peak current decreases with increases in temperature, while transition times tend to increase (see section 2.2). The relative insensitivity of direct-path power to temperature is shown in Figure 2-1. Figure 2-1 tracks the normalized average direct-path energy of a CMOS inverter simulated with one rising and one falling transition, with the power being directly proportional to the energy. As indicated in the figure, the direct-path energy of the inverter deviates by no more than 3% from the nominal amount over a one hundred degree temperature range.

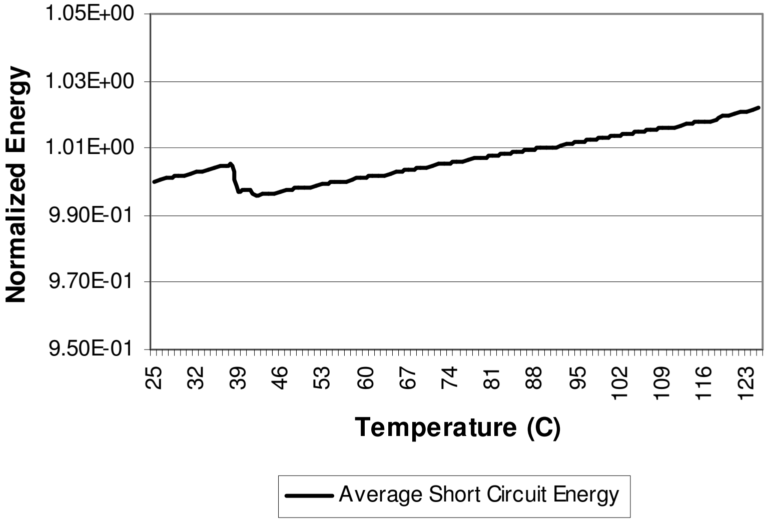


Figure 2-1: Normalized average short circuit energy for a CMOS inverter

When considering the final two components of power dissipation, it is well-known that dynamic power is generally insensitive to temperature change [8], [9]. This is supported by the accepted definition of dynamic power dissipation as shown in equation 2.2. In equation 2.2, C_L is the load capacitance of a gate, V_{DD} is the supply voltage, f is the clock frequency of the circuit, and $\alpha_{0->1}$ represents the probability that a clock event results in zero to one transition at the output of a gate [7]. Since all of the variables presented in equation 2.2 are independent of temperature, this directly implies that dynamic power is also independent of temperature. It is worth noting that the frequency of the clock is usually set by a phase-locked loop (PLL), which is designed to be insensitive to temperature.

$$P_{dynamic} = C_L V_{DD}{}^2 f \alpha_{0->1} \quad (2.2)$$

The equation for static power dissipation is shown in equation 2.3, where I_{leak} denotes the sub-threshold leakage current that exists between the supply rails when a transistor is inactive. It is this leakage current that gives static power dissipation its strong dependence on temperature, as detailed in this section.

$$P_{static} = I_{leak} V_{DD} \quad (2.3)$$

As discussed in [10], leakage currents are becoming a significant problem in circuit design as technology continues to scale. It is estimated by Berkeley predictive models in future technologies that leakage power will become so high that it can no longer be taken lightly [11]. Thus, when modeling temperature dependent parameters of a design, leakage power dissipation must be considered. The BSIM 3.3 model for the MOSFET drain current in the sub-threshold region is characterized in equation 2.4 and 2.5 [6].

$$I_{ds} = I_{s0}(1 - e^{(\frac{-V_{ds}}{V_t})})e^{\frac{V_{gs} - V_{th} - V_{off}}{nV_t}} \quad (2.4)$$

$$I_{s0} = \mu_0 \frac{W}{L} \sqrt{(\frac{q\varepsilon_{Si} N_{ch}}{2\phi_s}}) V_t^{\,2} \quad (2.5)$$

In these equations, V_t is the thermal voltage and is given by $K_B T/q$, where T is the temperature [6]. Although the BSIM model is complex, the scaling of drain current with temperature can be approximately reduced to $T^2 e^{1/T}$. Thus, leakage power is described as being exponentially dependent on temperature. In order to attempt to model this scaling behavior from a reference operating temperature, two published leakage models were tested. The first is from the work of [8] and is formulated directly from equations 2.4 and 2.5. This temperature dependent leakage model is shown equation 2.6, where β represents a curve-fitting parameter that is normally determined from SPICE simulation regressions.

$$I_{leak}(T) = I_{leak}(T_0) * T^2 e^{\frac{\beta}{T}} \quad (2.6)$$

Equation 2.6 predicts the leakage current at any arbitrary temperature, $I_{leak}(T)$, from a reference leakage current value, $I_{leak}(T_0)$. However, it was found to be very difficult and time-consuming to ascertain the value of β through SPICE simulations of an inverter using BSIM

SOI models. For this reason, a more-agreeable temperature dependent leakage model was chosen. Shown in equation 2.7, this model was specifically tested against BSIM SOI models and uses a second-order polynomial to describe the dependency [12]. The values for α_1 and α_2 are found using curve-fitting techniques.

$$I_{leak}(\Delta T) = I_{leak}(T_0)(1 + \alpha_1 \Delta T + \alpha_2 \Delta T^2) \quad (2.7)$$

According to equation 2.7, leakage power at an arbitrary temperature can be predicted using not the absolute temperature value but the departure from the reference temperature, ΔT. This model for leakage power more closely matched the actual values obtained from SPICE simulations.

2.1.2 Model Verification

To determine the values for the curve-fitting parameters in equation 2.7, a series of SPICE simulations were performed. Each test circuit was configured for a leakage power measurement. The inputs to the circuits were giving a static bit pattern, and the current through the supply was measured. The leakage power in watts is calculated by multiplying the current through the supply with the nominal supply voltage, which was 1.5 volts for the BSIM SOI model. The temperature parameter in SPICE was swept between 25 and 125 degrees Celsius, and the leakage power at each temperature was calculated. The test circuits for simulation were a number of the MUSE SOI standard cells in addition to some larger designs. Once all of the SPICE simulations were complete, the leakage power results from SPICE were collected into files and taken as input to a small curve fitting C program. For each test circuit, this program simply calculated the values of leakage power at every temperature and compared the calculated value with the SPICE value from the file. At every temperature node, the percentage of error between the SPICE value and the value generated from the model was saved. An average percentage of error was determined after all of the temperature nodes were visited. This average represented the error between SPICE and the model for one particular value of α_1 and α_2 . Iterations within a range of values for α_1 and α_2 were performed to arrive at the best possible error for each test circuit. The output of the program was a file containing a set of values for α_1 and α_2 that gave the best possible error for each test circuit. It was observed that the value for α_1 was exactly the same for all test circuits and that the value for α_2 varied very little. This was consistent with the work of [12]. Figure 2-2, Figure 2-3, and Figure 2-4 together show the quality of the prediction capability of the model versus the actual SPICE leakage power values. The test circuits for Figure 2-2, Figure 2-3, and Figure 2-4 are an inverter, a full adder, and a 16-bit ripple carry adder, respectively. The figures show that inaccuracy in the model is only present near the upper limit set on temperature. The model variables were set to 0.24 for α_1 and 0.0005 for α_2. Table 2-1 summarizes the leakage power error in the test circuits when using these values.

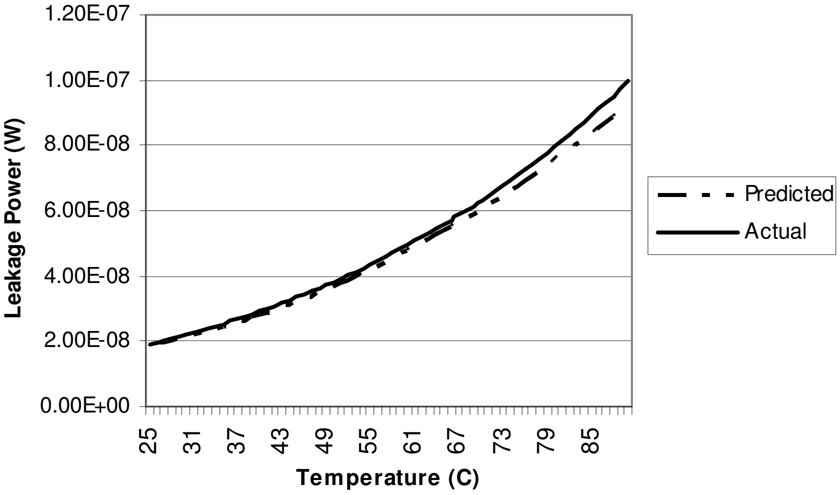


Figure 2-2: Predicted versus actual leakage lower for an inverter

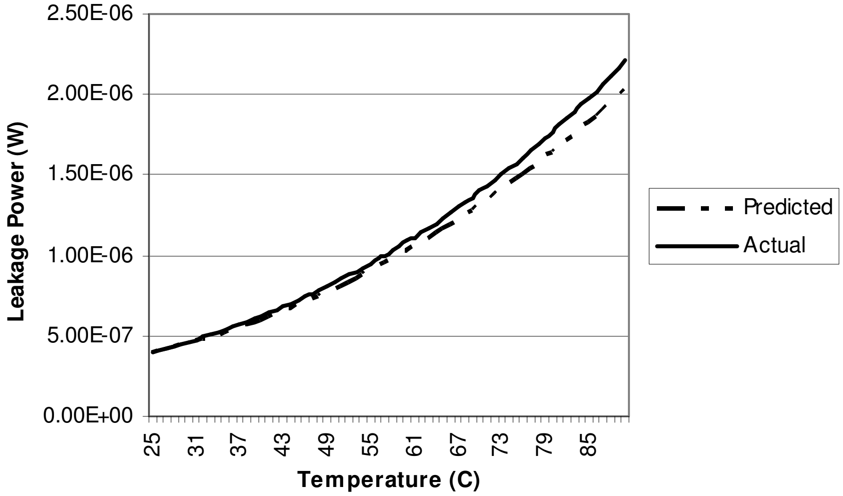


Figure 2-3: Predicted versus actual leakage Power for a full adder

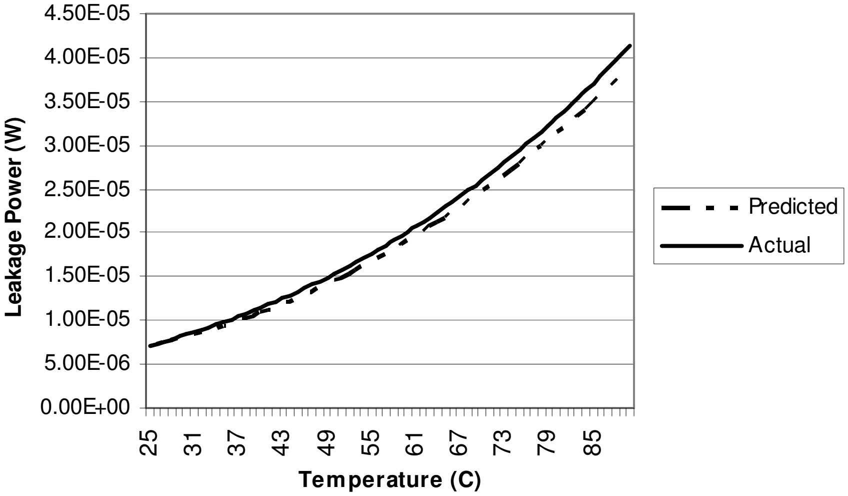


Figure 2-4: Predicted versus actual leakage power for a 16-bit ripple carry adder

Table 2-1: Leakage power model as compared to SPICE simulations

Cell Name	Average % of Error
INV	3.54
2-Input NOR	2.48
2-Input NAND	3.24
2-Input XOR	3.2
Full Adder	3.81
16-bit Ripple Carry Adder	4.70

2.2 Delay Temperature Dependence

Techniques for estimating the frequency achievable for a design at a nominal operating temperature are commonplace in the industry. Recently, interest has been invested in the research of methods for calculating full-chip thermal profiles. Using this thermal profile, the performance can be ascertained for specific temperatures at locations on the chip. The same concept can be applied to the tiers of a 3DIC. Particularly, the temperature profile across the tiers of a 3DIC can be used to estimate the slow down of the circuits on the each tier. This slow down is measured by the increase in combinational path delay due to temperature effects on MOSFETs. In a digital circuit, an increase in critical path delay has an inverse effect on the maximum clock frequency attainable. Therefore, it is vital to gain an understanding of the degree to which the speed of a 3D design derates from its synthesized speed target. Modeling the temperature dependence of delay in a digital design from a reference temperature is an efficient means of quickly estimating clock frequency requirements.

2.2.1 Model Derivation

The MOSFET drain current, I_D, can be expressed in terms of the alpha model shown in equation 2.8 [13]. In this equation, $\mu(T)$ and $V_{TH}(T)$ denote the temperature dependence of carrier mobility and threshold voltage, respectively.

$$I_D \; \alpha \; \mu(T)(V_{DD} - V_{TH}(T))^{\alpha} \quad (2.8)$$

Based on this equation for drain current, there is an inherent competition between the two temperature sensitive variables when temperature is varied. Since both carrier mobility and threshold voltage decrease as temperature increases, one works to increase drain current while the other works in the opposite manner. It was noted in [13] that, for supply voltages above one volt, the temperature dependence of carrier mobility tends to dominate. Hence, the drive current of a MOSFET has an inverse dependence on temperature, and because of this, the delay of a CMOS circuit will increase with temperature.

The model used to predict the temperature dependence of delay is largely based on [14], which showed that any combinational delay will hold constant over a wide range of technologies, temperatures, and voltages when normalized to the delay of an FO-4 inverter that is subject to the same changes. It is for this reason that the FO-4 inverter delay is often used as the metric for the speed of a technology. The term “FO-4” stands for “fan-out of four” and implies a circuit composed of an inverter loaded by four copies of itself. As opposed to measuring the delay increase with temperature for every circuit topology, one can predict with relatively high accuracy the degree to which an arbitrary circuit will slow down using the known slowdown of a FO-4 inverter at the same temperature. A prerequisite to adopting this approach is the formulation of an equation to predict the slow down of the FO-4 inverter. The delay model that incorporates temperature scaling is shown in equation 2.9 [8].

$$delay\ \alpha \frac{V_{dd} T^{\mu}}{(V_{dd} - V_t)^{\xi}} \qquad (2.9)$$

As in the case of the temperature dependent leakage model, μ and ξ are curve fitting parameters. The logic in place here is that, once the behavior of an FO-4 inverter across a temperature range is known, it is reasonable to assume that any combinational circuit will behave similarly. Thus, consistency between equation 2.9 and SPICE simulations is required to validate this temperature dependent delay model.

2.2.2 Model Verification

The FO-4 inverter was designed using five identical copies of the smallest inverter in the SOI standard library. To simulate this circuit in SPICE, the input was provided with a signal having exactly one rising and one falling transition. The transition time was set to 100 picoseconds, and the delay for both transitions was calculated with measurement statements in the netlist. The propagation delay was measured from the 50% points of the input and output waveforms. The average of the two delays across a temperature range of 25 to 125 degrees Celcius was used to determine the values for the curve fitting parameters of equation 2.9. A similar procedure as detailed in section 2.1.2 was performed to find the best average error between the model and the SPICE results. For the temperature range mentioned above, the best error was found to be 0.88%, which corresponds to the values of 0.09 and 8.51 for μ and ξ in equation 2.9, respectively. Figure 2-5 shows the predicted versus actual delay of the FO-4 inverter. The y-axis of Figure 2-5 is the delay normalized to the reference temperature. To verify that the FO-4 inverter is an excellent determinant for the temperature dependence of delay, additional circuits were simulated. Figure 2-6 and Figure 2-7 show the accuracy of the model for an AND gate and a 16-bit ripple carry adder. The average error across the entire range of temperatures for Figure 2-6 was just 1.89% and just 1.49% for Figure 2-7**.**

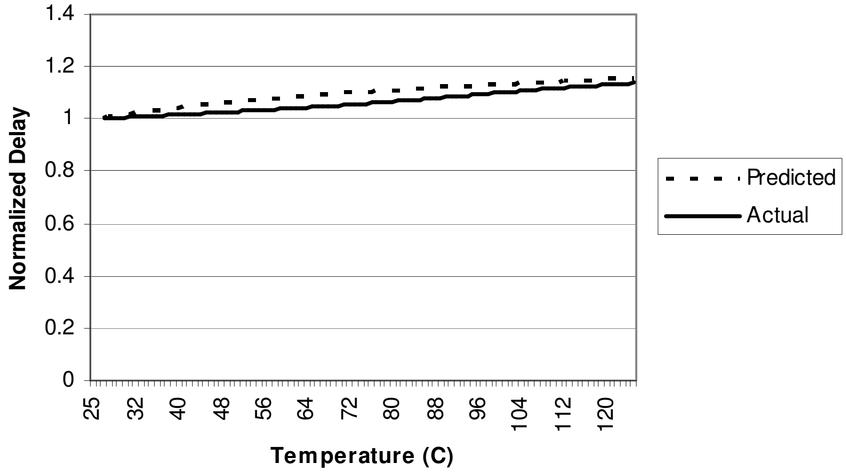


Figure 2-5: Predicted versus actual delay of an FO-4 inverter

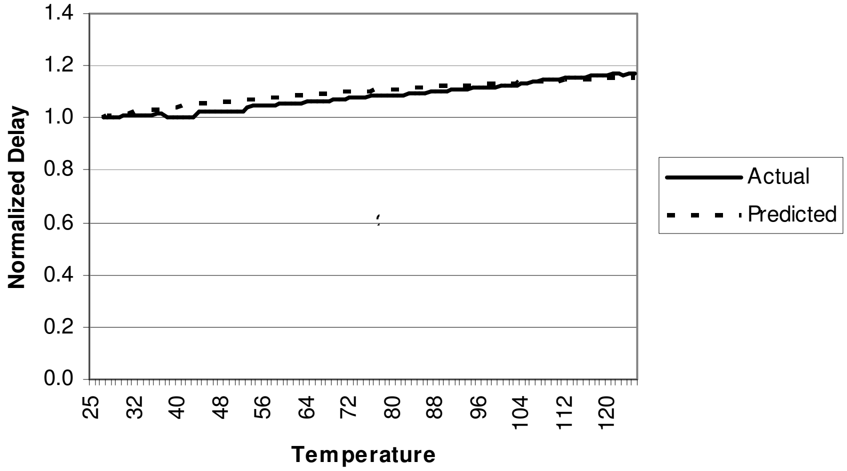


Figure 2-6: Predicted versus actual delay of an AND gate

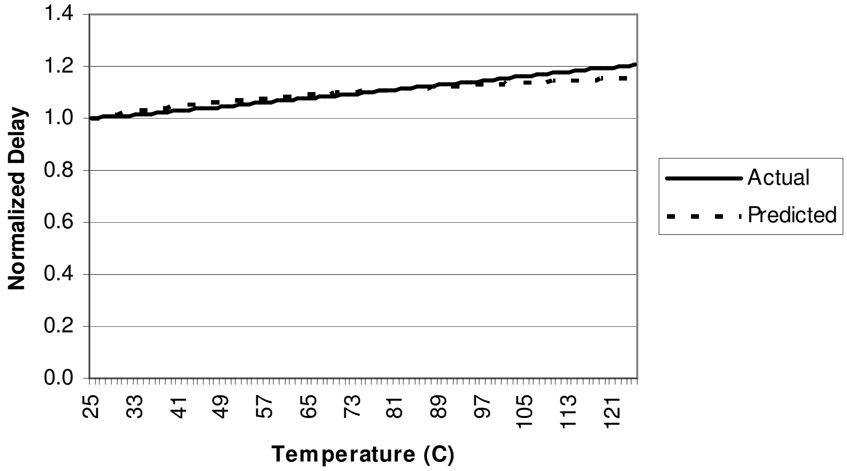


Figure 2-7: Predicted versus actual delay of a 16-bit ripple carry adder

3 Testcase Power Dissipation and Delay Estimation Flow

Realistic power dissipation and timing delay values for the ORPSOC test case are needed for evaluating the performance of a 3D design relative to a 2D one. Additionally, the previous section's formulation of temperature dependent models relies on accurate power and delay numbers at a reference temperature as a starting point for any predictive calculations. In this section, the procedure for characterizing the test case in terms of power dissipation and combinational path delays is detailed. For an increased level of accuracy, the parasitics of the wires in the layout of the test case (see sections 6.6 and 6.8) were incorporated into this procedure. For the purposes of analyzing both the 3D and 2D designs of the test case, the structure of the flows for obtaining power dissipation and delay values are generalized.

3.1 Initial Power Dissipation Estimation Flow

The process of finding realistic power dissipation values for a complex design reduces to realizing a switching activity profile that spans all of the nets in the design. Once the switching activity is known, the appropriate tools can be used to analyze the design for power dissipation. Recall from section 2.1.1 that switching activity of a gate is needed in the calculation of dynamic power dissipation. The value for the leakage power of a gate is constant (if using an average value) and is given in the standard cell library files once the library is characterized for power. Creating the library files is beyond the scope of this work and will not be covered here. It is assumed that every standard cell in the design has a leakage power section in the library files. The complete procedural flow for obtaining power dissipation of the ORPSOC test case is shown in Figure 3-1. An explanation of each step follows.

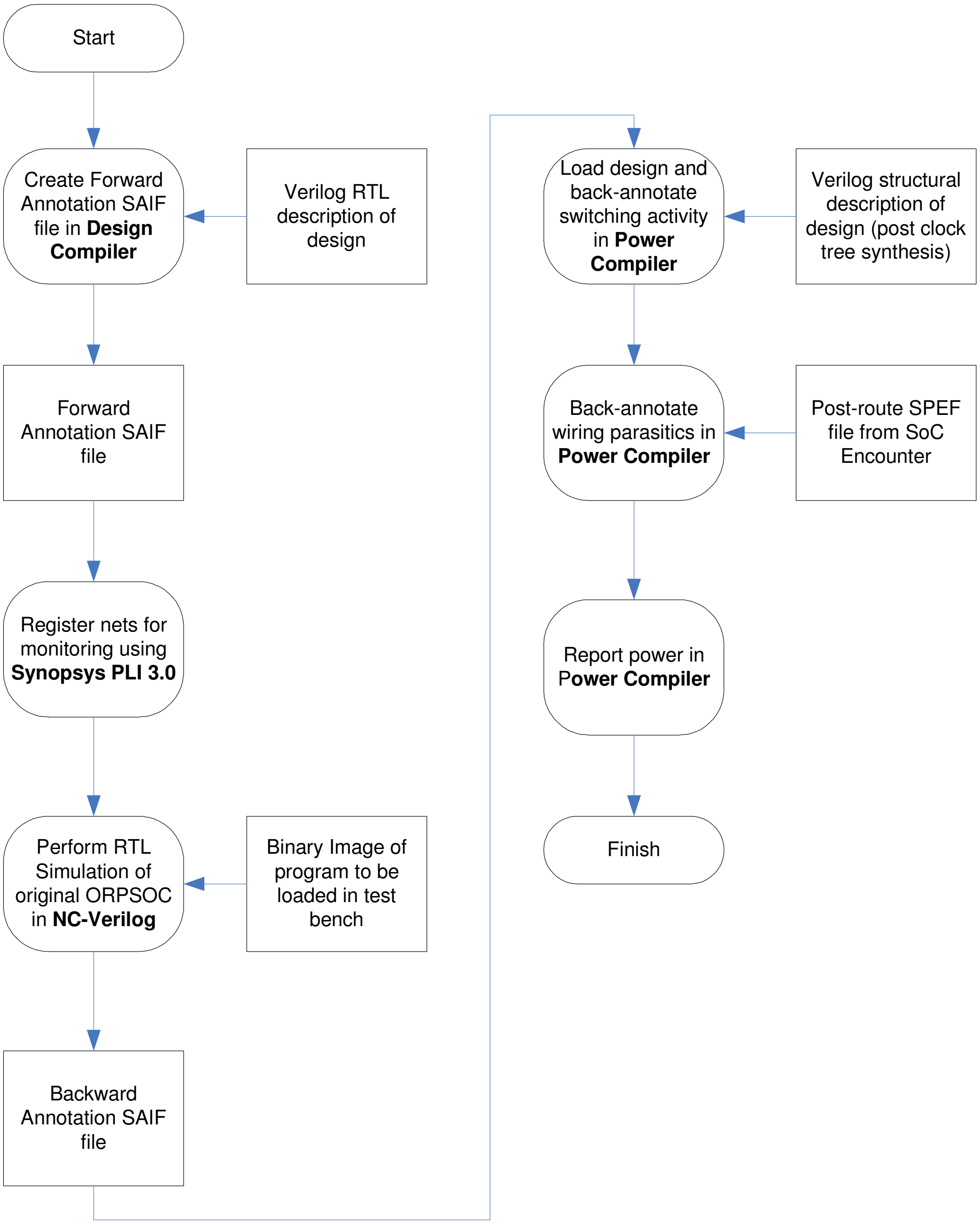
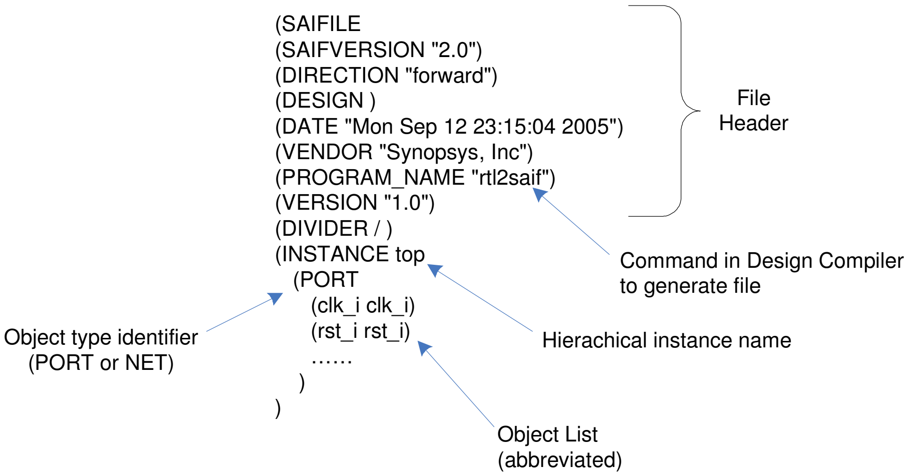

Start
Create Forward Annotation SAIF file in **Design Compiler**
Verilog RTL description of design
Forward Annotation SAIF file
Register nets for monitoring using **Synopsys PLI 3.0**
Perform RTL Simulation of original ORPSOC in **NC-Verilog**
Binary Image of program to be loaded in test bench
Backward Annotation SAIF file
Load design and back-annotate switching activity in **Power Compiler**
Verilog structural description of design (post clock tree synthesis)
Back-annotate wiring parasitics in **Power Compiler**
Post-route SPEF file from SoC Encounter
Report power in Power Compiler
Finish

Figure 3-1: Flow for obtaining the power dissipation of the ORPSOC test case

3.1.1 Forward Annotation SAIF File

A special type of file is used throughout this power flow to allow information exchange between tools. Known as switching activity interchange format, or SAIF, this Synopsys-developed file type streamlines the capturing and annotation of switching activity for RTL and gate-level designs. To begin, a forward annotation SAIF file is created in Design Compiler. Once written, this SAIF file contains a listing of all of the “synthesis-invariant” components of the design with no appended switching activity information. The classification of a port or net as being “synthesis-invariant” implies that it will be preserved during the synthesis process. Objects that are “synthesis-invariant” include hierarchical ports and sequential elements such as registers and memory arrays. The format of a forward annotation SAIF file is shown in Figure 3-2. A similar format will be used later in the flow for the back annotation SAIF file with the inclusion of switching activity information.



```
(SAIFILE
(SAIFVERSION "2.0")
(DIRECTION "forward")
(DESIGN )
(DATE "Mon Sep 12 23:15:04 2005")
(VENDOR "Synopsys, Inc")
(PROGRAM_NAME "rtl2saif")
(VERSION "1.0")
(DIVIDER / )
(INSTANCE top
  (PORT
    (clk_i clk_i)
    (rst_i rst_i)
    ……
  )
)
```

File Header
Command in Design Compiler to generate file
Hierachical instance name
Object type identifier (PORT or NET)
Object List (abbreviated)

Figure 3-2: Format of the forward annotation SAIF file

To write a forward annotation SAIF file in Design Compiler, the RTL description of the design must first be loaded and linked. Then, the “rtl2saif” command can be used to interpret the design, compile a list of synthesis-invariant objects, and finally write a forward annotation SAIF file. In an effort to exploit the existing RTL test bench that was packaged with the ORPSOC, the SAIF file was written at the OR1200 level of hierarchy.

3.1.2 Synopsys PLI 3.0

Now that the forward annotation SAIF file has been created, it must be recognized by a Verilog simulator before any switching activity can be captured. The mechanism to integrate the information contained within the forward annotation SAIF file with the Verilog simulator is the Synopsys PLI 3.0 [15]. “PLI” stands for programming language interface and is a feature of the Verilog compiler that allows the linking and execution of standard C/C++ programs within a Verilog source file. PLI is most useful for calling programming language functions within a test bench for statistical bookkeeping purposes. Synopsys provides Verilog PLI libraries to support the gathering of switching activity information for power analysis. NC-Verilog was chosen for Verilog simulation of the ORPSOC, but the Synopsys PLI libraries are functional for a wide range of simulators. For dynamic linking of the libraries, the “+loadpli1” command line argument is needed when invoking NC-Verilog. With the libraries loaded into the simulator, switching activity-specific functions appear as normal system tasks in the Verilog source files. A total of five Synopsys PLI functions are needed to capture the switching activity of a design during RTL simulation. These commands and their meanings are shown in Figure 3-3 as they appear in the test bench. With the link between the forward annotation SAIF file and the Verilog simulator established, RTL simulations can be performed to find realistic switching activities.

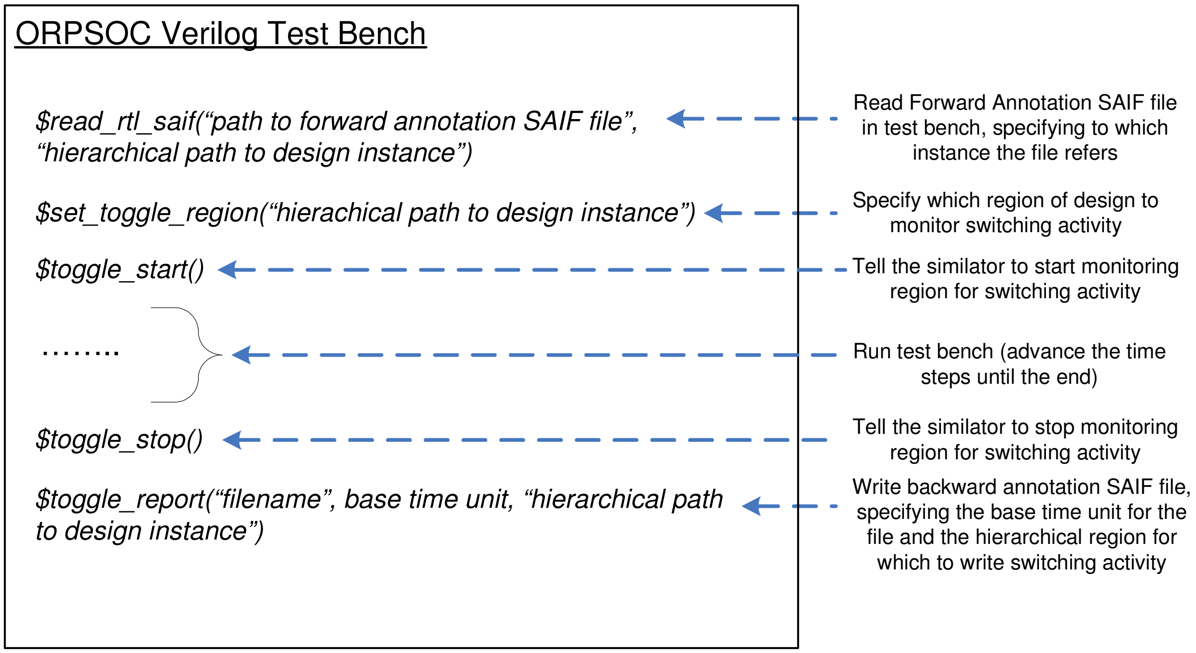
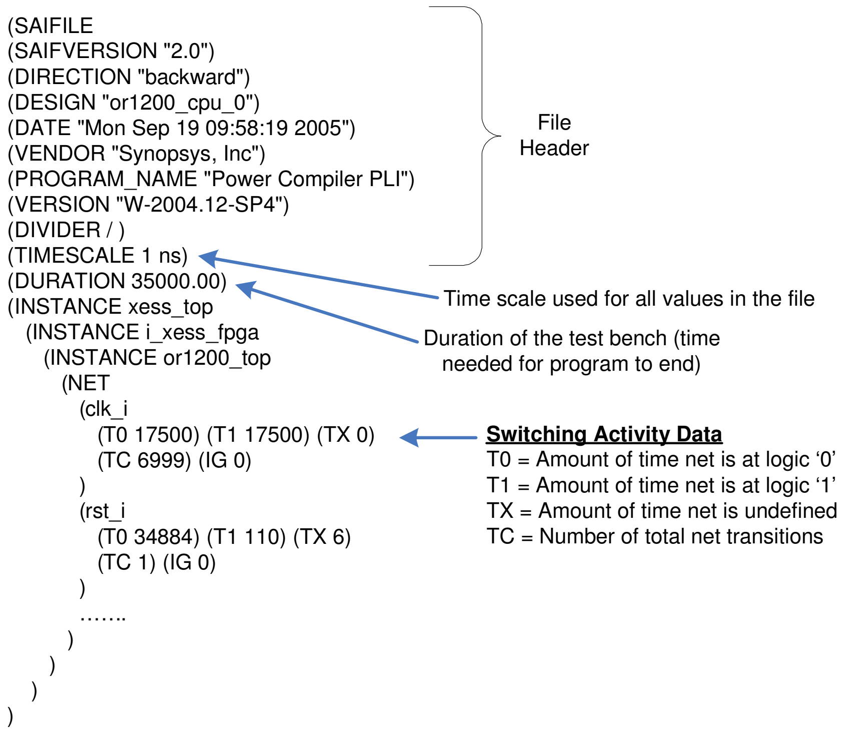

ORPSOC Verilog Test Bench
$read_rtl_saif("path to forward annotation SAIF file", "hierarchical path to design instance")
Read Forward Annotation SAIF file in test bench, specifying to which instance the file refers
$set_toggle_region("hierachical path to design instance")
Specify which region of design to monitor switching activity
$toggle_start()
Tell the similator to start monitoring region for switching activity
……..
Run test bench (advance the time steps until the end)
$toggle_stop()
Tell the similator to stop monitoring region for switching activity
$toggle_report("filename", base time unit, "hierarchical path to design instance")
Write backward annotation SAIF file, specifying the base time unit for the file and the hierarchical region for which to write switching activity

Figure 3-3: Synopsys PLI 3.0 functions in the test bench [15]

3.1.3 RTL Simulations with NC-Verilog

Bundled in the ORPSOC package is a suite of software with which to test designs. Most of these programs offer very basic functionality tests of the microprocessor. Distinct C programs exist for multiple-and-accumulate instructions, exceptions, MMU testing, system calls, and even the Dhrystone benchmark. These programs were compiled to OpenRISC microcode using the OpenRISC GNU toolchain [16]. Assuming a working installation of the toolchain, the provided makefiles for the software automate the compilation and image conversion from binary to hexadecimal, as needed by the Verilog simulator. Also included in the ORPSOC package is support for running RTL regression tests on the compiled software. The scripts for regression testing merely copy the hexadecimal image of the compiled program to another file that is loaded into the Flash memory model in the test bench. For the purposes of this work, these scripts were modified to include the required "+loadpli1" command line argument for the loading of the Synopsys PLI 3.0 at invocation of NC-

Verilog. Next, the bundled test bench for the ORPSOC was modified to include the functions shown in Figure 3-3. Finally, the test bench was simulated once for every compiled program using the OR1200 hardware configuration (i.e. disabled caches, ASIC-optimized multiplier, etc) and clock frequency of the test case. The embedded PLI functions in the test bench save a backward annotation SAIF file for each simulation, and consequently, the switching activity profile for each program is obtained. The differences between the backward and forward annotation SAIF files are noted in Figure 3-4**.** The instance hierarchy in the figure indicates that the switching activity was captured at the OR1200 level of hierarchy (denoted as “or1200_top” in the figure). The reason for this is that the test bench did not reflect the addition of the second OR1200 as used in the test case (see section 5.1). For the power analysis of the ORPSOC test case, it is assumed that the microprocessors have identical switching activity profiles.



```
(SAIFILE
(SAIFVERSION "2.0")
(DIRECTION "backward")
(DESIGN "or1200_cpu_0")
(DATE "Mon Sep 19 09:58:19 2005")
(VENDOR "Synopsys, Inc")
(PROGRAM_NAME "Power Compiler PLI")
(VERSION "W-2004.12-SP4")
(DIVIDER / )
(TIMESCALE 1 ns)
(DURATION 35000.00)
(INSTANCE xess_top
  (INSTANCE i_xess_fpga
    (INSTANCE or1200_top
      (NET
        (clk_i
          (T0 17500) (T1 17500) (TX 0)
          (TC 6999) (IG 0)
        )
        (rst_i
          (T0 34884) (T1 110) (TX 6)
          (TC 1) (IG 0)
        )
        …….
      )
    )
  )
)
```

File Header

Time scale used for all values in the file

Duration of the test bench (time needed for program to end)

Switching Activity Data
T0 = Amount of time net is at logic '0'
T1 = Amount of time net is at logic '1'
TX = Amount of time net is undefined
TC = Number of total net transitions

Figure 3-4: Format of the backward annotation SAIF file

3.1.4 Back Annotation of Switching Activity in Power Compiler

The tool used for power analysis of the ORPSOC test case was Synopsys' Power Compiler [17]. Power compiler operates from the same command line as Design Compiler, and in fact, just a simple license check-out is needed to enable it. Once the structural Verilog netlist is loaded into Power Compiler, the "read_saif" command can be executed to annotate the nets of the gate-level design with the switching activity from the backward annotation SAIF file. For the test case, switching activity annotation was performed for both OR1200 modules, necessitating the use of "read_saif" for both OR1200 designs in the hierarchy. The clock frequency in the design must be set to the same frequency with which RTL simulations were performed or the power estimation will not be accurate.

3.1.5 Back Annotation of Parasitics in Power Compiler

Back annotation of interconnect parasitics is necessary to consider the differences in interconnect power dissipation between 3D and 2D designs. Moreover, the increased capacitive loading from the interconnect wires often has a dramatic effect on the total power dissipation, making it doubly important to remove the notion of ideal wires in the design. Referring to section 6.6 and 6.8, the SPEF file was created for the sole purpose of parasitic back annotation. Using the "read_parasitics" command in Power Compiler, the parasitic routing information can be annotated onto the nets of the design. In the case of the 3D design, the merged 2D SPEF file (see section 6.8) is used for annotation of the top level partitioned netlist after the insertion of the clock tree (see section 6.7).

3.1.6 Report Power in Power Compiler

The final step in the power dissipation flow is the execution of the gate-level power analysis engine via the "report_power" command. Once this command is invoked, Power Compiler uses the annotated switching activity to compute the total power dissipation of the design. Since only the synthesis-invariant objects were monitored during the RTL simulations, the internal nets that changed during synthesis have no associated switching activity. For these nets, Power Compiler uses a zero-delay simulation to propagate switching activity [17]. As long as all of the primary inputs, hierarchical ports, and sequential nets (i.e. clocked elements that were synthesized to flip-flops) are covered by the back annotation SAIF file, every net in the design will be given a switching activity. The "report_saif" command can be used in Power Compiler to ensure that no nets were assigned the default value for switching activity. Upon completion of the power analysis, "report_power" prints a power breakdown that consists of the dynamic and leakage power dissipation. In order to infer the contributions to power dissipation from each module in the design, the “-hier” argument can be used with “report_power”. Due to the fact that a tier is viewed simply as a level of hierarchy in the netlist, the “-hier” option is useful in determining the amount of power dissipated on each tier of the 3DIC.

3.2 Initial Delay Estimation Flow

Contrary to the power dissipation estimation flow, the procedure to extract timing delays from the ORPSOC test case is relatively straightforward. An overview of the delay estimation flow is depicted in Figure 3-5.

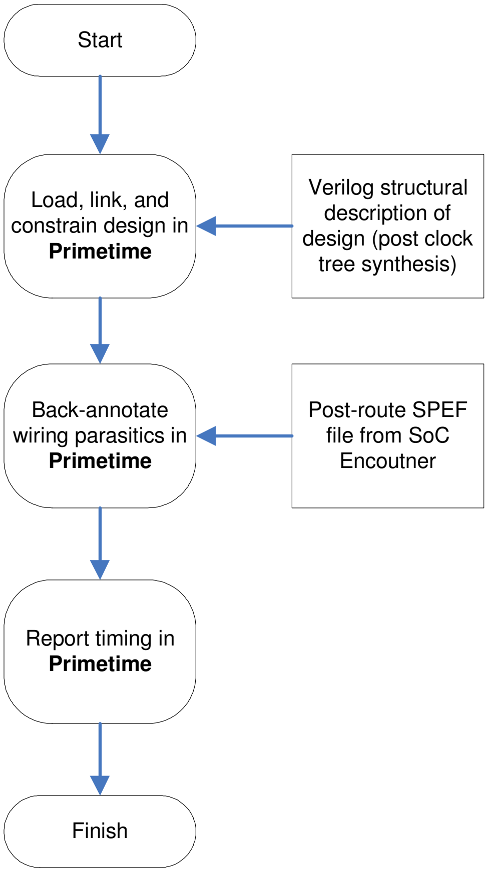

Start
Load, link, and constrain design in **Primetime**
Verilog structural description of design (post clock tree synthesis)
Back-annotate wiring parasitics in **Primetime**
Post-route SPEF file from SoC Encoutner
Report timing in **Primetime**
Finish

Figure 3-5: Flow for obtaining timing delay values of the ORPSOC test case

As noted in the figure, the timing analysis tool used was Synopsys' PrimeTime [18]. PrimeTime is a full-chip gate-level static timing analysis tool. To facilitate the use of the temperature dependent delay model developed in section 2.2, the longest register-to-register path delay for specific points in the test case was examined in PrimeTime. Identical to the use of Power Compiler and most other Synopsys tools, the structural Verilog netlist is first loaded and linked. Afterwards, the clock frequency, skew, and input/output loading constraints are specified. The same constraints used for synthesis are acceptable for use here (refer to section 6.1 for these constraints) and can even be entered verbatim given PrimeTime's use of Design Compiler commands. The emphasis in the use of PrimeTime is on the absolute delay values of combinational paths and not on timing slack, so accurate representation of clock skew is not a requirement. Following the constraining of the design, the "read_parasitics" command is reused to annotate the wiring information from the SPEF file. The PrimeTime command that starts the timing analysis is "report_timing". This command can be used in a myriad of ways, but in the absence of any arguments, the critical path of the design is returned. The longest delay can also be restricted on the basis of "to", "from", and/or "through" lists. When a "from" list is used as an argument, for instance, the longest delay starting from any of the nets in the list is computed. Similarly, a "to" list will find the longest delay ending at one of the nets in the list. This level of control with "report_timing" is exploited to determine the speedup from the use of 3D vias.

3.3 Converging to Final Power Dissipation and Delay Values

In the previous sections, temperature dependent models and tool flows for extracting initial power dissipation and timing delay values have been developed. These can then be used in conjunction with a thermal model of a 3DIC to ascertain the amount of performance degradation from the temperature gradient across the 3D stack. The thermal properties of 3DICs are more influential on circuit performance than that of conventional integrated circuits. This is due to the fact that there are multiple junction-to-ambient thermal resistances to consider in a 3DIC. As such, a large temperature gradient across the tiers of a 3DIC will cause the transistors on the top tier to consume more power and run more slowly than the transistors on the bottom tier. It is the aim of this section to provide a method for the estimation of performance degradation in the face of 3DIC thermal properties.

3.3.1 Thermal Model of a 3DIC

According to the well-known thermal and electrical phenomena duality, the average temperature drop across a tier of a 3DIC can be viewed as the difference between two node voltages. This temperature drop, ΔT, is related to the heat flow (power dissipation) of the tier, P, and the thermal resistance of the tier, θ, as indicated by equation 3.1.

$$\Delta T = P\theta = P\frac{L}{kA} \quad (3.1)$$

In this equation, the thermal resistance is expanded as a function of the thickness of the tier, L, the thermal conductivity of the tier, k, and the area of the tier, A. In keeping with the equivalent electrical circuit, the power dissipation of a tier is analogous to a current source with the current directed into the tier. The direction of the current source implies an assumption that the heat transfer in the tier is unidirectional and towards the heat-removing fixtures of the package. An average temperature gradient across the tier is also assumed by the single current source representing the total power dissipation of the tier. This thermal model does not account for the existence of localized “hot spots” that arise from non-uniform power density (P/A in equation 3.1) within the tier.

Figure 3-6 shows the simplified thermal model of a 3DIC used in this work. Based on the 3D FDSOI thermal model developed in [19], the left side of the figure shows the physical composition of a packaged 3DIC, and the right side shows its transformation to the equivalent electrical circuit. The tier thicknesses shown in the figure are calculated as the distance between the transistors on the respective tiers, since these are the power dissipating devices in the 3DIC. The remaining thicknesses are taken directly from the assumptions of [19]. The physical construction of a 3DIC (see section 4.1) dictates that there is only one current source injecting current into tier A and tier B. This allows for the treatment of these tiers as one equivalent resistive component with a thermal resistance labeled "$\theta_{\text{tier AB}}$" in the figure. The temperatures in the packaged 3DIC are shown as node voltages in the figure. Specifically, "$T_{\text{tier C}}$", "$T_{\text{tier B}}$", and "$T_{\text{tier A}}$" represent the average temperature seen by the transistors on the respective tiers of the 3DIC.

The bounds on the thermal conductivity, k, of the tiers of a 3DIC are a function of the 3D via density and the amount of metallization used on each tier. For instance, increasing the number of 3D vias connecting tier C with tier B will reduce the effective thermal resistance of tier C (shown as "$\theta_{\text{tier C}}$" in Figure 3-6). Likewise, a dense tier that uses more routing resources will exhibit a lower thermal resistance than a sparsely-placed tier having a higher ratio of oxide (poor thermal conductor) to metallization (good thermal conductor). The bounds on the thermal conductivity for a 3DIC fabricated in the FDSOI 3D process were reported in [19]. Given the 3D via density and tier area (see Figure 6-9) of the test case, these bounds combined with equation 3.1 were used to determine the associated thermal resistances in Figure 3-6. For the purposes of this work, the heat sink was assumed to be perfect having zero thermal resistance. Also, the same assumption for the conductivity of the epoxy in [19] was used here.

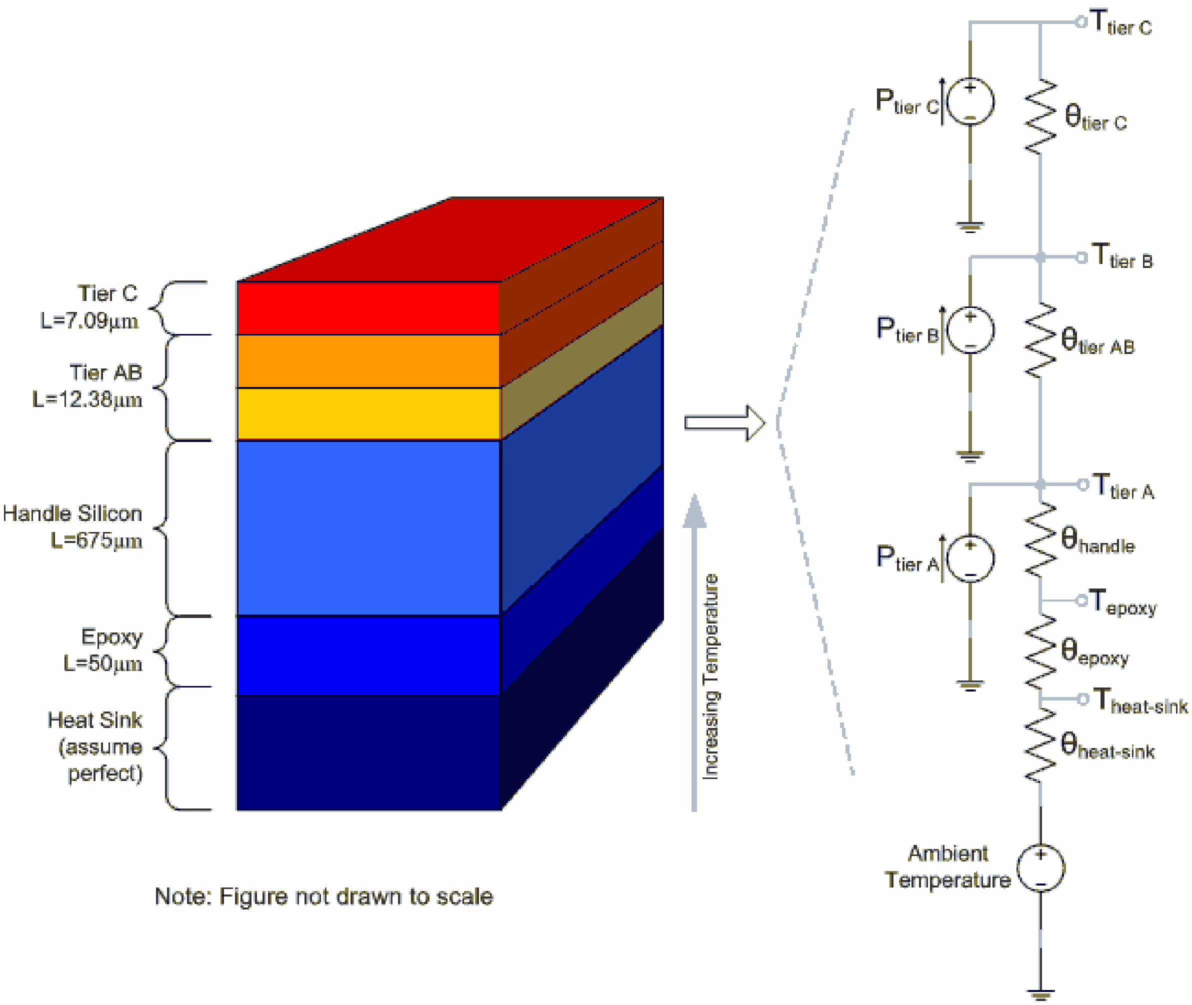


Figure 3-6: Simplified thermal model of a 3DIC

3.3.2 Temperature-Power Positive Feedback Loop

As noted in equation 3.1, the temperature across a material is directly proportional to the heat flow (power) through it. In turn, the power dissipation of a MOSFET has a dependence on temperature. More specifically, it was show in section 2.1.1 of this work that the leakage power of a transistor has an exponential response to temperature change. This inter-dependence on one another creates a positive feedback loop between the temperature and power dissipation of an integrated circuit. In fact, the phenomena known as "thermal runaway" can occur if no dampening is present in the feedback loop. This situation arises whenever the increased levels of heat generation caused by the feedback loop exceed the heat removal ability of the system [8]. In order for a system to be stable, the temperature-power positive feedback loop must converge upon a final temperature value. Using the thermal model in Figure 3-6 together with the predicted temperature scaling behavior of power dissipation, the extent of the feedback loop for a 3DIC can be analyzed. Once temperature convergence is achieved, the delay slow down and power dissipation increase per tier, as previously modeled in section 2, can be projected.

An example of the positive feedback loop as it pertains to this work is shown in Figure 3-7. To begin, the initial values for power dissipation are entered into the 3DIC thermal model to compute the initial temperature values. This computation is governed by the assumptions of the model in Figure 3-6 as well as equation 3.1. A flow for obtaining the power dissipation of the ORPSOC was previously discussed in section 3.1. Once the initial values for the temperatures of the 3DIC are known, the feedback loop is enabled. The temperature dependent model for leakage power is used to compute the new value for the leakage power on each tier. Next, these new leakage power numbers are added to the constant dynamic power values to find the total power dissipation for each tier of the 3DIC. The next iteration is initiated by re-computing the temperature values based on the updated total power dissipation. Iterations continue until temperature convergence is observed. No particular threshold was set for temperature convergence; instead, a visual inspection of the temperature change across iterations was performed. When enough cycles have been completed to achieve convergence, the final temperatures are used to compute the amount of the slow down of the combinational paths on each tier as per the temperature dependent delay model (see section 2.2). Slow down is measured as the percentage increase in delay as referenced from the ambient temperature. The predicted values for final power dissipation and slow down of the 3D test case are considered in the comparison to the 2D design later in this work.

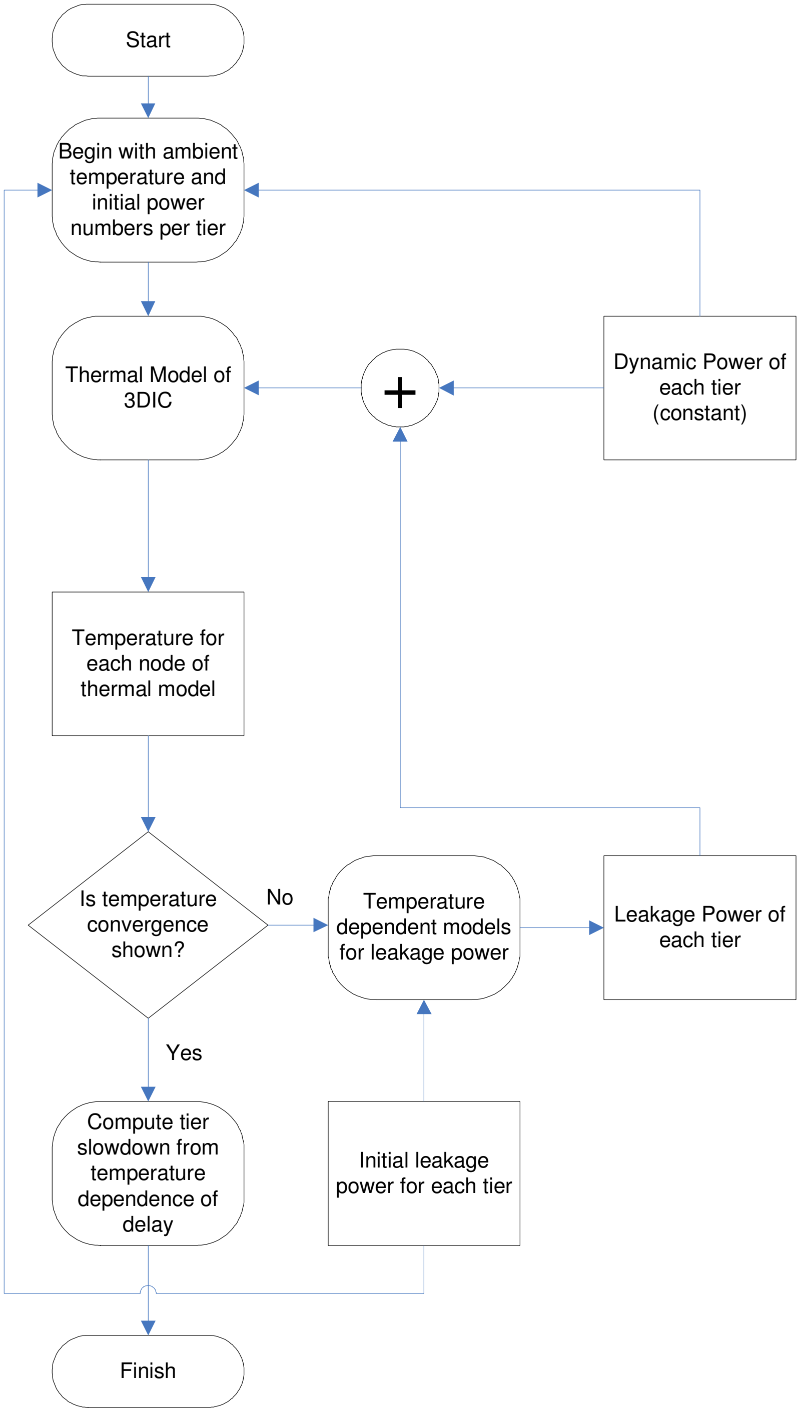
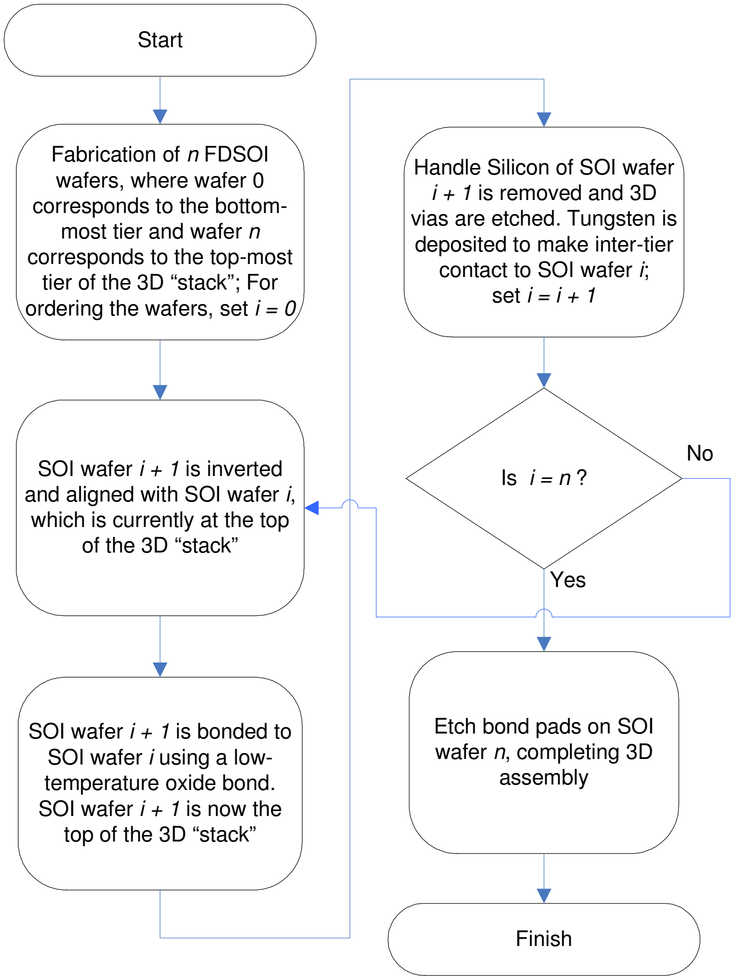

Start
Begin with ambient temperature and initial power numbers per tier
Thermal Model of 3DIC
+
Dynamic Power of each tier (constant)
Temperature for each node of thermal model
Is temperature convergence shown?
No
Yes
Temperature dependent models for leakage power
Leakage Power of each tier
Compute tier slowdown from temperature dependence of delay
Initial leakage power for each tier
Finish

Figure 3-7: Temperature-power positive feedback loop

4 The FDSOI 3D Process

4.1 3D Circuit Fabrication

MIT's Lincoln Laboratory (MITLL) has developed a 3DIC fabrication technology whereby circuit structures on multiple SOI substrates are combined to form a single integrated 3D circuit [20]. Each SOI substrate, or tier as it is referred to in the 3DIC process, is fabricated in the same manner as other FDSOI processes, but that is where the similarities with 2D integration end. Figure 4-1 provides a step-by-step outline of the FDSOI 3D process.



Start
Fabrication of *n* FDSOI wafers, where wafer 0 corresponds to the bottom-most tier and wafer *n* corresponds to the top-most tier of the 3D "stack"; For ordering the wafers, set *i = 0*
SOI wafer *i + 1* is inverted and aligned with SOI wafer *i*, which is currently at the top of the 3D "stack"
SOI wafer *i + 1* is bonded to SOI wafer *i* using a low-temperature oxide bond. SOI wafer *i + 1* is now the top of the 3D "stack"
Handle Silicon of SOI wafer *i + 1* is removed and 3D vias are etched. Tungsten is deposited to make inter-tier contact to SOI wafer *i*; set *i = i + 1*
Is *i = n* ?
No
Yes
Etch bond pads on SOI wafer *n*, completing 3D assembly
Finish

Figure 4-1: Generalized breakdown of a 3D fabrication process

As shown in Figure 4-1, the initial step of the 3D process is the individual fabrication of the tiers. In this particular design environment, there are three tiers on which to place circuits, but the process shown is viable for *n* number of tiers. Setting *n* equal to 3 would make Figure 4-1 consistent with the MITLL FDSOI 3D process. After a SOI wafer is fabricated for each tier in the design, the wafer that is designated as tier 2 is inverted, or "flipped", and aligned with the wafer that is designated as tier 1. A low-temperature oxide bond is used to bond the two wafers together [20]. At this point, the handle silicon that is at the top of the 3D stack is removed from tier 2. Tier 2 is electrically connected to tier 1 by means of 3D vias that are etched through the oxides of tier 1 and tier 2. Tungsten is then deposited to fill the 3D vias and complete the electrical path between tier 1 and tier 2 [20]. It is important to note at this point that a single 3D via consumes all of the available routing tracks on tier 2 (since it is literally "punched" through the bottom of the SOI wafer) but only makes contact to the top metal layer on tier 1. Therefore, apart from some 3D-specific layers that assign the starting and ending points of a 3D via (see section 4.2), there can be no wiring metallization near a 3D via on tier 2. The same pattern of inversion, alignment, bonding, and 3D via creation is performed to attach tier 2 with tier 3. The routing restrictions in terms of 3D vias that apply to tier 2 also apply tier 3. However, one key difference here is that the 3D vias that provide connectivity between tier 2 and tier 3 end on the bottom-most metal layer in tier 2. Figure 4-2 shows an example of a three tier assembly at the conclusion of the above process. This figure also indicates the etching of the bond pads to expose the bottom-most metal layer on tier 3 for wiring bonding to the package. In an effort to maintain the naming conventions of the MUSE research group's 3DIC design flow (see section 6), the numbering of the tiers in the three tier assembly described above will be exchanged for characters to specify their relative position within the 3D stack (i.e. tier 1 becomes tier A, tier 2 becomes tier B, and tier 3 becomes tier C).

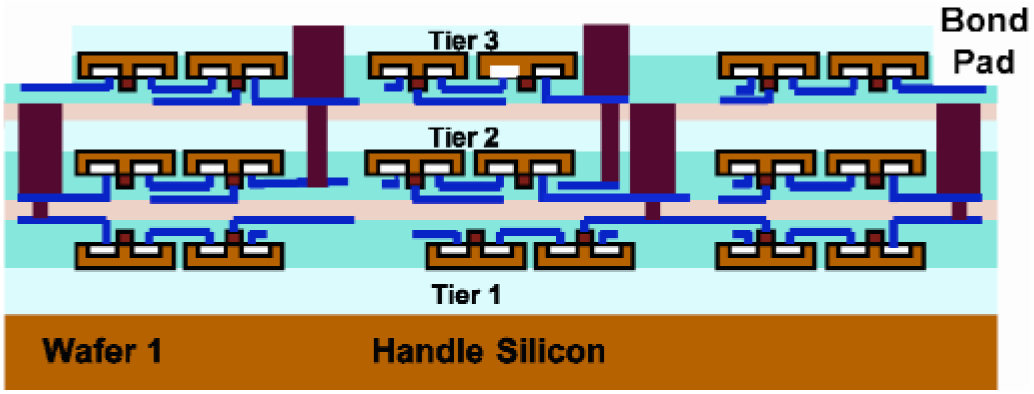


Figure 4-2: Example 3D circuit at the end of the FDSOI 3D process [20]

4.2 Changes to the 2D Design Environment

The MUSE research group has adapted 2D design tools to work with 3D process technology. Essentially, each tier of the 3DIC is designed in the standard 2D manner with the additional inclusion of extra layers to indicate the locations of 3D vias. As it relates to using the IC design software, the inversion of the tier B and tier C is transparent to the circuit designer. This is necessary so that 3D circuits can be visualized in the software in their finished state with direct vertical connections between the terminals of transistors in multiple tiers. Thus when viewing all three tier layouts simultaneously, the layers reserved for 3D vias on tier A exactly coincide with those on tier B, and the same can be said about 3D vias connecting tier B and tier C. To better illustrate this, Figure 4-3, Figure 4-4, and Figure 4-5 comprise the individual tier layouts of an inverter chain.

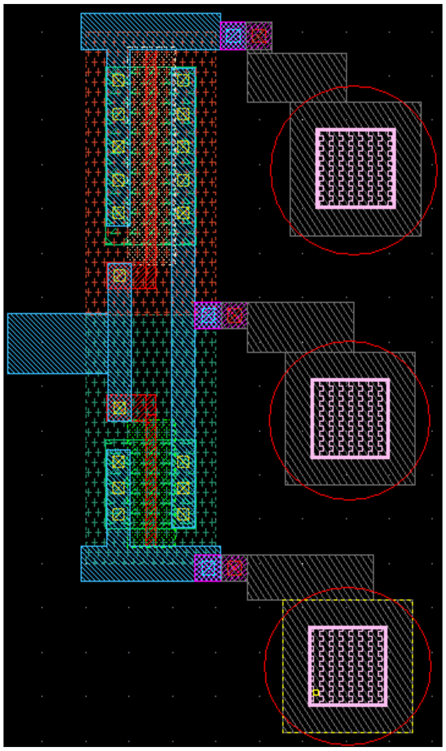


Figure 4-3: Tier A of the inverter chain

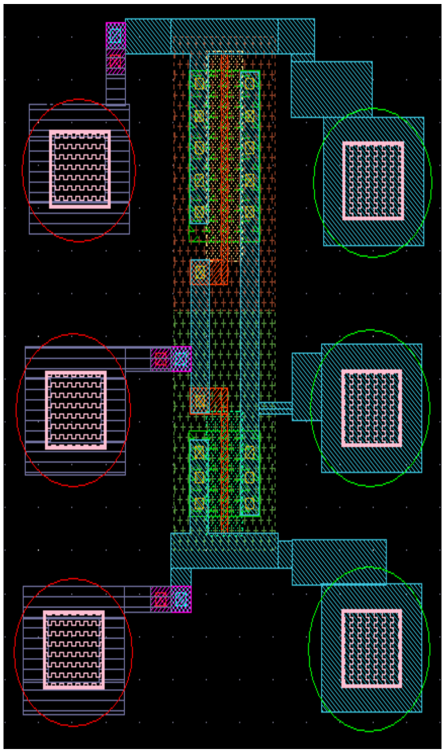


Figure 4-4: Tier B of the inverter chain

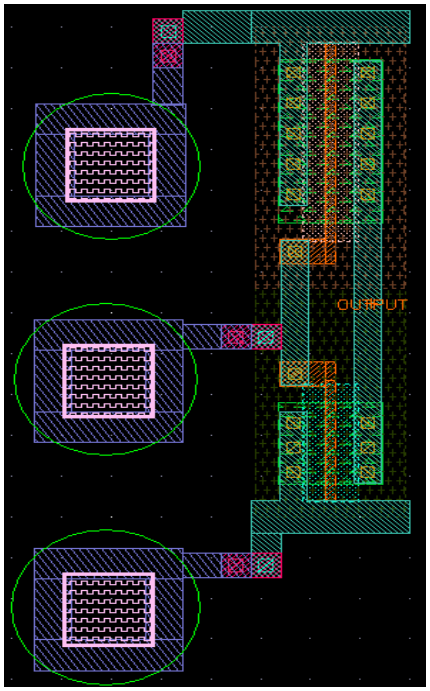


Figure 4-5: Tier C of the inverter chain

In the above figures, the colored circles are not actually a layer in the layout. The red colored circles show the locations of 3D vias between tier A and tier B. Using the nomenclature of the design kit that was developed by the MUSE research group, this type of via is called “VIA_AB”. Similarly, the green colored circles show the locations of 3D vias between tier B and tier C, each of which is known as “VIA_BC”. Each of the layouts in the Figure 4-3, Figure 4-4, and Figure 4-5 can be designed and viewed independently to reduce complexity. Alternatively, the designer can work with all three tiers concurrently. This approach is especially important when verifying the location of the 3D via layers on each tier. Figure 4-6 provides an example of the inverter chain layout when viewing all three tiers.

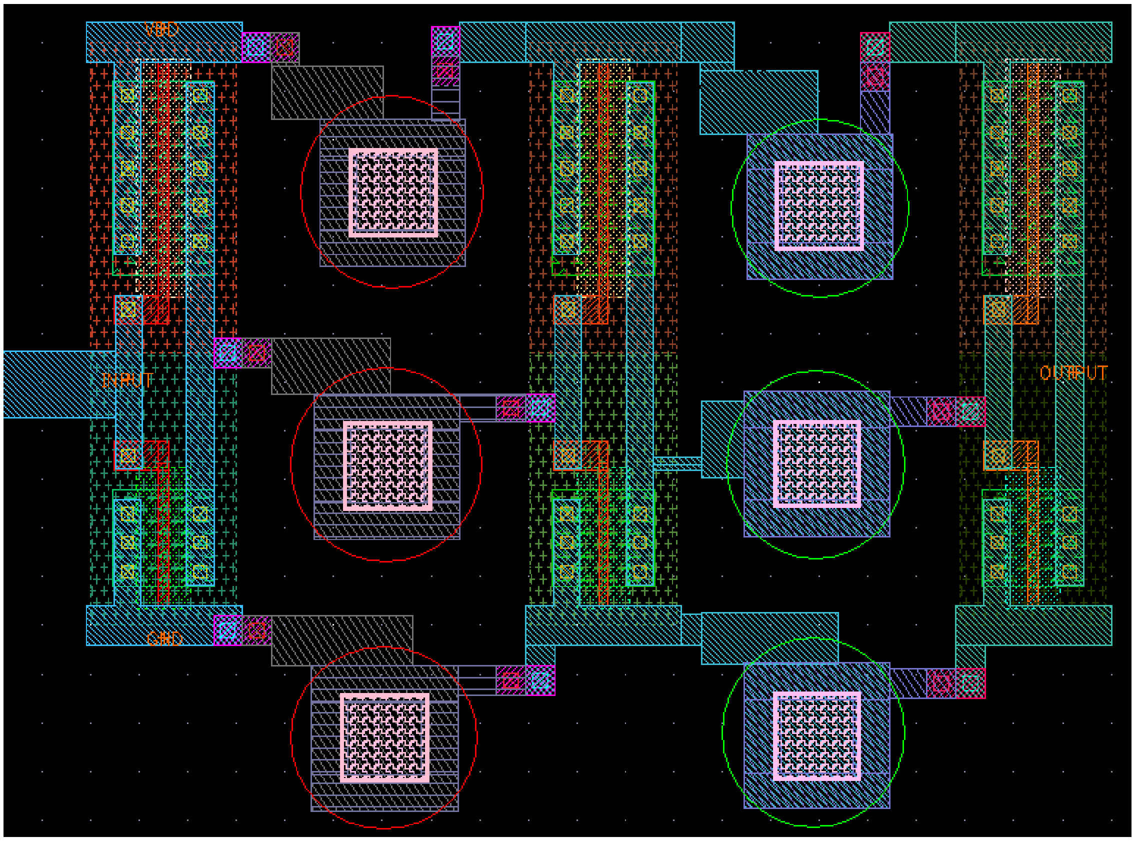


Figure 4-6: Inverter chain layout showing all tiers

As evidenced from Figure 4-6, the 3D via layers, which are once again highlighted by the colored circles, are in the exact same position on each tier. Thus, the output of the first inverter is electrically connected to the input of the second inverter through a VIA_AB. In turn, the output of the second inverter is connected to the input of the third inverter through a VIA_BC, and the inverter chain is completed on tier C. In a departure from 2D layout style, the designer must ensure that nets crossing tier boundaries, such as the internal nets of the inverter chain, have 3D via layers at identical coordinates on each connecting tier of the net. Known in the 3DIC design flow as “via alignment” (see section 6.5), this stage of the layout can be automated through scripting for standard cell-based designs.

5 The ORPSOC Architecture

5.1 Architecture Selection and Modification

An OpenRISC-based design was selected as a test case for 3DICs due to its open source status and wide range of applications. Opencores.org, a well-known supporter of open source hardware intellectual property (IP) cores, administers a repository for all things OpenRISC. Among the various implementations of the OpenRISC microprocessor is a design that is specifically targeted as a starting point for system-on-chip development: the ORPSOC [2]. With the increasing popularity of system-on-chip solutions in the integrated circuit design industry, the ORPSOC offers a useful and interesting case for 3D integration. The ORPSOC follows a standardized definition of an OpenRISC based system, lending to it the distinction as a "reference platform". The purpose of the ORPSOC is to allow for the rapid creation of OpenRISC-based system-on-chip designs with reduced verification time. Included in the current versioning system (CVS) package of the ORPSOC is a set of testing software (the use of which is discussed in section 3.1) and several IP cores that comprise the architecture.

The ORPSOC in its original form consisted of the OpenRISC 1200 microprocessor (OR1200), an Ethernet media access control (MAC) module, a universal asynchronous receiver/transmitter (UART) 16550 module, and several peripheral interfaces. Among the interfaces is functionality for support of SRAM, Flash Memory, audio, PS/2, VGA, and JTAG. The focus of this work was on the basic components necessary for the design of a microprocessor-based system and not on the bevy of possible peripheral configurations. Therefore, the OR1200 and its use of SRAMs for instruction and data memory storage were only considered in this study. The program instructions were originally transferred to the OR1200 through a behavioral Flash Memory model. In order to consolidate the whole of the microprocessor system onto a 3DIC, the Flash memory was replaced with an SRAM capable of storing the largest program, and it is assumed that the SRAM is loaded with the same program instructions as the Flash memory model. Upon further investigation into the architecture of the ORPSOC, it was also discovered that several master microprocessors could be easily interfaced with the slave SRAM modules by means of the extra ports on the "Wishbone Traffic Cop", the arbitration unit for communication between the peripherals (in this work, the only peripherals were SRAM modules) and the microprocessor. To guarantee an interesting case for heat (power dissipation) generation in 3DICs, a second OR1200 was instantiated as a master in the system. As will be previously discussed, high levels of power dissipation are detrimental to the performance of a 3DIC and to assess the merit of 3D technology without considering the power of a design would be foolishly optimistic. It is assumed that the second OR1200 has an identical workload as the original OR1200 (see section 3.1 for workload creation). After the above modifications to the ORPSOC were made to achieve a synthesizable, realistic integrated design, the test case was finalized. A high level block diagram of the ORPSOC test case for 3DICs is shown in Figure 5-1**.** A definition of each functional unit shown in Figure 5-1 follows in this section.

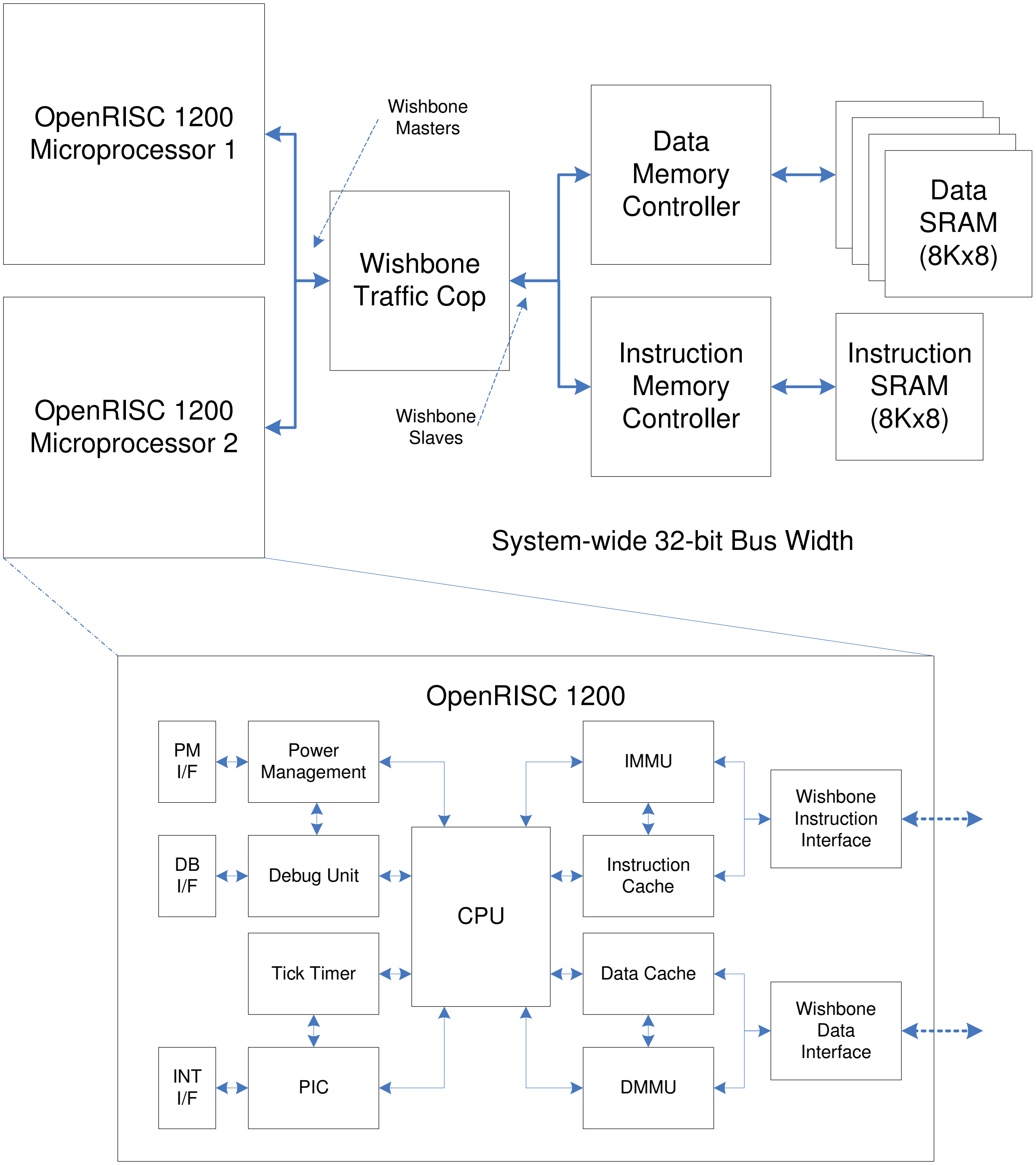


Figure 5-1: High level block diagram of the ORPSOC test case [2]

5.2 OpenRISC 1200 Microprocessor (OR1200)

The OR1200 is a part of the OpenRISC 1000 family of the processor cores. It is a 32-bit scalar load and store RISC with applications in high performance networking, embedded, automotive, and portable computer environments [2]. The processor is realized as a 5-stage integer pipeline with virtual memory support. Instruction and data caches are present in the microprocessor, and both default to 1-way direct-mapped 8KB caches. The memory management units (MMUs) are implemented for data (DMMU) and instructions (IMMU) as 64-entry, 1-way direct-mapped translation look-aside buffer (TLB). Other integrated functionality includes real-time debug capability, tick timer, programmable interrupt controller (PIC), and power management support. The OR1200 communicates with the external world through interfaces for power management, debug, interrupts, and the wishbone system-on-chip interconnection standard. A block diagram of the OR1200 architecture is also included in Figure 5-1**.**

5.2.1 Central Processing Unit (CPU)

As denoted by its name, the CPU is the central component of the OR1200. It implements a strict 32-bit interface for instruction fetches and data loads and stores. The organization of the CPU is show in Figure 5-2**,** and the accompanying descriptions of the functional units follow.

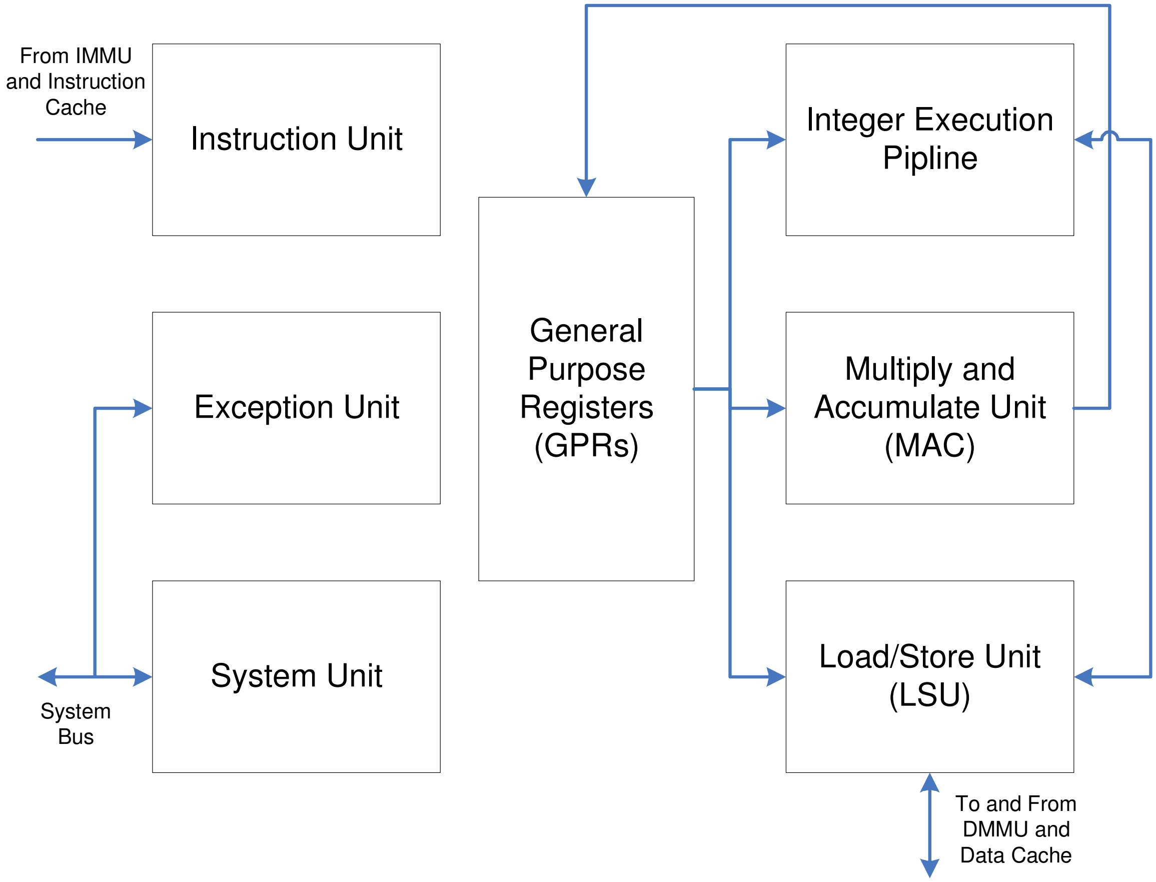

From IMMU and Instruction Cache
Instruction Unit
Exception Unit
System Unit
System Bus
General Purpose Registers (GPRs)
Integer Execution Pipline
Multiply and Accumulate Unit (MAC)
Load/Store Unit (LSU)
To and From DMMU and Data Cache

Figure 5-2: High level block diagram of the CPU [2]

5.2.1.1 Instruction Unit

The instruction unit is responsible for realizing the instruction pipeline, fetching instructions from memory, dispatching instructions to the executions units, and ensuring that operation finish in-order. The bandwidth of instruction dispatch is a maximum of one per clock cycle (scalar). This condition is only fulfilled if an execution unit is always available for dispatch and if there are no withstanding data dependencies in the current program flow.

5.2.1.2 General-Purpose Registers (GPRs)

The OR1200 contains 32 general-purpose 32-bit registers. These registers are implemented as two synchronous dual-port memories organized as 32 words with a bit length of 32 each. For the 3D design of the test case, the register file is implemented in generic fashion given the lack of access to IP memory modules. The generic register file simply models the dual-port memories as a unified two-dimensional array that, once synthesized, is implemented by a total of 1024 flip-flops. As a result, the OR1200 consumes more power and area than if real memories were used for the register file.

5.2.1.3 Load/Store Unit (LSU)

The load/store unit is responsible for the exchange of all data between the GPRs and the internal bus of the CPU. It is realized in the CPU as a discrete execution unit. On issuance of load and store instructions, the LSU first determines if any data dependencies are present before performing memory operations.

5.2.1.4 Integer Execution Pipeline

The integer execution pipeline can implement the standard array of arithmetic, compare, logical, rotate, and shift instructions. Most of these instructions can finish execution in one clock cycle. This is the same unit found in all modern microprocessors.

5.2.1.5 Multiply and Accumulate (MAC) Unit

The MAC unit executes digital signal processing (DSP) multiply and accumulate operations. These operations take two operands of 32 bits each with a 48-bit accumulator. Since the MAC unit is fully pipelined, a new operation can begin on each clock cycle. The OR1200 can be configured to use a generic multiplier or an ASIC-optimized multiplier in this unit. The latter was chosen for use in the ORPSOC test case as it is a better fit for the platform.

5.2.1.6 System and Exception Units

The overriding purpose of the system unit is to implement the special-purpose registers (SPRs) of the CPU. As shown in Figure 5-2, it is connected to the system bus of the CPU. Also connected to the system bus is the exception unit, which is involved in the handling of system calls, external interrupt requests, internal exceptions, and internal errors.

5.2.2 Instruction and Data Caches

The OR1200 offers a number of different configurations for direct-mapped caches. The maximum and minimum size cache possible for the OR1200 is 8KB and 1KB, respectively. That being said, any cache in that range of sizes would consume an unrealistic amount of power and area if implemented solely using flip-flops. Without access to IP memory modules, the instruction and data caches of the OR1200 were disabled for the test case. A disabled cache acts as a rudimentary feed-through unit for the design studied in this work.

5.2.3 Instruction and Data Memory Management Units (MMUs)

The MMUs of the OR1200 perform effective-to-physical address translation. Since the caches are physically-tagged, the MMUs are positioned between the CPU and the cache for the data and instruction memory hierarchy. Thus, communication between the MMUs and the CPU is essential for every load or store instruction in the program. The instruction MMU (IMMU) and data MMU (DMMU) are separate, discrete units of the OR1200, but both default to a direct-mapped 64-entry TLB. Each MMU features a page size of 8KB and an all-inclusive page protection scheme. The default configuration was used for both MMU in the test case design.

5.2.4 OpenRISC Wishbone Interfaces

All data and instruction exchange with external sources and the OR1200 is administered across the Wishbone compliant interfaces. The Wishbone interconnection architecture specifies a common interface for use in system-on-chip designs. Consequently, all of the core components of the ORPSOC include support for this interface to enable seamless multi-component integration. The OR1200, with its 32-bit processing capability, only supports a 32-bit Wishbone interface. This implies that the widths for data and instructions buses are fixed to 32-bits. With disabled caches internally, the OR1200 makes considerable use of the Wishbone interfaces. In fact, practically every instruction fetch and data load or store requires access to external memory via the Wishbone interfaces.

5.3 Wishbone Traffic Cop

The wishbone traffic cop is the module of the ORPSOC test case that arbitrates data transfer between a single wishbone-compliant master and a single wishbone-compliant slave. In this capacity, it is the glue logic between the peripherals of a system and the OR1200. Referring to Figure 5-1, the connections to the wishbone traffic cop are such that the two OR1200 modules are masters and the two memory controllers are slaves. When either OR1200 requires a memory access, a request is signaled to the wishbone traffic cop, which makes the decision to accept or deny the request. In the case of concurrent requests, priority for access to the instruction or data memory is given to the first OR1200. Since all data flow to and from the memories of the ORPSOC must pass through the wishbone traffic cop, it is important to limit the affect of the bottlenecking nature of this unit. As shown in this work, this is achieved through intelligent partitioning and floorplanning of the 3DIC.

5.4 Instruction and Data Memory Controller

The purpose of the memory controllers in this design is two-fold. First, each memory controller establishes a 32-bit interface to memory for reading and writing. Secondly, the instruction and data memory controllers interpret activity over the wishbone interface for store operations and package load operations from the memory such that it conforms to the wishbone standard. Recall that the Flash memory model was exchanged for an SRAM to house the program instructions in the test case. Thus, the instruction and data memory controllers are actually just two instantiations of an SRAM controller. The data memory controller interfaces to the SRAMs needed by both OR1200 modules to execute any arbitrary program. Throughout this work, any memory module not used for instruction storage is referred to as a data SRAM. The instruction memory controller, on the other hand, provides an interface to the read-only instruction SRAM.

5.5 Instruction and Data SRAMs

The only parts of the ORPSOC test case that are not mapped to standard cells are the memories. As evidenced by Figure 5-1, there are five total SRAMs in the system, each of which is organized as 8KB words with byte-wide words lengths (labeled as “8Kx8” in the figure). Four of the memories are designated as data SRAMs, which the OR1200 modules use to store any data during program execution. Each data SRAM contributes one byte to the width of the data bus and is addressed using the entire 13-bit address bus. The remaining memory in the system is the substitution for the Flash memory model of the original ORPSOC architecture and is used as a read-only instruction storage space.

The SRAMs are treated as black box components for design synthesis with no timing annotations. In order for the SRAMs to have a physical footprint in the placement and routing of the test case, a library exchange format (LEF) file was created. LEF is a standard specification for representation of an integrated circuit layout such that propriety is maintained. This LEF file is an abstract of the actual proposed layout of the SRAM, as it only contains information regarding the bounding box area, the metallization used, and the pin locations of the SRAM. Figure 5-3 shows the proposed layout of the SRAM with colors to indicate the locations of the actual components of the SRAM. Figure 5-4 provides an overview of the information presented in the LEF file for the same SRAM. The area filled with the colored pattern symbolizes the routing blockages of the first three metal layers. These routing blockages constrain all wires crossing the bounding box of the SRAM to the fourth and fifth metal layers in a 5-metal layer process. The dimensions of the SRAM were estimated to be 980 μm by 1450 μm. The pin locations, routing resources used, and area of the SRAM were approximated as the final attributes of a SRAM currently in design for the FDSOI 3D process at North Carolina State University. The SRAM design is a modification of the one used in [21].

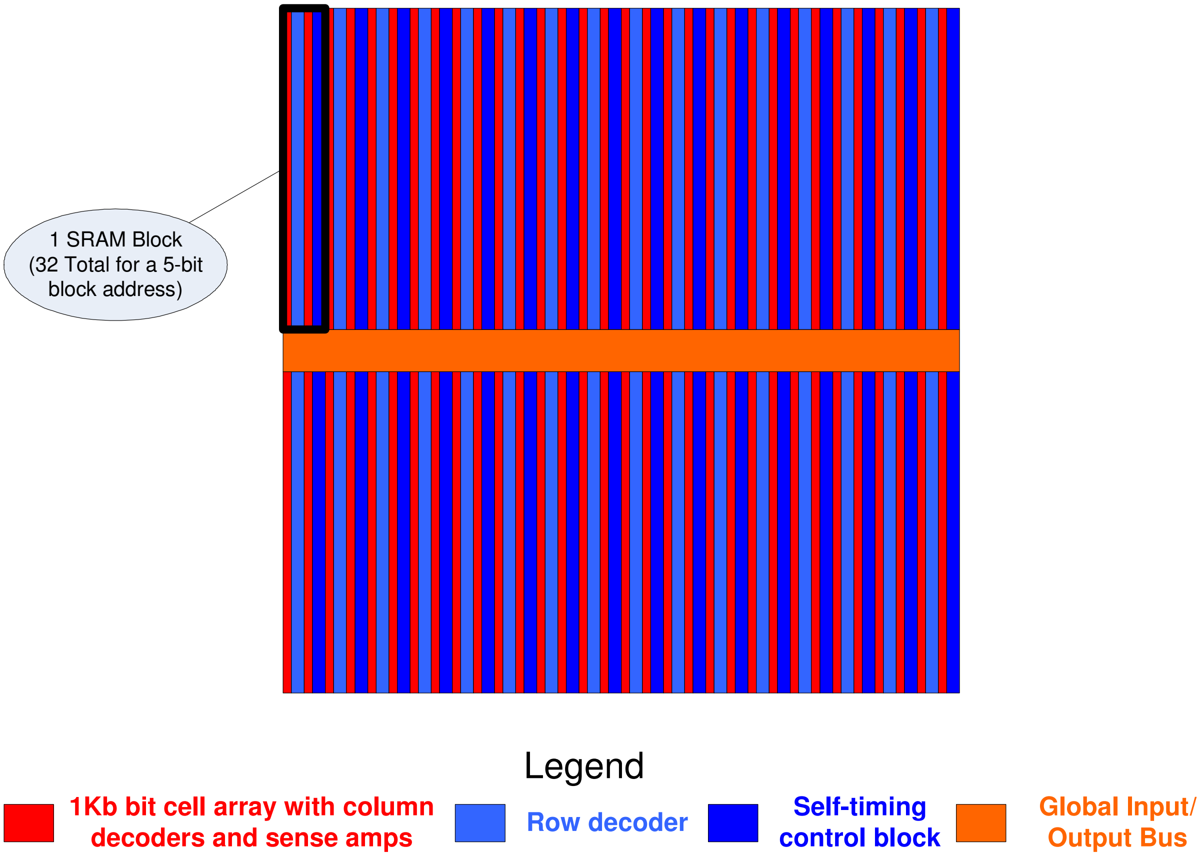

1 SRAM Block
(32 Total for a 5-bit block address)
Legend
1Kb bit cell array with column decoders and sense amps
Row decoder
Self-timing control block
Global Input/ Output Bus

Figure 5-3: High level view of the SRAM layout

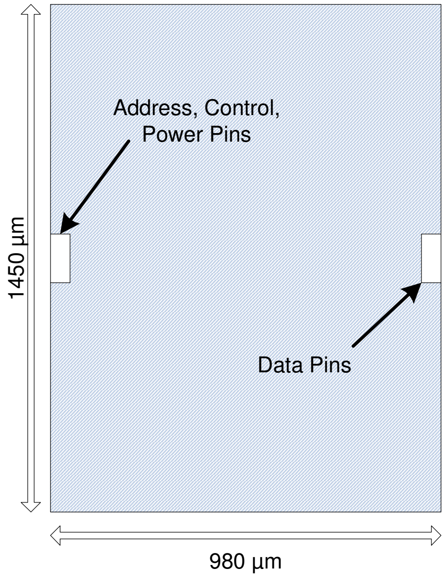


Figure 5-4: Graphical representation of the SRAM LEF file

6 Testcase Physical Design

Even with standard cell-based designs, the 3DIC design flow is a very time consuming and complex process. One can no longer simply rely on the synthesis, placement, and routing tools to generate a valid layout for any arbitrary standard cell-mapped design. A mixture of custom automatic and manual steps is necessary to couple the 3D integration technology (see section 4.2) with the pre-existing 2D IC design tools. The flow described in this section has previously been verified by passing all layout-versus-schematic (LVS) checks for a 3D fast Fourier transform (FFT) design awaiting production at MITLL [22]. For the purposes of this work, some additional measures were taken to ensure a valid result from this flow. These measures were necessary because the design flow was originally intended to function only with pure standard cell-based designs. As explained in section 5, the synthesized test case contains macros in the form of memory blocks in addition to standard cells. Blindly applying the unaltered 3DIC design flow used for the FFT chip would generate 3D layouts with design rule violations.

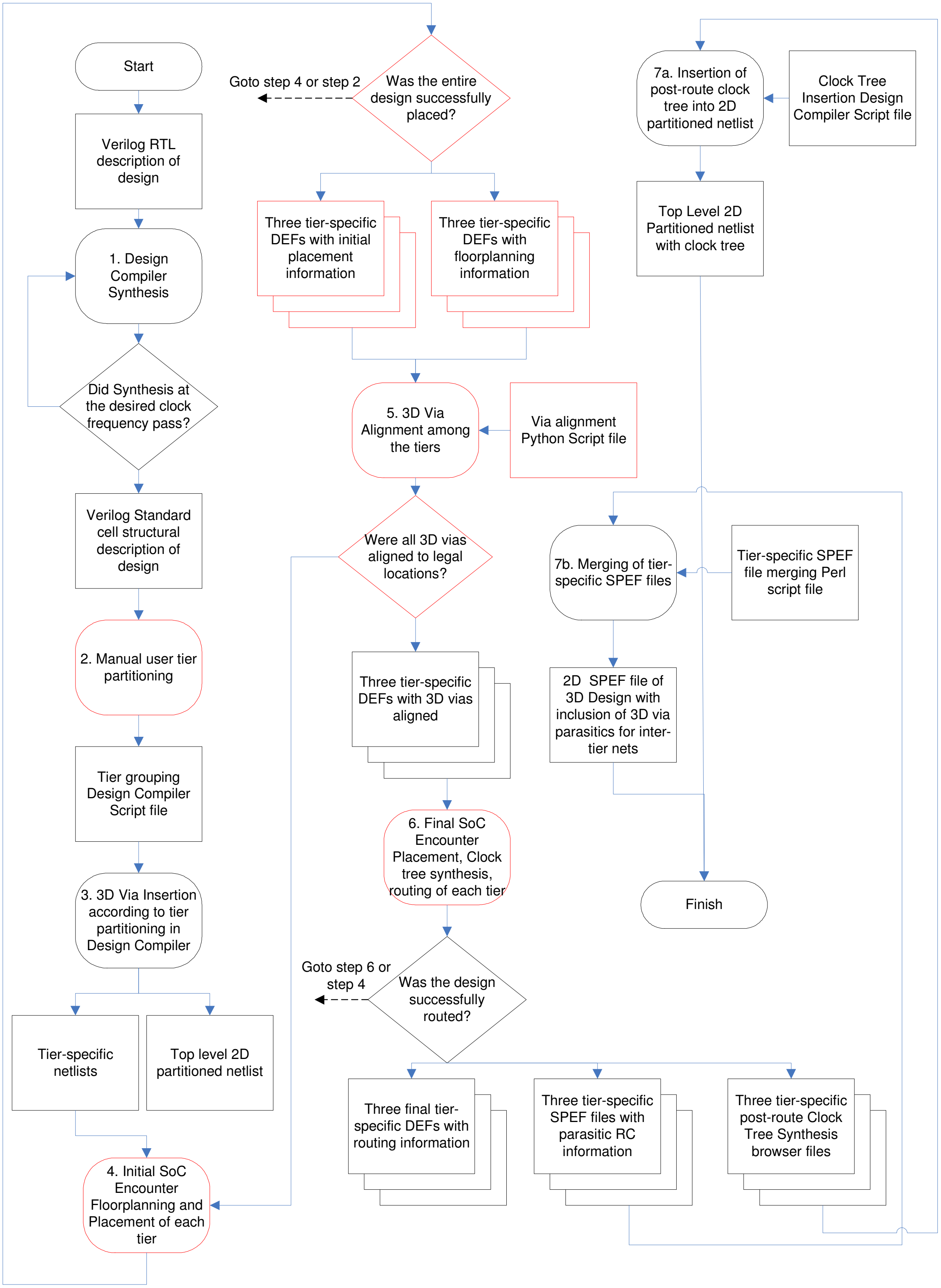

Start
Verilog RTL description of design
1. Design Compiler Synthesis
Did Synthesis at the desired clock frequency pass?
Verilog Standard cell structural description of design
2. Manual user tier partitioning
Tier grouping Design Compiler Script file
3. 3D Via Insertion according to tier partitioning in Design Compiler
Tier-specific netlists
Top level 2D partitioned netlist
4. Initial SoC Encounter Floorplanning and Placement of each tier
Goto step 4 or step 2
Was the entire design successfully placed?
Three tier-specific DEFs with initial placement information
Three tier-specific DEFs with floorplanning information
5. 3D Via Alignment among the tiers
Via alignment Python Script file
Were all 3D vias aligned to legal locations?
Three tier-specific DEFs with 3D vias aligned
6. Final SoC Encounter Placement, Clock tree synthesis, routing of each tier
Goto step 6 or step 4
Was the design successfully routed?
Three final tier-specific DEFs with routing information
Three tier-specific SPEF files with parasitic RC information
Three tier-specific post-route Clock Tree Synthesis browser files
7a. Insertion of post-route clock tree into 2D partitioned netlist
Clock Tree Insertion Design Compiler Script file
Top Level 2D Partitioned netlist with clock tree
7b. Merging of tier-specific SPEF files
Tier-specific SPEF file merging Perl script file
2D SPEF file of 3D Design with inclusion of 3D via parasitics for inter-tier nets
Finish

Figure 6-1: The logical progression through the 3DIC design flow

Figure 6-1 provides an overview of the sequence of steps in the 3DIC design flow. For this flow chart and the others similar to it, rounded shapes denote actions or processes. Some of these will be automatically performed through scripting in order to guarantee a valid result while still others will require manual user interaction with the inputs to the step. Rectangular shapes in the flow charts are used to denote the outputs of a step, and these often take the form of statistical reports or files used in later parts of the flow. Specifically for Figure 6-1, shapes that are outlined in red mark a modification to the original 3DIC design flow that was necessary for use with the ORPSOC test case. For each numbered process, of which there are eight in Figure 6-1, an in depth definition follows below.

6.1 Design Compiler Synthesis

The 3DIC design flow begins with the synthesis of the RTL description of the design. The synthesis tool that was used in this work was Synopsys Design Compiler [23]. Design Compiler provides a means of specifying the loading on the external ports, the clock frequency, and an estimation of the amount of clock uncertainty in order to produce the appropriate structural description of the design. This structural description removes all instances of behavioral modeling and replaces them with standard cell instances.

In order to compile a design in the synthesis tool, a valid standard cell library must first be identified and linked to the tool. This is necessary so that the tool can determine which standard cells to choose when transforming the RTL description. Once the standard cell library is loaded, the entire hierarchy of the RTL description is interpreted and linked by the tool. At this point, important synthesis variables must be set. The user must define a clock for the design at the desired frequency and some notion of clock skew must be defined as the uncertainty of the clock. To produce a realistic synthesis result, it is also desirable to set the delays and loading on the ports of the design, since any combinational path from an input or to an output must be included in the slack calculations of the tool. Once these variables are set, the tool can attempt to synthesize the design at the specified clock frequency. Timing reports after this compilation process are used to determine if the synthesis was successful. The critical path in the design must meet the setup time constraints of the associated register with non-negative slack. Similarly, the shortest path in the design must meet the hold time constraints of the associated register with non-negative slack. Once these conditions are met, the standard cell-mapped hierarchy of the design is written to a file by the tool.

For the design of the ORPSOC test cases, the SOI standard cell library previously designed by the MUSE research group was used. The clock frequency was set to an aggressive 100 MHz (10 nanosecond period), and the clock uncertainty was set to 200 picoseconds. It was assume that the ports of the design were connected to flip-flops. The inputs were assigned a delay of 300 picoseconds (simulating the delay of the input flip-flop), and the output setup time was assigned a value of 100 picoseconds. After compilation, the timing reports showed that the critical path met the constraints with exactly zero slack.

6.2 Manual User Tier Partitioning

In order to move the design from the 2D realm to a 3D one, decisions must be made about the position of each hierarchical block among the tiers of the 3DIC. Normally, the 3DIC design flow dictates that this is performed automatically by min-cutsize partitioning. Partitioning in this manner ensures that the area of the design is balanced among the partitions and that the partitioning result contains the least number of nets that cross partition boundaries. In terms of a 3DIC, a “partition” is a physical tier in the 3D stack. Subsequently, any nets that cross partition boundaries require the instantiation of 3D vias in the design.

Hence, the original MUSE 3DIC tier partitioning scheme sought to minimize the number of 3D vias. Concerns about the size, manufacturing yield, RC parasitics, and the required place and route blockages of the 3D vias were the motivation for this method of 3D partitioning.

For the purposes of better harnessing the advantages of 3DICs, more control must be given to the user when performing tier partitioning. Since 3DICs offer the possibility of short vertical wires, the methodology behind the 3D design of the test case was to exploit the vertical axes for large bus routing. These large busses, which could potentially become very long in a strict 2D placement, are made very short in wire length if the interconnecting modules are placed exactly one tier apart. During the placement and routing stage of the design flow, this will help reduce the local tier routing congestion created by these busses. As an example, Figure 6-2 shows a 2D layout and connectivity of five blocks. As shown in the figure, all five blocks are connected by a single bus, and long a net length is needed to fully connect the blocks (indicated by the dashed route).

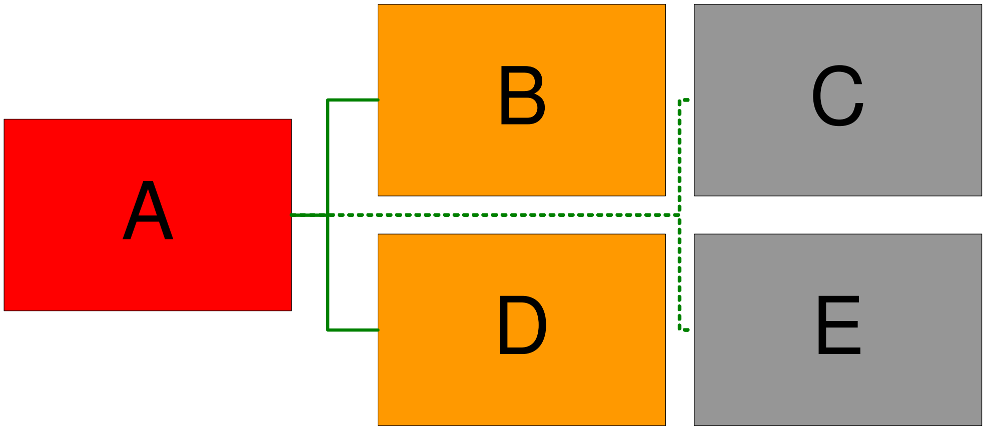


Figure 6-2: Example 2D rectilinear net length interconnecting five blocks

To help reduce the net length, the blocks in Figure 6-2 could be transitioned to 3D as shown in Figure 6-3**.** In this figure, the block labeled “A” is placed on tier B, with tier A and tier C each containing two of its interconnecting blocks. The long dashed route in the 2D example (Figure 6-2) is now shortened to the length of a 3D via (indicated by the thick green vertical line) plus the length of the local rectilinear route on tier C. For large, complex integrated systems, the direct path to blocks “C” and “E” in Figure 6-2 may be impossible due to tremendous routing congestion. Thus, if one can ensure the relative 3D locations of the blocks in Figure 6-3, the total net length can be greatly reduced. This was the general partitioning strategy applied to the design of test case.

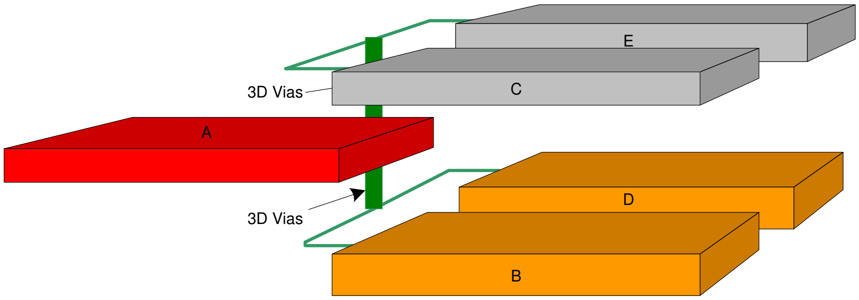


Figure 6-3: Example 3D net length interconnecting five blocks

The output of this stage of the design flow is a Design Compiler script file. The script file contains exactly three “group” commands. After the structural description of the design has been read and linked by Design Compiler, the “group” commands create a new level of hierarchy for each tier and place all partitioned modules for each tier in the associated level of hierarchy. Using Figure 6-3 as an example, the “group” commands would instruct Design Compiler to place block “A” in the tier B level of hierarchy. Similarly, block “B” and “D” would be placed in the tier A level of hierarchy and block “C” and “E” would be placed in the tier C level of hierarchy.

As detailed in section 5, the test case contains many data and instruction buses with widths as large as 32-bits. This presents the opportunity to use 3D vias for realizing short, vertical busses. The OpenRISC CPUs were quickly identified as being the components having the highest bus connectivity. Therefore, the CPUs were prime candidates for placement on tier B, giving them a central location among the 3D stack. This central location is important so that the components sharing the same bus with the CPUs can be partitioned to either tier A or tier C. If the CPUs were partitioned to any tier other than tier B and the goal was still to place the signal busses on the vertical axis, one of two things would occur. First, all of the components sharing a bus with the CPUs would be partitioned to the same tier (one vertically above or below the CPUs), creating an module area imbalance among the tiers (the last tier would be very sparse). Secondly, the components would be split among the two remaining tiers, creating a situation where some of the busses traverse two tier boundaries (since the CPUs would no longer be centralized), which would double the vertical length component of those nets. Thus, to maintain a high degree of density in tier A and tier C and to ensure that the vertical separation is as small as possible, the partitioning of components must be kept fairly balanced. In terms of the test case, this meant assigning all of the components connected to CPU 1 to tier A and all of the components connected to CPU 2 to tier C. The choice of tiers here is arbitrary since CPU 1 and CPU 2 are identical in their implementation.

Other components of the ORPSOC test case that have a high degree of external connections include the wishbone traffic cop (TC), the data memory controller, the instruction memory controller, and the data and instruction SRAMs themselves. Partitioning the TC to tier B was logical since it is connected to the wishbone interface modules of CPU 1 and CPU 2, which reside on tier A and tier C, respectively. Using the same reasoning for partitioning the CPUs to tier B, the data memory controller was partitioned to tier B to enable SRAM connectivity directly above and below it. For balance issues, the data SRAMs were divided evenly among tier A and tier C. This leaves just the instruction memory controller and the instruction SRAM, which were both delegated to tier B in the interest of balance once again. Figure 6-4 summarizes the complete partitioning scheme of the test case. In this diagram, the thick blue lines crossing the horizontal red dashed line represent the communication paths utilizing 3D vias. This 3D organization is attractive from a performance standpoint since the CPUs are "sandwiched" between the memory hierarchies with which they constantly communicate. Moreover, the data memory controller is theoretically residing only the length of a 3D via away from the data SRAMs. This would not be possible with 2D partitioning, as there would undoubtedly need to be trade-offs between the lengths of the busses.

Prior to settling with the organization seen in Figure 6-4, two other partitioning schemes were first considered. These schemes sought to consolidate the data SRAMs on a single tier and partition the CPUs on another tier, leaving their rest of the OR1200 components to be partitioned to the remaining tier. However, it was quite obvious that the tier dimensions of a 3DIC having four SRAMs on a single tier were too large, and consequently, a great deal of area was wasted in the other tiers. Still, the partitioning scheme shown in Figure 6-5 was considered. In this organization, one of the data SRAMs was moved from tier A to tier B in an attempt to meet density requirements (explained in section 6.4). Despite this change, the density was still unsatisfactory (varying by more than 10% between the tiers), so the more symmetrical approach shown in Figure 6-4 was selected.

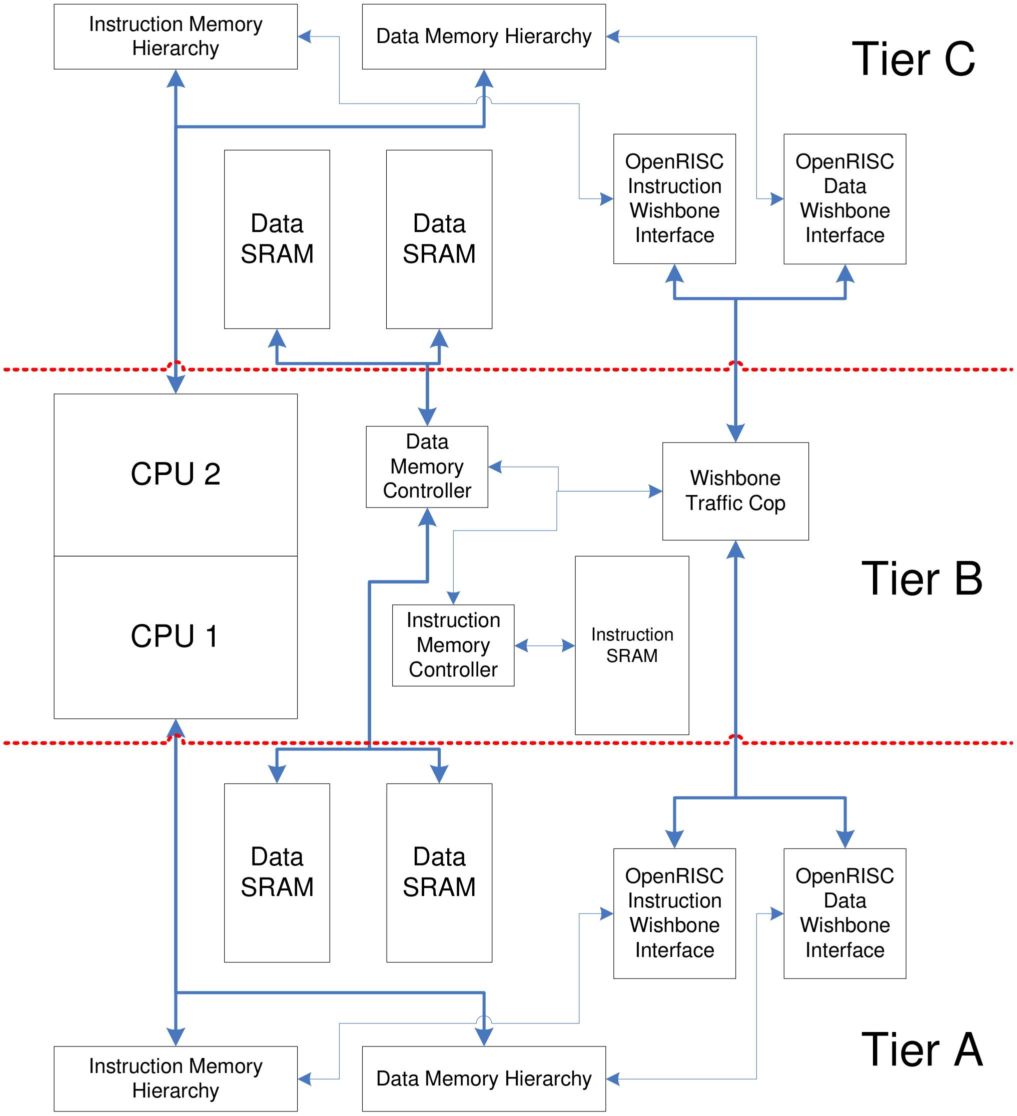

Instruction Memory Hierarchy
Data Memory Hierarchy
Tier C
Data SRAM
Data SRAM
OpenRISC Instruction Wishbone Interface
OpenRISC Data Wishbone Interface
CPU 2
Data Memory Controller
Wishbone Traffic Cop
CPU 1
Instruction Memory Controller
Instruction SRAM
Tier B
Data SRAM
Data SRAM
OpenRISC Instruction Wishbone Interface
OpenRISC Data Wishbone Interface
Instruction Memory Hierarchy
Data Memory Hierarchy
Tier A

Figure 6-4: Partitioning scheme employed for the ORPSOC test case

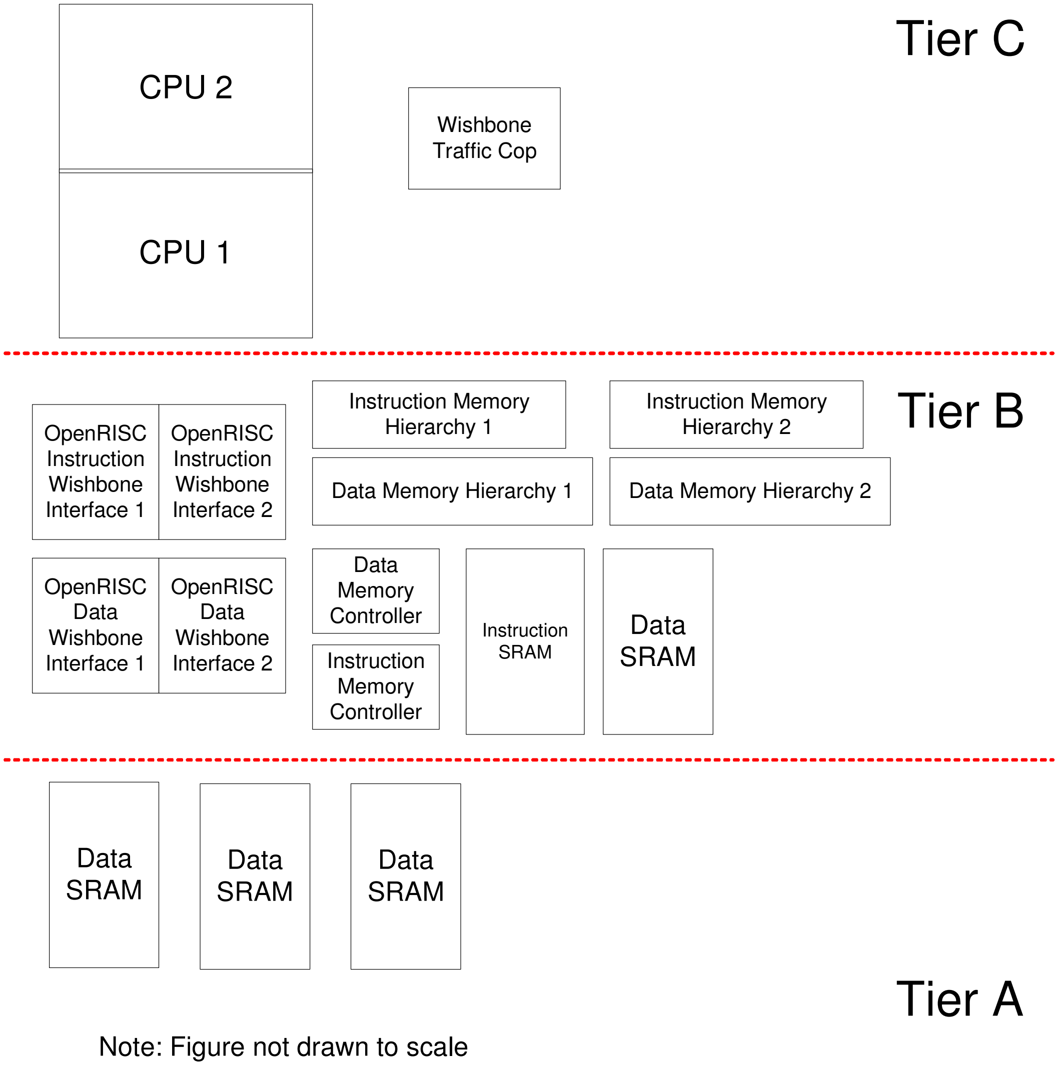

Tier C
CPU 2
CPU 1
Wishbone Traffic Cop
Tier B
OpenRISC Instruction Wishbone Interface 1
OpenRISC Instruction Wishbone Interface 2
Instruction Memory Hierarchy 1
Instruction Memory Hierarchy 2
Data Memory Hierarchy 1
Data Memory Hierarchy 2
OpenRISC Data Wishbone Interface 1
OpenRISC Data Wishbone Interface 2
Data Memory Controller
Instruction Memory Controller
Instruction SRAM
Data SRAM
Data SRAM
Data SRAM
Data SRAM
Tier A
Note: Figure not drawn to scale

Figure 6-5: Another partitioning strategy considered for the test case

6.3 3D Via Insertion according to User Partitioning

The tier partitioning that was defined in the previous step is taken as the input of the 3D via insertion step. This process is entirely automated by Design Compiler scripts. Essentially, the 2D structural description of the design that was the result of the original Design Compiler synthesis (see section 6.1) is once again read by Design Compiler. The connections among the standard cells in the design are analyzed based on the tier grouping script that was the output of the previous step. 3D vias are inserted wherever nets cross tier boundaries. Tier boundaries are determined by simply considering the additional levels of hierarchy that were defined by the tier grouping script. For instance, if a standard cell in the tier A level of hierarchy and a standard cell in the tier B level of hierarchy are connected by a net, 3D via insertion for this net is required. The actual inclusion of a 3D via in the netlist is achieved by means of special standard cells in the library. For the SOI standard cell library used by MUSE, "VIA_AB", "VIA_BC", "VIA_AC", and "VIA_ABC" are the cell names containing 3D vias. A two letter suffix such as "AB" indicates a standard cell containing a 3D via with via layers for tier A and tier B. "VIA_ABC" is the only cell containing via layers for all three tiers. Usually reserved for global signals such as reset and clock, this standard cell would be used for instances where a net has connections on all three tiers.

Figure 6-6 shows the process executed by the Design Compiler scripts to insert 3D vias while preserving the functionality of the design. In this example, standard cell "A" is contained within the tier A level of hierarchy and standard cell "B" is contained within the tier B level of hierarchy. Due to "Net_AB", 3D via instantiation is needed in the design. First, the appropriate 3D via standard cell is created and the connections between the standard cells and "Net_AB" are broken. According to Figure 6-6, the "VIA_AB" standard cell is required to complete the net. Next, a new net is created for each side of the 3D via. Finally, the standard cell ports are connected to the appropriate net, and the path from standard cell "A" to standard cell "B" is restored. The Design Compiler scripts exhaustively search through the design and perform these actions until all 3D vias have been inserted.

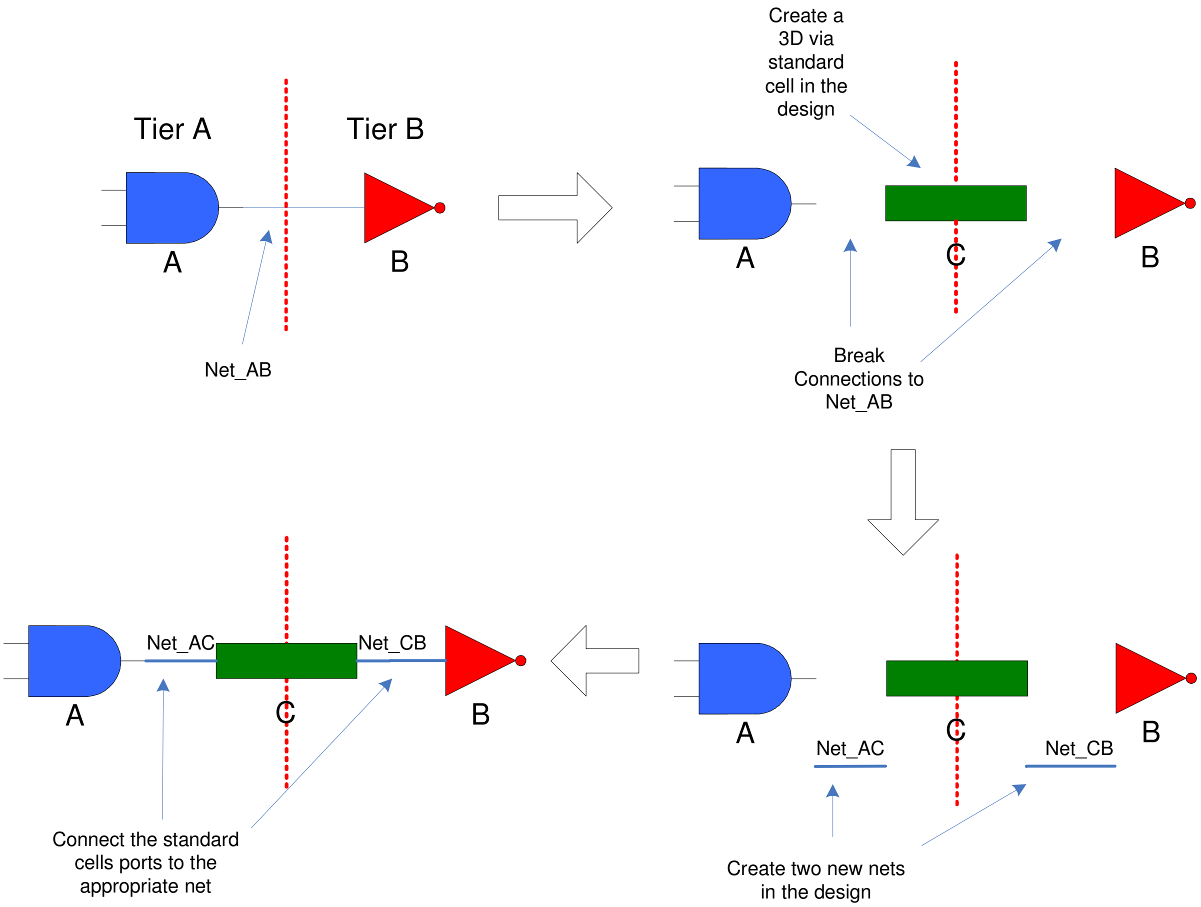

Tier A
Tier B
A
B
Net_AB
Create a 3D via standard cell in the design
A
C
B
Break Connections to Net_AB
A
Net_AC
C
Net_CB
B
Connect the standard cells ports to the appropriate net
A
C
B
Net_AC
Net_CB
Create two new nets in the design

Figure 6-6: 3D via insertion into the netlist

Once the 3D via insertion phase is complete, the scripts include commands to write four new netlist files. Three of these netlists are the tier-specific structural descriptions of the design. The tier-specific netlists are created by simply writing the hierarchy of each tier to a file. The only changes to the hierarchy of each tier in these files are the instances of 3D vias standard cells. The other netlist file that is written is termed the top level 2D "partitioned" netlist. This file contains the levels of hierarchy for the all of the tiers but excludes the 3D vias from the design. It specifies the interconnections between the tiers, where as the tier-specific netlists only show connections to 3D vias. The top level 2D "partitioned" netlist is used at a later point in the design flow for back-annotation of the RC wiring parastitics, including that of the 3D vias.

6.4 Initial SoC Encounter Floorplanning and Placement

SoC Encounter is a full-featured automatic place and route tool of the Cadence IC design suite [24]. During the initial floorplanning and placement stage of the design flow, the goal was to perform a coarse placement in SoC Encounter of the standard cells and macros on each tier. Since SoC Encounter provides no mechanism for 3D placement, each tier is individually placed irrespective of one another. However, it is still not advisable to let the placement tool be unrestricted with its 2D placement of each tier. Using the partitioning strategy that was developed in section 6.2, the modules are first floorplanned such that the components connected by vertical wires overlap in the 3D stack. For example, consider Figure 6-7 in which block "A" is connected to block "B" in the netlist. The diagram at the left shows a bad 3D floorplanning choice for tier A and tier B, since the blocks are not well aligned horizontally. A placement and subsequent routing generated from this floorplan would result in unreasonably long net lengths for those nets connected to 3D vias. Conversely, the diagram at the right shows a better floorplanning choice in which the blocks are well aligned. The placement and routing for this case would produce drastically reduced net lengths. The situation grows even more complex when considering all three tiers of the 3DIC. Therefore, it is paramount to understand the implications of 2D placement in a 3D space. Otherwise, it stands to reason that the entire benefit of 3D chips would be negated, and the 2D IC would still offer better performance.

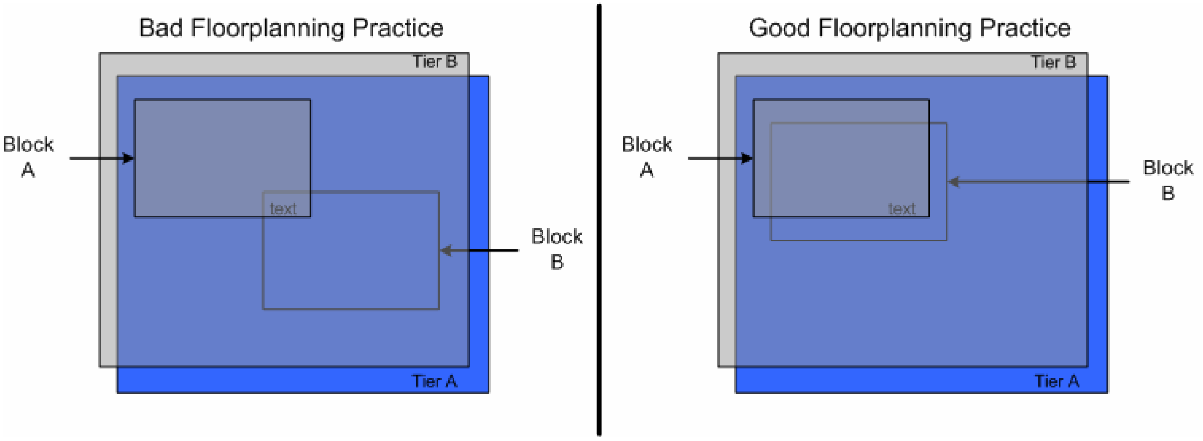


Figure 6-7: Example of good and bad floorplanning for 3DICs

With a good floorplanning strategy in mind, initial floorplanning and placement of each tier can move forward. Figure 6-8 provides an overview of the procedure used in SoC Encounter. Since the dimensions of each tier are identical in a 3D process, one must first determine the suitable area to use for each tier. In this work, placement density was used as the metric to arrive at the values for the tier dimensions. With five available routing layers, 95% area density was the goal for the design of the test case. It should be noted that the dimensions of the tiers are set by the tier having the greatest total module area. Since SoC Encounter estimates the core dimensions needed (assuming a width-to-height ratio of 1) to place a design during the import process, a quick survey of these numbers for all three tiers can identify the tier of greatest module area. In SoC Encounter, the core density of this tier was set to just under 95% to account for clock tree synthesis and power planning, which both work to decrease the number of legal placement locations. Next, the power planning phase is performed. This consists of adding power rings around the core and each macro of the design. It also includes adding power stripes across the core. The next step after power planning is the manual floorplanning of the modules. It is during this step that the practices introduced by Figure 6-7 are carried out. SoC Encounter allows the user to drag-and-drop the module blocks over the area of the core for preferred placement. The tool will then take these floorplan “guides” as inputs to the placement engine and place the contents of each module to the best of its ability within bounding box of the associate guide. Modules without 3D vias are not floorplanned so that SoC Encounter can fully optimize their placement. At the conclusion of floorplanning, a low effort placement run is executed. Given that a second, finer placement is performed later in the design flow, a low effort placement is favored here in the interest of runtime. The placement result is analyzed to ensure that the cells of the design did not stray too far from their floorplan guides. If SoC Encounter had difficulty adhering to the floorplan or if some cells were unable to be placed, iterations in the floorplanning and placement loop may be necessary. The area of the core, and therefore the area of every tier, may be increased to try to solve placement problems. In the worst-case scenario, an alternate partitioning scheme may be needed.

Once a suitable placement is achieved, two forms of Design Exchange Format (DEF) files are exported from the tool. One form contains only the floorplanning information of the design, such as the power planning and the location of any floorplan guides or fixed macro cells. The other type of DEF file contains the floorplanning information as well as the placement locations. DEF is a specification used by IC design tools for representing layout information in a human-readable format. These files are taken as inputs in the next stage of the design flow.

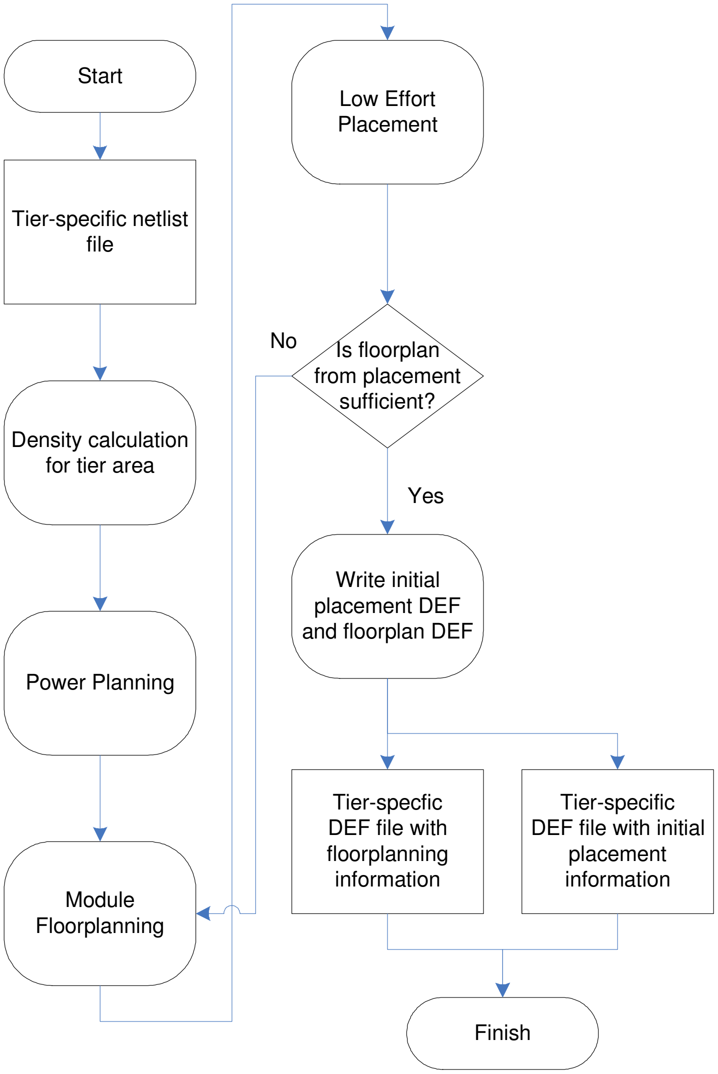

Start
Tier-specific netlist file
Density calculation for tier area
Power Planning
Module Floorplanning
Low Effort Placement
No
Is floorplan from placement sufficient?
Yes
Write initial placement DEF and floorplan DEF
Tier-specfic DEF file with floorplanning information
Tier-specific DEF file with initial placement information
Finish

Figure 6-8: Procedure for initial placement of tier-specific netlists

Given the partitioning of the test case as discussed in section 6.2, the tier with the highest module area was tier B. This was mostly due to the areas required by the OpenRISC CPUs, the largest modules in the design. Nonetheless, the partitioning scheme was more than sufficient because the areas required on each tier to obtain 95% density were very close in magnitude. This means that there was not a significant amount of “wasted” area in tier A and tier C. In fact, the placement results from this step showed greater than 90% density for all three tiers. When the final placement is performed at a later stage in the design flow, the density will further increase due to clock tree insertion.

Beginning with the floorplanning of tier B, the design of the ORPSOC test case sought to divide the regions of the tier between the components for the first OR1200 and the components of the second OR1200. If these divisions are held constant across the tiers, the modules with 3D vias can be adequately aligned. Figure 6-9 shows the core size and module floorplanning of tier B in SoC Encounter. There a few key points of interest to notice in this figure. First, the size of a tier on the 3DIC was 2520 µm by 2524 µm, which translates to a tier area of about 6.36 mm^2. When considering all three tiers, the total core area of the 3DIC was 19.08 mm^2. The figure also shows the locations of example power rings and stripes in the floorplan. The vertical power stripes do not cross the bounding box of the instruction SRAM, since for this design it is assumed that the SRAM imposes a total blockage of the first three metal routing layers The routing blockages of the SRAM are an integral factor of the via alignment stage of the design flow (see section 6.5).

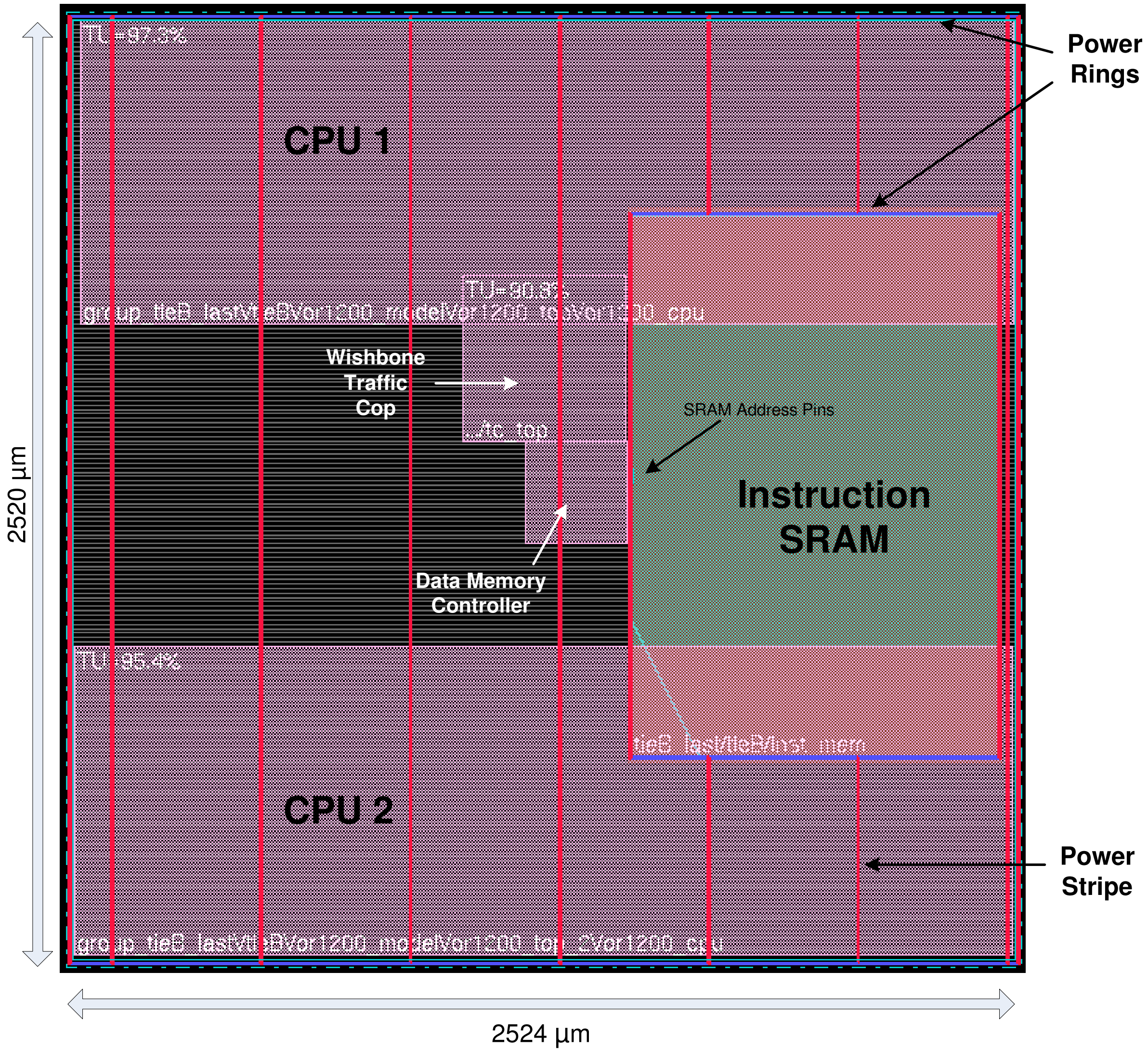


Figure 6-9: Core size and module floorplanning for tier B of the ORPSOC test case

Referring to Figure 6-9 again, the regional division among the two OR1200 modules is apparent. Particularly, the components of the first OR1200 are floorplanned to the top portion of the core, with the components of the second OR1200 occupying the bottom portion of the core. In tier B, this condition is fulfilled by the location of CPU 1 (top) and CPU 2 (bottom). The centered position of the wishbone traffic cop and the data memory controller in Figure 6-9 is also interesting. The reasons for the positioning of these modules in the floorplan are clearer after the consideration of the floorplans for tier A and tier C, which are shown in Figure 6-10 and Figure 6-11, respectively. The instruction memory controller does not appear as a floorplan object in Figure 6-9 given that its connectivity is isolated to tier B.

Recall from the manual partitioning step (see section 6.2) that tier A contains the components of the first OR1200 and two data SRAMs. Likewise, tier C is partitioned the remaining two data SRAMs and the components for the second OR1200. Since the CPUs of the design communicate directly with the memory management units (MMUs) as revealed in section 5, it is important to floorplan these modules such that they are aligned with the correct CPU on tier B. Notice in Figure 6-10 and Figure 6-11 that the MMUs (i.e., the data MMU and the instruction MMU) are assigned to either the top or bottom portions of the core. Thus, the floorplanning guide for CPU 1 in Figure 6-9 is aligned with the floorplanning guides for the DMMU 1 and IMMU 1 in Figure 6-10. The same condition exists for CPU 2, DMMU 2, and IMMU 2. The modules that are connected to each MMU on the tiers themselves do not need to be manually floorplanned since they do not contain 3D vias, so it is more efficient to allow SoC Encounter to optimize their placement.

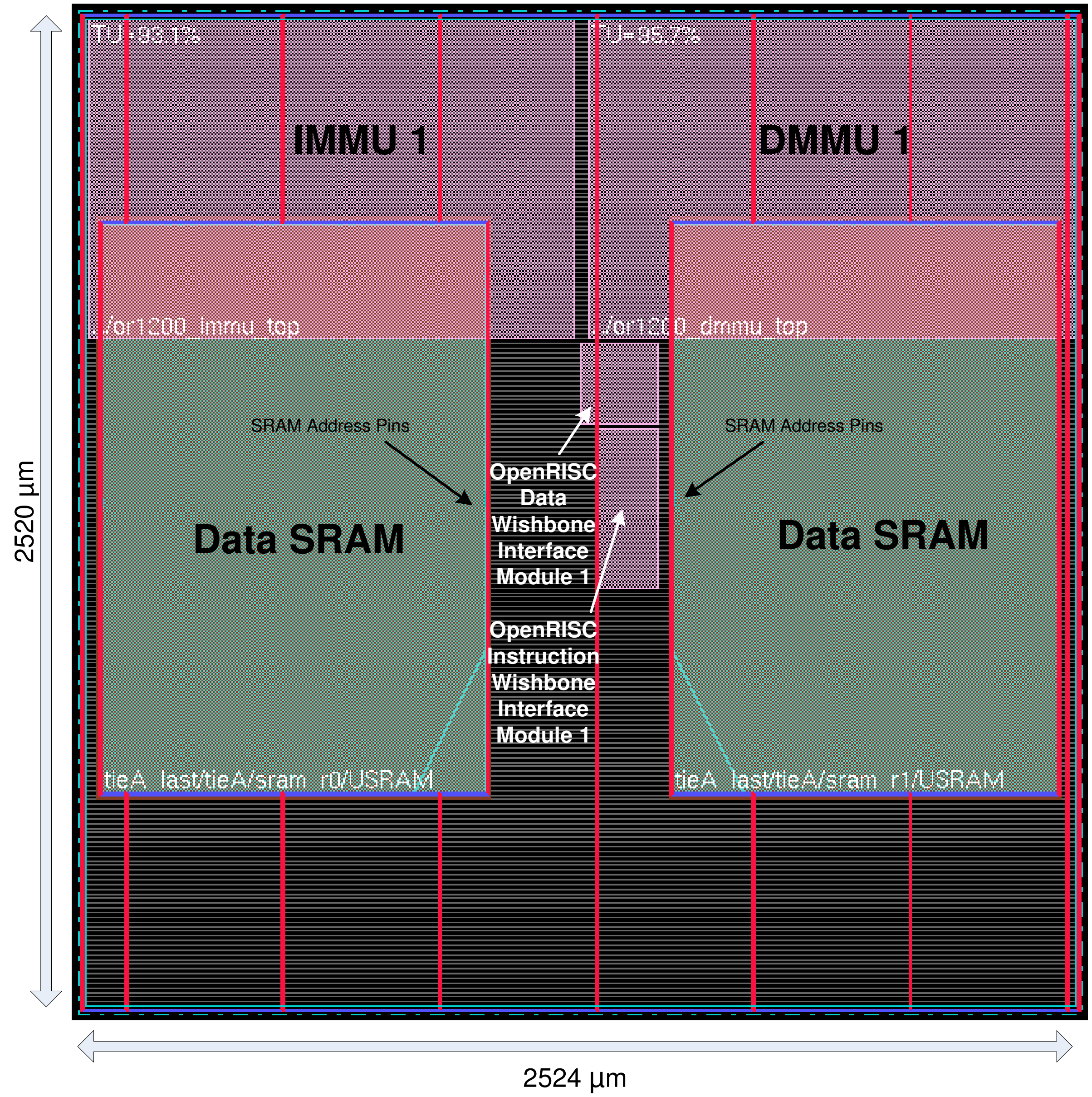

TU=93.1%
TU=95.7%
IMMU 1
DMMU 1
/or1200_immu_top
/or1200_dmmu_top
SRAM Address Pins
SRAM Address Pins
Data SRAM
Data SRAM
OpenRISC Data Wishbone Interface Module 1
OpenRISC Instruction Wishbone Interface Module 1
tieA_last/tieA/sram_r0/USRAM
tieA_last/tieA/sram_r1/USRAM
2520 µm
2524 µm

Figure 6-10: Core size and module floorplanning for tier A of the ORPSOC test case

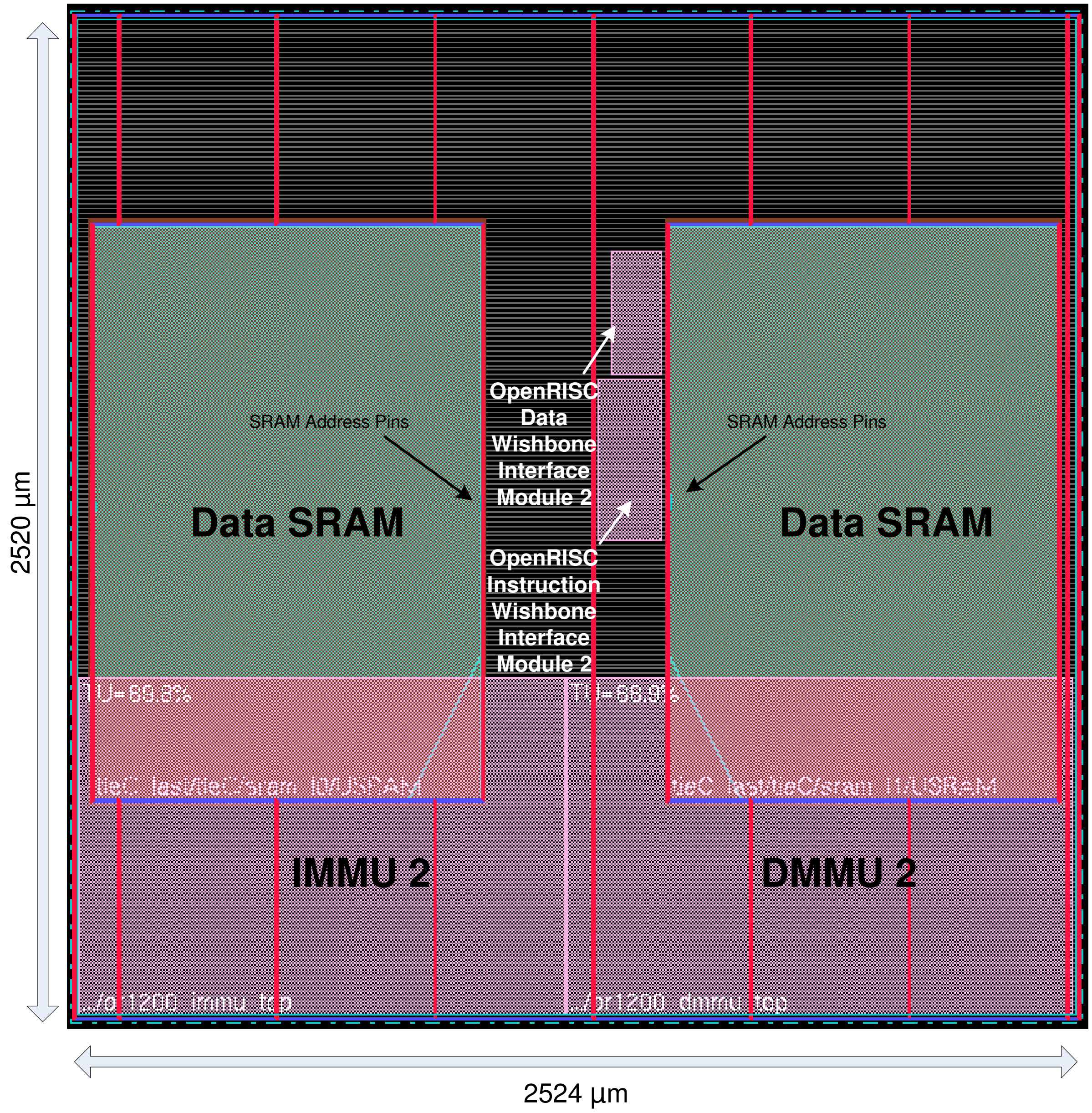

2520 µm
OpenRISC Data Wishbone Interface Module 2
SRAM Address Pins
SRAM Address Pins
Data SRAM
Data SRAM
OpenRISC Instruction Wishbone Interface Module 2
TU=69.8%
TU=66.9%
tieC_last/tieC/sram_10/USRAM
tieC_last/tieC/sram_11/USRAM
IMMU 2
DMMU 2
.../or1200_immu_top
.../or1200_dmmu_top
2524 µm

Figure 6-11: Core size and module floorplanning for tier C of the ORPSOC test case

According to the architecture of the test case, the wishbone traffic cop (see Figure 6-9) communicates with the OpenRISC wishbone interface modules (both the instruction and data ones in Figure 6-10 and Figure 6-11), the data memory controller, and the instruction memory controller. The latter two reside on the same tier as the wishbone traffic cop, allowing SoC Encounter to naturally minimize the lengths of the nets between these components. The OpenRISC wishbone interface modules, however, reside on the tiers above and below the wishbone traffic cop, so the position of these floorplan guides is significant. As verified by the module locations in Figure 6-9, Figure 6-10, and Figure 6-11, the OpenRISC wishbone interface modules and the wishbone traffic cop are all focused in the center of the core area along the edge of the right-most SRAM. A placement that adheres well to these floorplan guides will minimize the interconnect length of the busses between these components. Likewise, the position of the data memory controller relative to the data SRAMs is ideal. The floorplanning of the data memory controller was performed so as to minimize the distance between the 3D vias and the pins of the data SRAMs, which are denoted as "SRAM address pins" in Figure 6-10 and Figure 6-11. Moreover, as discussed in section 6.5, in order to maximize the number of legal 3D via locations, the bounding boxes of the SRAMs on each tier were fixed to one of two positions. For instance, the origin of the data SRAMs in Figure 6-10 and Figure 6-11 share the same set of coordinates. Similarly, the instruction SRAM in Figure 6-9 is perfectly aligned with the right-most data SRAMs in tier A and tier C. With the position of the SRAMs fixed, the data memory controller on tier B is brought close in proximity to the SRAM pins due to its central location. This ensures very short net lengths between the memory controller and the SRAMs. It is worth noting here that the interconnect length between the wishbone traffic cop and the data memory controller is not sacrificed in favor of this floorplanning choice. Referring to Figure 6-9, the guides for the data memory controller and wishbone traffic cop abut on one side yet are still aligned with the components to which they are vertically connected on tier A and tier C.

The potential performance benefits of the floorplanning techniques mentioned above are squandered if the placement result from SoC Encounter does not sufficiently adhere to the floorplan guides. It is entirely possible for SoC Encounter to produce a placement that does not resemble the floorplan if the modules are too dense to be fully contained within their guides. If this occurs, iterations involving the initial floorplanning and placement are necessary to arrive at the desired placement. During the design of the test case, it was observed that the quality of the placement result was strongly dependent on the locations of the SRAMs. In particular, placing the data SRAMs on tier A and tier C too close to one another tended to displace the OpenRISC wishbone interface modules away from the middle of the core. This occurrence was unwelcome given the increase in interconnect length to the wishbone traffic cop on tier B, which was placed in the center of the tier without complications. The solution to this was to separate the data SRAMs horizontally on each tier such that the placement did not show any noticeable migration from the floorplan.

Figure 6-12**,** Figure 6-13**,** and Figure 6-14 collectively show the results of the initial placement for all three tiers. Comparing these figures with the floorplans in Figure 6-9, Figure 6-10, and Figure 6-11 reveals that SoC Encounter largely maintained the initial floorplan in its placement of the tiers. The MMUs on tier A and tier C were entirely too large to fit in a quadrant of the core as intended by the floorplans, but this was expected due to the overlap with the SRAMs. In fact, these figures clearly show the influence that the large, fixed SRAMs have on the quality of the placement. Figure 6-12 shows that the floorplan for tier B was almost exactly duplicated in the placement, where as Figure 6-13 and Figure 6-14 show slight but acceptable displacement from the floorplan for tier A and tier C.

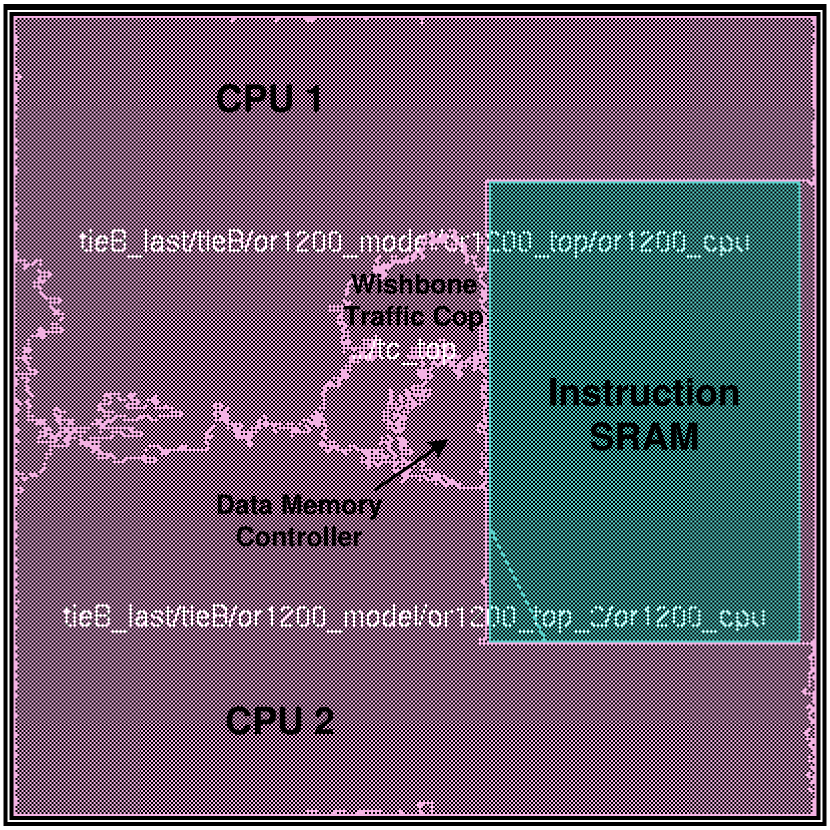

CPU 1
tieB_last/tieB/or1200_model/or1200_top/or1200_cpu
Wishbone
Traffic Cop
tc_top
Instruction
SRAM
Data Memory
Controller
tieB_last/tieB/or1200_model/or1200_top_2/or1200_cpu
CPU 2

Figure 6-12: Initial placement result for tier B of the ORPSOC test case

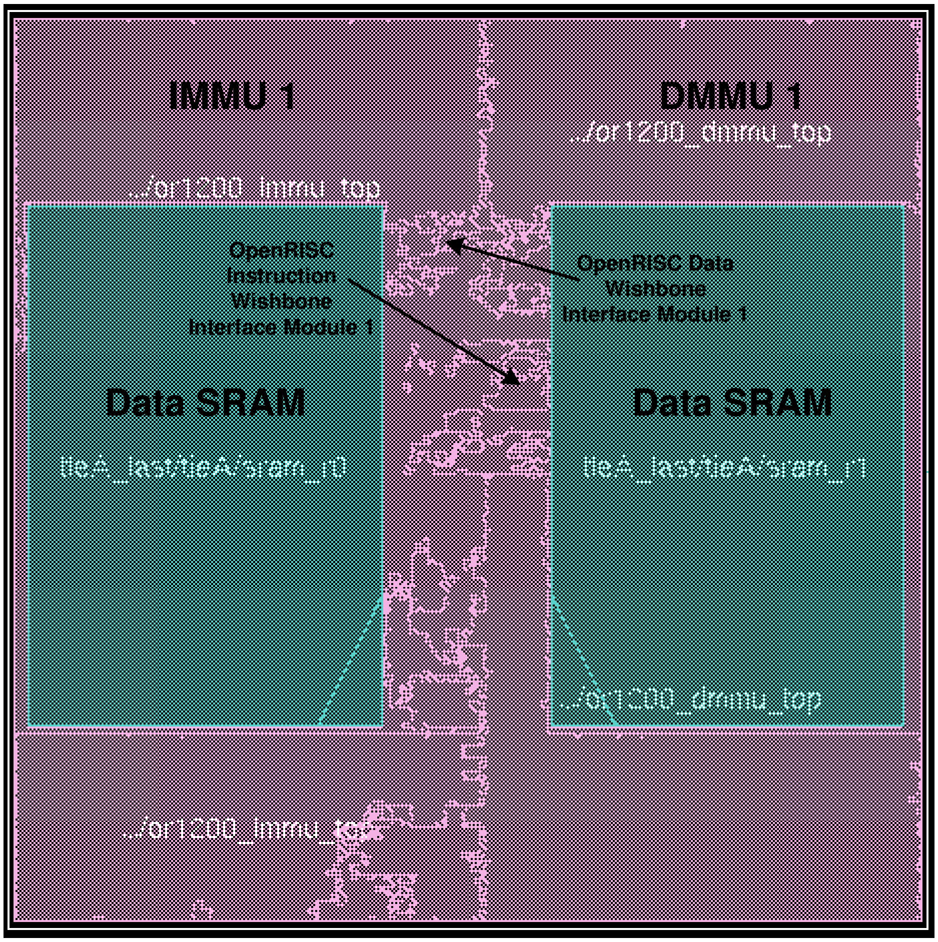


Figure 6-13: Initial placement result for tier A of the ORPSOC test case

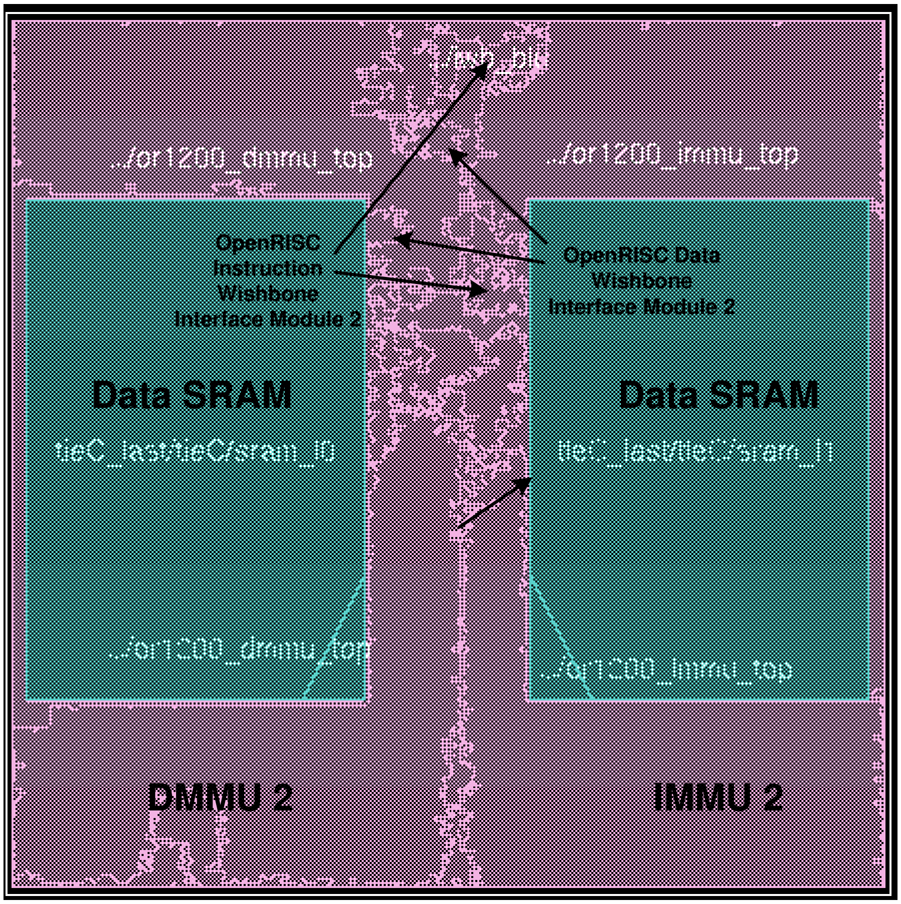


Figure 6-14: Initial placement result for tier C of the ORPSOC test case

6.5 3D Via Alignment

Due to the fact that the 2D placement of each tier is performed without any knowledge of the remaining tiers, the standard cells that comprise 3D vias must be aligned to ensure a functional layout. The vehicle that enables via alignment is a Python script written by the MUSE research group. The script takes as input the three sets of DEF files that were exported from SoC Encounter after the initial placement of the tiers. Regardless of whether or not the via alignment process was completely successful, the script writes a new DEF file per tier that reflects the alignment in the coordinates of the 3D via standard cells. When this step was performed for the 3DIC FFT design, the via alignment was very trivial since the design was composed solely of standard cells. The script simply found the nearest location on each tier to place the 3D via standard cells while considering the incidence of power stripes and fixed standard cells in the DEF files (i.e. geometry disallowing the placement of 3D vias). The situation was more complex with the test case given the existence of the fixed SRAM blocks. To handle this added complexity, the Python script was generalized to respect the bounding box of the SRAM blocks and the accompanying power rings as illegal locations for 3D via standard cells. Furthermore, the bounding box of the SRAM from one tier disallowed the placement of 3D via standard cells across all three tiers. This was the driving force behind completely overlapping the SRAMs between the tiers such that the total area that was removed from the list of legal locations was minimized to twice the area of an SRAM block. If the SRAM blocks were not fixed to one or two position on all three tiers, nearly the entire area of the core would have been blocked, and 3D via alignment would have been very difficult to achieve.

With the amount of 3D via blockage generated from the SRAMs being so large, it was necessary to create good floorplans in the previous stage of the design flow. An initial placement from a poor floorplan would not overlap the modules with 3D vias well, and consequently, the concentration of 3D via standard cells in each tier would be spaced too far apart. During the via alignment stage, this could possibly translate into 3D via alignment locations being "snapped" to the edge of the SRAM blocks. Behavior of this nature is inefficient because the "snapping" action could entail a relocation of the 3D via standard cells to the opposite side of the SRAM blocks, making the associated net lengths exorbitantly long. A manual inspection of the aligned DEF files in SoC Encounter is needed to guarantee the limited existence of any such alignment results. Additionally, if the Python script failed to align all of the 3D via standard cells, the unfixed cells are positioned at the origin in the aligned DEF file. If any 3D via standard cells are found at the origin after importing the DEF file into SoC Encounter, the design flow needs to be rolled back to the floorplanning and initial placement step (see section 6.4), or in the extreme case, a different partitioning strategy may be necessary (see section 6.2).

The 3D via alignment of the test case was made easier by sufficiently spacing the SRAM blocks on each tier. This allowed the modules with vertical connectivity in the center of the core (i.e. between the SRAM blocks) to remain in relatively the same position from floorplanning to initial placement, reducing the distance between the logically connected 3D via standard cells on each tier. To illustrate this, Figure 6-15 captures a zoomed version of the core centers for tier A and tier B after via alignment. The via-aligned DEF files that are written in this stage were used in the making of Figure 6-15. 3D via standard cells on tier B were initially placed inside of the bounding box of the left-most SRAM on tier A because this area was not reserved for a SRAM on tier B. After alignment, the 3D vias were displaced by the bounding box of the SRAM on tier A to the positions shown in the figure. Nonetheless, the ratio of displaced to non-displaced 3D via standard cells is small, so the short vertical interconnects between the tiers are not being inhibited by poor placement of the 3D vias.

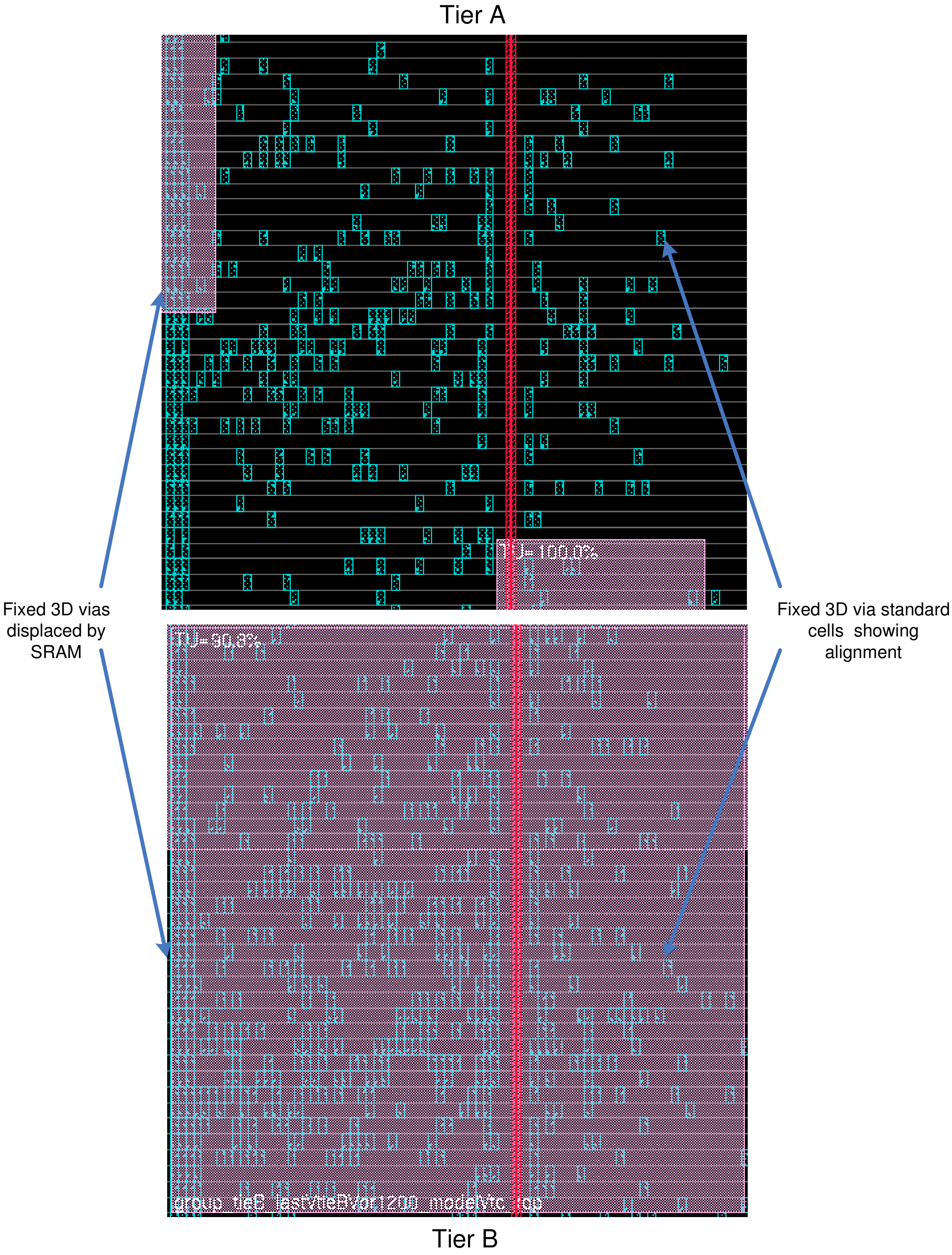
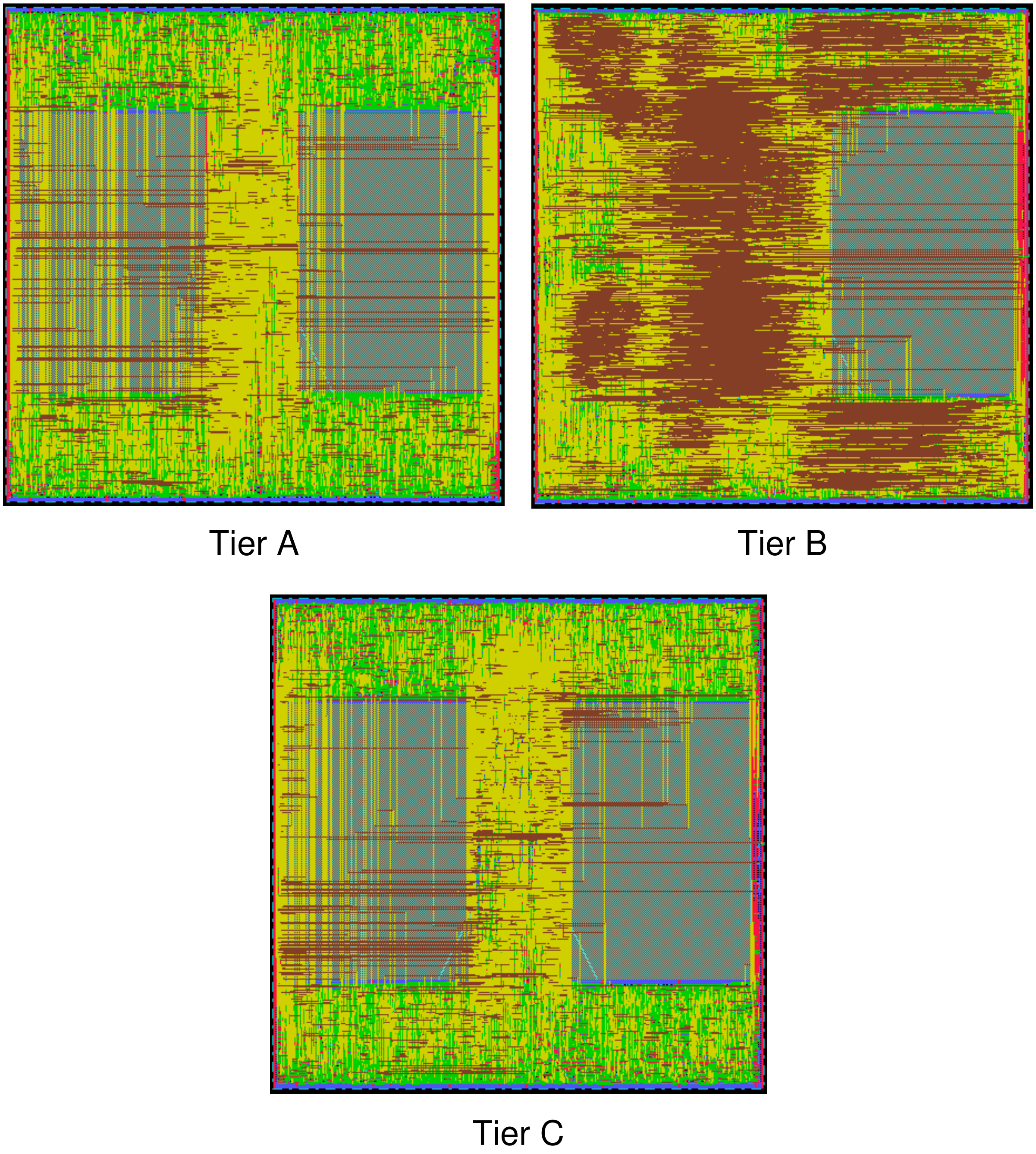


Figure 6-15: Center of core area for tier A and tier B after 3D via alignment

6.6 Final SoC Encounter Placement and Routing

Once the 3D vias have been aligned and fixed to their locations, the standard placement and routing flow may begin. The aligned DEF that was created in the previous step of the design flow is imported into SoC Encounter. From there, a medium effort placement is performed. SoC Encounter will inherently attempt to minimize the net lengths to the fixed 3D via standard cells, so no additional floorplanning is necessary. After placement has completed, the clock tree for each tier is synthesized to a particular specification. For the ORPSOC test case, the target clock skew was an aggressive 50 ps. Due to the complexity of the required clock tree and the number of clock sinks in each tier of the design, the actual clock skew after synthesis as reported by SoC Encounter was closer to 90 ps. After the clock tree has been synthesized, global and detailed routing of each tier is performed. The preference of the MUSE research group is to use “WRoute” that is packaged with SoC Encounter. Five routing layers were needed to route such a high density design. Figure 6-16 shows the tiers of the test case after successful routing runs. This figure will be referenced later in this work to compare the relative amount of routing congestion with that of the 2D design.



Tier A

Tier B

Tier C

Figure 6-16: Routed tiers of the ORPSOC test case

Once routing is finished, three sets of files (one set for each tier) are exported from SoC Encounter. The final DEF file of each tier is saved for later use. Also, the post-route clock tree browser file of each tier, which is needed to insert the clock buffers into the netlist, is saved. Finally, a standard parasitic exchange format (SPEF) file is exported for each tier. The SPEF file specifies a format to represent the resistive and capacitive parasitic data of the wires in the design. This last file will be used for back-annotation of the RC wiring parasitics during the analysis of the 3D design.

6.7 Insertion of Post-route Clock Tree into the Netlist

Throughout the entire design process in SoC Encounter, standard cells were only added during clock tree synthesis. To maintain consistency between the physical design and the netlist that describes the design's functionality, the clock tree must be inserted into the netlist. This removes the ideal clock from driving the clock pins of the synchronous components in the design and replaces the ideal clock at every sink node with a path through the synthesized clock tree. In the 3DIC design flow, clock tree insertion into the netlist is performed automatically using a Python script and the tier clock tree browser files exported after final routing in SoC Encounter. The Python script itself creates a Design Compiler script that contains the commands to instantiate the clock tree cells and assemble the tree within the design. Essentially, the Python script analyzes the three clock tree browser files to find of the hierarchy of the clock tree. This hierarchy is then transferred to the top level 2D "partitioned" netlist in Design Compiler. Recall from section 6.2 that the top level 2D "partitioned" netlist already encapsulates the hierarchy of all three tiers prior to this action. Afterwards, Design Compiler writes the new top level 2D "partitioned" netlist that includes all of the inverters and buffers of the clock tree on each tier. Now all that is needed is accurate RC parasitic information from the 3D design, which is addressed in the next stage of the design flow.

6.8 Merging of Tier-Specific SPEF Files

Just as the clock trees of the individual tiers were “merged” onto one netlist in the previous stage, the wire parasitics must be merged together in order for back-annotation and subsequent analysis in 2D design tools to be plausible. This phase of the design flow is automated by means of script written as part of the research work of MUSE. Termed the “spef merger”, this perl script is very simple in its concept. The “spef merger” simulataneously analyzes the SPEF files of all three tiers and creates a new SPEF file that is the result of the merging process. The merging process involves searching for all instances of 3D via standard cells in the tiers and adjoining the RC parasitics of each net that is connected to a 3D via. Adjoining is accomplished by the annotation of the resistive and capacitive contribution from the physical 3D via. As a result, the nets with vertical connectivity appear as flat 2D nets with extra RC parasitics to account for the inclusion of 3D vias in the 3DIC fabrication process [25]. Since the “spef merger” was designed specifically for the 3DIC design flow outlined in Figure 6-1, the merged 2D SPEF file that is produced by the script preserves the naming conventions of the top level 2D “partitioned” netlist. Together with the clock tree insertion into this netlist, the merged SPEF file provides all of the design information generated by the post-synthesis use of SoC Encounter.

7 Results

The design flow for the 2D ORPSOC test case diverged from the documented 3D design flow at the manual user partitioning step. From there, the placement and routing was fully automated in SoC Encounter. Placement, clock tree synthesis, and routing were performed under the same assumptions as the design of the 3DIC. Likewise, the wiring parasitics were exported from SoC Encounter in the familiar format of a SPEF file. The post-clock tree synthesis netlist was also saved for use in the procedures for obtaining delay and power dissipation values (see section 3). Figure 7-1 shows the 2D design after detailed routing. When compared to the 3D layouts in Figure 6-16, the increased usage of metal 4 and metal 5 (designated by the yellow and brown regions, respectively) in the 2D design is indicative of a higher degree of routing congestion.

In order to compare the 2D and 3D designs, the highest clock frequency achievable must first be determined. For this, the flow for obtaining delay values in PrimeTime was used to analyze the critical path as well as other key paths in both designs. The inverse of the critical path delay was used as an approximation for the maximum clock frequency of each design. Once the clock rate is known, power dissipation tests were performed using the Power Compiler/NC-Verilog flow (see section 3.1). Subsequently, the power distribution among the tiers of the 3D design, together with the delay values, was coupled with the 3DIC thermal model for full temperature convergence and performance degradation analysis. The temperature effects as outlined in this work were considered in the quantification of the performance advantages of migrating the ORPSOC test case from the 2D realm. Reported in this section are the detailed results of the 2D and 3D design comparison.

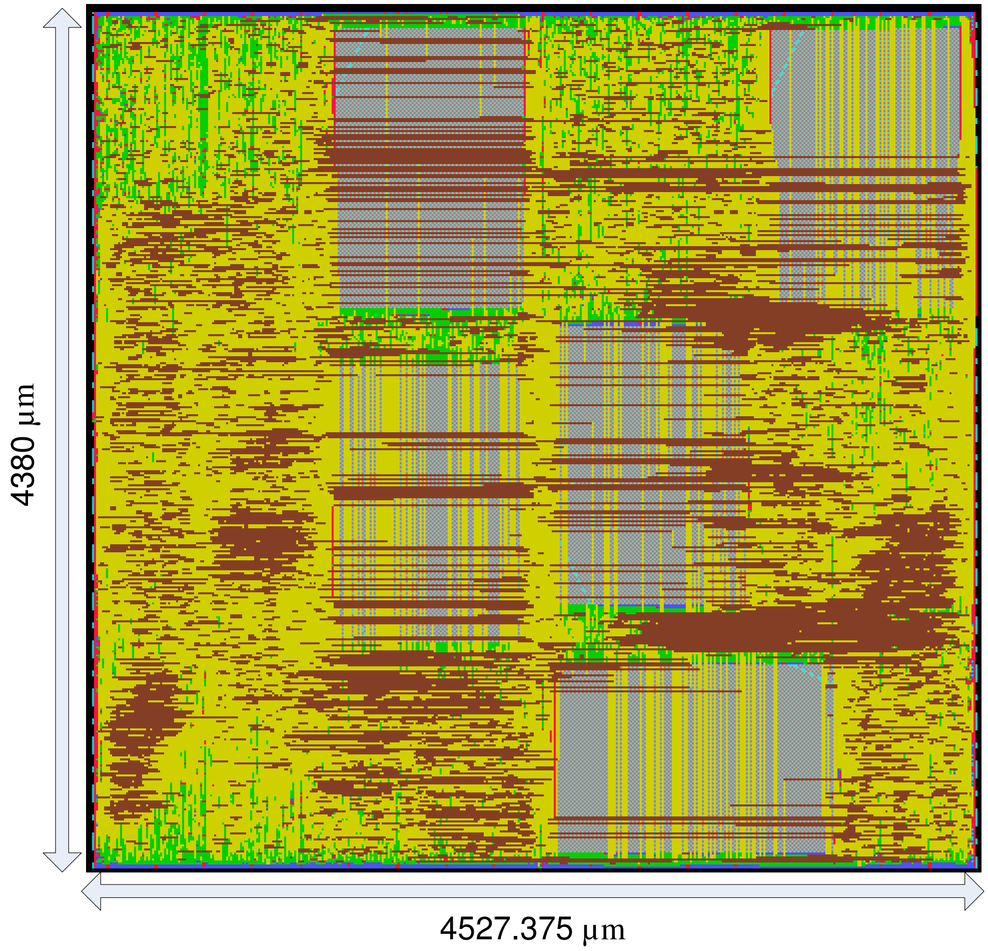


Figure 7-1: The 2D ORPSOC test case after detailed routing

7.1 2D and 3D Path Delay Comparison

Table 7-1 reports the data collected from PrimeTime along with the percentage of delay reduction when moving from 2D to 3D. As evidenced by the table, the 3D design improves upon the critical path delay by 29%, which translates to a maximum clock frequency of 56 MHz as opposed to 40 MHz for that of the 2D design. Additionally, the efficiency of intelligent 3D partitioning and floorplanning is apparent in large gap between the memory path delays of the 2D and 3D designs. These memory path delays represent the summation of the largest register-to-register delays between the CPUs and the SRAMs. The delays are further categorized by SRAM type (data or instruction) and path direction (to/from memory). Here, an important qualification to make is that the numbers shown in Table 7-1 represent the propagation delay before any consideration of performance degradation due to temperature. How the data in Table 7-1 derates with temperature is addressed in section 7.3.

Table 7-1: Path delay data from the 2D and 3D designs

		2D	3D	% Reduction
	Critical Path	25.13 ns	17.87 ns	28.89
Data Memory	To SRAM	33.07 ns	23.0 ns	30.45
	From SRAM	36.6 ns	24.86 ns	32.07
Instruction Memory	To SRAM	34.01 ns	23.27 ns	31.58
	From SRAM	37.82 ns	24.01 ns	36.51

7.2 2D and 3D Power Dissipation Comparison

Three distinct switching activity profiles were used in the calculation of the power dissipation of the designs. These profiles provide a realistic range of switching activities for the OR1200 modules. Recall from section 3.1.3 that one simulation for each piece of software available was performed, and subsequently, a back annotation SAIF file was generated from every simulation. The nets of each design were annotated in Power Compiler based upon the single most active program, the merged switching activity of all of the programs, and the estimated absolute worst case switching activity for the OR1200 modules. The merged switching activity profile describes the weighted average of the switching activities for all of the simulations. To facilitate this, Power Compiler provides a convenient "merge_saif" command whereby the user can annotate the design with multiple SAIF files, giving an appropriate weight for each file. Hence, the switching activity profiles from the single most active program and from the merging process were annotated onto the designs by means of back annotation SAIF files. The worst case switching activity profile, however, was not derived from any simulation. Instead, all of the nets apart from those in the clock tree were annotated with a switching rate of 0.3. This translates to each net having a 30% chance of switching during the clock period. All clock tree nets in the designs transition twice during a clock period, and therefore, these nets were annotated with the expected switching rate of 2.0. It was assumed for the worst case switching profile that the reset pin was held at a constant logic '0'. As previously mentioned, the clock frequency of each design was set to the inverse of the associated critical path delay. Table 7-2, Table 7-3, and Table 7-4 show the results for all of the power dissipation tests. In the case of the 3D design, the power dissipation is further broken into the individual tier contributions. Interestingly, the 3D design had a 30% higher clock frequency, yet it still dissipated less total power than the 2D design. A closer look at the power dissipation from interconnects revealed a contribution of about 29% for the 2D design, favoring 3D integration to reduce the role of interconnects. As a result, the 3D design was able to consume less power while operating at a higher frequency. Any change to the power figures in Table 7-2, Table 7-3, and Table 7-4 due to temperature is accounted for in section 7.3 of this work.

Table 7-2: Power dissipation data for the single most active program profile

Power Dissipation Type	2D	3D	% Reduction
Tier A Dynamic	▨	1.01697 W	▨
Tier B Dynamic	▨	1.66585 W	▨
Tier C Dynamic	▨	1.00697 W	▨
Total Dynamic	**3.798 W**	**3.68 W**	**3.11**
Tier A Leakage	▨	23.1242 mW	▨
Tier B Leakage	▨	16.0516 mW	▨
Tier C Leakage	▨	23.1242 mW	▨
Total Leakage	**62.4 mW**	**62.3 mW**	**0.002**
Total Combined	**3.8604 W**	**3.7423 W**	**3.11**

Table 7-3: Power dissipation data for the merged switching activity profile

Power Dissipation Type	2D	3D	% Reduction
Tier A Dynamic	▨	0.876308 W	▨
Tier B Dynamic	▨	1.37415 W	▨
Tier C Dynamic	▨	0.86597 W	▨
Total Dynamic	**3.1981 W**	**3.116 W**	**2.6**
Tier A Leakage	▨	23.1242 mW	▨
Tier B Leakage	▨	16.0516 mW	▨
Tier C Leakage	▨	23.1242 mW	▨
Total Leakage	**62.4 mW**	**62.3 mW**	**0.002**
Total Combined	**3.2605 W**	**3.1783 W**	**2.6**

Table 7-4: Power dissipation data for the worst case switching activity profile

Power Dissipation Type	2D	3D	% Reduction
Tier A Dynamic	▨	1.28697 W	▨
Tier B Dynamic	▨	2.16415 W	▨
Tier C Dynamic	▨	1.26697 W	▨
Total Dynamic	**4.8881 W**	**4.718 W**	**3.5**
Tier A Leakage	▨	23.1242 mW	▨
Tier B Leakage	▨	16.0516 mW	▨
Tier C Leakage	▨	23.1242 mW	▨
Total Leakage	**62.4 mW**	**62.3 mW**	**0.002**
Total Combined	**4.9505 W**	**4.7803 W**	**3.5**

7.3 Temperature Convergence and Performance Degradation Analysis

For the 3DIC thermal model used in this work (see section 3.3.1), the average of the maximum and minimum thermal conductivity of a tier as reported in [19] was used to calculate the thermal resistance of tier C and the combined thermal resistance of tier A and tier B (labeled as “$\theta_{\text{tier C}}$” and “$\theta_{\text{tier AB}}$” in Figure 3-6). The thermal resistances of the tiers are relatively high due to the lack of thermal vias in the design. Thermal vias, which are composed of two paired 3D vias connecting all three tiers, can be inserted into the design to remove more heat and thus achieve greater thermal conductivity. However, this comes at the expense of design routeability. At the density level of the 3D design, it would not be possible to insert enough thermal vias to significantly affect the thermal conductivity and still guarantee routeability.

The power dissipation breakdowns in Table 7-2, Table 7-3, and Table 7-4 were taken as input to the temperature-power positive feedback loop (see section 3.3.2) to compute final tier temperatures and performance degradation for each of the three switching activity profiles. The assumptions for the 3DIC thermal model are displayed in Table 7-5.

Table 7-5: Assumptions for the 3DIC thermal model

Parameter	Value	Base unit
Thickness of Tier C	7.09	μm
Thickness of Tier AB	12.38	μm
Thickness of Silicon Handle	675	μm
Thickness of Epoxy	50	μm
Conductivity of Tier C	2.5	W/(m-k)
Conductivity of Tier AB	3	W/(m-k)
Conductivity of Silicon	148	W/(m-k)
Conductivity of Epoxy	3.3	W/(m-k)
Thermal Resistance of Heat Sink	0	K/W
Area per Tier	6.36	mm^2
Ambient Temperature	25	°C

Four iterations of the feedback loop calculations were performed to observe temperature convergence, although for the test case, there was minimal temperature increase beyond the first iteration. The small amount of temperature variation in the remaining iterations was expected given the relatively miniscule amount of leakage power as compared to the dynamic component. As indicated in the power dissipation tables, the leakage power is no more than 2% of the total power for all of the switching activity profiles. Figure 7-2**,** Figure 7-3**,** and Figure 7-4 present the temperatures of the 3DIC thermal model across the iterations. The temperature for the heat sink is not shown in these figures, as it is assumed to be at the ambient temperature with zero temperature dropped across it.

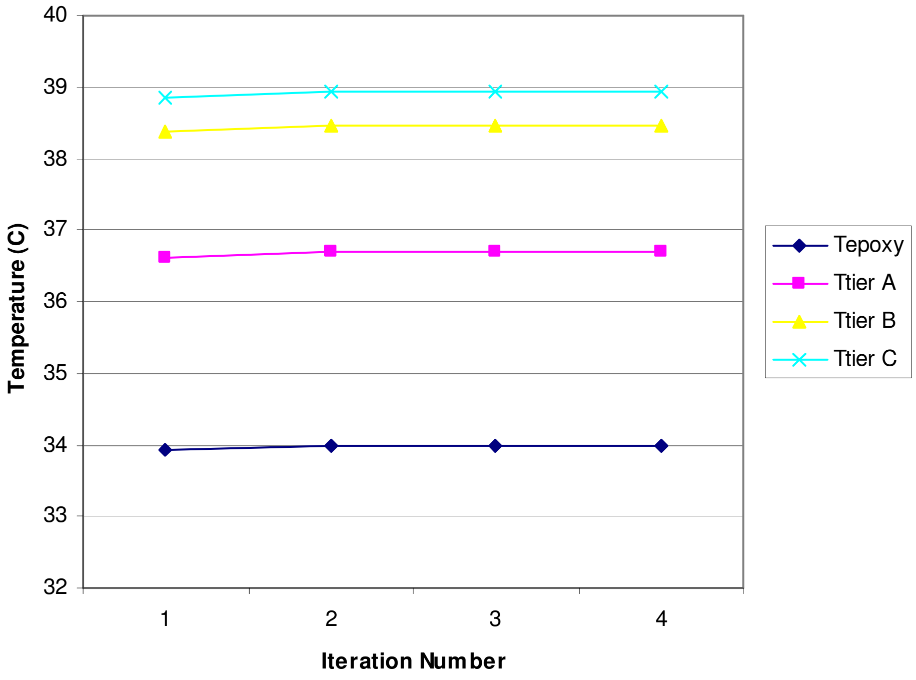


Figure 7-2: Thermal model temperatures for the single most active program profile

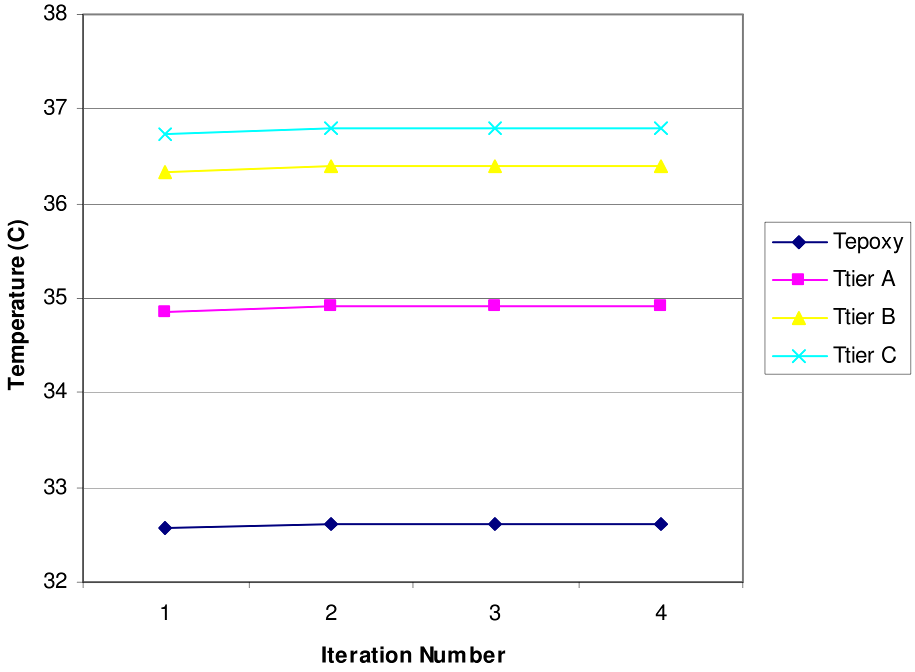


Figure 7-3: Thermal model temperatures for the merged switching activity profile

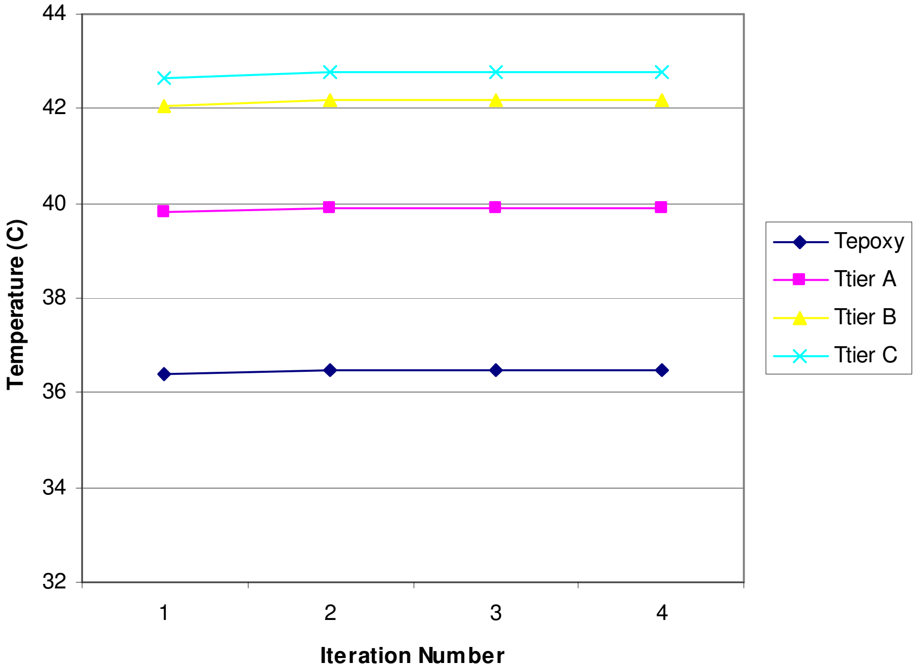


Figure 7-4: Thermal model temperatures for the worst case switching activity profile

Since most standard cell libraries are characterized for timing and power at one specific temperature node, it is useful to interpret the amount of delay and power increase from this single temperature. Extra measures such as the incorporation of the 3DIC thermal model into the design flow would have to taken if there is a substantial percentage of error between the initial and final values for delay and power. Shown in Figure 7-5 is the reaction of leakage power to the departure from the characterized temperature (equivalent to the ambient temperature in the 3DIC thermal model). As developed in this work, the leakage power is strongly dependent on temperature variations. For the test case, this strong dependence manifests in a 31 – 51% increase after temperature convergence. This change in leakage power, however, did not translate into a noticeable impact on the total power dissipation, which increased by only 0.6 – 0.66% for each switching activity profile. Therefore, the power dissipation at the characterized temperature for the test case was an excellent estimation of the power dissipation after consideration of a thermal model. Nonetheless, the error in the initial value of the power dissipation increases as leakage becomes more of a factor in the total power of a 3DIC.

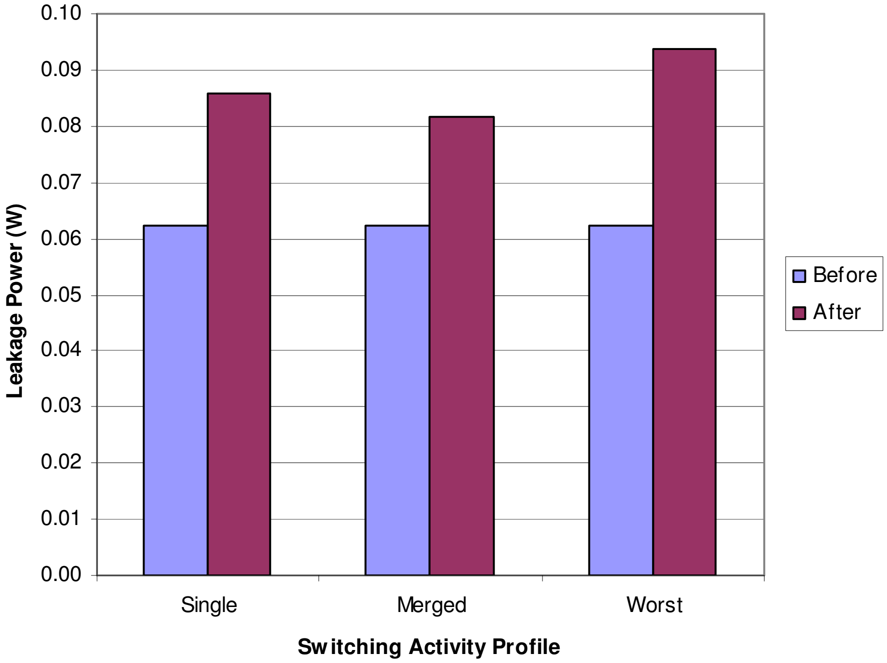


Figure 7-5: Change in total leakage power from ambient temperature

Whereas the temperatures of the 3DIC could be increased to very high levels before the leakage power would become a significant portion of the total power, the speed of the transistors would begin to suffer long before these temperature levels are reached. As evidenced by Figure 7-6**,** the temperature increase from ambient introduces a maximum of 5% combinational path delay increase on tier C, which is by virtue of the thermal model the hottest of the tiers.

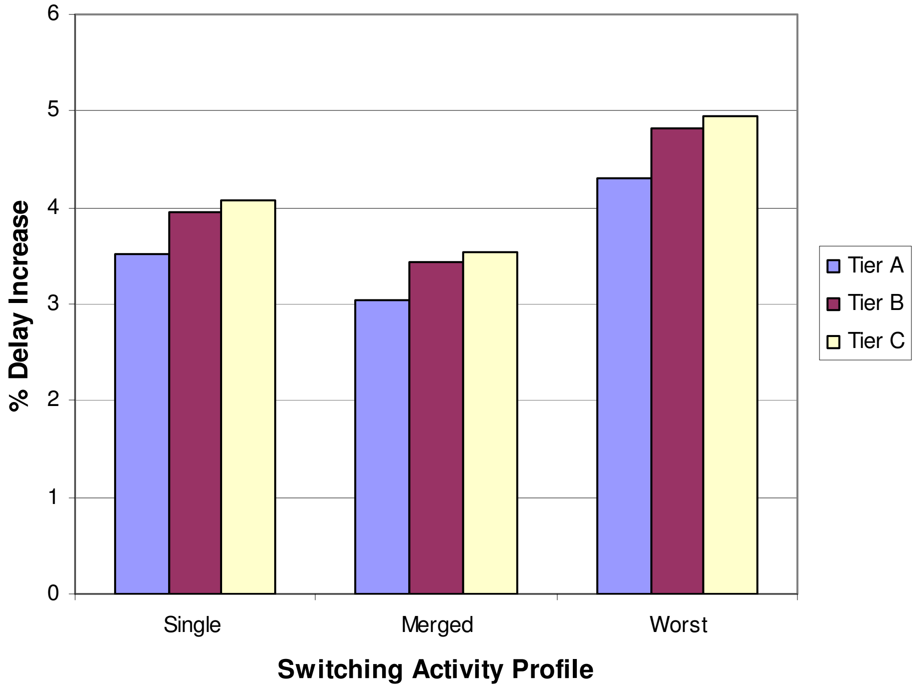


Figure 7-6: Percentage of tier delay increase from ambient temperature

Even if the critical path was fully contained within tier C (a worst-case assumption), this delay degradation does not surmount the critical path improvement over the 2D design. The 2D IC thermal model can be viewed as having a single tier with three times the area as tier A with the same amount of heat flowing through it. Figure 7-7 shows this equivalent thermal model for the 2D IC. The area of each component of the cross-section of the model was calculated from the dimensions of the final 2D layout (see Figure 7-1). The total area for the 2D version was slightly greater than three times that of the 3DIC, at 19.82 mm^2.

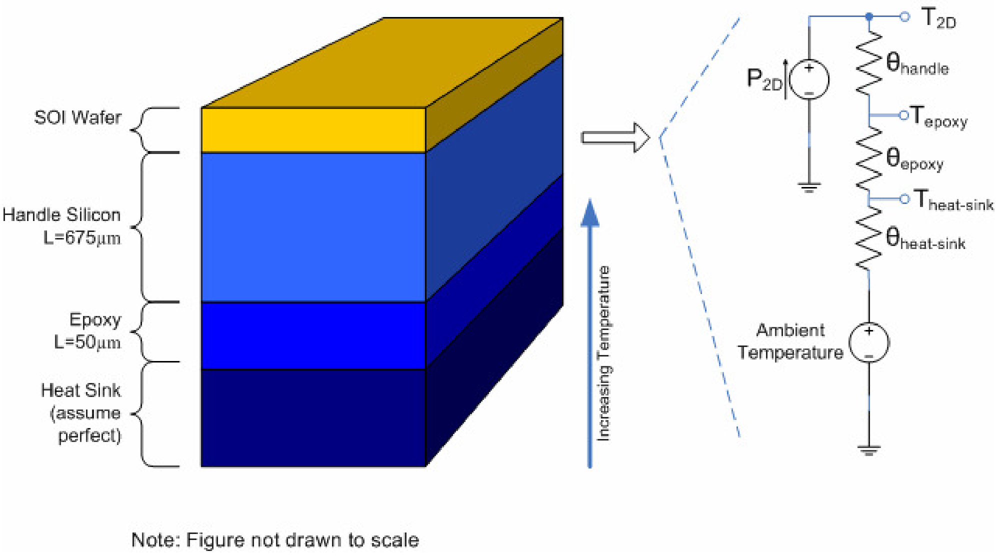


Figure 7-7: Thermal model of the 2D IC

Using the model shown in Figure 7-7 with the power dissipation values in Table 7-2, Table 7-3, and Table 7-4, the same process used to calculate temperature convergence and performance degradation for the 3DIC was performed for the 2D version. As expected, the dependence of total power on temperature was negligible, since the 2D design has the same insignificant leakage power component as the 3D design. In terms of the delay increase with temperature, the 2D design did not slow down from its ambient operating frequency as much as seen in the 3D case. However, the difference between the two designs was not show-stopping for the 3DIC. Figure 7-8 depicts the delay increase (in percent) of the hottest tier of the 3D design versus the 2D design. As indicated in the figure, the difference in the percentage of delay increase reached a maximum of just over 3%.

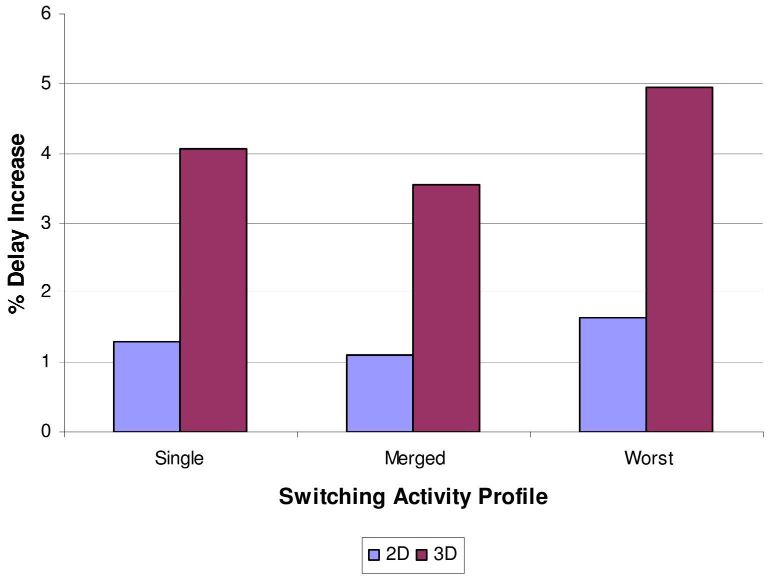


Figure 7-8: Delay increase comparison between tier C of the 3D design and the 2D design

Additionally, the delay increase at each switch activity profile from Figure 7-8 was used to scale the original path delay values from PrimeTime (see Table 7-1) to produce a worst-case speed comparison between the 2D and 3D design after temperature convergence. This comparison is labeled as "worst-case" because the highest temperature seen in the 3DIC (tier C) was used to calculate the delay increase for all combinational paths in the design irrespective of their location within the 3D "stack". The scaled values from Table 7-1 are shown in Table 7-6. The central aspect of this table is that critical path improvement was only diminished by 2.3% due to the higher temperatures of the 3DIC.

Table 7-6: Path delays of each design after temperature dependence

		2D	3D	% Reduction
	Critical Path	25.54 ns	18.75 ns	26.59
Data Memory	To SRAM	33.61 ns	24.14 ns	28.18
	From SRAM	37.2 ns	26.09 ns	29.87
Instruction Memory	To SRAM	34.56 ns	24.42 ns	29.34
	From SRAM	37.82 ns	25.2 ns	33.37

8 Conclusion

The design of a 3DIC based on the OpenRISC Reference Platform System-on-chip has been documented in this work. It was shown that, through the use of manual tier partitioning and floorplanning, the intrinsic advantages of 3D vias can be harnessed. When comparing the 3D test case with a functionally equivalent 2D design, steps were taken to acknowledge the non-uniform temperature profile between the tiers of the 3DIC. Temperature dependent predictive models for power dissipation and combinational path delay were derived and verified in this work to enable the use of the power-temperature positive feedback loop. Utilizing realistic switching activity profiles as generated from simulations of actual microprocessor workloads, iterative calculations were performed about this feedback loop to observe temperature convergence of the 3DIC thermal model. The increased tier path delays predicted by the temperature dependent models after temperature convergence did not appreciably diminish the improvement in critical path delay of nearly 29% over the 2D design. After temperature convergence was observed, the improvement to the critical path delay was still 26.59%. Moreover, power dissipation analysis revealed that the 3D design consumed, on average, 3% less power than the 2D design while running at the higher clock frequency dictated by the reduction in critical path delay. Both in the case of power dissipation and timing delays, the analyses at the characterized operating temperature provided reliable performance estimation even after consideration of the 3DIC thermal model and the power-temperature positive feedback loop.

For the current and future interconnect-limited integrated circuit technologies, 3D integration presents a compelling alternative to the conventional methods of circuit integration. In fact, it is the belief of the MUSE research group that the 3DIC design flow presented in this work has applications in the development of novel memory structures. One can imagine 3D integration being used to meet the density demands of future memory devices at a more matured process feature size. 3DICs, however, reach new heights with respect to design challenges. Additional complexities will soon be introduced from the outlook on leakage power scaling and from the ever-increasing power density of integrated circuits. Indeed, 3D partitioning and floorplanning irrespective of thermal effects is rapidly approaching its viability limits. In its place, true 3D thermally-aware placement will be necessary to manage the speed difference between the tiers of the 3DIC as well as any power dissipation and temperature requirements. How this problem will be solved currently is and will remain to be a research topic for years to come. Nevertheless, the design time needed for the manual partitioning strategy used in this work to find an optimal solution is insurmountable, and the need for such a solution to maximize the benefits of 3D vias will only be amplified in the upcoming technology generations.

References

[1] *International Technology Roadmap for Semiconductors,* Semiconductor Industry Association, 2003.

[2] *OpenRISC Reference Platform System-on-a-Chip and OpenRISC 1200 IP Core Specification,* available from Opencores.org at http://www.opencores.org/projects.cgi/web/or1k/orpsoc.

[3] V. Suntharalingam, *et al.*, “Megapixel CMOS Image Sensor Fabricated in Three-Dimensional Integrated Circuit Technology,” *Intl. Solid State Circuits Conf. Dig. of Tech. Papers*, Feb. 2005, pp. 356-357.

[4] A. Rahman and R. Reif, “Thermal Analysis of Three-Dimensional (3-D) Integrated Circuits (ICs),” *Proceedings of the IEEE International Interconnect Technology Conference,* June 2001, pp. 157-159.

[5] A. Vassighi, A. Keshavarzi, S. Narendra, G. Schrom, Y. Ye, Se. Lee, G. Chrysler, M. Sachdev, and V. De, “Design Optimizations for Microprocessors at Low Temperature,” *Proceedings of the 41st Annual Conference on Design Automation,* June 2004, pp 2-5.

[6] UC Berkeley Device Group, “BSIM3v3.3 MOSFET Model,” University of California, Berkeley, CA, July 2005.

[7] J. Rabaey, A. Chandrakasan, and B. Nikolić, Digital Integrated Circuits: A Design Perspective. New Jersey: Prentice Hall, 2003.

[8] W. Liao, L. He, K. Lepak, “Temperature and Supply Voltage Aware performance and Power Modeling at Microarchitecture Level,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* July 2005, pp. 1043-1048.

[9] K. Kanda, K. Nose, H. Kawaguchi, T. Sakurai, “Design Impact of Positive Temperature Dependence on Drain Current in Sub-1-V CMOS VLSIs,” *IEEE Journal of Solid State Circuits,* October 2001, pp. 1559-1562.

[10] W. Jiang, V. Tiwari, E. de la Iglesia, and A. Sinha. "Topological Analysis for Leakage Prediction of Digital Circuits," *ASP-DAC/VLSI Design 2002*, 2002, p. 39.

[11] C. Kim, and K. Roy. "Dynamic VTH Scaling Scheme for Active Leakage Power Reduction," *2002 Design, Automation and Test in Europe Conference and Exhibition*, 2002, p. 163.

[12] H. Su, F. Liu, A. Devgan, E. Acar, S. Nassif, “Full Chip Leakage Estimation Considering Power Supply and Temperature Variations,” *Proceedings of IEEE International Symposium on Low Power Electronics and Design, 2003.* pp. 78-81.

[13] A. Bellaouar, A. Fridi, M. I. Elmasry, and K. Itoh, “Supply Voltage Scaling for Temperature Insensitive CMOS Circuit Operation,” *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing,* March 1998, pp. 415.

[14] R. Ho, K. Mai, M. Horowitz, “The Future of Wires,” *Proceedings of the IEEE,* April 2001, pp. 491.

[15] Synopsys, *Synopsys PLI 3.0,* available in Core Synthesis Tools product, Release W-2004.12-SP4, 2004.

[16] *OpenRISC GNU Toolchain,* project website available at http://www.opencores.org/projects.cgi/web/or1k/gnu_toolchain_port.

[17] Synopsys, *Power Compiler User Guide,* Release V-2004.06, June 2004.

[18] Synopsys, *PrimeTime,* information available at http://www.synopsys.com/products/analysis/primetime_ds.html.

[19] W. Davis, H. Hua, S. Melamed, “Thermal Bounday Study for the MITLL 3D 0.18 μm FDSOI Process,” North Carolina State University, Raleigh, NC, September 2005, pp. 1-5.

[20] *3DIC Design Notes and Examples,* included in the MITLL Low-Power FDSOI CMOS Process release, December 2004, pp. 1-4.

[21] N. Walker, “Integrated Circuit Module Generation,” *Master’s Thesis,* University of California, Berkeley, CA, 1999.

[22] W. R. Davis, J. Wilson, S. Mick, J., X. Jian, H. Hao, C. Mineo, A. M. Sule, M. Steer, P. D. Franzon, “Demystifying 3D ICs: The Pros and Cons of Going Vertical,” *Design & Test of Computers, IEEE,* Volume 22, Issue 6, pp. 498 – 510.

[23] Synopsys, *Design Compiler*, information available at http://www.synopsys.com/products/logic/design_compiler.html.

[24] Cadence, *SoC Encounter*, information available at http://www.cadence.com/products/digital_ic/soc_encounter/index.aspx.

[25] C. Mineo, “Clock Tree Insertion and Verification for 3D Integrated Circuits,” *Master’s Thesis,* North Carolina State University, Raleigh, NC, September 2005, pp. 30 – 33.