行政院國家科學委員會專題研究計畫 成果報告

針對 3D 整合之電子設計自動化技術開發--子計畫五：應用在驗證與測試 3D IC 整合過程中以計算智慧為基礎的測試向量產生方法(2/2)

研究成果報告(完整版)

計 畫 類 別 ： 整合型
計 畫 編 號 ： NSC 99-2220-E-009-039-
執 行 期 間 ： 99 年 08 月 01 日至 100 年 07 月 31 日
執 行 單 位 ： 國立交通大學電信工程學系（所）

計 畫 主 持 人 ： 溫宏斌

計畫參與人員： 碩士班研究生-兼任助理人員：黃宣銘
碩士班研究生-兼任助理人員：張家慶
碩士班研究生-兼任助理人員：陳韋廷
碩士班研究生-兼任助理人員：張竣惟
碩士班研究生-兼任助理人員：林玗璇
碩士班研究生-兼任助理人員：顧鈞堯
碩士班研究生-兼任助理人員：林昱澤
碩士班研究生-兼任助理人員：許凱華
博士班研究生-兼任助理人員：廖千慧

中 華 民 國 100 年 10 月 31 日

中文摘要： 多核心單晶片系統為實現高效能嵌入式系統的可靠方式，而三維積體電路為現在階段實作多核心系統的最佳技術。但是三維積體電路本身仍存在許多議題等待著我們來探討。其中，最常被人關注的即是良率與高功耗密度問題。這是因為在堆疊的過程中可能會產生缺陷。除此之外，矽穿孔除了會占面積也可能是會導致缺陷的來源。還有三維多核心處理器由於高功耗密度問題，易有大量的能量消耗。因此，除何降低耗能便成為另一個主要議題。

首先 在考慮矽穿孔使用數目的限制下，我們提出一個快速的兩階段演算法，來決定掃描鏈的串接順序。第一階段，先使用貪婪演算法稱之為多片段錯誤嚐試法，得到一組初始解。第二階段為得到一組最佳解，會利用三維平坦化與三維鬆弛化來降低連線或功耗成本與符合矽穿孔使用數目限制。實驗結果顯示我們所提出的演算法，可達到與基因演算法相差不多的效能，並且效率比基因演算法快一百倍以上。這也顯示我們的方法可實際應用於三維積體電路掃描鏈的設計。

三維積體電路為實作出高效能嵌入式系統的最佳技術，但其耗能問題可能會導致不理想的表現。因此，針對耗能最小化，許多利用動態壓頻調整法被提出。然而，大部份先前的研究團隊使用的都是固定式核心對應法，留下了許多可再降低耗能的進步空間。因此，另一個研究議題便是在考慮核心間資料傳輸延遲時間，提出降低耗能的任務排程演算法。此演算法結合動態重覆對應法來提高耗能節省率。實驗結果顯示，我們所提出的演算法，其耗能節省率較先前的演算法高出十六個百分比。除此之外，我們的演算法不僅比整數線性規劃快上一千倍以上，還可達到與整數線性規劃解相差不多的耗能節省率。

英文摘要： To fulfill high-performance demands on embedded systems, MPSoC (Multiprocessor System-on-a-Chip) design methodology arises as a new paradigm where 3D integration is the state-of-the-art enabling technique. However, many issues wait being resolved to enable the popularization of 3D stacking. The most common issues include yield loss and high power density. The die-stacking steps may introduce defects. Also through-silicon vias (TSVs) will incur additional area overhead and may become another source of defects. Besides, since a 3D multi-core processor often consumes excessive energy, leading to a problem of high power density, energy efficiency becomes its paramount concern.

First, this work addresses the problem of scan-chain ordering under a limited number of TSVs constraints by presenting a fast two-stage algorithm as a solution. To enable three-dimensional (3D) optimization, a greedy algorithm, referred to as the multiple fragment heuristic, is modified to derive a good initial solution at stage one. Stage two initiates two local refinement techniques, 3D planarization and 3D relaxation, to reduce the wire or power cost and to relax the number of TSVs in use to meet the constraint, respectively. Experimental results show that the proposed algorithm results in comparable performance to a Genetic-Algorithm (GA) method but it runs at least two-orders faster, which makes it more practical for TSV-constrained scan-chain ordering for 3D-IC designs.

To achieve high-performance computing on embedded systems, three-dimensional (3D) multi-core processors have become a promising alternative where energy efficiency is crucial to its success. Many heuristics applying Dynamic
Voltage and Frequency Scaling (DVFS) techniques were proposed for energy minimization. However, most of the previous works were built upon a fixed task-to-core mapping where many slack spaces can be further improved. Therefore, the other goal in this work is to propose two dynamic remapping strategies to enhance an energy-aware task-scheduling algorithm considering transmission cost. Experimental results show that the energy-saving rate of the best strategy is 16 percent higher than the previous work on average. Moreover, compared to an ILP solution, the enhanced algorithm can run at least three-order faster while achieving comparable performance on total energy consumption.

行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告
□ 期中進度報告

應用在驗證與測試 3D IC 整合過程中以計算智慧為基礎的測試向量產生方法

計畫類別：□ 個別型計畫　　■ 整合型計畫
計畫編號：NSC　99-2220-E-009-039
執行期間：98 年 8 月 1 日至 100 年 7 月 31 日

計畫主持人：溫宏斌
共同主持人：
計畫參與人員：廖千慧 黃宣銘 陳韋廷 張家慶 林昱澤 許凱華 林玗璇
張竣惟 顧鈞堯

成果報告類型(依經費核定清單規定繳交)：□精簡報告　■完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管
計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年■二年後可公開查詢

執行單位：國立交通大學電機工程學系/電信工程研究所
中　華　民　國　100　年　10　月　31　日

摘 要

多核心單晶片系統為實現高效能嵌入式系統的可靠方式，而三維積體電路為現在階段實作多核心系統的最佳技術。但是三維積體電路本身仍存在許多議題等待著我們來探討。其中，最常被人關注的即是良率與高功耗密度問題。這是因為在堆疊的過程中可能會產生缺陷。除此之外，矽穿孔除了會占面積也可能是會導致缺陷的來源。還有三維多核心處理器由於高功耗密度問題，易有大量的能量消耗。因此，除何降低耗能便成為另一個主要議題。

首先 在考慮矽穿孔使用數目的限制下，我們提出一個快速的兩階段演算法，來決定掃描鏈的串接順序。第一階段，先使用貪婪演算法稱之為多片段錯誤嚐試法，得到一組初始解。第二階段為得到一組最佳解，會利用三維平坦化與三維鬆弛化來降低連線或功耗成本與符合矽穿孔使用數目限制。實驗結果顯示我們所提出的演算法，可達到與基因演算法相差不多的效能，並且效率比基因演算法快一百倍以上。這也顯示我們的方法可實際應用於三維積體電路掃描鏈的設計。

三維積體電路為實作出高效能嵌入式系統的最佳技術，但其耗能問題可能會導致不理想的表現。因此，針對耗能最小化，許多利用動態壓頻調整法被提出。然而，大部份先前的研究團隊使用的都是固定式核心對應法，留下了許多可再降低耗能的進步空間。因此，另一個研究議題便是在考慮核心間資料傳輸延遲時間，提出降低耗能的任務排程演算法。此演算法結合動態重覆對應法來提高耗能節省率。實驗結果顯示，我們所提出的演算法，其耗能節省率較先前的演算法高出十六個百分比。除此之外，我們的演算法不僅比整數線性規劃快上一千倍以上，還可達到與整數線性規劃解相差不多的耗能節省率。

關鍵字：*矽穿孔 ; 掃描測試 ; 核心對應法; 任務排程; 動態壓頻調整*

Abstract

To fulfill high-performance demands on embedded systems, MPSoC (Multiprocessor System-on-a-Chip) design methodology arises as a new paradigm where 3D integration is the state-of-the-art enabling technique. However, many issues wait being resolved to enable the popularization of 3D stacking. The most common issues include yield loss and high power density. The die-stacking steps may introduce defects. Also through-silicon vias (TSVs) will incur additional area overhead and may become another source of defects. Besides, since a 3D multi-core processor often consumes excessive energy, leading to a problem of high power density, energy efficiency becomes its paramount concern.

First, this work addresses the problem of scan-chain ordering under a limited number of TSVs constraints by presenting a fast two-stage algorithm as a solution. To enable three-dimensional (3D) optimization, a greedy algorithm, referred to as the multiple fragment heuristic, is modified to derive a good initial solution at stage one. Stage two initiates two local refinement techniques, 3D planarization and 3D relaxation, to reduce the wire or power cost and to relax the number of TSVs in use to meet the constraint, respectively. Experimental results show that the proposed algorithm results in comparable performance to a Genetic-Algorithm (GA) method but it runs at least two-orders faster, which makes it more practical for TSV-constrained scan-chain ordering for 3D-IC designs.

To achieve high-performance computing on embedded systems, three-dimensional (3D) multi-core processors have become a promising alternative where energy efficiency is crucial to its success. Many heuristics applying Dynamic Voltage and Frequency Scaling (DVFS) techniques were proposed for energy minimization. However, most of the previous works were built upon a fixed task-to-core mapping where many slack spaces can be further improved. Therefore, the other goal in this work is to propose two dynamic remapping strategies to enhance an energy-aware task-scheduling algorithm considering transmission cost. Experimental results show that the energy-saving rate of the best strategy is 16 percent higher than the previous work on average. Moreover, compared to an ILP solution, the enhanced algorithm can run at least three-order faster while achieving comparable performance on total energy consumption.

Keyword: TSV; scan testing; core mapping, task scheduling, DVFS

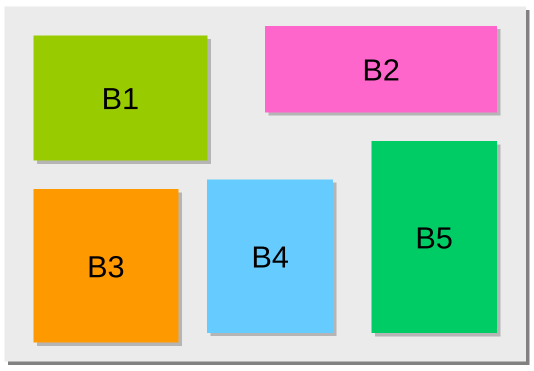
Table of Content

List of Figures

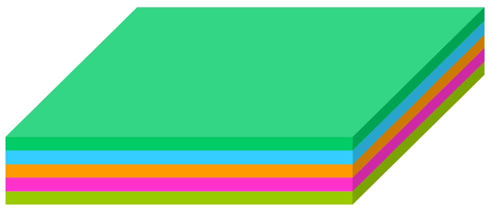
List of Tables

Chapter 1
Introduction

The next generation of integrated micro-system technologies enables ever increasing functionality and performance by utilizing the 3rd dimension. 3D integration of designs can bring together the virtues of overall performance, heterogeneous integration and miniaturization. International Technology Roadmap of Semiconductor (ITRS) points out that 3D integration is one of the most promising solutions to sustain the performance improvement beyond 65nm. Figure 1.1(a) illustrates an example of the integrated micro-system composed of five individual functional blocks. Traditionally, these five blocks are integrated in 2D packaging or printed wiring board (PWB). In the fashion of 3D architecture, each block can be built in the separate layer and stacked one-by-one vertically as shown in Figure 1.1(b). Apparently, the form-factor (i.e. X and Y dimensions) of the micro-system shrinks significantly and the overall and worst-case interconnect length can be also reduced.



(a)



(b)

Figure 1.1: 2D and 3D integration of micro-systems

Potential advantages of 3D integration technology captured significant attention. However, many issues wait being resolved to enable the popularization of 3D stacking. Since most of these issues vary significantly according to its application and the technology used, they needs inspection and evaluation one by one. However, the most common issues that have been targeted frequently include thermal management, yield, uncommon die size, cost and inadequate infrastructure for design, equipment and processing where the thermal management and yield issues capture more interest of research and are further elaborated into details.

In this technology, after wafers or ICs are fabricated, devices are stacked in 3D and interconnected by through-silicon-via (TSV). Therefore, IC stacking can be performed either at wafer level or die level. Figure 1.2 shows the wafer-level stacking technology. Note that through-silicon-vias can go through either bulk silicon or SiO2. Recently, a great amount of effort has been devoted to this line of research & development both in academy and industry. Among all vertical-integration techniques, through-silicon via provides the best timing and power performance for interconnection. However, TSVs typically incur additional area overhead and may become another source of defects [6]. Therefore, considering yield loss and area cost, the number of TSVs in use is typically limited in a 3D Integrated Circuit (IC) design.

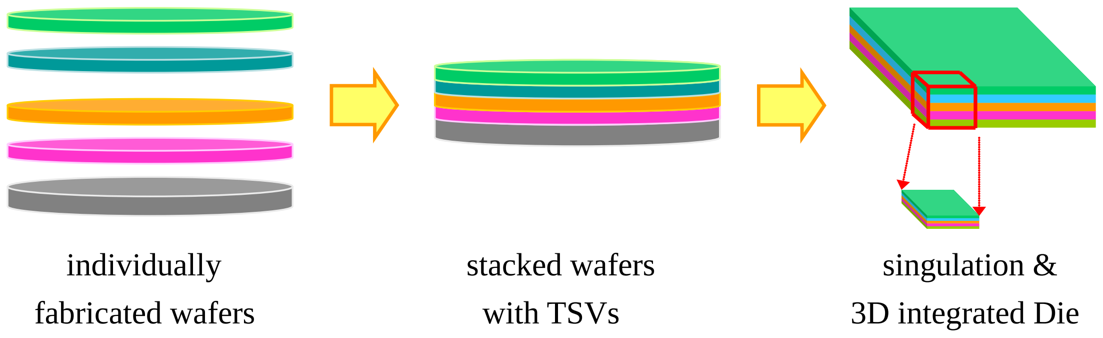


Figure 1.2: 3D IC stacking technology

According to the prediction of International Technology Roadmap for Semiconductors (ITRS), the era of tera-scale embedded systems is approaching [24], in which having numerous processing elements on a single chip has been the mainstream and strongly advocated by both the academy and industry. To fulfill high-performance demands on embedded systems, MPSoC (Multiprocessor System-on-a-Chip) design methodology arises as a new paradigm where 3D integration is the state-of-the-art enabling technique since it can benefit from shorter interconnect delay, footprint, performance and heterogeneous technology mixing.

Potential advantages of 3D integration technology captured significant attention. However, 3D multi-core system has a severe thermal issue due to high power density. High temperature spots worsen the system reliability and cause failure. The problem of consuming tremendous amount of energy is more severe on high-performance computing systems. Therefore, the minimization of power consumption has become a paramount concern for present large-scale 3D multi-core systems.

Nowadays, energy-efficiency is crucial to low-power design and high-performance computing. Many previous researches focus on energy minimization that can be applied at both the behavior level and the physical level. Many physical design solutions are proposed for this issue, including microchannel liquid cooling [28], floorplanning [29] and thermal TSVs [30]. Among all the high-level techniques are more effective than the low-level ones for energy minimization especially on 3D multi-core systems, such as thermal-aware task scheduling [31] and power-aware task scheduling [32]. More advanced techniques for energy efficiency are proposed and can be classified into Voltage selection (VS) (also called voltage scheduling) [33] and power management (PM) [34]. Both techniques mainly target the system-level energy saving while VS is more attractive than PM in general [35]. One of VS scheduling, Dynamic voltage and frequency Scaling (DVFS) scheduling algorithms has become more popular recently.

1.1 Research Goal

Both pre-bond testing and post-bond testing are important for improving the yield of 3D ICs. Scan-chain design is the most prevailing Design-for-Testability (DFT) technique which aims to reduce the difficulty of testing on the Circuit Under Test (CUT). Experimental results in [17] also suggested that the more TSVs in use in the scan chain, the less wire cost. Such observation combined with the TSV induced yield loss indicates an important tradeoff between wire cost and the number of TSVs in use. Therefore, a constraint of TSVs in use must be considered for a 3D-IC design. This work addresses the problem of scan-chain ordering under a limited number of Through-Silicon Vias (TSVs) constraints by presenting a fast two-stage algorithm as a solution.

In addition, since a 3D multi-core processor often consumes excessive energy, leading to a problem of high power density [26] [27], energy efficiency becomes its paramount concern. Therefore, the minimization of power consumption has become a paramount concern for present large-scale 3D multi-core systems. In our work, we also focus on energy minimization for 3D multi-core architecture.

1.2 Method

For enabling pre-bond testability, Lewis et al. [18] proposed a scan-island based design and Kumar et al. [19] proposed a hyper-graph based partitioning for pre-bond 3D IC testing. Additionally, several scan-ordering approaches for 3D IC post-bond testing were accordingly proposed in [17]. VIA3D uses the fewest number of TSVs to alleviate TSV impact on the scan-stitching wire. MAP3D first maps all scan FFs onto one single layer, followed by the 2D scan-chain reordering technique. OPT3D considers TSV impact during cost computation for scan-stitching wire. OPT3D outperforms the other two in terms of total wire cost. However, scan-induced power dissipation is not considered by such work and is also an important issue for 3D ICs. A Genetic Algorithm (GA) method was then proposed in [17] where the runtime issue remains unresolved and solution quality is unstable. Hence, a fast 3D scan-chain design is presented in this work to simultaneously consider wire and power costs.

In this work, TSV-constrained scan-chain ordering is first analyzed and formulated into a Traveling Salesman Problem (TSP). Later, a fast algorithm is developed to minimize the scan-stitching wire and/or scan-induced power dissipation, to simultaneously satisfy the constraint on the number of TSVs in use for 3D-IC designs. Our algorithm consists of two phases: First, we construct an initial simple path through all scan FFs using a modified greedy algorithm, the multiple fragment heuristic, via a dynamic closest pair data structure FastPair. Second, we propose two new techniques, 3D planarization and 3D relaxation, to minimize the wire/power cost and to reduce the TSV number, respectively.

For 3D multi-core processors, many previous researches focused on energy minimization. Many behavioral-level solutions were also proposed [31-41] for 3D multi-core systems, such as thermal-aware task scheduling [32] and power-aware task scheduling [33]. More advanced techniques that can be classified into Voltage Selection (VS) (also called voltage scheduling) [34] and Power Management (PM) [35], mainly target the system-level energy saving where VS is more attractive than PM in general [36].

Particularly, one of VS scheduling, Dynamic Voltage and Frequency Scaling

(DVFS) scheduling algorithm, has prevailed recently. Many heuristics applying Dynamic Voltage and Frequency Scaling (DVFS) techniques were proposed for energy minimization. However, most of the previous works were built upon a fixed task-to-core mapping where many slack spaces can be further improved. Therefore, in this work, we propose two dynamic remapping strategies to enhance an energy-aware task-scheduling algorithm considering transmission cost.

1.2.1 First year

According to the formulation for the TSV constrained scan-chain ordering problem, two approaches are proposed in [17]. One approach is developed on the basis of Genetic Algorithm (GA), and the other is based on Integer Linear Programming (ILP). Although the GA approach may possibly find the near-optimal solution, the quality of one identified solution cannot be guaranteed. Moreover, the ILP approach, which will find the optimal cost, may not be able to produce a feasible solution within a limited time. The experimental result in [17] shows a lower-bound value on the total scan-stitching wire cost, which was obtained quickly through the ILP approach without providing a detailed ordering of scan FFs.

From a practical perspective, a fast algorithm needs to be developed that will overcome the runtime issue. Therefore, we propose a fast two-stage algorithm. In stage 1, we convert the 3D scan-chain ordering problem into a TSP problem. Then, a tour-construction heuristic [20] with the support of a particular closest-pair data structure, FastPair, [21] is used to stitch a simple path as an initial solution. During stage 2, local refinement by 3D planarization and constraint-solving by 3D relaxation minimize the total cost and reduce the number of TSVs in use, respectively. Figure 1.3 shows the overall flow.

We present problem formulations of TSV-constrained scan chain ordering for 3D-IC designs, with three different objectives:

- Wire-cost minimization
- Power-cost minimization
- Wire-and-power cost minimization

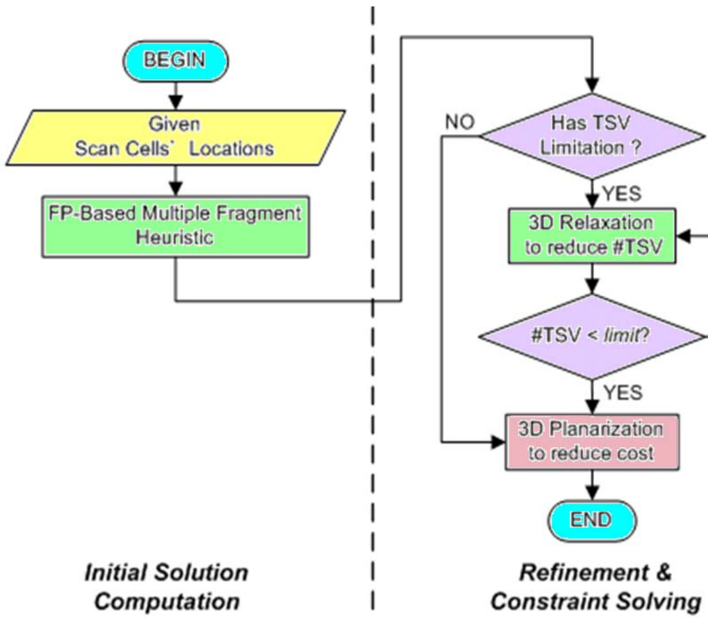

BEGIN
Given Scan Cells' Locations
FP-Based Multiple Fragment Heuristic
NO
Has TSV Limitation ?
YES
3D Relaxation to reduce #TSV
#TSV < limit?
NO
YES
3D Planarization to reduce cost
END
Initial Solution Computation
Refinement & Constraint Solving

Figure 1.3: 3D-MFH flow

As a result, the contributions of this work can be summarized as:

- Formulate scan-chain ordering considering TSV constraints into a modified TSP problem.
- Propose a greedy algorithm for scan-chain ordering of 3D-IC designs to simultaneously minimize wire and power costs.
- Demonstrate that the proposed algorithm can be practically used while supporting multiple scan chains.

1.2.2 Second year

To achieve high performance on embedded systems, 3D multi-core architecture has become a promising alternative. Besides, efficiency in energy consumption is also crucial to enable high-performance computing. Mapping and scheduling of many-core utilization has been known as a NP-complete problem, and thus, many heuristics were proposed for energy-aware schedules using various dynamic voltage and frequency scaling (DVFS) techniques including an energy-efficient time-constrained task-scheduling algorithm considering transmission cost for minimizing the total energy consumption.

In this work, the core problem is to find a schedule with the best energy-efficiency on a 3D-multi-core architecture. Figure 1.4 shows the system overview of the timing-and-resource constrained scheduler. Inputs to a scheduler include a task graph, a timing constraint, a resource constraint and an energy model. All tasks after scheduling must be assigned into one core with a correct execution order. Moreover, energy minimization is the objective of a scheduler where the energy-saving rate is computed to estimate the energy- efficiency of schedulers.

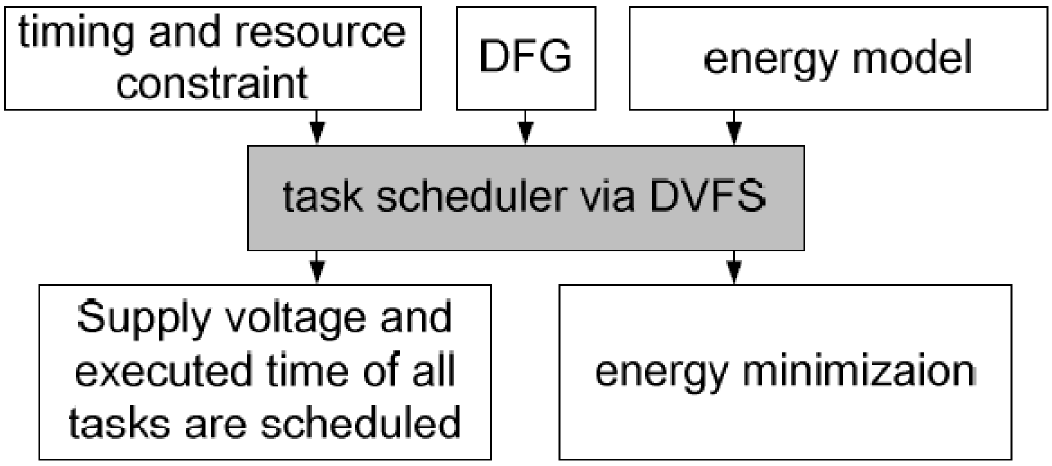

timing and resource constraint
DFG
energy model
task scheduler via DVFS
Supply voltage and executed time of all tasks are scheduled
energy minimizaion

Figure 1.4: system overview of task scheduler

Wu et al. [41] proposed an energy-efficient task scheduling algorithm on top of [40] via DVFS at the system level and formulated a priority gain function considering both gains and losses for selecting tasks to scale down its frequency.

Built on top of the previous task-scheduling algorithms [41], two dynamic task-to-core mapping strategies, Dynamic Remapping (DR) and Iterative Dynamic Remapping (IDR), are proposed to reduce slack slots and to improve Energy-Saving Rate (ESR). Experimental results show that ESR of the algorithm with the IDR strategy is 16 percent higher than the previous work [41] on average. Moreover, compared to an ILP solution, both two proposed strategies can run at least three-order faster and achieve comparable performance on energy saving. Figure 1.5 shows the flow of the DR or IDR strategy. There are two rounds in the DR strategy where task-to-core mapping and voltage scaling are performed in both round.

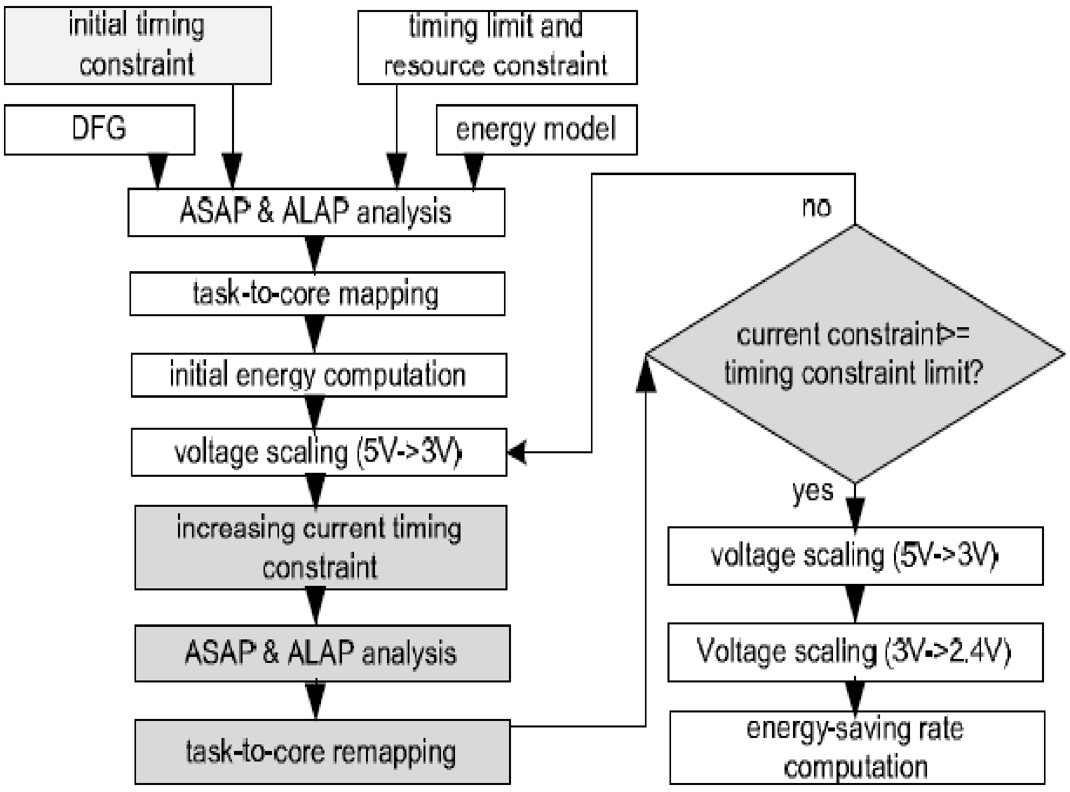

initial timing constraint
timing limit and resource constraint
DFG
energy model
ASAP & ALAP analysis
task-to-core mapping
initial energy computation
voltage scaling (5V->3V)
increasing current timing constraint
ASAP & ALAP analysis
task-to-core remapping
no
current constraint>= timing constraint limit?
yes
voltage scaling (5V->3V)
Voltage scaling (3V->2.4V)
energy-saving rate computation

Figure 1.5: Design flow of scheduling with dynamic remapping methods

Chapter 2
Fast scan-chain ordering for 3D-IC designs under through-silicon-via (TSV) constraints

2.1 Introduction

Interconnect along with scaling technology plays an important role in deciding circuit performance. Structural Three-Dimensional (3D) integration is emerging as a promising solution to reduce the length of long interconnects across circuits [1]. Moreover, 3D integration provides many other advantages over the traditional Two-Dimensional (2D) implementation, such as better packaging efficiency and higher transistor density. These advantages, collectively, not only provide significant performance improvement but also alleviate the problems caused by long interconnects [2], [3], [4], [5]. Among all vertical-integration techniques, Through-Silicon Via (TSV) provides the best timing and power performance for interconnection. However, TSVs typically incur additional area overhead and may become another source of defects [6]. Therefore, considering yield loss and area cost, the number of TSVs in use is typically limited in a 3D Integrated Circuit (IC) design.

On the other hand, scan-chain design is the most prevailing Design-for-Testability (DFT) technique which aims to reduce the difficulty of testing on the Circuit Under Test (CUT). In order to guarantee high fault coverage on complex designs, the CUT is modified during the synthesis stage to enhance its controllability and observability. All Flip-Flops (FFs) are replaced by multiplexed-input scan FFs with multiple operation modes. During the test mode, i.e., when a signal test is activated, the values of one test pattern are shifted to scan FFs of the scan chain in sequel. Later, the pattern is applied to the combinational logic through the primary inputs under the function mode. The response values are finally captured at the primary outputs and shifted out through the scan chain once again under the test mode. Scan testing reduces the sequential problem into a combinational problem; thus, high coverage can be efficiently achieved.

Although scan FFs enhance the testability on the CUT, the stitching wire of a scan chain can be long and may deteriorate signal integrity or even violate the timing constraint. Therefore, scan-chain ordering, referring to the order decision for scan FFs based on physical information, is widely studied. Many layout-based techniques [7], [8], [9] have been shown to reduce the scan-stitching wire effectively.

Test power has always been a concern of scan testing. It depends on the characteristics of test patterns as well as shift operations. Higher logic switching activities in the combinational logic usually stem from ATPG patterns and corresponding LFSR without considering the functionality of the circuit. The scan-shift operation also causes the high toggle rate during testing. Generally, different methods reported to solve the power-related problem in the CUT, such as power-aware test pattern generation [10], test-pattern-filling technique [11], scan-chain partitioning [12], and scan-chain ordering [13], [14], [15], [16]. Among all solutions, scan-chain ordering offers several advantages over other techniques, including no negative effects in the test application time and fault coverage, and can be easily combined to the design flow with other power reduction techniques.

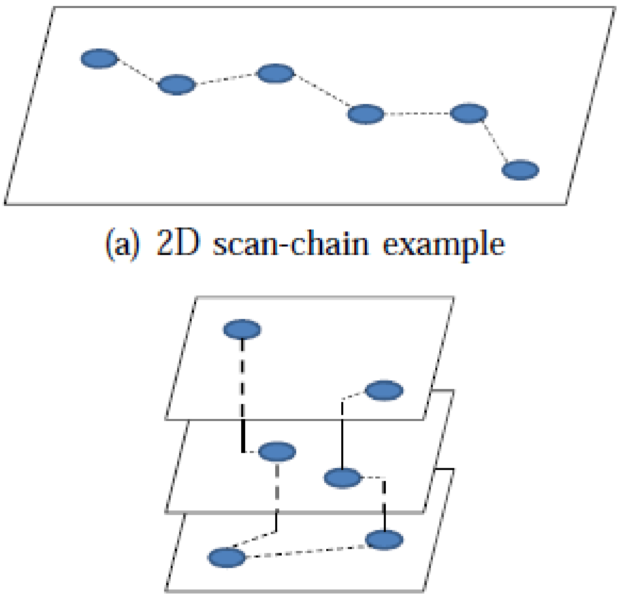


Figure 2.1: Comparison between 2D and 3D scan-chain designs

To further study interconnects on 3D IC designs, Yuan et al. [17] showed that the scan-stitching wire length in a multi-layer circuit is shorter compared with that in the planar circuit, as shown in Figure 1. Experimental results in [17] also suggested that the more TSVs in use in the scan chain, the less scan-stitching wire cost. Such observation combined with the TSV induced yield loss indicates an important tradeoff between the scan-stitching wire and the number of TSVs in use. Therefore, a constraint of TSVs in use must be considered for a 3D-IC design.

Both pre-bond testing and post-bond testing are important for improving the yield of 3D ICs. For enabling pre-bond testability, Lewis et al. [18] proposed a scan-island based design and Kumar et al. [19] proposed a hyper-graph based partitioning for pre-bond 3D IC testing. Additionally, several scan-ordering approaches for 3D IC post-bond testing were accordingly proposed in [17]. VIA3D uses the fewest number of TSVs to alleviate TSV impact on the scan-stitching wire. MAP3D first maps all scan FFs onto one single layer, followed by the 2D scan-chain reordering technique. OPT3D considers TSV impact during cost computation for scan-stitching wire. OPT3D outperforms the other two in terms of total wire cost. However, scan-induced power dissipation is not considered by such work and is also an important issue for 3D ICs. A Genetic Algorithm (GA) method was then proposed in [17] where the runtime issue remains unresolved and solution quality is unstable. Hence, a fast 3D scan-chain design is presented in this work to simultaneously consider wire and power costs.

In this work, TSV-constrained scan-chain ordering is first analyzed and formulated into a Traveling Salesman Problem (TSP). Later, a fast algorithm is developed to minimize the scan-stitching wire and/or scan-induced power dissipation, to simultaneously satisfy the constraint on the number of TSVs in use for 3D-IC designs. Our algorithm consists of two phases: First, we construct an initial simple path through all scan FFs using a modified greedy algorithm, the multiple fragment heuristic, via a dynamic closest pair data structure FastPair. Second, we propose two new techniques, 3D planarization and 3D relaxation, to minimize the wire/power cost and to reduce the TSV number, respectively. Experiments show the practicality of our algorithm by producing comparable scan-stitching wire length (and total power dissipation) to the GA method with a two-order speedup on average.

As a result, the contributions of this work can be summarized as:

- Formulate scan-chain ordering considering TSV constraints into a modified TSP problem.
- Propose a greedy algorithm for scan-chain ordering of 3D-IC designs to simultaneously minimize wire and power costs.
- Demonstrate that the proposed algorithm can be practically used while supporting multiple scan chains.

The rest of this work is organized as follows: In Section 2.2, we present problem formulations of TSV-constrained scan-chain ordering for 3D-IC designs, with three different objectives:

- Wire-cost minimization
- Power-cost minimization
- Wire-and-power cost minimization

In Section 2.3, a multiple fragment heuristic with the support of FastPair is implemented to obtain good initial solution. The process of 3D planarization to minimize scan-stitching wire cost (or scan-induced power dissipation), and the 3D relaxation process to reduce TSV numbers are detailed, respectively. Section 4.1 presents the experimental results, which include a comparison between our algorithm and a GA method under TSV constraints in terms of numerous performance metrics and runtime over a variety of benchmark circuits. Finally, in Section 5 we draw our conclusion and outline future work.

2.2 Problem formulation of scan-chain ordering for TSV-constrained 3D-IC designs

In this section, we formulate the scan-chain ordering problem for 3D-IC designs with three different objectives: (1) to minimize the scan-stitching wire cost to avoid routing congestion and timing violation; (2) to reduce the scan-induced power dissipation on testing to avoid damage and reliability degradation to the CUT; and (3) to simultaneously consider wire and power costs. First we briefly describe the traditional scan-ordering problem for wire minimization and we define a new model for TSV-constrained 3D-IC designs. We then provide a literature review of the power issue for scan reordering and define a new problem for 3D power-optimized scan ordering. Finally, the problem is formulated by simultaneously considering the wire and power costs.

2.2.1 Wire-cost minimization problem

The traditional problem of planar (2D) scan-chain ordering to minimize scan-stitching wire cost can be formulated into:

Input: CUT C with n scan FFs $\{c_0, c_1,\ldots, c_{n-1}\}$ and their locations $\{(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$

Output: Scan-FF ordering is formed as $\langle c_{\pi(0)}, c_{\pi(1)}, \ldots, c_{\pi(n-1)} \rangle$ such that the total cost of scan-stitching wire is minimized.

$$\sum_{i=1}^{n-1} |x_{\pi(i)} - x_{\pi(i-1)}| + |y_{\pi(i)} - y_{\pi(i-1)}| \quad (1)$$

In Equation (1), $x_{\pi(i)}$ and $y_{\pi(i)}$ denote the x and y coordinates of the i_{th} scan FF in the scan-FF ordering, respectively. All scan FFs are placed on the same plane and the cost of scan-stitching wire is defined as the sum of the Manhattan distances between two consecutive FFs, c_i and c_{i+1}, in this formulation. However, since FFs can be located across different layers for 3D-IC designs, the TSV cost for connecting two cross-layer FFs needs to be considered and the layer information of FFs needs to be included.

$$\{(x_0, y_0, L_0), (x_1, y_1, L_1), \ldots, (x_{n-1}, y_{n-1}, L_{n-1})\}$$

The total cost of scan-stitching wire is modified as follows:

$$\sum_{i=1}^{n-1} |x_{\pi(i)} - x_{\pi(i-1)}| + |y_{\pi(i)} - y_{\pi(i-1)}| + C_{TSV} \times |L_{\pi(i)} - L_{\pi(i-1)}| \quad (2)$$

In Equation (2), C_{TSV} denotes the equivalent scan-stitching wire cost for one TSV connecting two consecutive layers. Generally, C_{TSV} can be defined as the height of one TSV. Moreover, considering manufacturability and yield loss, the total number of TSVs in use becomes a constraint to this problem and can be expressed as

$$N_{TSV} = \sum_{i=1}^{n-1} |L_{\pi(i)} - L_{\pi(i-1)}| \quad (3)$$

According to the modified formulation for the TSV constrained scan-chain ordering problem, two approaches are proposed in [17]. One approach is developed on the basis of Genetic Algorithm (GA), and the other is based on Integer Linear Programming (ILP). Although the GA approach may possibly find the near-optimal solution, the quality of one identified solution cannot be guaranteed. Moreover, the ILP approach, which will find the optimal cost, may not be able to produce a feasible solution within a limited time. The experimental result in [17] shows a lower-bound value on the total scan-stitching wire cost, which was obtained quickly through the ILP approach without providing a detailed ordering of scan FFs.

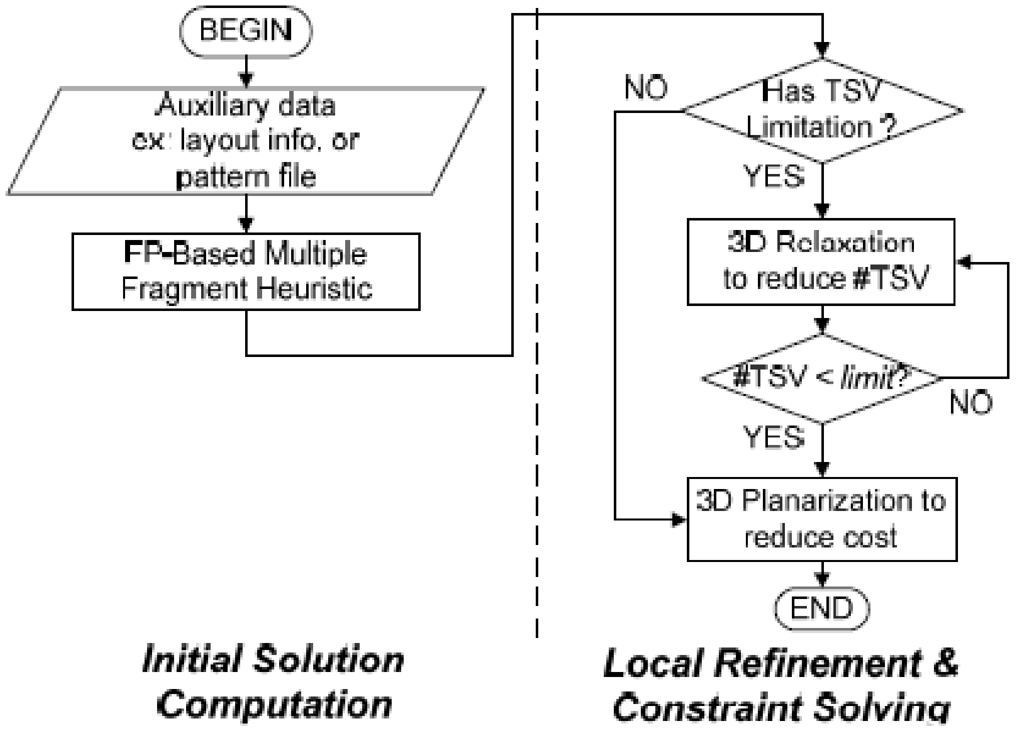

BEGIN
Auxiliary data ex: layout info, or pattern file
FP-Based Multiple Fragment Heuristic
Has TSV Limitation ?
NO
YES
3D Relaxation to reduce #TSV
#TSV < limit?
NO
YES
3D Planarization to reduce cost
END
Initial Solution Computation
Local Refinement & Constraint Solving

Figure 2.2: Flow of proposed scan reordering algorithm

From a practical perspective, a fast algorithm needs to be developed that will overcome the runtime issue. Therefore, we propose a fast two-stage algorithm. In stage 1, we convert the 3D scan-chain ordering problem into a TSP problem. Then, a tour-construction heuristic [20] with the support of a particular closest-pair data structure, FastPair, [21] is used to stitch a simple path as an initial solution. During stage 2, local refinement by 3D planarization and constraint-solving by 3D relaxation minimize the total cost and reduce the number of TSVs in use, respectively. Figure 2.2 shows the overall flow. Additional details are given in Section 2.3.

2.2.2 Power-cost minimization problem

In the second problem, the goal of scan-chain ordering is to find an ordering of scan FFs with minimal power dissipation originating from scan-shift operations. Integrating scan-chain ordering techniques into the current design flow (while maintaining the original fault coverage and test application time) is straight-forward. The only challenge is that the power-optimized scan-chain ordering depends on a fixed set of test patterns generated by Automatic Test Pattern Generation (ATPG). Therefore, in this section we briefly introduce the background of power consumption induced by scan testing and then formulate this problem for TSV-constrained 3D-IC designs.

1) Estimation of Power Dissipation: Previous power-optimized ordering techniques focus on both the ***total power*** and the ***peak power*** consumption. The total power consumption is the sum of power consumed during testing and the peak power consumption is the highest power consumption used among all test patterns. Therefore, the dynamic power consumption can be expressed as:

$$\mathrm{P} = 0.5 \cdot C_{ld} \cdot V_{dd}^2 \cdot \mathrm{F} \cdot \mathrm{S} \qquad (4)$$

where P is the dynamic power consumption, C_{ld} is the load capacitor, V_{dd} is the supply voltage, S is the switching activity, and F is the clock frequency, respectively.

According to Equation (4), the power consumption during scan-shift operations is highly correlated with the switching activities in the CUT. In practice, it is time-consuming to count the exact number of all switching activities in the CUT, but the number of scan-chain transitions and the triggered transitions of logic elements in CUT are proven highly correlated in [11]. In other words, the number of transitions in the scan chain is a good estimation for total switching activities in the CUT.

Total switching activities in the CUT during scan-shift operations depend on the transitions in the scan chain and the corresponding positions. Thus, the number of Weighted Transitions (WT) can be defined as follows,

$$\mathrm{WT} = \sum (size - position)$$

where WT represents the real switching activities in the CUT, size is the total number of scan FFs, and position is indexed from the different beginning locations between the input vector and output response. Hence, every transition in the input vector or the output response has its own weight to reflect the real condition. Defined below are several necessary notations used in the weight transitions throughout the remainder of the paper:

- $\{c_0, c_1, \ldots, c_{n-1}\}$: n scan FFs in the CUT C.
- $O = \langle c_{\pi(0)}, c_{\pi(1)}, \ldots, c_{\pi(n-1)} \rangle$: Scan-chain ordering with n scan FFs.
- $V = \{v_0, v_1, \ldots, v_{n-1}\}$: n-bit input pattern where v_i is scanned in the scan FF c_i during scan testing. Therefore, $\langle v_{\pi(0)}, v_{\pi(1)}, \ldots, v_{\pi(n-1)} \rangle$ represents an

input pattern with respect to a given scan chain ordering.

- R = {$r_0, r_1, \ldots, r_{n-1}$}: n-bit output response where r_i is scanned out from the scan FF c_i during scan testing. Therefore, $\langle r_{\pi(0)}, r_{\pi(1)}, \ldots, r_{\pi(n-1)} \rangle$ represents an output response with respect to a given scan chain ordering.

Given the notations, the weighted transitions of an input, vector V and an output response R can be defined, respectively:

$$\mathrm{VWT(V)} = \sum_{i=1}^{n-1} i \cdot (v_{\pi(i)} \oplus v_{\pi(i-1)}) \tag{5}$$

$$\mathrm{RWT(R)} = \sum_{i=1}^{n-1} (n-i) \cdot (r_{\pi(i)} \oplus r_{\pi(i-1)}) \tag{6}$$

where VWT(V) and RWT(R) are denoted as the weighted transitions for the input vector V and the output response R; the exclusive-or $\oplus$ operator checks the difference between two adjacent bits. i and (n−i) represent different weighting rules for scan-in and scan-out operations respectively. Generally, Equations (5) and (6) can be easily extended into the following equations form test patterns:

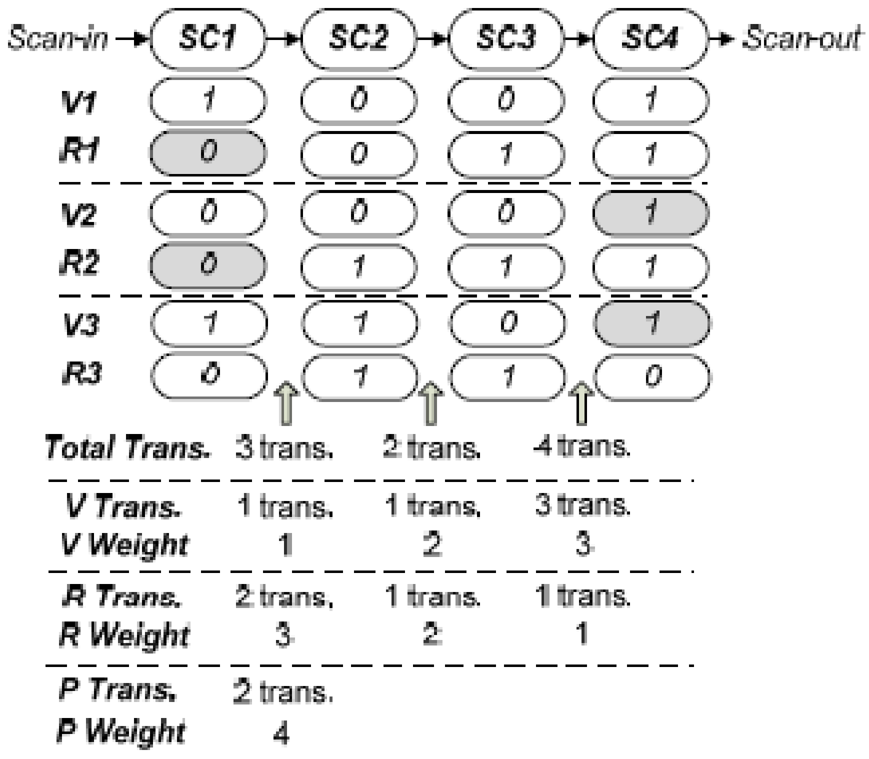
$$\mathrm{VWT}(V^1, V^2, \ldots, V^m) = \sum_{j=1}^{m} \sum_{i=1}^{n-1} i \cdot (v^j_{\pi(i)} \oplus v^j_{\pi(i-1)}) \tag{7}$$

$$\mathrm{RWT}(R^1, R^2, \ldots, R^m) = \sum_{j=1}^{m} \sum_{i=1}^{n-1} (n-i) \cdot (r^j_{\pi(i)} \oplus r^j_{\pi(i-1)}) \tag{8}$$

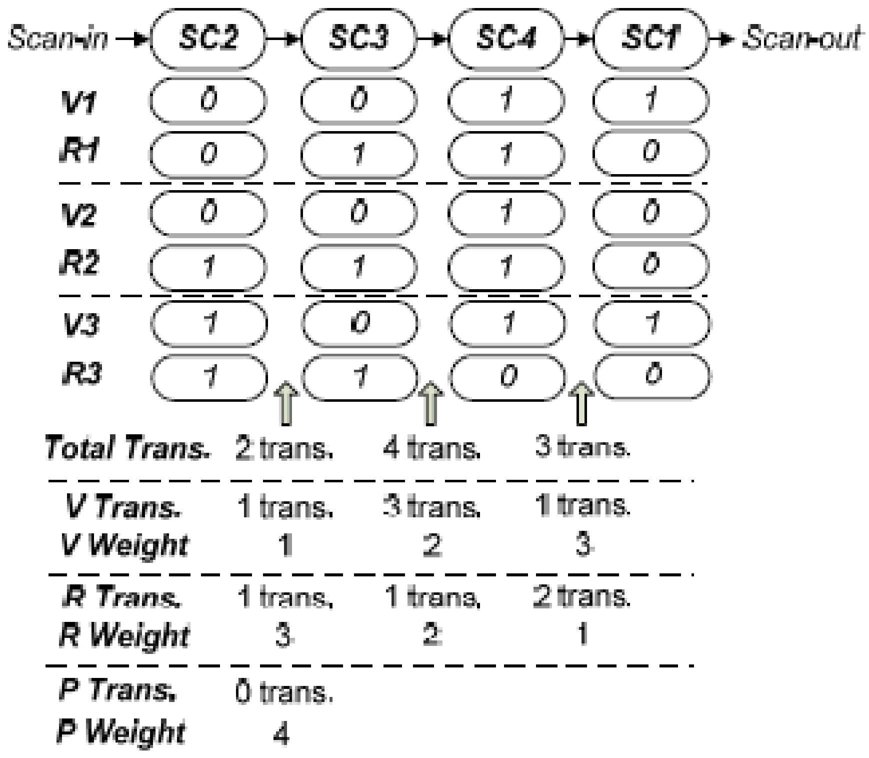
V^j and R^j are the jth input vector and the jth output response in the set of m test patterns, respectively, and the $v^j_{\pi(i)}(r^j_{\pi(i-1)})$ is the bit being scanned in the ith scan FF of the chain ordering, located at the jth input vector (jth output response).

In addition to scan-in and scan-out transitions, peak transitions are also taken into account to determine the total weighted transitions. A peak transition occurs when there is a difference between the last-out bit of the jth output response and the first-in bit of the (j + 1)th input vector. Since a peak transition causes all scan FFs to toggle, the weight of the peak transitions is the length of the scan chain. The weighted peak transition is denoted by PWT defined as:

$$\mathrm{PWT} = \sum_{j=1}^{m-1} n \cdot (r^j_{\pi(n-1)} \oplus v^{j+1}_{\pi(0)}) \tag{9}$$



(a) initial ordering



(b) power-optimized ordering

Figure 2.3: Calculations for weighted transitions

Consequently, the total weighted transition TWT can be viewed as TWT = VWT + RWT + PWT.

Figure 2.3 shows two examples of calculated total weighted transitions. The CUT with four scan FFs uses two scan-chain ordering, and three test patterns are applied during scan testing. Hence, the total transitions, the transitions for input vectors, the transitions for output responses, the peak transitions and the corresponding weights in different positions are shown in Figure 2.3. In Figure 2.3(a), the scan chain has an initial ordering (1, 2, 3, 4). Thus, VWT($\{V^1, V^2, V^3\}$) = 1・1+1・2+3・3 = 12, RWT($\{R^1, R^2, R^3\}$) = 2 ・ 3 + 1 ・ 2 + 1 ・ 1 = 9, and PWT = 2 ・ 4 = 8. The total weighted transitions TWT is 12 + 9 + 8 = 29. However, Figure 3(b) shows a power-optimized ordering (2, 3, 4, 1) by scanning in the same test patterns. Thus, VWT($\{V^1, V^2, V^3\}$) = 1 ・ 1 + 3 ・ 2 + 1 ・ 3 = 10, RWT($\{R^1, R^2, R^3\}$) = 1 ・ 3 + 1 ・ 2 + 2 ・ 1 = 7, and PWT = 0 ・ 4 = 0. The total weighted transitions TWT is 11 + 7 + 0 = 18. Therefore, the total power reduction rate is 38 percent and the number of peak transitions is reduced from 2 to 0.

2) Formulation for TSV-constrained 3D-IC Designs: The problem of scan-chain ordering to minimize the scan-shift power dissipation can be formulated into:

Input: CUT C with n scan cells $\{c_0, c_1, \ldots, c_{n-1}\}$, their layer information $\{L_0, L_1, \ldots, L_{n-1}\}$, and a fixed set of m test patterns $\{V^1, R^1, V^2, R^2, \ldots, V^m, R^m\}$.

Output: Scan-cell ordering is formed as follows $\langle c_{\pi(0)}, c_{\pi(1)}, \ldots, c_{\pi(n-1)} \rangle$ such that the total weighted transitions TWT($\{V^1, R^1, V^2, R^2, \ldots, V^m, R^m\}$) is minimized under a TSV constraint

Compared with the scan-wire minimization problem, we are only concerned with the layer information of the scan FFs since the problem is not related to their geometric locations or the objective function. Therefore, we only need to consider total TSV cost by using Equation (3).

Regarding the formulation for the power-minimization concerning TSV-based 3D-IC designs, Giri et al. from [22] also used a GA approach to solve this problem. However, it is time-consuming and unstable, which can impair quality solutions. Therefore, we propose a similar flow, as illustrated in Figure 2.2, to solve this power-optimization problem. At the beginning, we establish a look-up table storing the pair-wise cost to avoid the high complexity of calculations. Since the objective involves the transition positions in the scan chain, there are several modifications in the proposed algorithm. Further details are provided in Section 2.3.

2.2.3 Wire-and-power cost minimization problem

Two previous 3D-IC scan-chain ordering problem (with different objectives) are reviewed. One is to minimize the total cost of scan-stitching wire cost; the other is to minimize the scan-induced power cost during testing. In a more advanced case, we would like to simultaneously consider wire and power costs. Cost function in this new problem is combined from the wire and power cost function.

The problem of scan-chain ordering to minimize the power and wire cost simultaneously can be formulated into:

Input: CUT C with n scan cells $\{c_0, c_1, \ldots, c_{n-1}\}$, their layer information $\{L_0, L_1, \ldots, L_{n-1}\}$, and a fixed set of m test patterns $\{V^1, R^1, V^2, R^2, \ldots, V^m, R^m\}$.

Output: Scan-cell ordering is formed as follows $\langle c_{\pi(0)}, c_{\pi(1)}, \ldots, c_{\pi(n-1)} \rangle$ such that the combined cost $((1 - \alpha) \times$ wire cost $+ \alpha \times$ power cost) is minimized under a TSV constraint.

The same flow illustrated in Figure 2.2 is used again to solve the combined-cost optimization problem. Experimental results in Section IV will also show that the proposed algorithm can efficiently minimize the combined cost when ordering scan FFs.

2.3 A fast scan-chain ordering

In this section, the proposed algorithm is elaborated with respect to different objectives, including wire-cost minimization in Section 2.3-A, power-cost minimization in Section 2.3-B, and wire-and-power (combined) cost minimization in Section 2.3-C, respectively.

2.3.1 Minimizing wire cost

According to Figure 2, in stage 1, a state-of-the-art tour-construction heuristic used in TSP problems, multiple fragment heuristic [20], is incorporated. Moreover, a dynamic closest-pair data structure, FastPair [21], is implemented to facilitate the considerable computation of pair-wise cost in the tour-construction heuristic. An initial solution is obtained in stage 1 and sent to stage 2 to perform 3D planarization to optimize the total wire cost and 3D relaxation to reduce the total number of TSVs in use.

1) Initial Ordering Computation: First, a solid initial ordering of scan FFs needs to be constructed in stage 1. We solve this problem by using the multiple fragment heuristic. This heuristic finds the shortest edge between the endpoints of two different paths until all points are connected. Each point is initialized as a one-point path. Then, the legal min-cost edge will be found by the closest-pair data structure. At the same time, the useless point will be deleted from the point set. Finally, this procedure iterates until a simple path is derived and all points are stitched. The multiple fragment heuristic is shown as follows in Algorithm 2.1.

Algorithm 2.1 Multiple Fragment Heuristic

```
1  for each FF $c_i \in C$
2      do $endpoint(c_i) \leftarrow c_i$
3  while $|C| > 2$
4      do $(c_i, c_j) \leftarrow closet\_pair(C)$
5         $c_x \leftarrow endpoint(c_i)$
6         $c_y \leftarrow endpoint(c_j)$
7         $endpoint(c_x) \leftarrow c_y$
8         $endpoint(c_y) \leftarrow c_x$
9         if $c_x \neq c_i$
10           then delete $c_i$ from $C$
11        if $c_y \neq c_j$
12           then delete $c_j$ from $C$
```

The **for** loop of line 1-2 sets the endpoint of each FF to itself. The endpoint of each cell is updated and checks to see if illegal conditions occur. The **while** loop of lines 3-12 iteratively links the scan FFs to derive a final simple path until the number of the point set C is less than or equal to two. Figure 2.4 (a) is an example with no connection in the original graph. The first shortest edge between node 2 and node 3, denoted by (2,3) (i.e., the minimum cost), is linked and shown as the dotted line. Under two given sub-paths, 1-2-3 and 4-5-6 shown as the solid lines, Figure 2.4(b) shows the selection of the next shortest edges from the remaining five points. The gray nodes are not considered since any link to these nodes violates the simple-path property. Therefore, in this example, edge (3, 4) is chosen as a new link to connect sub-path 1-2-3 and 4-5-6. This process iterates until only two nodes (node 1 and node 7 in this example) are left in the CUT. As a result, a simple path connecting all points by (n － 1) edges is derived and forms an initial ordering of the scan chain, shown as

Figure 2.4(c).

In Algorithm 2.1, the closest-pair computation in line 4 realized by different implementations can result in different performances. To the best of our knowledge, FastPair is one of the best data structures and first proposed for handling dynamic closest-pair problems with pair-wise cost functions [21]. It behaves similar to the neighbor heuristic where each point stores its own nearest neighbor, but differs from the creation of initial neighbor values. Before exploring FastPair, we first outline the concept of the neighbor heuristic where each point p stores its nearest point from the point set S based on the following equation:

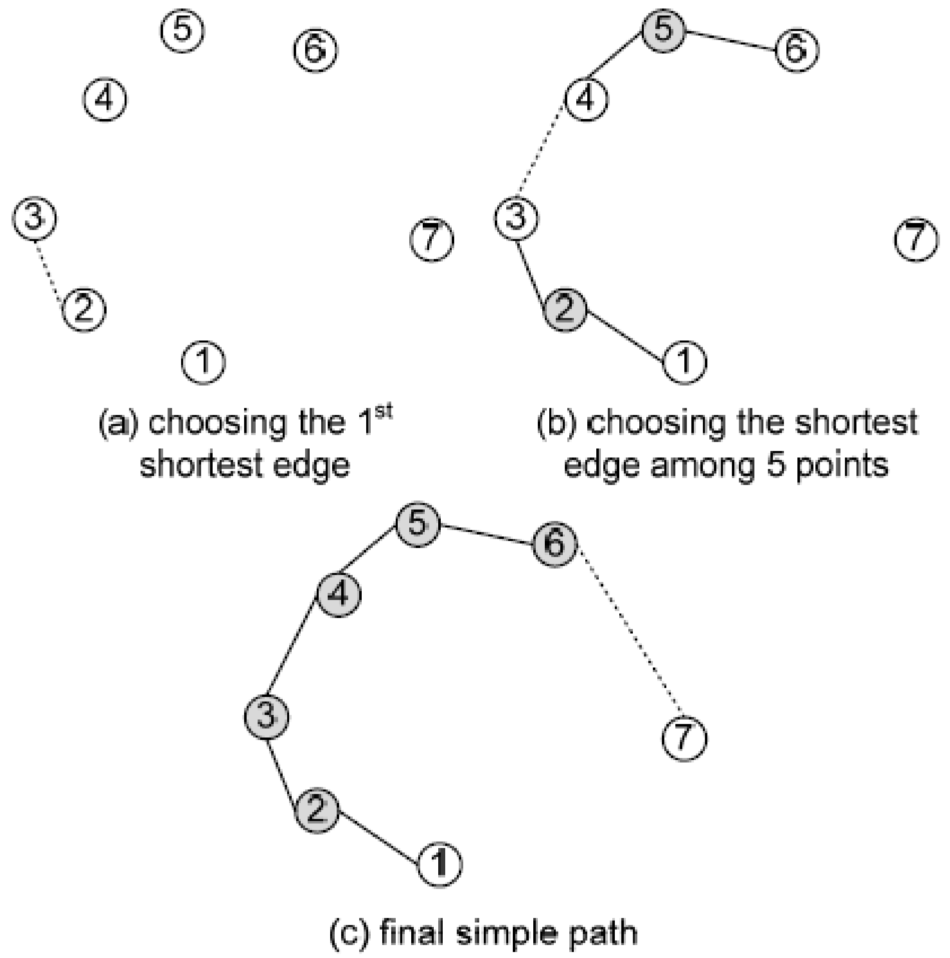
$$\mathrm{d(p)} = min_{q \in S-\{p\}} D(p, q) \tag{10}$$

where D(p, q) is a user-defined function and computes the distance between scan FFs. That is,

$$\mathrm{D(p,\ q)} = |\mathrm{x_p} - \mathrm{x_q}| + |\mathrm{y_p} - \mathrm{y_q}| + \mathrm{c_{TSV}} \times |\mathrm{L_p} - \mathrm{L_q}| \tag{11}$$

Three operations of **insertion**, **deletion**, and **query** are employed by the neighbor heuristic to maintain nearest neighbors. A query scans over distances and selects the smallest one. This process is illustrated in Figure 2.5. Figure 2.5 (a) shows the initialization of one neighbor heuristic. The nearest neighbors of nodes 1, 6, and 7 are nodes 2, 5, 6, respectively; the node pairs (2, 3) and (4, 5) are the closest nodes to each other. After deleting node 5, node 4 and node 6 need to update their closest nodes to be node 3 and node 4, respectively. The corresponding result is illustrated as Figure 2.5 (b).

FastPair is developed on the basis of the neighbor heuristic with some improvements. Instead of computing all possible distances, FastPair is initialized as a single directed path. This structure has the advantage of requiring only one update after deleting one node, which differs from the neighbor heuristic. Figure 2.6 shows an example. In Figure 2.6 (a), a single directed path is formed as 1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 4 $\rightarrow$ 5 $\rightarrow$ 6 $\rightarrow$ 7. In the beginning, node 1 checks all other points and finds the min-cost point. Then, node 2 checks nodes 3, 4, 5, 6, and 7 without node 1. Finally, node 6 only checks one node, node 7, and connects to it. Therefore, such initialization can be explained by a graph that depicts how each node finds its closest node by only checking the nodes that have not been connected. The FastPair heuristic only updates



(a) choosing the 1st shortest edge
(b) choosing the shortest edge among 5 points
(c) final simple path

Figure 2.4: Multiple Fragment Heuristic example

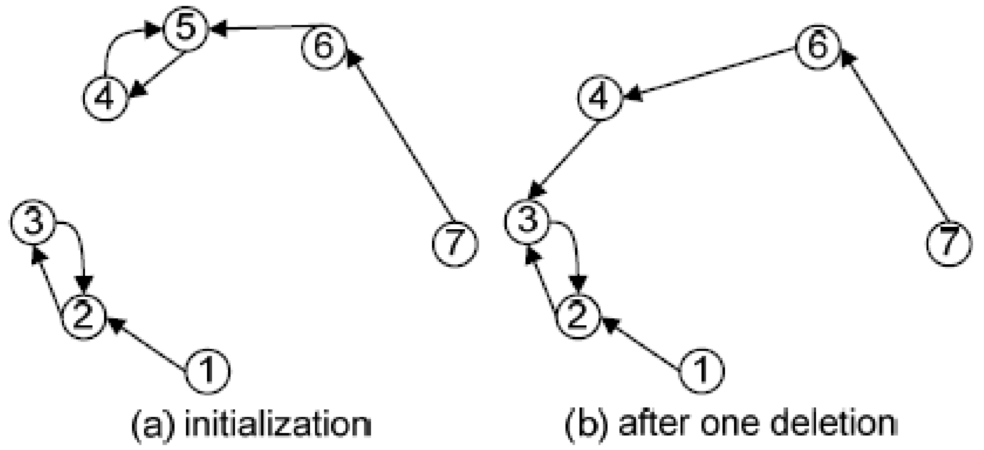


Figure 2.5: Illustration of the neighbor heuristic

the closest node for node 4 after deleting node 5 in Figure 2.6 (b). Overall, the FastPair heuristic runs in the time complexity of O(n2) and outperforms the neighbor heuristic empirically according to [21]. A comparison of run time for deleting an object and querying the closest pair among several different closest pair data structures is thoroughly surveyed; FastPair is known so far to be the best one for many applications. Therefore, considering time efficiency, FastPair and its operations are incorporated when developing our multiple-fragment-heuristic-based algorithm.

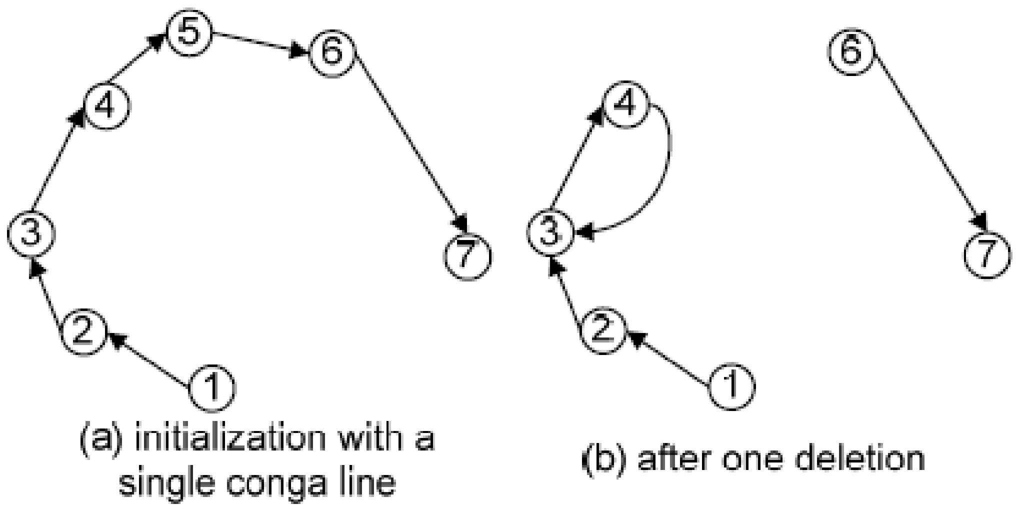


Figure 2.6: Illustration of the application of the FastPair method

2) Local Refinement and Constraint Solving: After obtaining the initial solution, the second stage of our algorithm applies two strategies to optimize total wire cost and/or to relax TSVs in use. Figure 2.7 (a) shows an initial path with the un-optimized wire cost. In the study of the optimization theory, 2D **planarization** is one common technique to reduce the total cost in the TSP problem. The key idea behind this is to **planarize** a graph and remove all cross edges on the plane. A modified tour with cross-edge removal results in a shorter cost than that from the initial tour. Figure 2.7 (b) shows such an example. Cross edges, (2, 6) and (3,7) are replaced by edges (2,3) and (6,7).

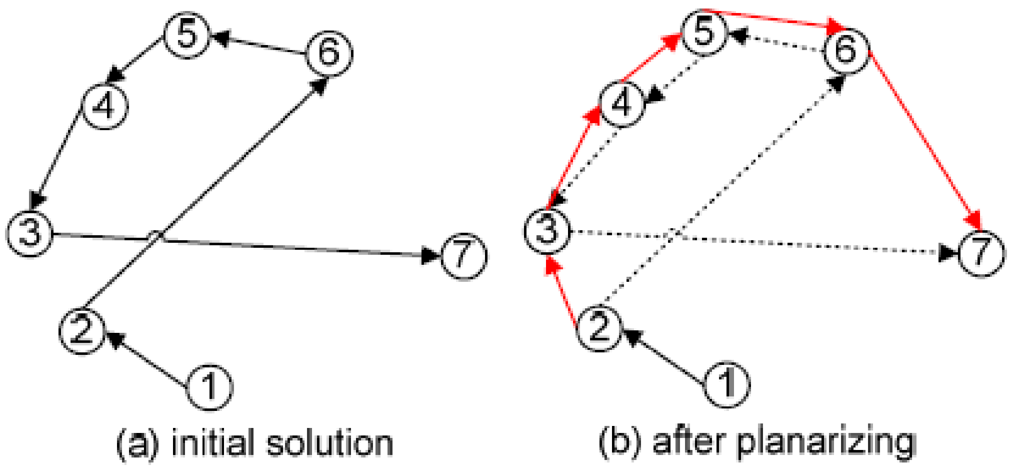


Figure 2.7: Example of six-point Planarization

From a different point of view, such an operation can be viewed as the reverse of a fragment of one path, i.e., the sequence of node traversal. Reversing the fragment from 2 → 6 → 5 → 4 → 3 → 7 into 2 → 3 → 4 → 5 → 6 → 7 can effectively reduce the total wire cost. To generalize this idea, we reverse any possible fragment with the edge length from 1 to (n − 1) and test if such reversion can reduce cost. Such a local-refinement technique is termed 3D planarization and runs in the time complexity of O (k1n2), where k1 denotes the constant number of iterations. After constructing the initial solution, a small k1 ≪ n is usually suff
good optimization in our experiment.

The key reason for using the above technique is to avoid checking the cross edges in the 3D space. Hence, Figure 2.8 shows two examples of using 3D planarization to reduce the total wire cost. In Figure 8, L1, L2, L3 and L4 represent the first, second, third and fourth layers, respectively, and the connection of two scan FFs residing in two consecutive layers requires one TSV. Figure2. 8 (a) shows an example with refinements of both the wire cost and the total number of TSVs in use. The left part of Figure 2.8 (a) represents an initial scan-chain ordering: 1 → 2 → 3 → 4 → 5. After reversing the fragment 2 → 3 → 4 to 4 → 3 → 2, shown as the right part in Figure 2.8 (a), the total wire cost is reduced to a saving of two TSVs: one from the replacement of 1 → 2 to 1 → 4 and the other from the replacement of 4 → 5 to 2 → 5. Similarly, Figure 2.8 (b) shows another example with the refinement over the wire cost. After reversing the fragment 2 → 3 to 3 → 2, no TSV can be saved, but the total wire cost can be reduced.

A similar constraint-solving technique, 3D relaxation, is proposed to reduce the number of TSVs in use and to satisfy the TSV constraint. 3D relaxation reverses all fragments of 1 to (n-1) edges again to find the best reduction of TSVs in use until the target number is achieved. Later, 3D planarization is also performed to further reduce the total wire cost but it uses no extra TSVs. Figure 2.9 shows an example that illustrates the 3D relaxation process. The left part of Figure 9 represents an initial scan-chain ordering: 1 → 2 → 3 → 4 → 5. After reversing the fragment 2 → 3 → 4 to 4 → 3 → 2, the total number of TSVs in use can be effectively reduced and shown as the right part in Figure 2.9. Two new reversed fragments, 1 → 2 to 1 → 4 and 4 → 5 to 2 → 5, save six TSVs in use.

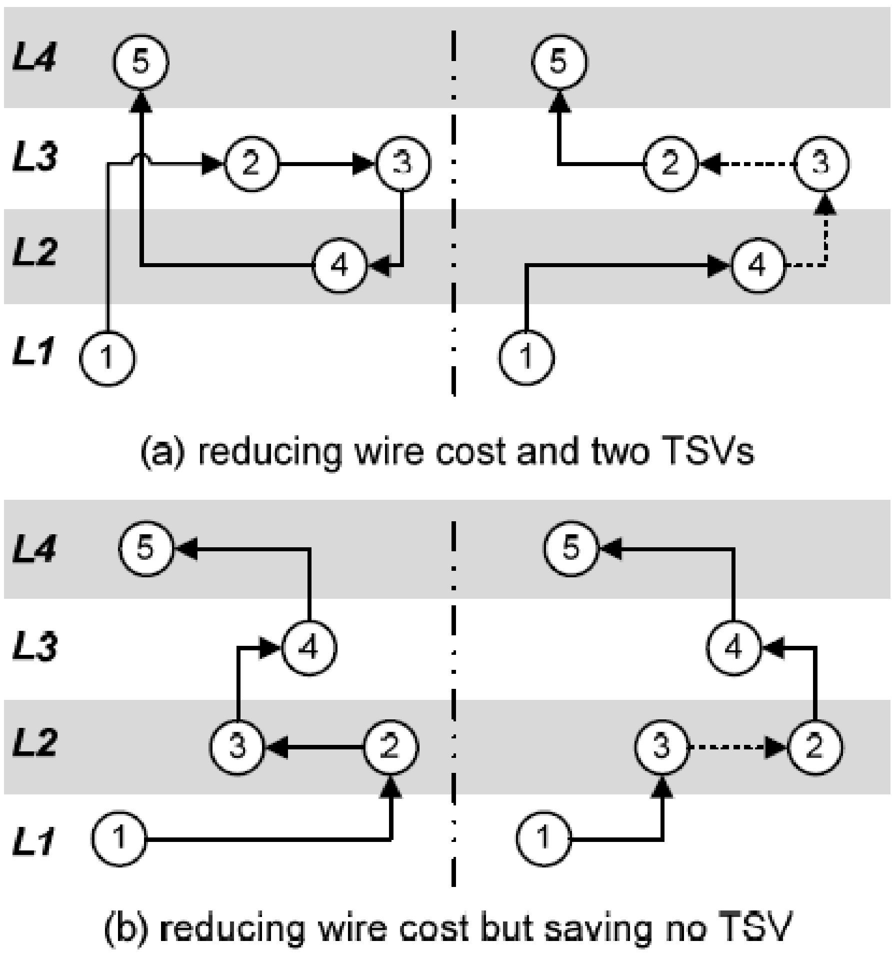

L4
L3
L2
L1
(a) reducing wire cost and two TSVs
(b) reducing wire cost but saving no TSV

Figure 2.8: Example of 3D Planarization

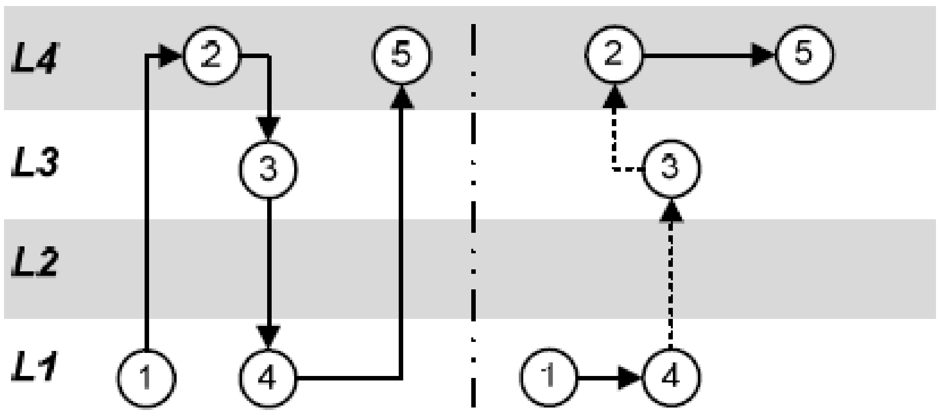


Figure 2.9: Example of 3D Relaxation

The time complexity for the constraint solving technique is also O (k_2n^2) and k_2 depends on the number of relaxations on the TSVs in use, i.e., the difference between the initial and the target number of TSVs in use. Therefore, the total complexity in the second phase is $T(n) = O(k_1n^2)+O(k_2n^2) = O(n^2)$. To sum up, the proposed two-phase scan-chain ordering is more efficient than the previous work [17] in terms of time and it can consider the TSV constraints simultaneously.

2.3.2 Minimizing power cost

We use a similar flow to solve the power-cost minimization problem, and address the differences in wire-cost minimization in this section. According to the problem formulation, the pattern information is an input to the algorithm and the objective is to minimize the total weighted transitions. Again, the computation for the total weighted transitions requires the knowledge of an initial scan-chain ordering and the position information. However, the scan-induced transitions between scan FFs are available in the beginning.

For computing an initial solution, we change the user-defined function D(p, q) in Equation (10) to count the scan-induced transitions between scan FFs considering m test patterns. That is,

$$D(p,q) = \sum_{j=1}^{m}[(v_p^j \oplus v_q^j) + (r_p^j \oplus r_q^j)] \tag{12}$$

where the notations have the same definition as those in Section II. Since each computation of Equation (12) costs O(m) due to m test patterns, a look-up table that stores the pair-wise transitions is established to avoid repeated calculations in the proposed algorithm. After constructing the scan-chain ordering, the sum of the total transitions between scan FFs is minimized by the multiple fragment heuristic. Furthermore, we improve the total weighted transitions by rotating it n times and choose the best solution. Figure 2.3 shows an example where the total scan-induced transitions between scan FFs are accumulated in the first row (Total Trans). Although the sum of the total transitions are the same, the power-optimized ordering shown in Figure 3(b) has better total weighted transitions by rotating the initial ordering SC1 $\rightarrow$ SC2 $\rightarrow$ SC3 $\rightarrow$ SC4 three times into SC2 $\rightarrow$SC3 $\rightarrow$SC4 $\rightarrow$SC1.

Although the construction of the look-up table takes more time than the cost computation in the scan-stitching wire minimization problem, the time complexity is still O (n^2) and it outperforms the technique proposed in [22].

2.3.3 Minimizing wire-and-power cost simultaneously

In the problem, the wire and power costs are optimized simultaneously to determine the scan-chain ordering. The flow to solve the combined-cost optimization problem is similar to the previous problems. According to the problem formulation, the inputs to the algorithm are test patterns and layout information, and the objective is to optimize the combined cost including the wire and power costs.

For computing the initial solution, we combine the user-defined function D(p, q) in Equation (11) and Equation (12) to count the combined cost between scan FFs considering m test patterns. That is:

$$D(p,q) = (1-\alpha) \times D_{wire}(p,q) + \alpha \times D_{power}(p,q) \qquad (13)$$

where D_{wire} (p, q) and D_{power} (p, q) are shown in Equation (12) and Equation (11). User-defined coefficient α ranges from 0 to 1. When α = 0, this problem becomes a pure wire cost minimization problem. While α = 1, only power cost is considered, as in the power-cost minimization problem.

Chapter 3
Enhancing Energy-Efficient Task Scheduling on 3D Multi-Core Processors by Dynamic Remapping

3.1 Introduction

According to the prediction of International Technology Roadmap for Semiconductors (ITRS), the era of tera-scale embedded systems is approaching [24], in which having numerous processing elements on a single chip has been the mainstream and strongly advocated by both the academy and industry [25]. To fulfill high-performance demands on embedded systems, MPSoC (Multiprocessor System-on-a-Chip) design methodology arises as a new paradigm where 3D integration is the state-of-the-art enabling technique. However, since a 3D multi-core processor often consumes excessive energy, leading to a problem of high power density [26] [27], energy efficiency becomes its paramount concern.

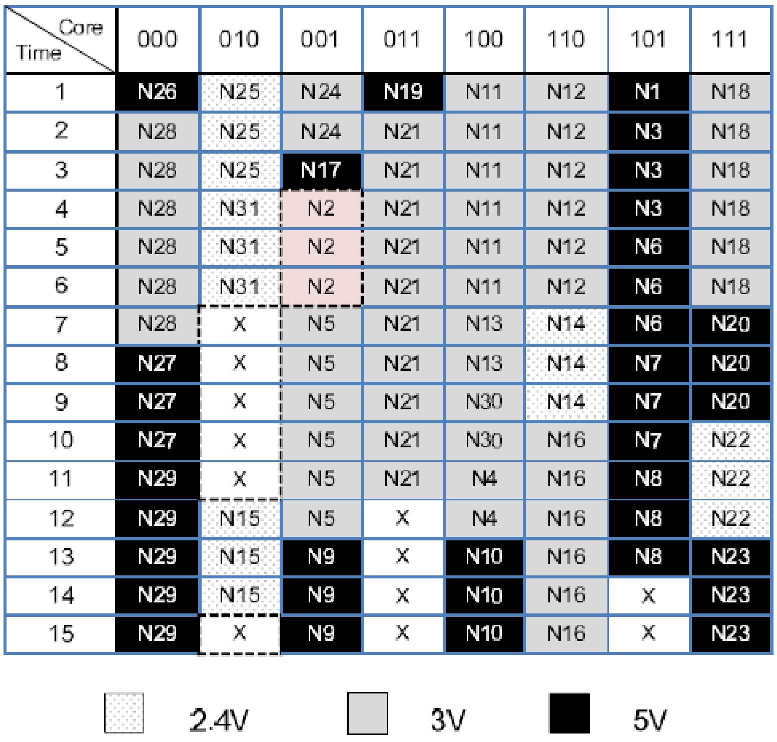
Many previous researches focused on energy minimization at the physical level including micro-channel liquid cooling [28], floorplanning [29] and thermal TSVs [30]. Moreover, behavioral-level solutions were also proposed [31-41] for 3D multi-core systems where high-level techniques are typically more effective than the low-level ones on energy minimization [31], such as thermal-aware task scheduling [32] and power-aware task scheduling [33]. More advanced techniques that can be classified into Voltage Selection (VS) (also called voltage scheduling) [34] and Power Management (PM) [35], mainly target the system-level energy saving where VS is more attractive than PM in general [36].

Particularly, one of VS scheduling, Dynamic Voltage and Frequency Scaling (DVFS) scheduling algorithm, has prevailed recently. The first DVFS scheduling technique proposed in [37] assigns different operational voltages to each task and lowers the clock speed to bring about large power reduction. Other DVFS scheduling algorithms, typically implemented into Integer Linear Programming (ILP), suffer from the scalability problem [38] [39]. The approach in [40] defines a priority function to determine the order of candidate tasks for changing supply voltage. The priority function only considers power gain, mobility and computation density for each task independently while neglecting the overall gains and losses from scaling down the frequency of one task candidate. To alleviate such problem, Wu et al. from [41] proposed an energy-efficient task scheduling algorithm via DVFS at the system level and formulated an priority gain function considering both gains and losses for selecting tasks to be scaled. Figure 3.1 (a) shows the result in [41] for scheduling 31 tasks on 3D processors with eight cores considering transmission cost under a timing constraint 15. In summary, all the previous works used fixed task-to-core mapping strategies where many slack spaces can be further utilized.

To take Figure 3.1 (a) for example, an exploration of the slack slots is conducted after applying DVFS. Due to the fixed task-to-core mapping, many time slots (denoted a .x. in Figure 3.1 (a)) can be further utilized. For example, if we move task N2 from core 001 to core 010 using a slower frequency as shown in Figure 3.1 (b), the remaining spaces can be better utilized and thus the energy-saving rate is improved.

Built on top of the previous task-scheduling algorithms [40] [41], two dynamic task-to-core mapping strategies, **Dynamic Remapping (DR)** and **Iterative Dynamic Remapping (IDR)**, are proposed to reduce slack slots and to improve Energy-Saving Rate (ESR). Experimental results show that ESR of the algorithm with the IDR strategy is 16 percent higher than the previous work [41] on average. Moreover, compared to an ILP solution, both two proposed strategies can run at least three-order faster and achieve comparable performance on energy saving.

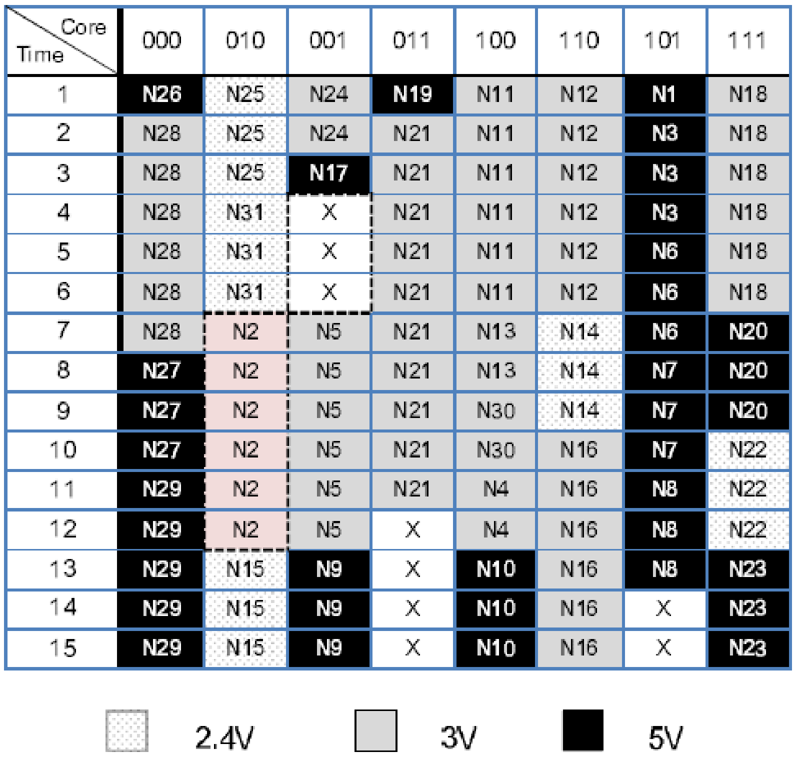
The rest of this work is organized as follows. Section 3.2 formulates the problem of this work. In Section 3.3, the framework of task-to-core mapping and scheduling with DVFS based on [40] and [41] is presented. Two dynamic task-to-core remapping strategies are elaborated in Section 3.4. Section 4.2 provides the experimental result to show the energy efficiency of modified algorithms compared with a previous work [41] and an ILP solution. Finally, Section 5 concludes this paper.



Core / Time	000	010	001	011	100	110	101	111
1	N26	N25	N24	N19	N11	N12	N1	N18
2	N28	N25	N24	N21	N11	N12	N3	N18
3	N28	N25	N17	N21	N11	N12	N3	N18
4	N28	N31	N2	N21	N11	N12	N3	N18
5	N28	N31	N2	N21	N11	N12	N6	N18
6	N28	N31	N2	N21	N11	N12	N6	N18
7	N28	X	N5	N21	N13	N14	N6	N20
8	N27	X	N5	N21	N13	N14	N7	N20
9	N27	X	N5	N21	N30	N14	N7	N20
10	N27	X	N5	N21	N30	N16	N7	N22
11	N29	X	N5	N21	N4	N16	N8	N22
12	N29	N15	N5	X	N4	N16	N8	N22
13	N29	N15	N9	X	N10	N16	N8	N23
14	N29	N15	N9	X	N10	N16	X	N23
15	N29	X	N9	X	N10	N16	X	N23

▨ 2.4V ⬜ 3V ⬛ 5V

(a) fixed task-to-core mapping (ESR=31.49%)



Core / Time	000	010	001	011	100	110	101	111
1	N26	N25	N24	N19	N11	N12	N1	N18
2	N28	N25	N24	N21	N11	N12	N3	N18
3	N28	N25	N17	N21	N11	N12	N3	N18
4	N28	N31	X	N21	N11	N12	N3	N18
5	N28	N31	X	N21	N11	N12	N6	N18
6	N28	N31	X	N21	N11	N12	N6	N18
7	N28	N2	N5	N21	N13	N14	N6	N20
8	N27	N2	N5	N21	N13	N14	N7	N20
9	N27	N2	N5	N21	N30	N14	N7	N20
10	N27	N2	N5	N21	N30	N16	N7	N22
11	N29	N2	N5	N21	N4	N16	N8	N22
12	N29	N2	N5	X	N4	N16	N8	N22
13	N29	N15	N9	X	N10	N16	N8	N23
14	N29	N15	N9	X	N10	N16	X	N23
15	N29	N15	N9	X	N10	N16	X	N23

▨ 2.4V ⬜ 3V ⬛ 5V

(b) dynamic remapping (ESR=34.37%)

Figure 3.1: Examples for different task-to-core mapping strategies

3.2 Problem formulation

In this work, the core problem is how to find a schedule which can achieve the best energy efficiency on 3D multi-core processors. Figure 3.2 (a) is such a sample schedule which assigns 31 tasks to eight cores on a 3D processor under a timing constraint 20. Input information required by a schedule includes a task graph, a timing constraint, a resource constraint and an energy model. All tasks after scheduling must be assigned to one core in a correct execution order. Moreover, energy minimization is the objective for schedule where energy-saving rate is defined to approximate the energy efficiency of the computed schedule.

3.2.1 Data flow graph

One of the input data for scheduling is an unscheduled task graph. A task graph is also called a Data Flow Graph (DFG) that usually describes the behavior of design. Figure 3.2 (b) shows an example of a task graph with 31 nodes. Precedent constraint refers to the situation that a node v_i connected by a directed edge to a node v_j under the constraint that v_j can start execution if and only if v_i finishes completely.

3.2.2 Timing versus Resource Constraint

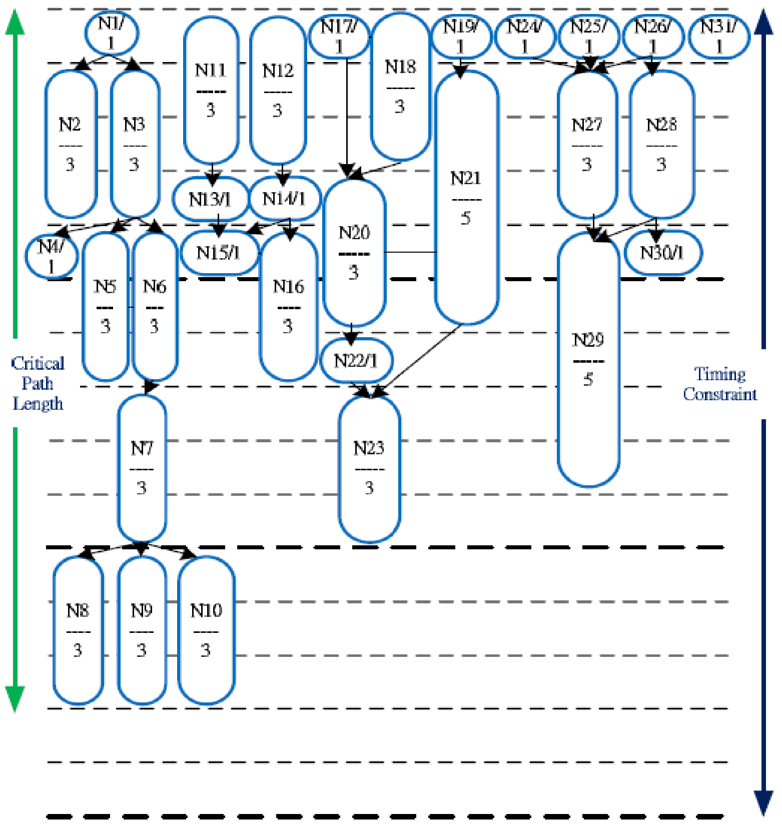
A Critical Path of a DFG is defined as the longest path that the summation of execution time of the nodes in the path is the maximum among all paths. In our work, the timing constraint can be specified by the user but is required to be larger than the length of the critical path.

3D multi-core processors are illustrated as Figure 3.3 where both the number of cores per layer and the number of layers are parameters in our work. The transmission cost between any two cores is also considered and can be specified by users. We denote the transmission cost on the same core as α, to the neighboring core on the same layer as β, and to a neighboring core on the neighboring layer as γ. Given Figure 3.2 and Figure 3.3, task N17 is assigned to core 010, and task N20 (a successor of task N17) is assigned to core 100. The transmission cost between these two tasks is $1 \times \beta + 1 \times \gamma$.

Step \ Core	000	010	001	011	100	110	101	111
1	N12	N17	N26	N25	N24	N19	N18	N1
2	N12	N17	N26	N25	N11	N19	N18	N1
3	N12	N17	N26	N25	N11	N21	N18	N3
4	N12	x	N28	N31	N11	N21	N18	N3
5	N12	x	N28	N31	N11	N21	N18	N3
6	N12	x	N28	N2	N11	N21	N18	N6
7	N12	x	N28	N2	N11	N21	N27	N6
8	N12	x	N28	N2	N13	N21	N27	N6
9	N12	x	N28	N2	N13	N21	N27	N7
10	N14	x	N30	N2	N20	N21	x	N7
11	N14	x	N30	N2	N20	N21	N29	N7
12	N14	x	N30	N5	N20	N21	N29	N8
13	N16	x	x	N5	N22	N4	N29	N8
14	N16	x	x	N5	N22	N4	N29	N8
15	N16	x	x	N9	N23	N10	N29	N8
16	N16	x	x	N9	N23	N10	N29	N8
17	N16	x	x	N9	N23	N10	N29	N8
18	N16	x	x	N9	N23	N10	N29	N8
19	N15	x	x	N9	N23	N10	N29	N8
20	N15	x	x	N9	N23	N10	N29	N8

░ 2.4V ▒ 3V ■ 5V

(a) scheduling result



(b) a DFG example

Figure 3.2: Example for DFG and scheduling result of 31 tasks

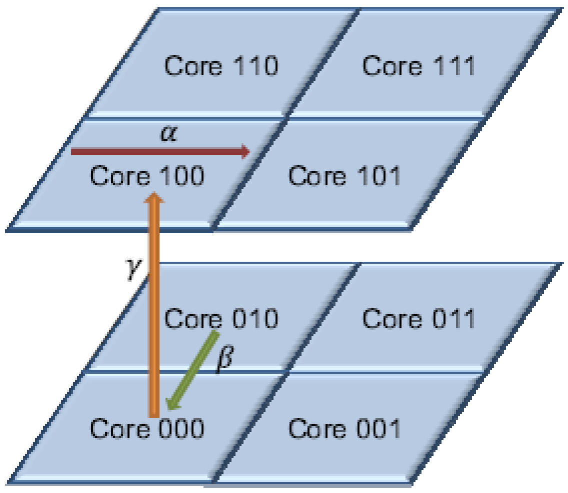


Figure 3.3: Transmission cost in a 3D multi-core processor

3.2.3 Energy model

To minimize energy consumption, an energy model needs to be incorporated into the DVFS technique. The number of the allowable voltage levels determined by the designer's preference and the manufacturing technology. Figure 3.4 shows voltage versus latency curve from [31]. In [41], the energy model in Table 3.1 was proposed based on [31] and includes 3 voltage levels: 5V, 3V and 2.4V. In this work, energy consumption can be defined as the execution delay multiplied by the power. We use such energy model to perform the voltage scaling. As shown in the energy model, the increase of execution-delay for each task from 5V to 3V and 3V to 2.4V are the same and the gain of energy reduction by lowering down a task from 5V to 3V and 3V to 2.4V are 17t and 3.2t, respectively.

Voltage(V)	Delay	Power(W)	Energy(J)
5	$1t$	25	$25t$
3	$2t$	4	$8t$
2.4	$3t$	1.6	$4.8t$

Table 3.1: Energy model

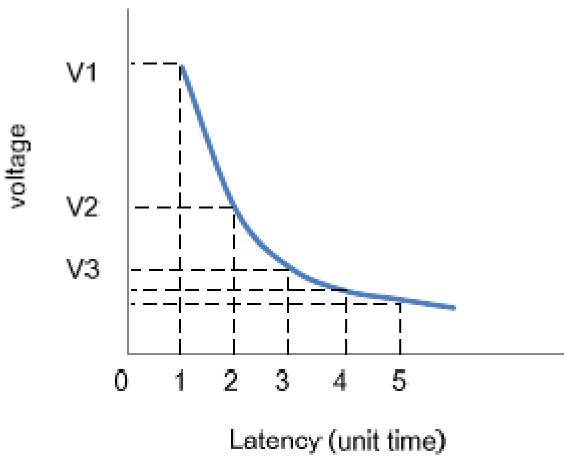


Figure 3.4: Voltage versus latency curve

Under the timing and resource constraints, selecting supply voltages for each task to minimize the energy consumption is crucial to an energy-aware scheduler. At the beginning, all tasks are assumed to run at 5V. After a fixed task-to-core mapping, many time slots are available. Therefore, all tasks compete for a limited number of spaces to achieve better energy efficiency. Then, Energy-Saving Rate (ESR) is computed according to Equation (14) to estimate the energy efficiency of a given schedule.

$$\text{ESR} = \frac{E_{init}(5V) - E_{final}(5V, 3V, 2.4V)}{E_{init}(5V)} \times 100\% \tag{14}$$

3.3 Baseline algorithm under a fixed task-to-core mapping

Wu et al. [41] proposed an energy-efficient task scheduling algorithm on top of [40] via DVFS at the system level and formulated a priority gain function considering both gains and losses for selecting tasks to scale down its frequency. Using their algorithm [41] as a baseline, we further propose two dynamic task-to-core remapping strategies to reduce slack slots and acquire energy saving. In this section, we first overview the baseline scheduling algorithm (denoted as ORI) in [41] and explain its key components (core mapping and voltage scaling) in details.

Figure 3.5 shows the overall flow of the baseline scheduling algorithm. First, before the task-to-core mapping, the earliest possible time (As Soon As Possible, ASAP [42]) and the latest possible time (As Late As Possible, ALAP [42]) for each operation are computed. Second, task-to-core mapping decides the core that a task runs on and its execution order. After task-to-core mapping, the initial energy of each task t 5V can be derived. Later, a task candidate set is computed based on a gain function and tasks with the highest rankings take turn to be selected for voltage scaling. Last, the energy-saving rate is derived to evaluate the energy efficiency of the schedule computed by the ORI algorithm. The pseudocode of the ORI algorithm is shown below.

```
ORI $algorithm$ :
1.  for $each\ task\ N_i \in DFG$ do
2.    $E_i \leftarrow ASAP\_analysis(N_i)$
3.    $L_i \leftarrow ALAP\_analysis(N_i)$
4.  end for
5.  for $each\ step_i\ under\ timing\ constraint$ do
6.    for $each\ core \in 3D\ multi-core\ architecture$ do
7.      $R \leftarrow find\_ready\_task()$
8.      $S \leftarrow task-to-core\_mapping()$
9.    end for
10. end for
11. $Energy_i \leftarrow compute\_initial\_energy()$
12. $C \leftarrow construct\_candidate\_for\_voltage\_scaling()$
13. while $C \neq \phi$ do
14.   $L \leftarrow priority\_of\_candidate\_list()$
15.   $C_i \leftarrow candidate\_selection\_with\_highest\_priority(L)$
16.   $V_i \leftarrow voltage\_scaling\_of\_C_i$
17.   $S \leftarrow Rescheduling()$
18.   $C \leftarrow update\_candidate\_set()$
19. end while
20. $Energy_f \leftarrow compute\_final\_energy()$
21. $ESR \leftarrow compute\_energy-saving\_rate()$
```

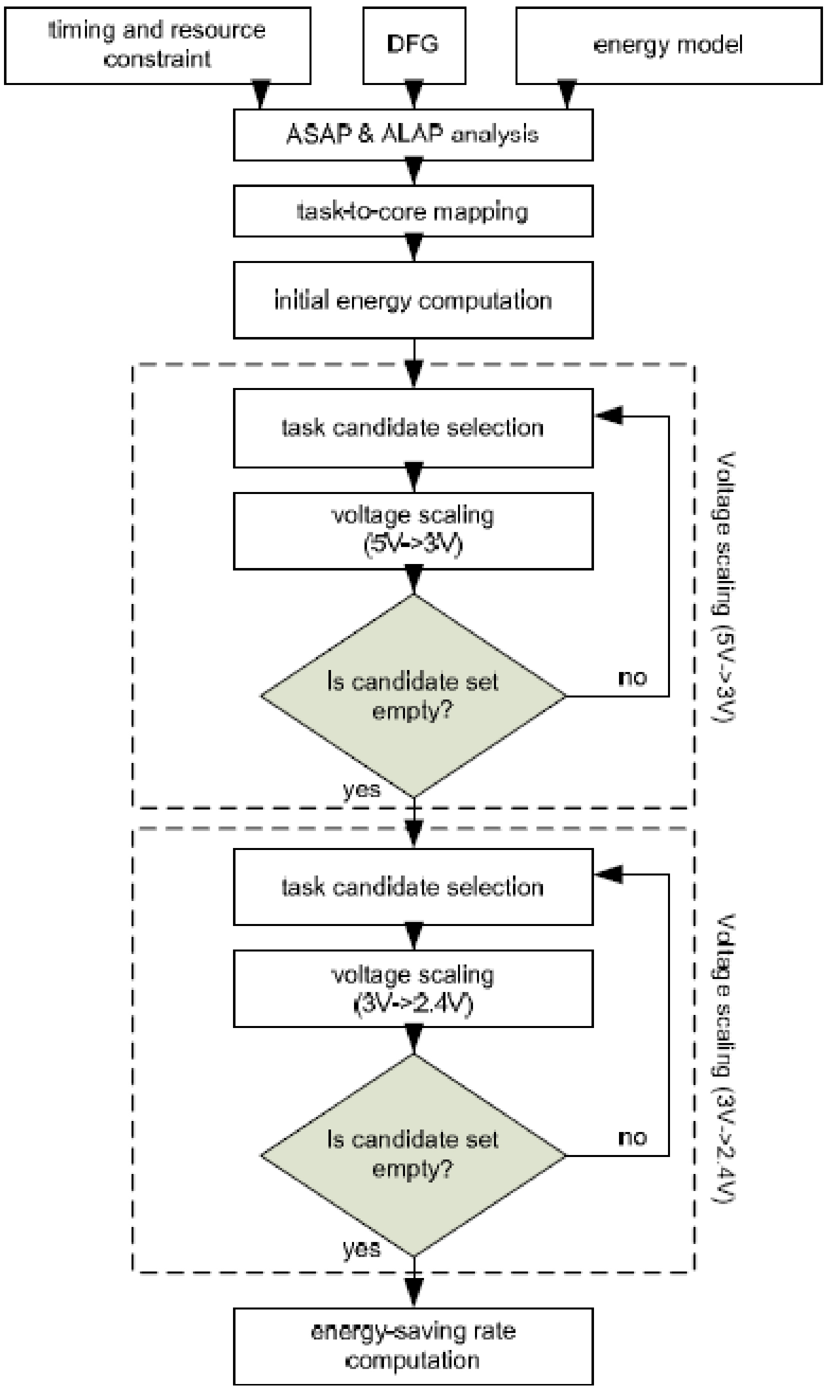


Figure 3.5: Design flow of baseline scheduling

3.3.1 Task-to-core mapping

In the baseline algorithm, task-to-core mapping is computed by List-Scheduling-based approach [42] [43] (through a dynamic priority list). The ready tasks are stored in the dynamic priority list for each time slot of each core. A task is ready to be mapped to a core if and only if all of its predecessors are mapped and the core mapping of such task satisfies its transmission constraint. The priority of tasks mapping to core is decided by their mobility:

$$mobility = ALAP - ASAP \tag{15}$$

A task with a lower mobility has the higher priority to be mapped. Mapping each task in the dynamic priority list to its free core needs to consider the dependence of the given DFG. Furthermore, a task is preferred to be mapped to the same core with its predecessor to avoid generating additional dependency.

3.3.2 Voltage scaling

After the task-to-core mapping, the remaining space can be further reduced by rescheduling tasks using DVFS which changes the voltage level and increases the execution time for a task. All tasks with space_rate ≥ 1 are selected to be the candidate for voltage scaling where the space_rate is:

$$space\_rate = \frac{\text{mobility}}{delay} \tag{16}$$

Variable delay in Equation (16) denotes the execution delay of a task. If a task with space rate < 1, there are not enough slack slots for scaling down the frequency of a task. After constructing the task candidate set, each task with the highest gain value is selected for voltage scaling.

Another key problem is how to decide the priority of each candidate using the gain function, where the gain function of the task candidate C_i can be defined as:

$$\text{Gain}_{overall}(C_i) = \text{Gain}(C_i) - \text{Lose}(PS_i) \tag{17}$$

Gain (C_i) in Equation (18) denotes the gain (energy reduction) after scaling down the voltage of C_i, while Lose (PS_i) in Equation (19) denotes scaling down the voltage of one candidate may block the possible energy gains from its predecessors and successors. α and β are parameters specified in the energy model and PS_i are predecessors and successors of C_i. Hence, the gain function considers the both gains and losses for selecting the task to be scaled down.

$$\mathrm{Gain}(C_i) = \alpha \times Delay_{C_i} \tag{18}$$

$$\mathrm{Lose}(PS_i) = \beta \times Delay_{PS_i} \tag{19}$$

Figure 3.2 (a) shows the scheduling result of 31 tasks by the baseline algorithm on a 3D processor with eight cores. The energy-saving rate (ESR) is computed according to Equation (1) and the final ESR after voltage scaling by the ORI method is 51.79 percent.

3.4 Dynamic task-to-core remapping strategies

From the previous example in Figure 3.2, after a fixed mapping, many time slots are still available. We are motivated to explore utilization of the remaining slack slots. Therefore, we propose two dynamic task-to-core remapping strategies and integrate them with the ORI algorithms. The following sections will elaborate each respective strategy in details.

3.4.1 Dynamic remapping (DR)

The idea of the Dynamic Remapping (DR) strategy comes from observing the execution of the ORI algorithm under a fixed core mapping. After voltage scaling, the distribution of tasks on cores can be more non-uniform. If we apply the task-to-core remapping after voltage scaling, the task density of cores becomes more uniform and may acquire more energy saving.

Figure 3.6 shows the flow of the DR strategy. There are two rounds in the DR strategy where task-to- core mapping and voltage scaling are performed in both round. Input information required by this strategy includes a initial timing constraint used for the first round and a timing-constraint limit used for the second round. Moreover, the initial timing constraint must be less than or equal to the timing-constraint limit. In the first round of the DR strategy, the task-to-core mapping and voltage scaling (5V →

3V) is performed under the initial timing constraint. Only 5V → 3V voltage scaling is applied to prevent the failure of task-to-core remapping later. After the first round of voltage scaling, the changed ASAP time and ALAP time of each task are updated. In the second round, mobility of each task need to be recomputed and then the task-to-core remapping and voltage scaling (5V→ 3V and 3V → 2.4V) are applied under the timing-constraint limit, respectively. Task-to-core remapping is also computed by List-Scheduling-based approach [42] [43] and the priority of tasks remapping to core is decided according to their mobility. After the task-to-core remapping, the distribution of executed tasks on each core can be more uniform.

Figure 3.7 and Figure 3.8 show the example of scheduling 31 tasks with the DR strategy on 3D processors with eight cores. The initial timing constraint is set to be 16 (starting from 1.05 critical path length) and the timing constraint limit is set to be 20. Figure 3.7 (a) and Figure 3.7 (b) show the first round of the DR strategy including task-to-core mapping and voltage scaling with 5V → 3V under a given timing constraint 16, respectively. After the voltage scaling in the first round, the distribution of executed tasks on each core is unbalanced. Especially on core 010, 18 slack slots are available. In the second round, the task-to-core remapping is applied to the result in Figure 3.8 (a) under the timing constraint limit 20, then voltage scaling 5V→ 3V and 3V → 2:4V are applied as shown in Figure 3.8 (b). Comparing Figure 3.7 (b) with Figure 3.8 (a), after the task-to-core remapping in the second round, the distribution of executed tasks on each core is found more balanced. Hence, a higher energy-saving rate is achieved after the voltage scaling in the second round. The final energy-saving rate of the DR strategy is 55.32 percent more than that from the ORI method on the same case.

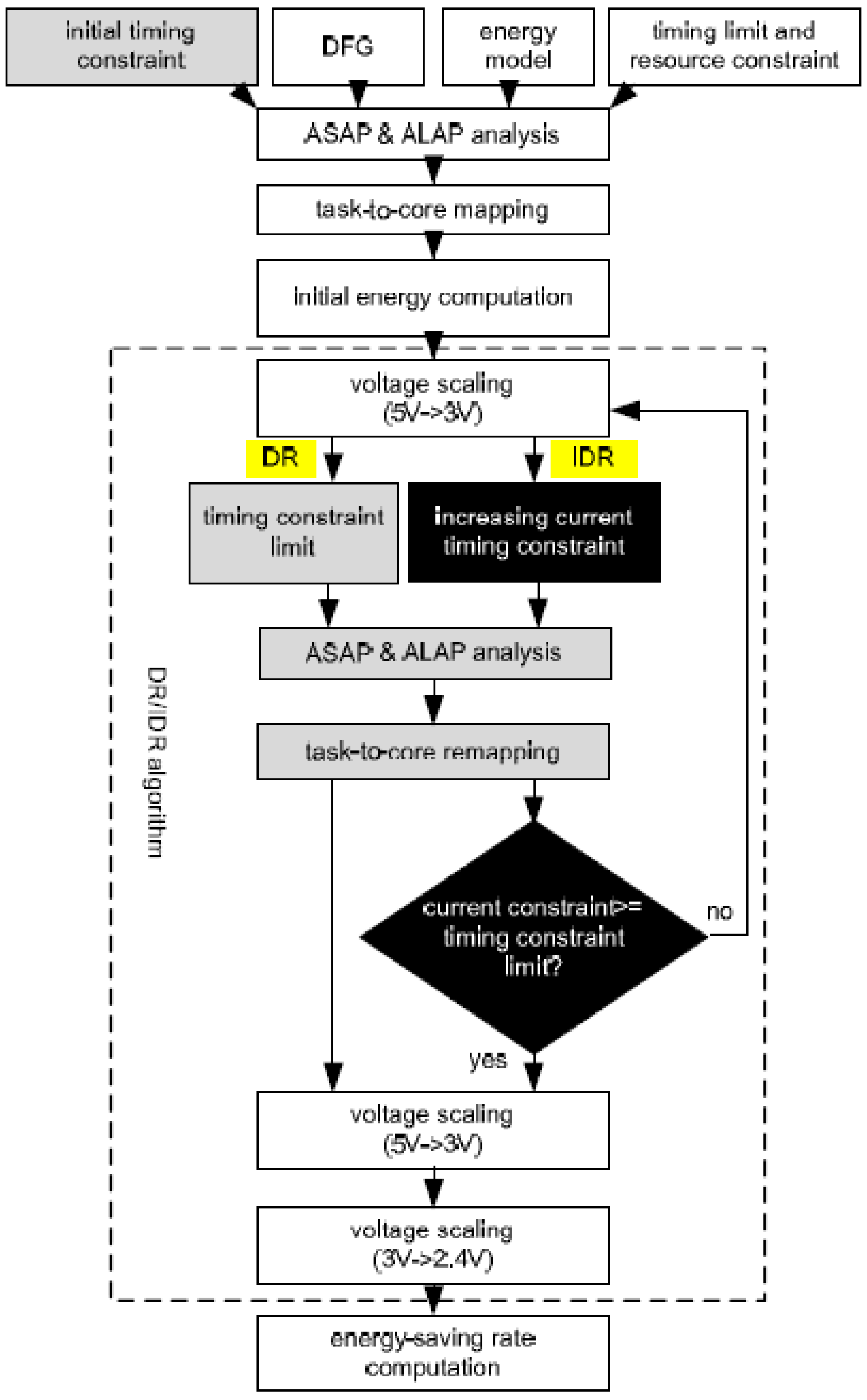
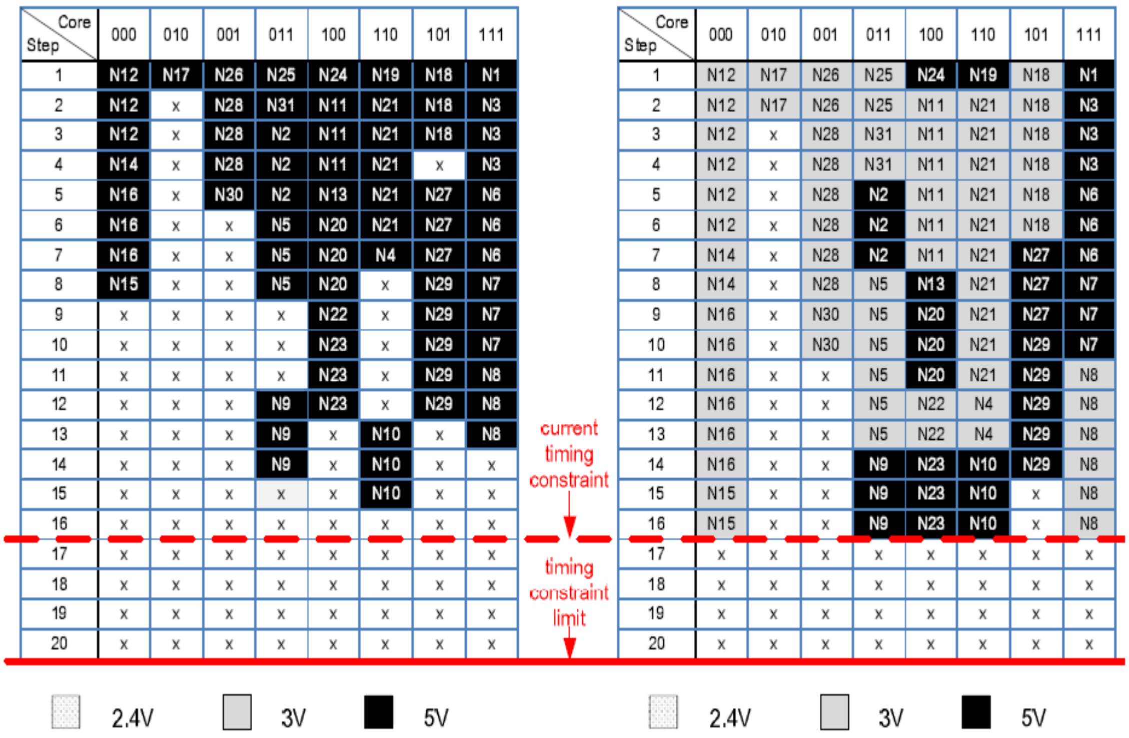

initial timing constraint
DFG
energy model
timing limit and resource constraint
ASAP & ALAP analysis
task-to-core mapping
Initial energy computation
voltage scaling (5V->3V)
DR
IDR
timing constraint limit
increasing current timing constraint
ASAP & ALAP analysis
task-to-core remapping
DR/IDR algorithm
current constraint>= timing constraint limit?
no
yes
voltage scaling (5V->3V)
voltage scaling (3V->2.4V)
energy-saving rate computation

Figure 3.6: Design flow of DR/IDR



current timing constraint
timing constraint limit
2.4V
3V
5V

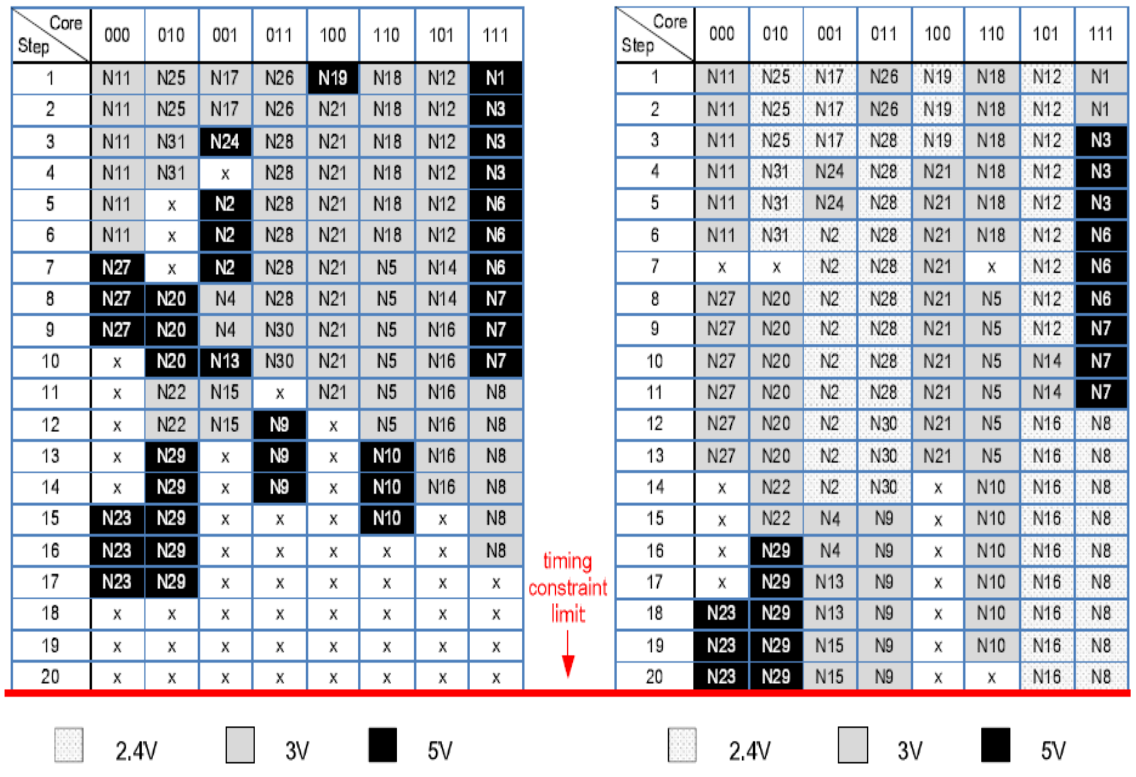
Step \ Core	000	010	001	011	100	110	101	111
1	N12	N17	N26	N25	N24	N19	N18	N1
2	N12	x	N28	N31	N11	N21	N18	N3
3	N12	x	N28	N2	N11	N21	N18	N3
4	N14	x	N28	N2	N11	N21	x	N3
5	N16	x	N30	N2	N13	N21	N27	N6
6	N16	x	x	N5	N20	N21	N27	N6
7	N16	x	x	N5	N20	N4	N27	N6
8	N15	x	x	N5	N20	x	N29	N7
9	x	x	x	x	N22	x	N29	N7
10	x	x	x	x	N23	x	N29	N7
11	x	x	x	x	N23	x	N29	N8
12	x	x	x	N9	N23	x	N29	N8
13	x	x	x	N9	x	N10	x	N8
14	x	x	x	N9	x	N10	x	x
15	x	x	x	x	x	N10	x	x
16	x	x	x	x	x	x	x	x
17	x	x	x	x	x	x	x	x
18	x	x	x	x	x	x	x	x
19	x	x	x	x	x	x	x	x
20	x	x	x	x	x	x	x	x

(a) task-to-core mapping

Step \ Core	000	010	001	011	100	110	101	111
1	N12	N17	N26	N25	N24	N19	N18	N1
2	N12	N17	N26	N25	N11	N21	N18	N3
3	N12	x	N28	N31	N11	N21	N18	N3
4	N12	x	N28	N31	N11	N21	N18	N3
5	N12	x	N28	N2	N11	N21	N18	N6
6	N12	x	N28	N2	N11	N21	N18	N6
7	N14	x	N28	N2	N11	N21	N27	N6
8	N14	x	N28	N5	N13	N21	N27	N7
9	N16	x	N30	N5	N20	N21	N27	N7
10	N16	x	N30	N5	N20	N21	N29	N7
11	N16	x	x	N5	N20	N21	N29	N8
12	N16	x	x	N5	N22	N4	N29	N8
13	N16	x	x	N5	N22	N4	N29	N8
14	N16	x	x	N9	N23	N10	N29	N8
15	N15	x	x	N9	N23	N10	x	N8
16	N15	x	x	N9	N23	N10	x	N8
17	x	x	x	x	x	x	x	x
18	x	x	x	x	x	x	x	x
19	x	x	x	x	x	x	x	x
20	x	x	x	x	x	x	x	x

(b) voltage scaling (5V→3V)

Figure 3.7: First stage of DR strategy under a timing constraint 16



timing constraint limit
2.4V
3V
5V

Step \ Core	000	010	001	011	100	110	101	111
1	N11	N25	N17	N26	N19	N18	N12	N1
2	N11	N25	N17	N26	N21	N18	N12	N3
3	N11	N31	N24	N28	N21	N18	N12	N3
4	N11	N31	x	N28	N21	N18	N12	N3
5	N11	x	N2	N28	N21	N18	N12	N6
6	N11	x	N2	N28	N21	N18	N12	N6
7	N27	x	N2	N28	N21	N5	N14	N6
8	N27	N20	N4	N28	N21	N5	N14	N7
9	N27	N20	N4	N30	N21	N5	N16	N7
10	x	N20	N13	N30	N21	N5	N16	N7
11	x	N22	N15	x	N21	N5	N16	N8
12	x	N22	N15	N9	x	N5	N16	N8
13	x	N29	x	N9	x	N10	N16	N8
14	x	N29	x	N9	x	N10	N16	N8
15	N23	N29	x	x	x	N10	x	N8
16	N23	N29	x	x	x	x	x	N8
17	N23	N29	x	x	x	x	x	x
18	x	x	x	x	x	x	x	x
19	x	x	x	x	x	x	x	x
20	x	x	x	x	x	x	x	x

(a) task-to-core remapping

Step \ Core	000	010	001	011	100	110	101	111
1	N11	N25	N17	N26	N19	N18	N12	N1
2	N11	N25	N17	N26	N19	N18	N12	N1
3	N11	N25	N17	N28	N19	N18	N12	N3
4	N11	N31	N24	N28	N21	N18	N12	N3
5	N11	N31	N24	N28	N21	N18	N12	N3
6	N11	N31	N2	N28	N21	N18	N12	N6
7	x	x	N2	N28	N21	x	N12	N6
8	N27	N20	N2	N28	N21	N5	N12	N6
9	N27	N20	N2	N28	N21	N5	N12	N7
10	N27	N20	N2	N28	N21	N5	N14	N7
11	N27	N20	N2	N28	N21	N5	N14	N7
12	N27	N20	N2	N30	N21	N5	N16	N8
13	N27	N20	N2	N30	N21	N5	N16	N8
14	x	N22	N2	N30	x	N10	N16	N8
15	x	N22	N4	N9	x	N10	N16	N8
16	x	N29	N4	N9	x	N10	N16	N8
17	x	N29	N13	N9	x	N10	N16	N8
18	N23	N29	N13	N9	x	N10	N16	N8
19	N23	N29	N15	N9	x	N10	N16	N8
20	N23	N29	N15	N9	x	x	N16	N8

(b) voltage scaling

Figure 3.8: Second stage of DR strategy under a timing constraint 20

3.4.2 Iterative dynamic remapping (IDR)

Based on the DR strategy, we further propose Iterative Dynamic Remapping (IDR), where multiple rounds of task-to-core remapping and voltage scaling are performed. Similar to the DR strategy, IDR also applies voltage scaling under the initial timing constraint in the first round and finally applies another round of voltage scaling under a timing constraint limit. However, DR and IDR differ from IDR having multiple rounds of task-to-core remapping and voltage-scaling applications.

Figure 3.6 shows the flow of the IDR strategy on top of the ORI method where the current timing constraint incrementally increases up to the timing-constraint limit in IDR at two black boxes. Hence, the task-to-core remapping and voltage scaling under a current timing constraint are applied iteratively in IDR. Same as DR, only 5V → 3V voltage scaling is applied for preventing the failures of the task-to-core remapping. In each round of task-to-core remapping and 5V →3V voltage scaling, the current timing constraint is relaxed until the current timing constraint reaches the timing-constraint limit. We perform the final round of voltage scaling (5V → 3V and 3V→ 2.4V) under the timing-constraint limit. The pseudocode of the DR/IDR algorithm (indicated by the dotted-line box in Figure 5 (b)) is shown below.

```
**DR/IDR** $algorithm$ :
1.  $T \leftarrow initial\ timing\ constraint()$
2.  $S \leftarrow voltage\ scaling(5V \rightarrow 3V)$
3.  **if** $DR()$
4.      $T \leftarrow timing\ constraint\ limit()$
5.      $A \leftarrow ASAP\_ALAP\_analysis()$
6.      $S \leftarrow task-to-core\_remapping()$
7.  **else if** $IDR()$
8.      **do**
9.          $T \leftarrow increasing\ current\ timing\ constraint()$
10.         $A \leftarrow ASAP\_ALAP\_analysis()$
11.         $S \leftarrow task-to-core\_remapping()$
12.     **while** $current\ timing\ constraint \leq timing\ constraint\ limit$
```

```
13. end if
14. $S \leftarrow voltage\ scaling(5V \rightarrow 3V \& 3V \rightarrow 2.4V)$
```

For example, scheduling 31 tasks with the IDR strategy on a 3D processor with eight cores is illustrated. In the first round of the IDR strategy under a initial timing constraint 16, task-to-core mapping and voltage scaling with 5V $\rightarrow$ 3V are performed as in Figure 3.7 (a) and Figure 3.7 (b). Then, the current timing constraint is relaxed to 17. In the second round, task-to-core remapping and voltage scaling with 5V $\rightarrow$ 3V under the timing constraint 17 are performed, respectively. In the following rounds (the third and fourth rounds), task-to-core remapping and voltage scaling with 5V $\rightarrow$ 3V are performed under the timing constraint 18 and 19. At last, when the current timing constraint is 20, we perform task-to-core remapping and voltage scaling (5V $\rightarrow$ 3V and 3V $\rightarrow$ 2.4V) in the final round. The final energy-saving rate of the IDR strategy is 63:17 percent more than that from the ORI method on the same case.

Chapter 4 Experimental Result

4.1 Fast scan-chain ordering for 3D-IC designs under through-silicon-via (TSV) constraints

4.1.1 Experimental setup

A reference flow for 3D scan designs is provided in [17]. We modify the flow as shown in Figure 10 and utilize commercial softwares to complete 3D scan designs. In Figure 10, we first partition an original design to N layers which minimizes the number of TSVs in use and balances the area between different layers. After obtaining N-layer designs, Design Compiler does logic synthesis for each layer design. Then, all FFs are placed in N-layer designs with scan FFs by First Encounter. Finally, all planar placements are combined into one single 3D placement, and outputs of all scan FF locations from Design Exchange Format (DEF) files are provided. Therefore, we obtain all layout information.

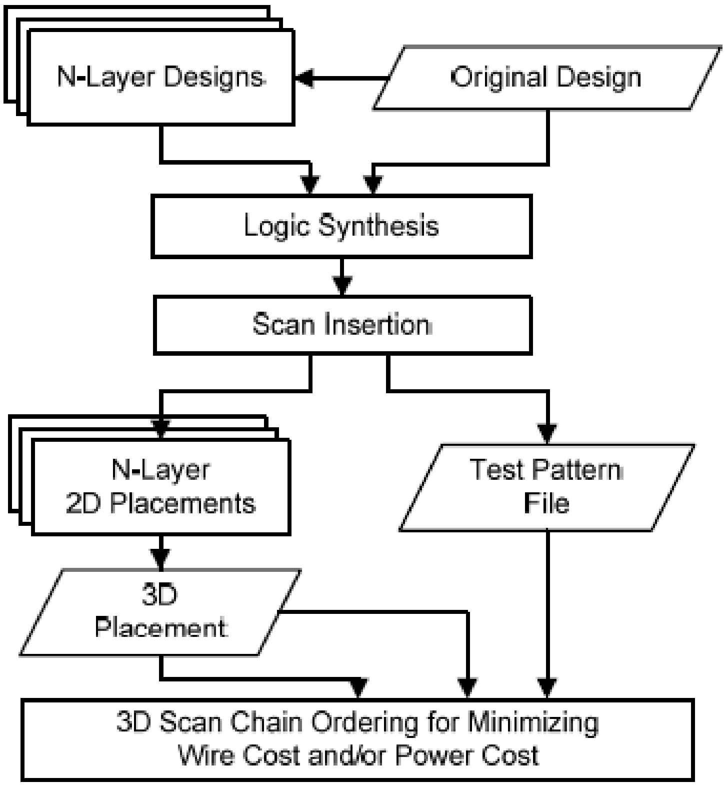


Figure 4.1: Proposed 3D scan design flow

In Figure 4.1, we also perform logic synthesis and scan insertion for the original design and then retrieve the test information via Standard Test Interface Language (STIL) files by using TetraMax. Two input data, including the layout information and test patterns, are achieved to reduce the scan induced power by performing the proposed algorithm. Using the same input data, we also perform the proposed algorithm by simultaneously considering wire and power costs.

For an objective comparison, the settings used in the previous GA approach [17] are employed here. The population size is set to 2000; the same operators are used, which include reproduction, crossover, and mutation. The GA stops when no more than 0.0001 percent improvement on the fitness score (i.e., the total scan-stitching wire length or total weighted transitions) can be obtained for the last 1000 generations.

Both the proposed 3D scan-chain ordering algorithm and the previous GA approach are exercised on a Linux machine with a Pentium Core Duo (2.4 GHz) processor and 4 GB memory. TSMC .18μm library is used and the height of a TSV is set as 10μm while the partitions for 3D ICs range from 3 to 5. ISCAS'89 benchmark circuits and two large 3D-IC designs from [23] are used to conduct experiments.

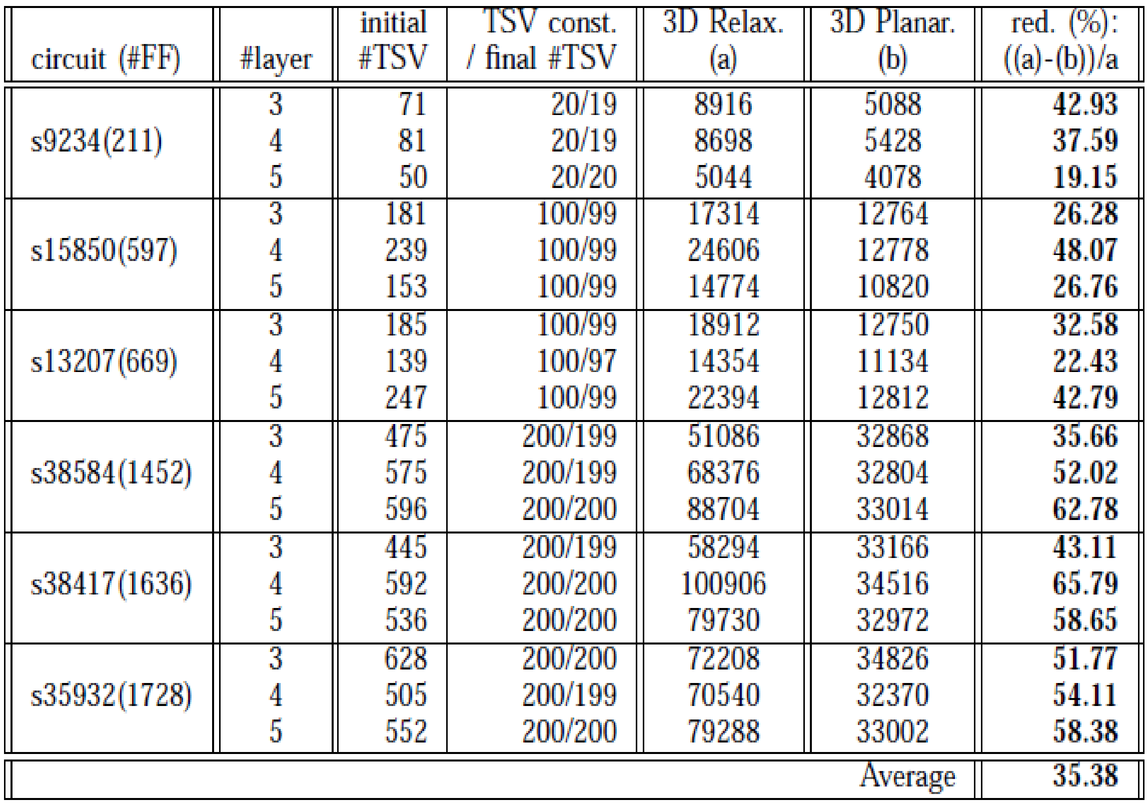
All experiments are divided into four parts. The first part minimizes the scan-stitching wire cost. The second part reflects the results of the scan-induced power dissipation reduction. The third part identifies the best scan-chain ordering while simultaneously considering wire-and-power costs. It is natural to use multiple scan chains to prevent long test-application time when the total number of scan FFs is large. Therefore, the last part applies the proposed algorithm to multiple scan chains on two large 3D-IC designs from [23].

4.1.2 Experimental results

4.1.2.1 Minimizing wire cost

Table 4.1 shows the preliminary performance of our algorithm with TSV constraints. The third column shows the TSV usage in the initial ordering. The fourth column reports the TSV constraints followed by the TSV usage in the wire-cost minimized ordering. The fifth and sixth columns show the total wire cost in μm after performing 3D relaxation and 3D planarization, respectively. Both solutions can satisfy the related TSV constraints. The seventh column reports the reduction rate from constraint-solving to local refinement during the second stage. Note that the total wire cost after stage one is not presented since the initial solution is usually not feasible due to an overuse of TSVs. That is, it is meaningless to compare such illegal wire cost with those from feasible solutions.

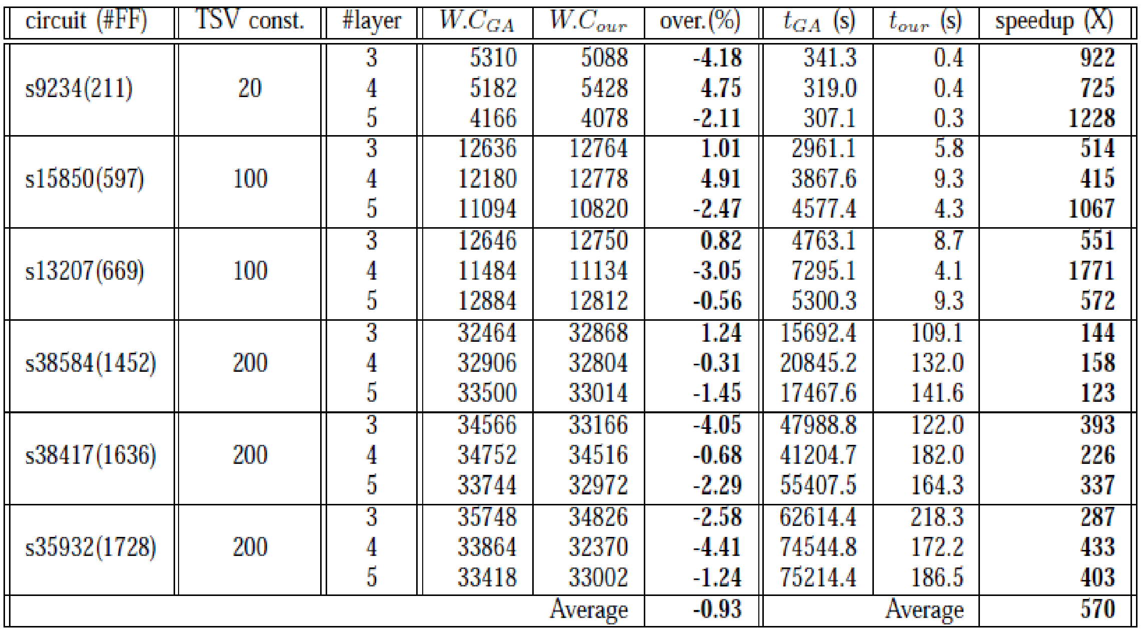
As a result, Table I shows a good reduction in the total number of TSVs on big circuits, such as s38584, s38417, and s35932. Meanwhile, the four-layer s38417 has the best reduction rate; 65.79 percent after constraint-solving. For all circuits, the 3D planarization technique achieves on average a 35.38 percent reduction in wire cost but it consumes slightly more time.



circuit (#FF)	#layer	initial #TSV	TSV const. / final #TSV	3D Relax. (a)	3D Planar. (b)	red. (%): ((a)-(b))/a
s9234(211)	3	71	20/19	8916	5088	**42.93**
	4	81	20/19	8698	5428	**37.59**
	5	50	20/20	5044	4078	**19.15**
s15850(597)	3	181	100/99	17314	12764	**26.28**
	4	239	100/99	24606	12778	**48.07**
	5	153	100/99	14774	10820	**26.76**
s13207(669)	3	185	100/99	18912	12750	**32.58**
	4	139	100/97	14354	11134	**22.43**
	5	247	100/99	22394	12812	**42.79**
s38584(1452)	3	475	200/199	51086	32868	**35.66**
	4	575	200/199	68376	32804	**52.02**
	5	596	200/200	88704	33014	**62.78**
s38417(1636)	3	445	200/199	58294	33166	**43.11**
	4	592	200/200	100906	34516	**65.79**
	5	536	200/200	79730	32972	**58.65**
s35932(1728)	3	628	200/200	72208	34826	**51.77**
	4	505	200/199	70540	32370	**54.11**
	5	552	200/200	79288	33002	**58.38**
					Average	**35.38**

Table 4.1: Effectiveness of 3D relaxation and 3D planarization on wire and TSV usage reduction

Table 4.2 reports the results of comparing the wire cost between a GA method and our algorithm under different TSV constraints for all benchmark circuits. The second column specifies the maximum number of TSVs that can be used. Note that the benchmark circuits of different scales are imposed with different TSV constraints, from 20 to 200. The third column shows the number of partitions for layers. The fourth and fifth columns show the total wire cost in μm after performing the GA method and the proposed algorithm, respectively. The sixth column shows the improvement ratio of our approach compared with the GA method. The seventh and eighth columns report run time in seconds for our algorithm and the GA method, respectively. The last column shows the speed-up rate of our algorithm compared with the GA method. The run time of the proposed algorithm is proportional to the number of iterations used to perform 3D planarization and circuit size. Although the proposed algorithm results in the slightly inferior total cost on s15850, it produces comparable or even better results than the GA method on all other larger circuits. Moreover, the proposed algorithm can run at least two-orders faster than the GA method.



circuit (#FF)	TSV const.	#layer	$W.C_{GA}$	$W.C_{our}$	over.(%)	t_{GA} (s)	t_{our} (s)	speedup (X)
s9234(211)	20	3	5310	5088	-4.18	341.3	0.4	922
		4	5182	5428	4.75	319.0	0.4	725
		5	4166	4078	-2.11	307.1	0.3	1228
s15850(597)	100	3	12636	12764	1.01	2961.1	5.8	514
		4	12180	12778	4.91	3867.6	9.3	415
		5	11094	10820	-2.47	4577.4	4.3	1067
s13207(669)	100	3	12646	12750	0.82	4763.1	8.7	551
		4	11484	11134	-3.05	7295.1	4.1	1771
		5	12884	12812	-0.56	5300.3	9.3	572
s38584(1452)	200	3	32464	32868	1.24	15692.4	109.1	144
		4	32906	32804	-0.31	20845.2	132.0	158
		5	33500	33014	-1.45	17467.6	141.6	123
s38417(1636)	200	3	34566	33166	-4.05	47988.8	122.0	393
		4	34752	34516	-0.68	41204.7	182.0	226
		5	33744	32972	-2.29	55407.5	164.3	337
s35932(1728)	200	3	35748	34826	-2.58	62614.4	218.3	287
		4	33864	32370	-4.41	74544.8	172.2	433
		5	33418	33002	-1.24	75214.4	186.5	403
				Average	-0.93		Average	570

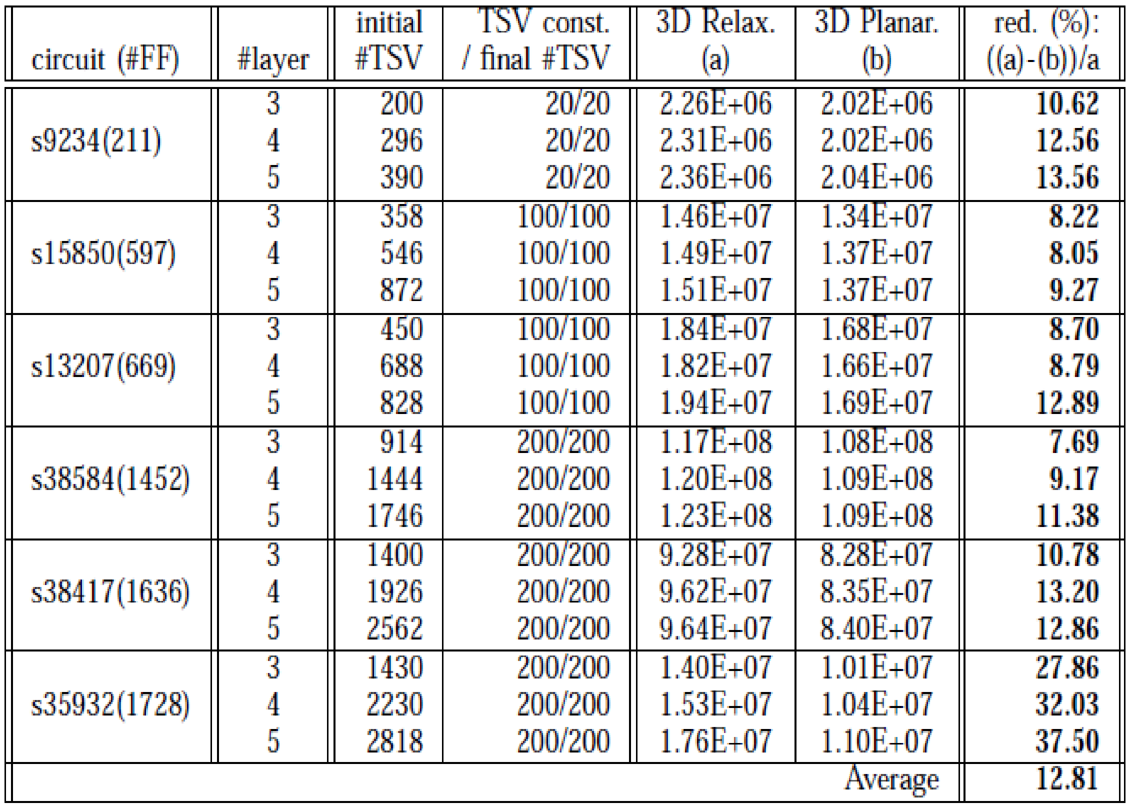
Table 4.2: Wire length and runtime comparison with different TSV constraint

4.1.2.2 Minimizing power cost

Table 4.3 shows the performance of our algorithm with the TSV constraint. In this case, the information for the number of layers for all circuits are shown. The third column shows the TSV usage in the initial ordering. The fourth column reports the TSV usage in the power-cost minimized ordering and the TSV constraints. The fifth and sixth columns show the total weighted transitions after performing 3D relaxation and 3D planarization, respectively. The seventh column also reports the reduction rate from two proposed techniques during stage 2.

Table 4.3 also shows good reduction in the total number of TSVs on the big circuits, especially on s35932. Particularly, the five-layer s35932 has the best reduction rate; 37.50 percent after 3D relaxation. Consequently, 3D planarization averages a reduction rate of 12.81 percent in total power cost.

Table4.4 reports the results of comparing the power costs between the GA method and our algorithm under different TSV constraints for all benchmark circuits. The second column specifies the limitation of TSVs in use. The third column show the number of partitions for layers. The fourth and fifth columns show the total weighted transitions after performing the GA method and the proposed algorithm, respectively. The sixth column also shows the improvement ratio of our approach compared with the GA method. The seventh and eighth columns report run time in seconds for our algorithm and the GA- method, respectively. The last column shows the speed-up rate of our algorithm compared with the GA method. The run time is related to the number of iterations used to perform the local refinement technique, look-up table construction, and circuit size. Again, the proposed algorithm results in comparable or even better results on all circuits, resulting in run time speeds that are at least two-orders faster than the GA method.



circuit (#FF)	#layer	initial #TSV	TSV const. / final #TSV	3D Relax. (a)	3D Planar. (b)	red. (%): ((a)-(b))/a
s9234(211)	3	200	20/20	2.26E+06	2.02E+06	**10.62**
	4	296	20/20	2.31E+06	2.02E+06	**12.56**
	5	390	20/20	2.36E+06	2.04E+06	**13.56**
s15850(597)	3	358	100/100	1.46E+07	1.34E+07	**8.22**
	4	546	100/100	1.49E+07	1.37E+07	**8.05**
	5	872	100/100	1.51E+07	1.37E+07	**9.27**
s13207(669)	3	450	100/100	1.84E+07	1.68E+07	**8.70**
	4	688	100/100	1.82E+07	1.66E+07	**8.79**
	5	828	100/100	1.94E+07	1.69E+07	**12.89**
s38584(1452)	3	914	200/200	1.17E+08	1.08E+08	**7.69**
	4	1444	200/200	1.20E+08	1.09E+08	**9.17**
	5	1746	200/200	1.23E+08	1.09E+08	**11.38**
s38417(1636)	3	1400	200/200	9.28E+07	8.28E+07	**10.78**
	4	1926	200/200	9.62E+07	8.35E+07	**13.20**
	5	2562	200/200	9.64E+07	8.40E+07	**12.86**
s35932(1728)	3	1430	200/200	1.40E+07	1.01E+07	**27.86**
	4	2230	200/200	1.53E+07	1.04E+07	**32.03**
	5	2818	200/200	1.76E+07	1.10E+07	**37.50**
					Average	**12.81**

Table 4.3: Effectiveness of 3D relaxation and 3D planarization on power and TSV usage reduction

circuit (#FF)	$TSV\,const.$	#layer	TWT_{GA}	TWT_{our}	over.(%)	t_{GA} (s)	t_{our} (s)	speedup (X)
s9234 (211)	20	3	1.99E+06	2.02E+06	**1.51**	584.2	2.1	**282**
		4	2.02E+06	2.02E+06	**0.00**	440.5	2.3	**195**
		5	2.03E+06	2.04E+06	**0.49**	380.6	2.2	**172**
s15850 (597)	100	3	1.36E+07	1.34E+07	**-1.47**	9522.3	23.4	**407**
		4	1.38E+07	1.37E+07	**-0.72**	8534.6	26.9	**317**
		5	1.36E+07	1.37E+07	**0.74**	8641.2	27.2	**317**
s13207 (669)	100	3	1.66E+07	1.68E+07	**1.20**	12591.3	35.1	**359**
		4	1.65E+07	1.66E+07	**0.61**	12612.5	34.9	**361**
		5	1.68E+07	1.69E+07	**0.60**	9722.1	42.9	**227**
s38584 (1452)	200	3	1.16E+08	1.08E+08	**-6.90**	65920.9	349.3	**189**
		4	1.17E+08	1.09E+08	**-6.84**	56979.5	410.4	**139**
		5	1.16E+08	1.09E+08	**-6.03**	62367.5	444.9	**140**
s38417 (1636)	200	3	8.86E+07	8.28E+07	**-6.55**	111766.0	486.5	**230**
		4	8.76E+07	8.35E+07	**-4.68**	106233.0	565.5	**188**
		5	8.94E+07	8.40E+07	**-6.04**	92206.0	602.6	**153**
s35932 (1728)	200	3	1.05E+07	1.01E+07	**-3.81**	188079.0	516.7	**364**
		4	1.08E+07	1.04E+07	**-3.70**	162181.0	604.2	**268**
		5	1.11E+07	1.10E+07	**-0.90**	145956.0	693.3	**211**
				Average	**-2.50**		Average	**251**

Table 4.4: Power dissipation and runtime comparison with different TSV constraint

4.1.2.3 Considering wire-and-power costs simultaneously

We further perform the proposed algorithm under TSV constraints while simultaneously considering wire and power costs. Table 4.5 and table 4.6 report the results of comparing the wire-and-power costs between the GA method and our algorithm under TSV constraints 20 for circuit s1423 (total number of FFs is 74) and circuit s5378 (total number of FFs is 179) with three-layer partitions. The first column specifies the coefficient α in Equation (13) from 0 to 1. The second and third columns show the total wire cost after performing the GA method and the proposed algorithm, respectively. The fourth column shows the improvement ratio of our approach to the GA method. The fifth and sixth columns show the total power cost after performing the GA method and the proposed algorithm, respectively. The seventh column also shows the improvement ratio of our approach compared with the GA method. The last column shows the speed-up rate of our algorithm compared with the GA method. Similarly, the proposed algorithm produce comparable or even better results with at least two-orders run time speedups to the GA method.

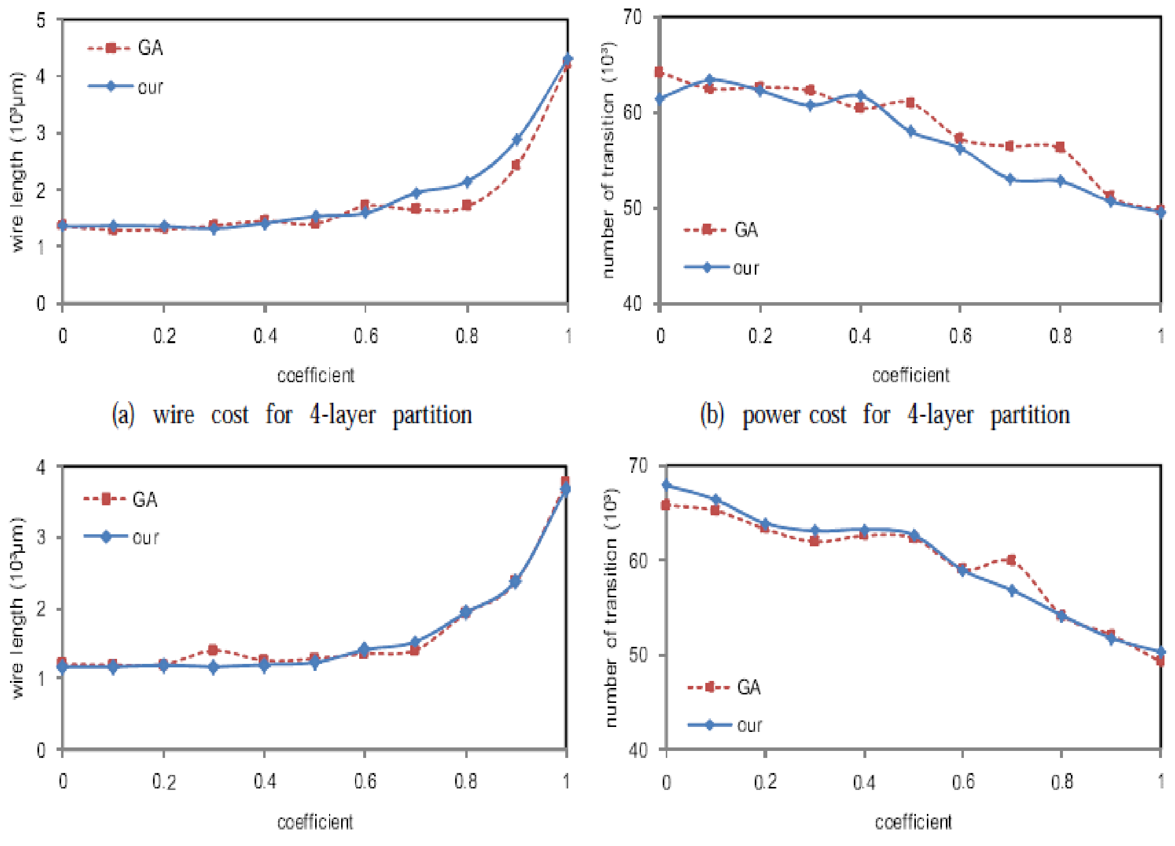
coefficient	$W.C_{GA}$ (um)	$W.C_{our}$ (um)	over.(%)	TWT_{GA}	TWT_{our}	over.(%)	speedup (X)
0	1284	1268	-0.01	66093	66391	0.00	1037.86
0.1	1462	1296	-0.11	64277	64199	0.00	637
0.2	1310	1248	-0.05	63439	64209	0.01	2976.33
0.3	1370	1300	-0.05	63225	63103	0.00	713.57
0.4	1376	1310	-0.05	62503	62353	0.00	647.29
0.5	1364	1342	-0.02	61645	61009	-0.01	3478.71
0.6	1540	1524	-0.01	58131	58103	0.00	470.63
0.7	1760	1888	0.07	57255	55067	-0.04	654.71
0.8	1676	1952	0.16	56007	54797	-0.02	529.13
0.9	2350	3028	0.29	52577	50541	-0.04	587.5
1	5342	5598	0.05	48559	48467	0.00	656.67
		Average	0.02		Average	-0.01	1125.33

Table 4.5: Wire-and-power cost and runtime comparison with different TSV constraint 20 on circuit s1423

coefficient	$W.C_{GA}$ (um)	$W.C_{our}$ (um)	over.(%)	TWT_{GA}	TWT_{our}	over.(%)	speedup (X)
0	3618	3602	0.00	925720	933772	0.01	381.41
0.1	3748	3484	-0.07	918654	930247	0.01	253.12
0.2	3628	3600	-0.01	908644	927748	0.02	274.88
0.3	3832	3446	-0.1	890707	927278	0.04	289.56
0.4	3764	3508	-0.07	893832	907906	0.02	268.35
0.5	4002	3856	-0.04	888702	883268	-0.01	241.11
0.6	4022	3862	-0.04	876088	882418	0.01	299.85
0.7	4796	4598	-0.04	841270	860036	0.02	361.86
0.8	5904	6088	0.03	824076	821180	0.00	449.92
0.9	8624	10002	0.16	786914	769509	-0.02	347.34
1	19418	20888	0.08	747609	744027	0.00	208.14
		Average	-0.01		Average	0.01	306.36

Table 4.6: Wire-and-power cost and runtime comparison with different TSV constraint 20 on circuit s5378

Figure 4.2 and Figure 4.3 compare the combined cost between the GA method and our algorithm on circuit s1423 and s5378 with four-layer and five-layer partitions, respectively. The maximum number of TSVs that can be used is 20 for both circuits. The X-axis denotes the coefficient α in Equation (13) ranging from 0 to 1. Figure 4.2(a), Figure 4.2(c), Figure 4.3(a), and Figure 4.3(c) show the wire costs under different co efficient values after performing the GA method and the proposed algorithm, respectively. Figure 4.2(b), Figure 4.2(d), Figure 4.3(b), and Figure 4.3(d) show the total weighted transitions under different coefficient values after performing the GA method and the proposed algorithm, respectively. According to Figure 4.2 and Figure 4.3, as the coefficient increases, power cost decreases but wire cost increases, and vice versa. The coefficient α can be adjusted by users at their discretion. Again, the proposed algorithm can result in comparable or even better wire and/or power cost compared with the result from the GA method.



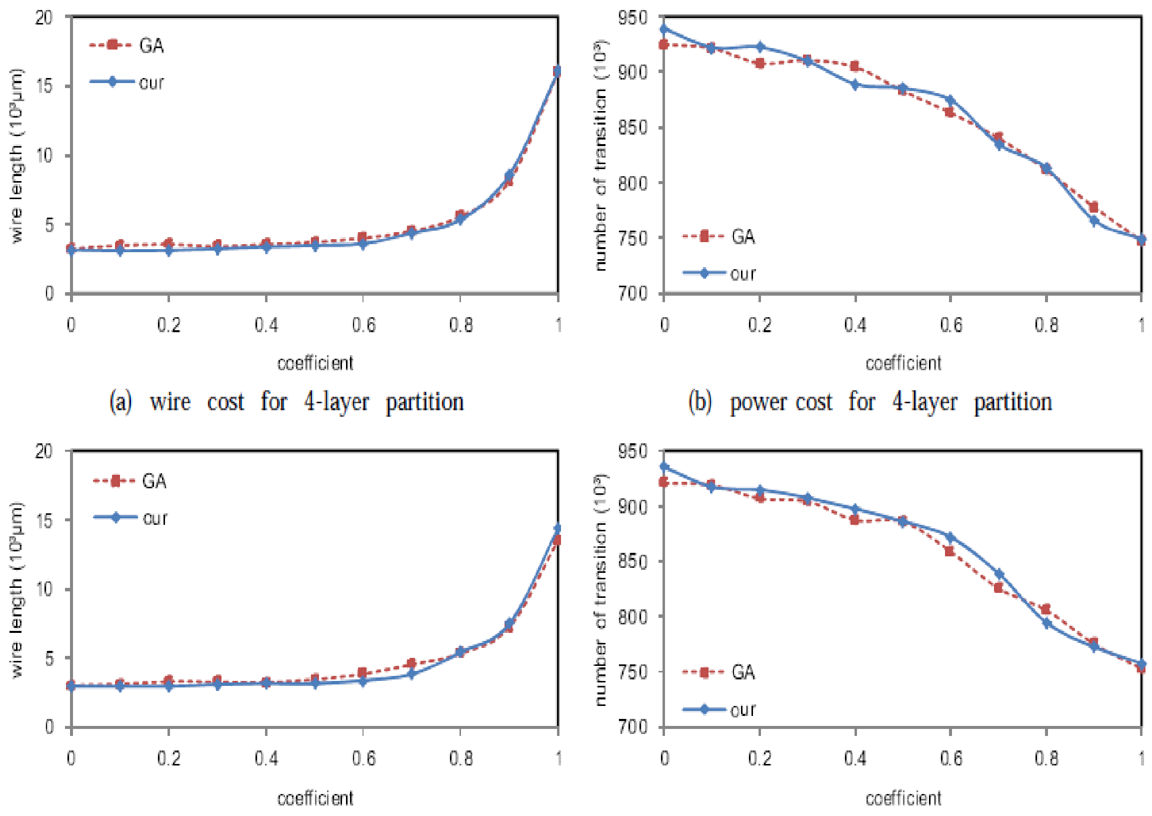
(a) wire cost for 4-layer partition

(b) power cost for 4-layer partition

(c) wire cost for 5-layer partition

(d) power cost for 5-layer partition

Figure 4.2: Comparison among GA and our algorithm for considering power and wire simultaneously on circuit s1423



(a) wire cost for 4-layer partition (b) power cost for 4-layer partition

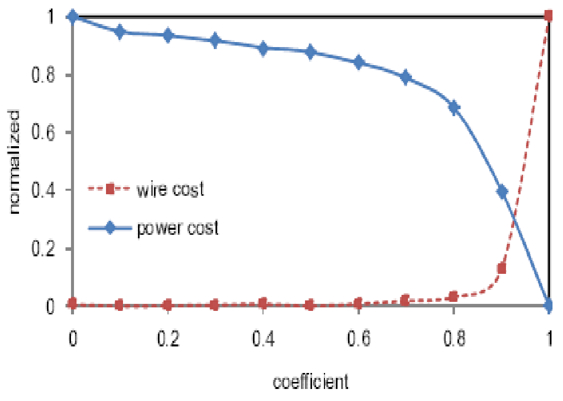
(c) wire cost for 5-layer partition (d) power cost for 5-layer partition

Figure 4.3: Comparison among GA and our algorithm for considering power and wire simultaneously on circuit s5378

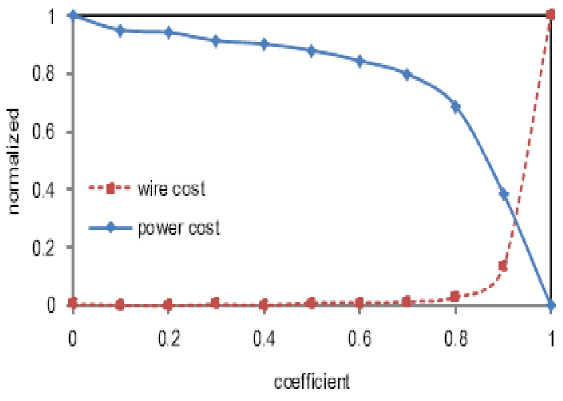
4.1.2.4 Multiple scan-chain ordering

It is common to use multiple scan chains to prevent long test-application time when the total number of scan FFs is great. Figure 4.4 and Figure 4.5 show the results of multiple scan-chain ordering when considering the combined costs under TSV constraints on two large 3D circuit designs, bench 2 and bench 7, from [23] with 4-layer and 5-layer partitions. The total number of FFs is 4095 in bench 2 (50 K gates with two clock domains) using 24 scan chains while the total number of FFs is 17983 in bench 7 (385 K gates with two clock domains) using 108 scan chains. The TSV constraint for each chain is set as 20 and the average number of FFs in each chain is under 200.

In Figure 4.3 and Figure 4.4, the X-axis denotes the coefficient α in Equation (13) from 0 to 1. The wire and power costs under different coefficients after performing the proposed algorithm are shown, respectively. As the coefficient increases, the power cost decreases but the wire cost increases. The combined cost for bench 2 and bench 7 can be computed in seconds; thus, the efficiency of the proposed algorithm is demonstrated. In summary, our algorithm is capable of supporting multiple scan chains after scan partitioning.

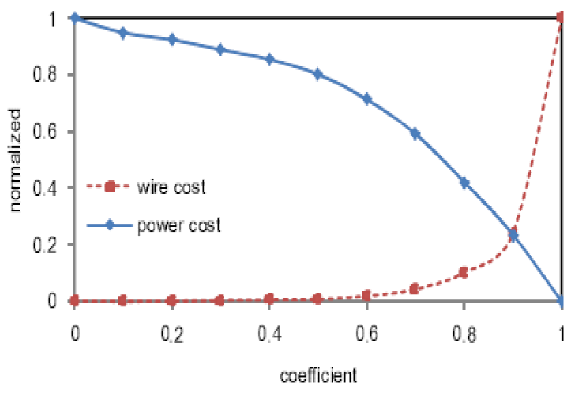


(a) wire and power cost for 4-layer partition

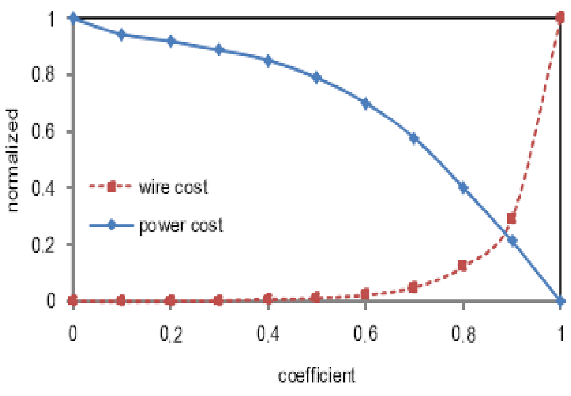


(b) wire and power cost for 5-layer partition

Figure 4.4: 24 scan chains for circuit bench2 (50K gates with 4095 FFs)



(a) wire and power cost for 4-layer partition



(b) wire and power cost for 5-layer partition

Figure 4.5: 108 scan chains for circuit bench7 (385K gates with 17983 FFs)

4.2 Enhancing energy-efficient task scheduling on 3D multi-core processors by dynamic remapping

4.2.1 Experimental setup

Our framework was implemented in C/C++ and executed on a Linux machine with a Pentium Core Duo (2.4 GHz) processor and 4 GB memory. The experiments were conducted on scheduling these task graphs from Standard Task Graph (STG) [44] on different 3D multi-core processors. We also implemented an ILP-based method for an optimal solution without considering the transmission costs between cores. The experimental result shows both two modified methods can outperform the ORI method and achieve comparable performance as the ILP. Especially, the energy-saving difference between IDR and ILP is only 2.54 percent on average. Later, we also compare the energy efficiency of our methods to the ORI method considering transmission cost. Experimental results also show that energy-saving rate from the proposed IDR algorithm is 16 percent higher than the ORI method in average. As a result, dynamic-remapping strategies are demonstrated to be more energy efficient than the fixed-mapping strategies.

4.2.2 Experimental result

4.2.2.1 Compare with Integer Linear Programming (ILP)

In this section, we cross compare the scheduling algorithms with the DR and IDR remapping strategies with an ILP-based method. We formulate each problem into an ILP instance without considering the transmission costs between cores.

We first tested our task scheduling algorithms and the ILP-based method to schedule task graphs with 50 tasks on 3D eight-core processors. The timing constraint was set from 1.2X to 1.5X Critical- Path Length (CPL) and 10 cases were randomly selected for each timing constraint. Table 2 shows the comparison of Energy-Saving Rate (ESR) of each method and the energy-saving-rate difference (△ESR) from three methods (ORI, DR and IDR) to the ILP-based method. The energy-saving-rate difference (△ESR) is defined as the difference between each respective methods and the ILP method as:

$$\Delta_{ESR} = \frac{ESR(ILP) - ESR(ORI, DR\ or\ IDR)}{ESR(ILP)} \times 100\% \qquad (20)$$

Table 4.7 demonstrates the efficiency of three methods. The energy-saving rate of ILP-based method is 55.24 percent on average. The energy-saving rate of ORI method is 48.60 percent and their difference is 12.40 percent on average. The energy-saving rate of DR method is 53.14 percent and their difference between DR and ILP is 3.82 percent on average. The energy-saving rate of IDR method is 53.87 percent and their between IDR and ILP is 2.54 percent on average. Experimental result shows our two dynamic remapping methods are superior to the baseline method by achieving a comparable performance as ILP. Besides, two proposed methods only take less than 1 second to complete, resulting in 300X to 11, 000X speedups when comparing to the ILP-based method.

timing	$ESR_{ILP}\%$	ESR_{ORI}	$(\Delta_{ESR})\%$	ESR_{DR}	$(\Delta_{ESR})\%$	ESR_{IDR}	$(\Delta_{ESR})\%$
$1.2CPL$	51.02	43.38	(14.77)	48.39	(5.03)	49.50	(2.95)
$1.3CPL$	56.20	49.98	(11.11)	53.79	(4.32)	54.66	(2.73)
$1.4CPL$	60.22	53.04	(11.79)	57.85	(3.97)	58.89	(2.18)
$1.5CPL$	63.41	59.25	(6.54)	61.90	(2.35)	62.50	(1.41)
average	**55.24**	**48.60**	**(12.40)**	**53.14**	**(3.82)**	**53.87**	**(2.54)**

Table 4.7: Energy-saving rate (ESR) and the ESR Difference (Δ_{ESR}) of different mapping strategies

4.2.2.2 Compare different strategies with transmission costs

In this section, we only compare the dynamic-remapping scheduling algorithms with the baseline algorithm when considering transmission cost. We further tested two dynamic remapping strategies on graphs with node size 50, 100, 300, 500 and 1000, and randomly selected 10 cases of each size. Table 4.8 shows the settings of 3D multi-core processors with transmission costs (α = 0, β = 2 and γ = 1). Timing constraints were set from 1.2X to 1.5X Critical-Path Length (CPL) for each case. For the DR and IDR methods, we set the initial timing constraint 1.05X CPL. For the IDR method, the timing constraint increases 0.1X CPL for each round of voltage scaling.

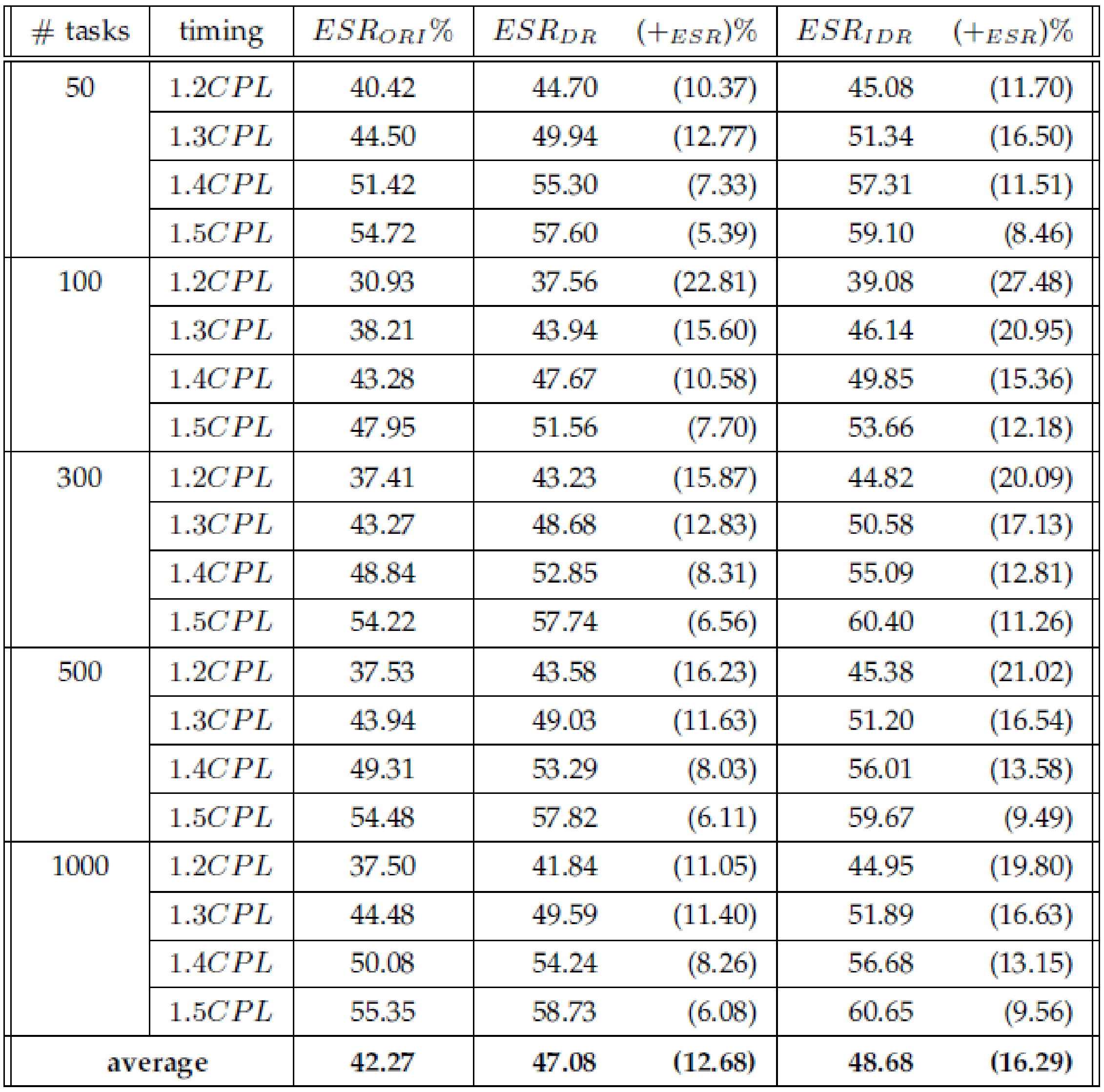
# Cores	# Layers	# Cores of each layer	# tasks
8	2	4	50 &100
16	4	4	300
24	3	8	500
48	3	16	1000

Table 4.8: Settings of 3D multi-core processors

Table 4.9 shows the Energy-Saving Rate (ESR) of each dynamic-remapping methods and the ESR improvement (+ESR) over the ORI method where the ESR improvement (+ESR) is defined as

$$+_{ESR} = \frac{ESR(DR\ or\ IDR) - ESR(ORI)}{ESR(ORI)} \times 100\% \qquad (21)$$

As shown in Table 4.9, the energy-saving rate of the ORI method is 42.27 percent. The energy-saving rate of the DR method is 47.08 percent and the energy-saving rate improvement between DR and ORI is 12.68 percent on average. The energy-saving rate of the IDR method is 48.68 percent and the energy-saving rate improvement between IDR and ORI is 16.29 percent on average. As a result, IDR is demonstrated to be the most energy-efficient scheduling algorithm.



# tasks	timing	$ESR_{ORI}\%$	ESR_{DR}	$(+_{ESR})\%$	ESR_{IDR}	$(+_{ESR})\%$
50	$1.2CPL$	40.42	44.70	(10.37)	45.08	(11.70)
	$1.3CPL$	44.50	49.94	(12.77)	51.34	(16.50)
	$1.4CPL$	51.42	55.30	(7.33)	57.31	(11.51)
	$1.5CPL$	54.72	57.60	(5.39)	59.10	(8.46)
100	$1.2CPL$	30.93	37.56	(22.81)	39.08	(27.48)
	$1.3CPL$	38.21	43.94	(15.60)	46.14	(20.95)
	$1.4CPL$	43.28	47.67	(10.58)	49.85	(15.36)
	$1.5CPL$	47.95	51.56	(7.70)	53.66	(12.18)
300	$1.2CPL$	37.41	43.23	(15.87)	44.82	(20.09)
	$1.3CPL$	43.27	48.68	(12.83)	50.58	(17.13)
	$1.4CPL$	48.84	52.85	(8.31)	55.09	(12.81)
	$1.5CPL$	54.22	57.74	(6.56)	60.40	(11.26)
500	$1.2CPL$	37.53	43.58	(16.23)	45.38	(21.02)
	$1.3CPL$	43.94	49.03	(11.63)	51.20	(16.54)
	$1.4CPL$	49.31	53.29	(8.03)	56.01	(13.58)
	$1.5CPL$	54.48	57.82	(6.11)	59.67	(9.49)
1000	$1.2CPL$	37.50	41.84	(11.05)	44.95	(19.80)
	$1.3CPL$	44.48	49.59	(11.40)	51.89	(16.63)
	$1.4CPL$	50.08	54.24	(8.26)	56.68	(13.15)
	$1.5CPL$	55.35	58.73	(6.08)	60.65	(9.56)
average		**42.27**	**47.08**	**(12.68)**	**48.68**	**(16.29)**

Table 4.9: Comparison of energy-saving rates (ESRs) and ESR improvement (+ESR)

Chapter 5 Conclusion

The scan-chain ordering problem is formulated as TSP problem and modified for 3D IC designs considering TSV constraints. A fast scan-chain ordering algorithm is developed, consisting of two stages: (1) initial solution computation; and (2) local refinement and constraint solving. To avoid high complexity of 3D optimization, we convert such a problem into a TSP problem and use a state-of-the-art algorithm, a multiple fragment heuristic, combined with a dynamic closest-pair data structure named FastPair to quickly derive a satisfactory initial solution. Two techniques, 3D planarization and 3D relaxation are also proposed to minimize the wire cost, (power cost or combined cost) and to relax the total number of TSVs in use, respectively. Experimental results show that the proposed algorithm can achieve comparable (<3 percent difference) or even better performance than that from a GA method, resulting in run time speeds at least two-orders faster on all ISCAS'89 benchmark circuits when considering TSV constraints. Moreover, our algorithm is shown to successfully scale to multiple scan chains for large designs from [23]. Therefore, the proposed algorithm can be practically used for scan-chain ordering of 3D IC designs.

As a result, the contributions of this work can be summarized as:

- Formulate scan-chain ordering considering TSV constraints into a modified TSP problem.
- Propose a greedy algorithm for scan-chain ordering of 3D-IC designs to simultaneously minimize wire and power costs.
- Demonstrate that the proposed algorithm can be practically used while supporting multiple scan chains.

To achieve high-performance computing on embedded systems, three-dimensional (3D) multi-core processors have become a promising alternative where energy efficiency is crucial to its success. Many heuristics applying Dynamic Voltage and Frequency Scaling (DVFS) techniques were proposed for energy minimization. However, most of the previous works were built upon a fixed task-to-core mapping where many slack spaces can be further improved. Therefore, in this work, we propose two dynamic remapping strategies to enhance an energy-aware task-scheduling algorithm considering transmission cost.

The other issue in this work, we apply two dynamic task-to-core remapping strategies, DR and IDR, on top of a baseline task-scheduling algorithm from [41].

Experimental results show the solutions from the two task-to-core remapping strategies are more close to the ILP solution where IDR only has 2.54 percent difference. Besides, two proposed methods can run 300X to 11,000X faster than the ILP method. Our experiments also show that the energy-saving rate of the IDR method can achieve 16 percent higher than that of the baseline algorithm on average. As a result, the scheduling algorithm with dynamic task-to-core remapping can result in more energy efficient scheduling than that with a fixed task-to-core mapping.

Based on such result, the scheduling algorithm with dynamic task-to-core remapping can result in more energy efficient scheduling than that with a fixed task-to-core mapping. We also implemented an ILP-based method for an optimal solution without considering the transmission costs between cores. The experimental results show IDR strategy can achieve comparable performance as the ILP and the energy-saving difference between IDR and ILP is only −2.54 percent on average. However, in our future work, we would like to further validate our IDR strategy considering transmission costs between cores. Hence, the problem is formulated into Quadratic Programming (QP) and compared to the IDR solution considering the transmission costs between cores. However, the scalability problem of QP-based method is more sever with transmission costs. Therefore, scheduling with a larger task graph is implemented into Genetic Algorithm (GA) and also compared to our IDR solution.

References

[1] J. W. Joyner, P. Zarkesh-Ha, J. A. Davis, and J. D. Meindl, “A three-dimensional stochastic wire-length distribution for variable separation of strata,” in Interconnect Technology Conference, 2000. Proceedings of the IEEE 2000 International, 2000, pp. 126 –128.

[2] J. Joyner and J. Meindl, “Opportunities for reduced power dissipation using three-dimensional integration,” in Interconnect Technology Conference, 2002. Proceedings of the IEEE 2002 International, 2002, pp. 148 – 150.

[3] C. Ababei, Y. Feng, B. Goplen, H. Mogal, T. Zhang, K. Bazargan, and S. Sapatnekar, “Placement and routing in 3d integrated circuits,” Design Test of Computers, IEEE, vol. 22, no. 6, pp. 520 – 531, nov.-dec. 2005.

[4] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon, “Demystifying 3d ics: the pros and cons of going vertical,” Design Test of Computers, IEEE, vol. 22, no. 6, pp. 498 – 510, nov.-dec. 2005.

[5] Y. Xie and Y. Ma, “Design space exploration for 3d integrated circuits,” in Solid-State and Integrated-Circuit Technology, 2008. ICSICT 2008. 9th International Conference on, 20-23 2008, pp. 2317 –2320.

[6] B. Noia, K. Chakrabarty, and E. J. Marinissen, “Optimization methods for post-bond die-internal/external testing in 3d stacked ics,” in INTERNATIONAL TEST CONFERENCE (ITC), 2010.

[7] M. Hirech, J. Beausang, and X. Gu, “A new approach to scan chain reordering using physical design information,” in Test Conference, 1998. Proceedings., International, 18-23 1998, pp. 348 –355.

[8] S. Makar, “A layout-based approach for ordering scan chain flip-flops,” in Test Conference, 1998. Proceedings., International, 18-23 1998, pp. 341 –347.

[9] P. Gupta, A. Kahng, and S. Mantik, “Routing-aware scan chain ordering,” in Design Automation Conference, 2003. Proceedings of the ASP-DAC 2003. Asia and South Pacific, 21-24 2003, pp. 857 – 862.

[10] S. Remersaro and X. et al., “Preferred fill a scalable method to reduce capture power for scan based designs,” in Proc. Int. Test Conf., 2006.

[11] S. R., R. Oruganti, and N. Touba, “Static compaction techniques to control scan vector power dissipation,” in VLSI Test Symposium, 2000. Proceedings. 18th IEEE, 2000, pp. 35 –40.

[12] P.Rosinger and B.-H. et al., “Scan architecture with mutually exclusive scan segment activation for shift and capture power reduction,” Proc. Transactions on Computer-Aided Design(TCAD), pp. 1142–1153, 2004.

[13] Y. Bonhomme, P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch, and A.

Virazel, “Design of routing-constrained low power scan chains,” in DATE’04: Proceedings of the conference on Design, automation and test in Europe. Washington, DC, USA: IEEE Computer Society, 2004, p. 10062.
[14] X.-L. Huang and J.-l. Huang, “A routability constrained scan chain ordering technique for test power reduction,” in Design Automation, 2006. Asia and South Pacific Conference on, 24-27 2006, p. 5 pp.
[15] B. Paul, R. Mukhopadhyay, and I. Gupta, “Genetic algorithm based scan chain optimization and test power reduction using physical information,” in TENCON 2006. 2006 IEEE Region 10 Conference, 14-17 2006, pp. 1 –4.
[16] Y.-Z. Wu and M.-T. Chao, “Scan-chain reordering for minimizing scan-shift power based on non-specified test cubes,” in VLSI Test Symposium, 2008. VTS 2008. 26th IEEE, april 2008, pp. 147 –154.
[17] X. Wu, P. Falkenstern, K. Chakrabarty, and Y. Xie, “Scan-chain design and optimization for three-dimensional integrated circuits,” J. Emerg. Technol. Comput. Syst., vol. 5, no. 2, pp. 1–26, 2009.
[18] D. Lewis and H.-H. Lee, “A scan island based design enabling prebond testability in die-stacked microprocessors,” in Test Conference, 2007. ITC 2007, IEEE International, oct. 2007, pp. 1 –8.
[19] A. Kumar, S. Reddy, I. Pomeranz, and B. Becker, “Hyper-graph based partitioning to reduce dft cost for pre-bond 3d-ic testing,” in Design, Automation Test in Europe Conference Exhibition (DATE), 2011, March 2011, pp. 1 –6.
[20] J. L. Bentley, “Experiments on traveling salesman heuristics,” in SODA ’90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1990, pp. 91–99.
[21] D. Eppstein, “Fast hierarchical clustering and other applications of dynamic closest pairs,” J. Exp. Algorithmics, vol. 5, p. 1, 2000.
[22] C. Giri, S. K. Roy, B. Banerjee, and H. Rahaman, “Scan chain design targeting dual power and delay optimization for 3d integrated circuit,” Advances in Computing, Control, and Telecommunication Technologies, International Conference on, vol. 0, pp. 845–849, 2009.
[23] “Taiwan integrated circuit computer-aided design (ic/cad) contest, http : //cad contest.cs.nctu.edu.tw/cad11/,” 2011.
[24] 'International Technology Roadmap for Semiconductors (ITRS)', http://www.itrs.net/
[25] Kahle, J. and Day, M. and Hofstee, H. and johns, C. and Maeurer, T.and Shippy, D.: 'Introduction to the Cell multiprocessor', IBM Journal of Research and Development, July, 2005, 49, (4/5), pp. 589-604

[26] Hung, W.-L. and Link, G. M. and Xie, Y. and Vijaykrishnan, N. and Irwin, M. J.: 'Interconnect and Thermal-aware Floorplanning for 3D Microprocessors'. Proc. Int. Symposium on Quality Electronic Design, Washington, DC, USA, 2006, pp. 98-104
[27] Puttaswamy, K.: 'Thermal Analysis of a 3D Die-Stacked High-Performance Microprocessor'. Proc. GLSVLSI, 2006
[28] Bakir, M.S. and King, C. and Sekar, D. and Thacker, H. and Dang, B. and Huang, G. and Naeemi, A. and Meindl, J.D.: '3D heterogeneous integrated systems: liquid cooling, power delivery, and implementation'. Proc. IEEE Custom Integrated Circuits Conference, 2008
[29] Cong, J. and Wei, J. and Zhang, Y.: 'A thermal-driven _oorplanning algorithm for 3D ICs'. Proc. IEEE Int. Conf. on Computer-Aided Design, 2004
[30] Cong, J. and Zhang, Y.: 'Thermal via planning for 3-D ICs'. Proc. IEEE Int. Conf. on Computer-Aided Design, 2005
[31] Raje, S. and Sarrafzadeh, M.: 'Variable voltage scheduling'. Proc. Int. symposium on Low power design, Dana Point, California, United States, 1995, pp. 9-14
[31] Zhou, X. and Xu, Y. nd Du, Y. and Zhang, Y. and Yang, J.: 'Thermal management for 3D processors via task scheduling'. Proc. Int. Conf. on Parallel Processing, 2008
[32] Sun, C. and Shang, L. and Dick, R.P.: 'Three-dimensional multi-processor system-on-chip thermal optimization'. Proc. Int. Conf. Hardware/Software Codesign and System Synthesis, 2007
[33] Chandrakasan, A. and Brodersen, R.: 'Low Power Digital CMOS Design'. Kluwer Academic Publishers, 1995
[34] Benini, L. and Bogliolo, A. and Micheli, G. D.: 'A survey of design techniques for system-level dynamic power management', IEEE Transactions on VLSI systems, June, 2000, pp. 299-316
[35] Hong, I. and Kirovski, D. and Qu, G. and Potkonjak, M. and Srivastava, M.: 'Power optimization of variable voltage core-based systems'. Proc. IEEE Design Automation Conference, 1998
[36] Yao, F., Demers, A. and Shenker, S.: 'A scheduling model for reduced CPU energy'. Proc. Int. Foundations of Computer Science, Oct 1995, pp. 374-382
[37] Ishihara, T. and Yasuura, H.: 'Voltage scheduling problem for dynamically variable voltge processors'. Proc. Int. Symposium on Low Power Electronics and Design, 1998
[38] Zhang Y. and Hu X. and Chen D.: 'Task scheduling and voltage selection for energy minimization'. Proc. IEEE Design Automation Conference, 2002
[39] Lin, Y.-R., Hwang, C.-T. and Wu, Allen C.-H.: 'Scheduling techniques for variable voltage low power designs', ACM Trans. Des. Autom. Electron. Syst., 1997,

2, (2), pp. 81-97

[40] Wu, C.-B., Lin, Y.-L.: 'Energy-Ef_cient Task Scheduling for DVFS-Multi-Core 3D IC'. Master thesis, National Tsing Hua University, 2010

[41] Gajski, D. D. and Dutt, N. D. and Wu, A. C.-H. and Lin, S. Y.-L.: 'Scheduling','High-Level Synthesis: Introduction to Chip and System Design'(Springer, 1992, 1st edn.), pp. 213-258

[42] Adam, T. L. and Chandy, K. M. and Dickson, J. R.: 'A comparison of list schedules for parallel processing systems'. ACM, Commun. 1974, 17, (12), pp. 685-690

[43] 'Standard Task Graph Set, Kasahara Laboratory Department of Electrical, Electronics and Computer Enginnering, Waseda University', http://www.kasahara.elec.waseda.ac.jp/schedule/index.html

[44] Chandrakasan, A. P. and Potkonjak, M. and Mehra, R. and Rabaey, J. and Brodersen, R.: 'Optimizing power using transformations', IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems., Jan, 1995, 14, (1), pp. 12-31

國科會補助計畫衍生研發成果推廣資料表

日期:2011/10/30

國科會補助計畫	計畫名稱: 子計畫五：應用在驗證與測試3D IC整合過程中以計算智慧為基礎的測試向量產生方法(2/2)	
	計畫主持人: 溫宏斌	
	計畫編號: 99-2220-E-009-039-	學門領域: 晶片科技計畫--整合型學術研究計畫
無研發成果推廣資料		

99 年度專題研究計畫研究成果彙整表

計畫主持人：溫宏斌	計畫編號：99-2220-E-009-039-
計畫名稱：針對 3D 整合之電子設計自動化技術開發--子計畫五：應用在驗證與測試 3D IC 整合過程中以計算智慧為基礎的測試向量產生方法(2/2)	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	1	1	100%		
		研討會論文	4	4	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	2	2	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	1	1	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	8	8	100%	人次	
		博士生	1	1	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	1. 陳韋廷、張家慶, U-tools 展示：3D-MFH, 2011. 2. W.T. Chen, C.C. Chiang, 3D-MFH, University Booth, IEEE/ACM Design Automation Conference, 2011.

🟥	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

國科會補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

   ■達成目標

   □未達成目標（請說明，以 100 字為限）

   　　□實驗失敗

   　　□因故實驗中斷

   　　□其他原因

   說明：

2. 研究成果在學術期刊發表或申請專利等情形：

   論文：■已發表 □未發表之文稿 □撰寫中 □無

   專利：□已獲得 ■申請中 □無

   技轉：□已技轉 □洽談中 ■無

   其他：（以 100 字為限）

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）（以 500 字為限）

   A.學術成就

   1.提出針對 scan-chain ordering 在考慮降低繞線長度或是功能消耗的演算法。

   2.在配合 DVFS(動態電壓頻率規劃)的技術下，設計一套 task scheduling 的演算法來降低 IC 的消耗能量。

   3.目前有四篇論文發表在國內會議中（The 22nd VLSI Design / CAD Symposium 2 篇，2011 VLSI Test Technology Workshop）

   4.另外有兩篇論文已投稿至國際期刊(IEEE TVLSI, IET Computer and Digital Techniques)

   三維積電路比傳統的二維積體電路有著許多顯著的優勢，但三維積體電路比二維積體電路面臨了更嚴重的熱問題，因為我們提出了 task scheduling 的演算法並配合 DVFS(動態電壓頻率規劃)的技術，調整每個 task 執行的電壓頻率，進而平衡三維積體電路中每一層的溫度分佈。

   B.技術創新：

   良率問題在三維積體電路中極為重要，而良率與製造成本息息相關，因此 Design for Testability 在三積體電路中更是不可或缺的一項技術，scan-chain based 的測試是 DFT 中常見的方法，我們提出了快速的演算法來幫助我們決定 scan-chain 的順序，用來降低繞線長度以及功能消耗。

   C.社會影響：

培養學生在三維積體電路這一領域的專業知識，以期畢業後可為業界注入新血。