計算機アーキテクチャ　98－13
設　計　自　動　化　65－13
（1993. 1. 21）

専用システムのためのHW/SW協調再設計の一手法

中村秀一　　安浦 寛人

九州大学 大学院総合理工学研究科 情報システム学専攻

〒816 春日市春日公園6-1

E-mail: {nakamura, yasuura}@is.kyushu-u.ac.jp

あらまし

HW(ハードウェア)/SW(ソフトウェア)協調設計(HWとSWの両方を同時に変更しつつ，目的にあったシステムを設計する)の一手法を提案する．本手法では，初期仕様として与えられたシステム(HW+SW)の記述から，ユーザの与える制約条件に適したシステムをインクリメンタルに得ることが可能となる．本報告ではHW/SW協調設計によるシステム再設計について，設計例を通してHW/SWトレードオフの決定機構と，本手法に要求される要素技術について考察する．

和文キーワード：VLSI，ASIC，ハードウェア/ソフトウェア協調設計，高位論理合成，

HW/SW Co-Operated-Redesign Method for Application Specific Systems

Shuichi Nakamura　Hiroto Yasuura

Department of Information Systems
Interdisciplinary Graduate School of Engineering Sciences
Kyushu University
Kasuga-shi, Fukuoka 816 Japan

E-mail: {nakamura, yasuura}@is.kyushu-u.ac.jp

Abstract

We propose a new method for HW(Hardware)/SW(Software) Co-Operated-Redesign with changing both hardware and software to be satisfied specifications under given constraints. In our method, hardware and software are improved incrementally from initialized HW and SW preserving its function. In this paper,we discuss a decision mechanism of HW/SW trade-off and design techniques required to this method through an example of Co-Operated-Redesign.

英文 key word: VLSI, ASIC, Hardware/Software Co-Design, High-level synthesis

1 はじめに

本来，システムの設計は，その構成要素である SW(ソフトウェア) と HW(ハードウェア) を総合的に設計し，両者の均整のとれたシステムを構築するのが理想である．しかし，今までは，HW(ハードウェア) のコストが高いために高速性が要求される分野以外では専用の HW は用いられず，汎用のプロセッサ上で SW の可変性を利用したシステムの構築がなされてきた．ところが，最近の CAD 技術，プロセス技術の発達により，HW 開発にかかるコストは低減しつつあり，さまざまなユーザの目的に適した ASIC（Application Specific IC）を低コストで実現することが可能となってきた．特に，HW 設計における CAD ツールの進歩により，現在は HW の論理合成からレイアウトまでの作業が，ほぼ自動化され，HW の設計も SW と同じような可変性を持つようになってきた．

上記のことから，システムの設計環境は，従来の HW が固定され，SW(ソフトウェア) で各ユーザの目的に対応していた状況から，SW にあわせて HW をも変更し，目的にあったシステムを構築するという状況になることが予想される．このような設計手法においては，HW と SW の境界をどのように決定するかがシステム設計の重要課題となる．我々は HW/SW 協調設計をこのような枠組の中での新しい設計手法として位置付ける．

本報告では，2 章で HW/SW 協調設計の一般的枠組を議論し，3 章ではここで提案する設計手法の概要，4 章で本手法を適用した設計例，5 章で HW/SW 協調設計についての利点や問題点を議論しする．

2 HW/SW 協調設計

2.1 HW/SW 協調設計の現状

従来の HW 設計では，特定の SW を意識せずに HW 設計がなされる．HW/SW 協調設計では，使用される SW に応じた HW 設計がなされ，システム全体の最適化が図られる．従来の HW 設計の流れと HW/SW 協調設計の流れを図 1に示す．

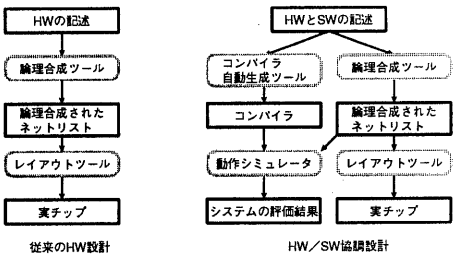

図 1: 従来の HW 設計と HW/SW 協調設計の流れ

HW/SW 協調設計が成立するためには以下のような技術の確立が必要とされる．

1. 実行またはシミュレーションによる検証が可能な HW と SW の記述手法．
2. HW の記述からの HW 自動生成技術．
3. HW の設計変更に応じた SW 用のコンパイラの自動生成技術．

これらの技術が確立してはじめて HW/SW 協調設計が可能となる．ここで，HW/ SW 協調設計に必要な技術の現状について述べる．

1. の HW と SW の記述手法については，すでに技術が確立しつつある．HW 記述は HDL(ハードウェア記述言語) の研究が各方面でなされており [1]，これらの記述によって動作レベルや論理レベルのシミュレーションが可能となっている．SW 記述は様々な高級言語が存在する．

2. の HW 自動生成技術については，下位の階層のツールから徐々に自動化され，現在は，論理合成からレイアウトまではほぼ自動化されており，HW の記述さえあれば，チップを実現することができる．

3. の設計変更に応じたコンパイラの自動生成技術については，現在各方面で研究が始まっており，近い将来成果をあげるものと思われる．[2] [3]

1. と 2. はすでに確立されつつあり，3. が確立されれば，HW/SW 協調設計が可能となる．

2.2 HW/SW 協調設計への課題

HW も SW もあるアルゴリズムにしたがって，一連の処理を行う．ただし，HW は手順の決まった一連の処理を高速に行うがそれ以外の処理は行うことができない．SW はある細粒度の処理を組み合わせて一連の処理を行うことができ，組み合わせ方によって多様な処理を行うことができるが，高速化の点では HW に劣る．マイクロプロセッサを用いたシステムでは，HW と SW のそれぞれの利点を生かして，システムの中で処理が決まっており，高速化したい部分は HW で，処理が可変な部分は SW で実現する．ここで重要であるのは，どの部分を HW で実現し，どの部分を SW で実現するかの切りわけである． 最近の CAD ツールの中には，SW を HW 化するものもある [4] [5] [6] [7]．しかし，これらのツールでは，SW は完全に HW 化されるため，その用途にしか使用されない HW を作成するためには有効であるが，SW によって可変な処理が行えることが重要であるマイクロプロセッサシステムには，有効ではない．したがって，マイクロプロセッサシステムをユーザの目的にあわせて変更するには，SW のある部分のうち HW 化したい部分だけ HW 化し，HW 化したくない可変な部分を SW として残すことができなければならない．

すなわち，HW/SW 協調設計には
「HW と SW を分ける境界を，任意の制約条件によって決定できること」
が要求される．

3 HW/SW 協調再設計手法

HW/SW 協調設計には 2 通りの手法が考えられる．まず 1 つは HW と SW を統一的な記述の上で取り扱い，その上で境界を決めるという手法であり，もう 1 つは HW と SW を別々な記述で行い，双方の情報を基礎に再設計を行って境界を決めるという手法である．前者では HW と SW を統一的に取り扱う新しい記述法が必要であり，かつ制約条件によって設計される HW と SW に柔軟性がありすぎ，境界によってシステムの HW と SW がドラスティックに変化する．後者では既存の HW と SW の記述法を用いればよく，それぞれの情報によって HW と SW をインクリメンタルに変化させることができる．

ここで提案する手法では，HW と SW を別々な記述の上で取り扱う．

本手法では既存の HW と SW の記述と設計に対する制約条件を入力とし，制約条件を満たすような HW と SW によるシステムを出力とする．また，HW と SW の記述法や，HW の記述からの HW 自動合成技術，HW の変更にともなう SW 用のコンパイラ自動生成技術が確立されており，本手法による設計に使用できることを仮定する．以下に本手法の設計手順を示す．

Step 1. プロセッサの初期仕様と目的にあった SW を用いてシステムの機能・動作を確認しながらシステムの仕様を決定．

Step 2. HW，SW に対する切りわけの制約条件を決定．

Step 3. SW の各ブロックの頻度情報や並列性を抽出．

Step 4. SW の情報によって HW ユニットを変更．

Step 5. 各 HW ユニットを組み合わせた場合の HW 量，実行時間の概略を評価．制約条件を満たさない場合は Step 4. に戻る．(再設計ループ)

Step 6. 組合せの中から制約条件に適した候補を選出し，全体の HW 合成を行い，正確に評価．また，HW の変更にともなって SW の変更部分の情報を提供．

3.1 Step 1. の概要

Step 1. では，初期仕様の HW に汎用プロセッサと SW に汎用プログラミング言語で書かれたプログラムを用いて，システムの機能・動作を早期に確認しながらシステムの仕様を決定する．ここで，初期仕様となる HW には，最新の技術で作られた性能が十分に良いものを公開された情報として用意する．

3.2 Step 2. の概要

Step 2. では，システムに対する HW と SW の境界の移動方向を決定する．システムの初期仕

様に対して HW の領域を大きくする場合と SW の領域を大きくする場合の 2 通りに分けて再設計を進める．(図 2)

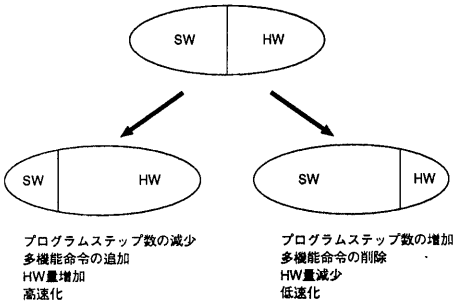

図 2: HW と SW の切りわけ

HW の領域を大きくする場合は目的の SW の実行時間 T(秒) を T_0(秒) 以下にすることを第一の制約条件とし，このとき，HW 量 HW(ゲート) は守らねばならない下限 HW_1(ゲート) 以下であればよいことにする．HW 量もできるだけ少ない方が良いわけであるが，設計時間の短縮のため，設計の途中であっても時間の制約条件を満たしており，かつ HW 量が上限 HW_0(ゲート) 以上を満たせば設計を終了することとする．すなわち，$T \leqq T_0$かつ $HW_0 \leqq HW \leqq HW_1$であれば設計を終了する．

また，SW の領域を大きくする場合は目的の HW 量を HW_0(ゲート) 以下にすることを第一の制約条件とし，このとき，SW の実行時間 T(秒) は守らねばならない下限 T_1(秒) 以下であればよいことにする．また，同様に設計の途中であっても HW 量の制約条件を満たしており，かつ実行時間が上限 T_0(秒) 以上を満たせば設計を終了することとする．すなわち，$HW \leqq HW_0$かつ $T_0 \leqq T \leqq T_1$であれば設計を終了する．また，規定しない限り，初期仕様の各処理機能を実現方法が異なっても再現できなければならない．すなわち，HW の削減によって，ある処理を行うことができなくなってはならない．このことは必ずしも同じ機能ユニットを使用するということではない．例えば，初期仕様が HW に乗算器を持っていて，HW を削減するために乗算器を削除したとしても，加算やシフト機能を用いれば乗算の処理は再現できる．

3.3 Step 3. の概要

Step 3. では，プログラムの中から HW の変更に必要な情報を抽出する．抽出する情報は使用するレジスタ数やデータフローグラフ，命令の頻度等がある．

3.4 Step 4. の概要

Step 4. では，SW から抽出した反復情報や並列性の情報に基づいて HW の変更を行う．HW の変更は制約条件が満たされるまで行われる．制約条件のうち主なものは「時間」と「HW 量」であるので，高速化を目的とする場合は,「時間」をパラメータにとって，HW の変更・評価を行い，「時間」の制約が満たされることを再設計終了の条件とする．また HW 量の削減を目的とする場合は,「HW 量」をパラメータにとって，HW の変更・評価を行い,「HW 量」の制約が満たされることを再設計終了の条件とする．

3.5 Step 5. の概要

Step 5. では，HW ユニットの評価を行う．ここでは，機能ユニットのクリティカルパスの評価やトレードオフを考慮する．ここで必要となる技術は機能ユニットの HW 量や動作速度を評価するツールである．ハイレベルシンセシスツールの進歩により，この部分は比較的簡単に設計・検証ができるようになった．また，各 HW ユニットを組み合わせた場合の HW 量とプログラムステップの概算を行う．

3.6 Step 6. の概要

Step 6. では，概算から制約条件に適した組合せの候補を選出し，HW 全体の合成を行い，正確に評価する．また，HW の変更にともなう SW の変更部分の情報を提供し，SW を変更する．

3.7 再設計ループ

再設計時にはStep 4からStep 5を制約条件が満たされるまで繰り返し行う．HW領域を大きくする，すなわち高速化を目的とする再設計時には次にあげる高速化手法を順に適用して再設計ループを繰り返す．

- A. SWのHW化によるプログラムステップ数の削減．
- B. 並列化．
- C. 動作周波数の向上．

これらの高速化手法のうち，動作周波数の向上は，デバイス技術や機能ユニット(機能ユニット)の性能(遅延時間)によって大きく左右される要素であり，本手法では機能ユニットの性能のみ考慮する．本手法では制約条件を満たすまでA.をまず適用し，A.の適用のみでは目標とする性能を達成できないときはB.を適用する．

また，SWの領域を大きくする，すなわちHW量の削減を目的とする再設計には次の手法を適用する．

- A'. HWのSW化によるHW量の削減．
- B'. レジスタの削減．
- C'. 機能ユニットの規模の削減．

本手法では制約条件を満たすまでA'.をまず適用し，A'.の適用のみでは目標とする性能を達成できないときはB',C'を順次適用する．

4 本手法による設計例–FFT用プロセッサ–

本手法によるFFT(Fast Fourier Transform: 高速フーリエ変換)用プロセッサ設計の例を示す．今回の再設計においては初期仕様のHW記述にはASIC Design System[8][9]上でのHW記述とし，論理合成をASIC Design System上で行い，その出力からHW量と動作周波数を算出した．SWの初期仕様はコンパイラ自動生成ツールがないため，アセンブリ言語とし，SWからの情報抽出を人手で行った．また，HWの変更によるSW変更はハンドコンパイルし，実行時間のシミュレーションも人手で行った．

4.1 初期仕様

今回のシステム設計に対してあらかじめ設計されている汎用プロセッサとアセンブリ言語によるFFTプログラムを初期仕様として与える．ここで，その諸元等を示す．もととなる汎用プロセッサはASIC Design System上で記述されたDLX(文献[10]参照)に準じた仕様である．FFTは離散フーリエ変換(式-1)を高速に計算するアルゴリズムであり，その処理の流れを図3に示す．

$$y_i = \sum_{k=0}^{n-1} x_k w_n^{ik} \quad (式-1)$$

表1: もととなる汎用プロセッサの諸元

動作周波数	22.34 MHz
ゲート数	32896ゲート 2入力NAND換算
汎用レジスタ	32bit,32本
1024点FFT 実行時間	200821(ステップ) 8.98 ms
算術演算HW	加算器, 左右1bitシフタ, 乗算器

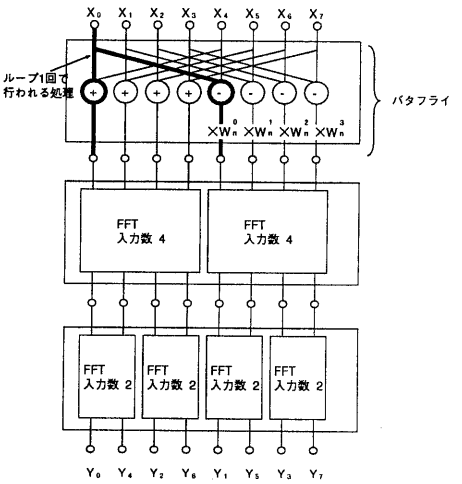

図3: FFTアルゴリズム(8点FFT)

4.2 制約条件の設定

今回の再設計では制約条件として以下のものを挙げる.
制約条件α . 高速化を目的とした再設計.

1. トランジスタ数の上限 (70,000 ゲート) ・下限 (50,000 ゲート)
2. プログラム実行時間の上限 (5msec 以内)

制約条件β . HW 量の削減を目的とした再設計.

1. トランジスタ数の上限 (15,000 ゲート)
2. プログラム実行時間の上限 (無限)・下限 (500msec 以内)

4.2.1 制約条件α . の下での再設計

制約条件α . の下で高速化を目的とした再設計を行った. まず, A. の SW の HW 化によるプログラムステップ数の削減を行った. 今回 FFT プログラムから抽出した命令列の中で, メモリアクセスのためのインデックスの計算にデータ依存関係があることが判明した. そこで, データ依存関係のため生じる遅延をなくすことによって高速化を行う. 具体的には右シフトの後に加算を実行する部分は依存関係による遅延があり, 1 命令化した. (図 4)

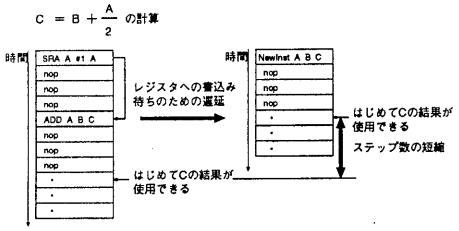

図 4: 2 命令の 1 命令化

また, ロード命令のインデックス修飾もレジスタ 2 つで行うように変更した. このことにより, 表 2のような性能向上が得られた.

しかし, ここまでの改良で目標とする性能を得られなかったので, 次の B. の並列化による高速化を行った. この並列化は VLIW 形式で行い,

表 2: A.SW の HW 化によるプログラムステップ数の削減

	もとの仕様	A. の変更後
HW 量	32896 ゲート	33442 ゲート
レジスタ数	32 本	32 本
動作周波数	22.34 MHz	22.37MHz
SW ステップ数	200821	175181
SW 実行時間	8.98 ms	7.69 ms

HW の変更点
右シフト後に加算を行う機能ユニットを追加 メモリアクセス時のインデックス修飾を 2 つのレジスタとイミィディエイトで行う

表 3: B. 並列化によるプログラムステップ数の削減

	もとの仕様	B. の変更後
HW 量	32896 ゲート	64753 ゲート
レジスタ数	32 本	32 本
パイプライン数	1 本	2 本
動作周波数	22.34 MHz	22.02MHz
SW ステップ数	200821	78373
SW 実行時間	8.98 ms	3.56 ms

HW の変更点
2 本のパイプラインで同時に 2 命令が実行可 (ただし, 分岐を除く) 4READ2WRITE のレジスタファイル

データフローグラフから並列度を求め, HW 量の制約条件を満たすようなパイプラインの数を決定した. 結果として, 表 3のような性能向上が得られた.

4.2.2 制約条件β. の下での再設計

制約条件β. の下で HW 量の削減を目的とした再設計を行った. まず, A'. の HW の SW 化による HW 量の削減を行った. 今回は乗算の SW 化を行い, 表 4のような結果が得られた.

しかし, ここまでの改良で目標とする性能を得られなかったので, 次に B'. のレジスタの削減による HW 量の削減を行った. 今回は HW 量の制約条件を満たすまでレジスタの削減を行い, 表 5

表 4: A'.HW の SW 化による HW 量の削減

	もとの仕様	A'. の変更後
HW 量	32896 ゲート	26806 ゲート
レジスタ数	32 本	32 本
乗算器	あり	なし
動作周波数	22.34 MHz	30.74MHz
SW ステップ数	2008221	11669621
SW 実行時間	8.98 ms	377.0 ms
HW の変更点		
乗算器の削除		

表 5: B'. レジスタの削減による HW 量の削減

	もとの仕様	B'. の変更後
HW 量	32896 ゲート	14302 ゲート
レジスタ数	32 本	8 本
乗算器	あり	なし
動作周波数	22.34 MHz	30.89MHz
SW ステップ数	200821	11763916
SW 実行時間	8.98 ms	381.1 ms
HW の変更点		
レジスタの削減		

のような結果が得られた.

表 6に制約条件α , β の下で再設計されたプロセッサの諸元を示す.

5 HW/SW 協調設計の問題点

今回の再設計を通して以下のような問題点が明らかになった.

1. HW/SW トレードオフ

HW 設計コストの減少にともなって, HW と SW との境界を揺らしながら, 実行する SW に適した HW 構成を, HW/SW 協調設計によって得られる. しかし, HW/SW 協調設計についてトレードオフをどのようにして決定するかは今後の大きな研究課題である.

2. 再設計方針の組合せ問題

再設計時においてもっとも重要であると考えられるのは, システムの再設計にあたっての方針の組合せによる変更方針の選択である. 例えば, 高速化を目的とする場合, SW の HW 化をせずに, 並列化のみを行った場合が制約条件を満たすこともあり得る. また, HW 量の削減を目的とする場合,

表 6: 各制約条件による再設計結果

もとの仕様	
命令	DLX に準じた 命令セット
HW 構成	パイプライン 1 本 2READ1WRITE の レジスタファイル 32 本 1 レジスタによるアドレス修飾 ALU, 乗算器, シフタ
HW 量	32896 ゲート
動作周波数	22.34 MHz
FFT プログラム 実行時間	200821 ステップ 8.98 ms
制約条件α. での再設計後	
命令	shift,add を 1 命令化した 命令を追加
HW 構成	パイプライン 2 本 (うち 1 つは分岐機能なし) 4READ2WRITE の レジスタファイル 32 本 2 レジスタによるアドレス修飾 ALU, 乗算器, シフタ 右シフト後の加算器
HW 量	64753 ゲート
動作周波数	22.02 MHz
FFT プログラム 実行時間	78373 ステップ 3.56 ms
制約条件β. での再設計後	
命令	乗算命令を SW ルーチン化 (1 命令→ 25 命令) (1 ステップ→ 561 ステップ)
HW 構成	パイプライン 1 本 2READ1WRITE の レジスタファイル 8 本 ALU, シフタ
HW 量	14302 ゲート
動作周波数	30.89 MHz
FFT プログラム 実行時間	11763916 ステップ 381.1 ms

HW の SW 化をせずに，レジスタの削減のみを行った場合が制約条件を満たすこともあり得る．したがって，このような方針の組合せの中で最良のものを短時間で発見できなければならない．

3. HW の変更にともなう SW の変更情報

HW の変更にともなって SW の変更がなされなければ，HW/SW 協調設計とはいえない．HW の変更情報を SW が利用できるようになれば，コンパイラ自動生成技術にも利用できる．

4. 基本 HW

ASIC を容易に再設計するためには，設計の開始点となる性能の良い HW の完全な記述が与えられる必要がある．さらには，その HW が，実際に物として供給されることも重要である．HW と SW の機能ブロックに関するライブラリを用意することもまた重要である．

6 おわりに

本報告では HW/SW 協調再設計の一手法を提案し，これに基づくによる専用プロセッサシステムの要素となる技術について考察し，設計例を示した．ASIC 技術の発展にともない，従来の HW と SW の設計手法を大きく見直す必要が生じている．本手法はこれまでの設計資産を有効に利用しつつ，システムへの新しい要求仕様に対応する設計手法になると考えられる．

謝辞

日頃ご討論頂く九州大学 大学院総合理工学研究科村上和彰講師，および安浦研究室の諸氏に感謝致します．
本研究は一部，京都高度技術研究所の新産学交流事業 EAGL の支援による．

参考文献

[1] 特集「ハードウェア記述言語–新しいシステム設計環境の実現に向けて–」安浦寛人，山田輝彦 編著，情報処理学会誌，VOL.33,NO.11, 通巻 333 号,1992 年 11 月

[2] “スーパースカラアーキテクチャ評価用ワークベンチ,” 赤星博輝，安浦寛人，村上和彰，弘中哲夫，DA シンポジウム'92 論文集，pp.77–80,1992 年 8 月．

[3] “アーキテクチャ評価用ワークベンチ,” 赤星博輝，安浦寛人，VLSI 設計技術研究会，1993 年 1 月，信学技報．

[4] “ASIP 用ハードウェア/ソフトウェア・コデザインシステム PEAS の実現とその評価,” 佐藤淳，Alauddin Alomary，本間啓道，中田武治，博多哲也，今井正治，引地信之，情処研報，DA64–11，pp.79–86,1992.

[5] “An Integrated Design Environment for Application Specific Integrated Processor,” Jun Sato, Masaharu Imai, Tetsuya Hakata, Alauddin Y.Alomary, and Nobuyuki Hikichi, Proceedings of ICCD'91, pp.414–417, Oct.1991.

[6] “Cyber:High Level Synthesis System from Software into ASIC,”Kazutoshi Wakabayashi High–Level VLSI Synthesis,Kluwer Academic Publishers,pp.127–151,1991.

[7] “Archtectual Synthesis for Medium and High Throughput Signal Processing with the new CATHEDRAL environment, ”Dirk Lanneer, Stefaan Note, Francis Depuydt, Marc Pauwels, Frrancky Catthoor, Gert Goossens, Hugo De Man High–Level VLSI Synthesis, Kluwer Academic Publishers, pp.27–54, 1991.

[8] “ASIC Design System Users's manual,” 株式会社 YHP システム技術研究所．

[9] “ASIC Design System Reference manual,” 株式会社 YHP システム技術研究所．

[10] “COMPUTER ARCHITECTURE A QUANTITATIVE APPROACH,” John L Hennessy and David A Patterson Morgan Kaufmann Publishers, Inc., 1990.