DTIC FILE COPY

(2)

AD-A220 837

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER<br>NW-LIS-89-10-05 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle)<br>Normalized Time and its Use in Architectural Design | | 5. TYPE OF REPORT & PERIOD COVERED<br>Technical |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s)<br>Sam Ho, Tom Holman, Larry Snyder | | 8. CONTRACT OR GRANT NUMBER(s)<br>N00014-88-K-0453 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Northwest Laboratory for Integrated Systems<br>University of Washington<br>Dept. of Comp. Science, FR-35 Seattle, WA 98195 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS<br>DARPA-ISTO<br>1400 Wilson Boulevard<br>Arlington, VA 22209 | | 12. REPORT DATE<br>October 1989 |
| | | 13. NUMBER OF PAGES<br>3 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)<br>Office of Naval Research - ONR<br>Information Systems Program - Code 1513: CAF<br>800 North Quincy Street<br>Arlington, VA 22217 | | 15. SECURITY CLASS. (of this report)<br>Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this report is unlimited.



17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

NW LIS, normalized analysis, greedy, architecture, microprocessor enhancements.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document reports on the idea of normalized analysis: that the cost of a modification is just as important as its benefits. This study has extended the model of normalized time to multiple groups of modifications. It then analyzes the results of the simplest, greedy, algorithm as a tool for selecting the best set of modifications.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# Normalized Time and its Use in Architectural Design

Sam Ho, Tom Holman, Larry Snyder

Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195

Technical Report 89-10-05
September 1989

# Normalized Time and its Use in Architectural Design

Sam Ho, Tom Holman, Larry Snyder

Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195

Technical Report 89-10-05
September 1989



This paper appears in the Proceedings of the twenty-seventh annual Allerton conference on Communication, Control, and Computing, Allerton Park, Illinois, September 1989.



Supported in part by Defense Advanced Research Projects Agency
under ONR Contract #N00014-88-K-0453.

# NORMALIZED TIME AND ITS USE IN ARCHITECTURAL DESIGN

S. Ho, T. Holman,[1] L. Snyder
University of Washington,
Seattle, Washington

## Introduction

Building better and faster computers is always the goal of computer design. To do this, designers often propose modifications and improvements to computers. Typically, these so-called improvements must also carry some cost, in additional size or complexity. All too often, only the benefits and not the costs are the subject of analysis. As an example, the Berkeley RISC design [Patterson] had a reduced instruction set, as well as register windows. The extra cost of the register windows was offset by the smaller control. But what then, if we had allocated this cost to, say, a carry-lookahead adder, or some other part? Would this have been a wiser choice?

Holman [1988] addressed this problem. The method of *normalized analysis* is a way of fairly resolving both the costs and benefits of a modification. [Holman 1989] A concrete example of such analysis is to ask:

> Do programs run faster on (parallel) computers when floating-point coprocessors are installed, or when the equivalent amount of hardware is used instead for additional processor elements?

We may repeat this question for each additional proposed modification, such as multipliers, shifters, *etc.*

This analysis allows determining whether a particular modification is, individually, cost-effective. In real designs, though, the number of potential modifications is not one, but many. Further, these changes may interact variously. A multiplier may obviate the need for a shifter, a shifter may duplicate part of a floating point unit, and so forth. We need an algorithm for taking the varied set of modifications, and choosing that set which, working in concert, provides the best cost-benefit ratio. We first extend the normalized analysis to the more understandable concept of *normalized time*. We then examine the effect of selecting multiple modifications with the simplest algorithm, the *greedy* algorithm.

## Model

First, we must define our model. We start with some *base architecture,* and then evaluate the time and cost, on a fixed problem. The normalized time is then their product.

[1] Currently with Sun Microsystems, Mountain View, CA

**Definition 1** *Fix the computation. Then define*

$$\begin{aligned} T_0 &\equiv \textit{Time} \\ C_0 &\equiv \textit{Cost} \\ T_0C_0 &\equiv \textit{Normalized Time.} \end{aligned}$$

*The subscript zero denotes the base architecture. The units, e.g. $sm^2$, are irrelevant, as we are only making comparisons here.*

To the base architecture, we then add modifications. We stipulate that, akin to Amdahl's law [Amdahl], some fraction $f$ is affected by the change, speeding it up by some factor $S$, and the rest is left undisturbed. We also stipulate that the change increases the cost by some fraction $c$.

As an example, a floating point coprocessor might produce a speedup of a factor of twelve, but only on sixteen percent of all instructions. It might also increase the cost, measured as chip area, by thirty-eight percent. (These figures are for relational operations in a bitonic sort on the Transputer T800. [Holman 1989])

**Proposition 1** *A modification $m$ affecting a fraction $f$, with speedup $S$ and cost $c$ obeys*

$$\begin{aligned} T &= T_0\left(1 - f + \frac{f}{S}\right) \\ C &= C_0(1+c). \end{aligned}$$

The comparison is then between the normalized times. In our floating point example above, we find the normalized time is 1.18 times larger with the coprocessor than without. The coprocessor is not used enough, in the relational operations of this case, to be worthwhile, as the cost exceeds the benefit.

We can combine modifications by summing the time and cost. For simplicity, let us assume that the modifications do not interact. Interacting combinations would have a speedup term for each possible combination, but would otherwise be similar.

**Proposition 2** *For a set of noninteracting modifications $m_i$, given $f_i$, $c_i$, $S_i$, we have*

$$\begin{aligned} T &= T_0\left[1 + \sum_i\left(-f_i + \frac{f_i}{S_i}\right)\right] \\ C &= C_0\left(1 + \sum_i c_i\right). \end{aligned}$$

## Two Are Better Than One

Before we consider the greedy algorithm, let us first examine the effect of the simplest combination: two noninteracting modifications combined. In this case, the cost is less than the product of the

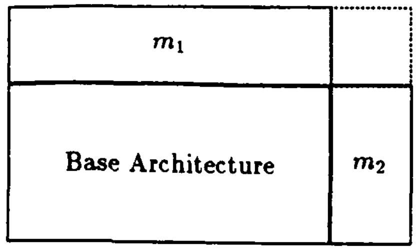


Figure 1: Representation of cross terms

```
algorithm greedy
X ← ∅
do
    for each i, m_i ∈ M
        compute TC
        if TC > T_0C_0 then X ← X ∪ {m_i}
    Base ← Base ∪ X
while X ≠ ∅
```

Figure 2: The Greedy Algorithm

two costs, relative to the base, individually. This is expressed by the inequality

$$1 + c_1 + c_2 < (1 + c_1)(1 + c_2).$$

Graphically, this is demonstrated in Figure 1. The product overestimates the cost by the dashed interaction term. A similar relationship holds for the time. Thus, we have

**Theorem 1** *The combined normalized time of two noninteracting modifications is less than the product of their separate normalized times.*

$$\frac{T_{12}C_{12}}{T_0C_0} \le \frac{T_1C_1}{T_0C_0}\frac{T_2C_2}{T_0C_0}.$$

## The Greedy Algorithm

Now we may consider the greedy algorithm, illustrated in Figure 2, for minimizing normalized time. The algorithm considers each modification in turn. All comparing favorably are added, becoming part of the new base architecture, and the process repeats until no (individually) favorable modifications remain.

Unfortunately, the greedy algorithm ignores the cross term described above.

**Theorem 2** *The greedy algorithm is suboptimal.*

As an example, take $f_1 = f_2 = 1/2$, $c_1 = c_2 = 1$, and $S_1 = S_2 = 5$. We then have

$$\frac{T_1C_1}{T_0C_0} = \frac{T_2C_2}{T_0C_0} = \frac{6}{5}$$
$$\frac{T_{12}C_{12}}{T_0C_0} = \frac{3}{5}.$$

Neither $m_1$ nor $m_2$, taken alone, is worthwhile. The algorithm will leave the base architecture untouched, yet the optimal set is both of $\{m_1, m_2\}$.

Nevertheless, the greedy algorithm is conservative, in the sense that every greedily chosen modification is also a member of the optimal set. This is because the cross term is always positive.

**Theorem 3** *The greedy algorithm is conservative.*

*Proof:* Let $G$ be the greedily chosen set, and $S$ the optimal set of modifications. Consider $G - S$. If nonempty, it must have normalized time less than one. Then, $S \cup (G - S)$ must have normalized time better than $S$, which is optimal, a contradiction. Therefore $G - S = \emptyset$, or $G \subseteq S$.

## Conclusion

We began with the idea of normalized analysis: that the cost of a modification is just as important as its benefits. We have extended the model of normalized time to multiple groups of modifications. We then analyzed the results of the simplest, greedy, algorithm as a tool for selecting the best set of modifications.

In doing so, we find that the greedy algorithm is provably a suboptimal algorithm, even for the very simple types of modifications considered here. Nevertheless, since it is a conservative algorithm, it is still useful as a starting point for further selection. By running the fast and simple greedy algorithm, we can select many of the same modifications that would be found by any better algorithm, thus reducing the number of choices that the other algorithm must make.

With this theoretical basis, and the results of an initial algorithm, it may now be possible for computer designers to select, in a more analytical manner, which of the multitude of potential modifications to include in a computer system.

## References

**Amdahl** G. M. Amdahl, "Validity of the single processor approach to achieving large-scale computing capabilities," in *Proc AFIPS Vol. 30*, pp. 483-465, 1967.

**Holman 1988** T. J. Holman, *Processor Element Architecture for Non-shared Memory Parallel Computers*, PhD Thesis, University of Washington, 1988.

**Holman 1989** T. J. Holman and L. Snyder, "Architectural Tradeoffs in Parallel Computer Design," in *Decennial Caltech Conference on VLSI*, 1989.

**Patterson** D. A. Patterson and C. H. Sequin, "A VLSI RISC," *Computer*, 15(9):8-21, 1982.