**AD-A261 002**

University
of Southern
California

**DTIC**
**S ELECTE**
FEB 2 2 1993
**B** D

**93-03627**
‖‖‖‖‖‖‖‖‖‖‖

*INFORMATION*
*SCIENCES*
*INSTITUTE* ／SI

93 2 19 117

# A Design of a Fast and Area Efficient Multi-input Muller C-element

Tzyh-Yung Wuu, USC-ISI

Sarma B. K. Vrudhula, UA-ECE

# REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE 1993, Jan | 3. REPORT TYPE AND DATES COVERED Research Report |
|---|---|---|

**4. TITLE AND SUBTITLE**

A Design of a Fast and Area Efficient Multi-input Muller C-element

**5. FUNDING NUMBERS**

MDA903-92-D-0020
MIP-9111206

**6. AUTHOR(S)**

Wuu, Tzyh-Yung, Vrudhula, Sarma B.K.

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA  90292-6695

**8. PERFORMING ORGANIZATION REPORT NUMBER**

ISI/RR-93-302

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

DARPA
3701 N. Fairfax Drive
Arlington, VA  22203-1714

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**

This document is approved for public release;
distribution is unlimited.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT (Maximum 200 words)**

A multi-input Muller C-element has frequently been used for joining signal transitions or completion time detection in self-timed circuits. This paper presents an n-input Muller C-element design which uses the multi-level login design technique and has a symmetric format for any integer $n \geq 2$. In comparison with series-parallel MOS structure implementations and C-element tree implementations, our design has fewer restrictions in terms of n, less path delay, less delay variance from inputs to output, and less area consumption. Experimental validation in this paper is based on an industrial standard cell library.

**14. SUBJECT TERMS**

Muller C-element, MOS, standard cells, self-timed circuits, signal transitions, path delay, cell area

**15. NUMBER OF PAGES**

11

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | UL |

# GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

**Block 1. Agency Use Only (Leave blank).**

**Block 2. Report Date.** Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

**Block 3. Type of Report and Dates Covered.** State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

**Block 4. Title and Subtitle.** A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title; add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

**Block 5. Funding Numbers.** To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

| | | | |
|---|---|---|---|
| C | - Contract | PR | - Project |
| G | - Grant | TA | - Task |
| PE | - Program Element | WU | - Work Unit Accession No. |

**Block 6. Author(s).** Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

**Block 7. Performing Organization Name(s) and Address(es).** Self-explanatory.

**Block 8. Performing Organization Report Number.** Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

**Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es).** Self-explanatory.

**Block 10. Sponsoring/Monitoring Agency Report Number.** (If known)

**Block 11. Supplementary Notes.** Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

**Block 12a. Distribution/Availability Statement.** Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

  DOD - See DoDD 5230.24, "Distribution Statements on Technical Documents."
  DOE - See authorities.
  NASA - See Handbook NHB 2200.2.
  NTIS - Leave blank.

**Block 12b. Distribution Code.**

  DOD - Leave blank.
  DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
  NASA - Leave blank.
  NTIS - Leave blank.

**Block 13. Abstract.** Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

**Block 14. Subject Terms.** Keywords or phrases identifying major subjects in the report.

**Block 15. Number of Pages.** Enter the total number of pages.

**Block 16. Price Code.** Enter appropriate price code *(NTIS only)*.

**Blocks 17. - 19. Security Classifications.** Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

**Block 20. Limitation of Abstract.** This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

# A Design of a Fast and Area Efficient Multi-input Muller C-element

**Tzyh-Yung Wuu**

Information Science Institute
Univ. of Southern California
wuu@lepton.isi.edu

**Sarma B. K. Vrudhula**
**(a.k.a. Sarma Sastry)**
ECE Dept.
Univ. of Arizona
sarma@kashi.ece.arizona.edu

## Abstract

A multi-input Muller C-element has frequently been used for joining signal transitions or completion time detection in self-timed circuits. This paper presents an $n$-input Muller C-element design which uses the multi-level login design technique and has a symmetric format for any integer $n \geq 2$. In comparison with series-parallel MOS structure implementations and C-element tree implementations, our design has fewer restrictions in terms of $n$, less path delay, less delay variance from inputs to output, and less area consumption. Experimental validation in this paper is based on an industrial standard cell library.

## 1   Introduction

A Muller C-element [5] is used as a basic component in the design of speed-independent circuits. A C-element is functionally equivalent to an SR latch. Under the assumption of unbounded gate delays it is not possible to guarantee that S and R will not be 1 simultaneously. This problem does not arise with a C-element [4]. The output of a two input C-element will equal the value of the inputs after both inputs have reached the same value; otherwise the output remains unchanged. That is, if $i_1$ and $i_2$ are the two inputs and $O$ is the output, then the defining equation of the C-element is $O = i_1 \cdot i_2 + O \cdot i_1 + O \cdot i_2$ [5]. A two input C-element can be viewed as a logical *and* of two *events*, where an *event* can be a 0-1 or 1-0 transition [7]. This behaviour is shown in Figure 1.

A C-element is commonly used for *joining* signal transitions to signal the completion of an operation [1, 2, 3, 4, 6]. For example, the 16 output computational block in Figure 2 will require a 16 input C-element to join all 16 completion signals in order to generate one signal transition to indicate the completion of the block.

The output of an n-input C-element is 1 if all the inputs are 1 and it is 0 if all the inputs are 0; otherwise its value remains unchanged [6]. The state diagrams for two-input and three-input C-elements are shown in Figure 3, where the state is labeled "inputs/output". The initial state of a C-element is having all inputs and its output zero. This is denoted by the double circle in the state diagram.
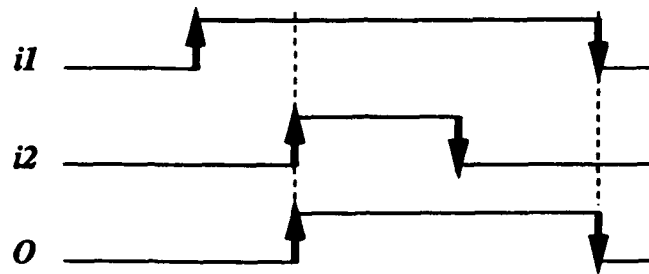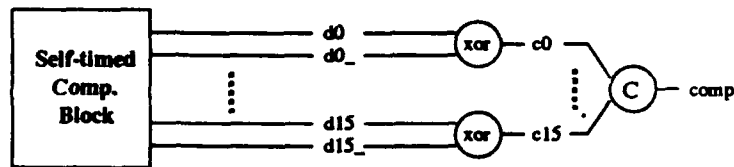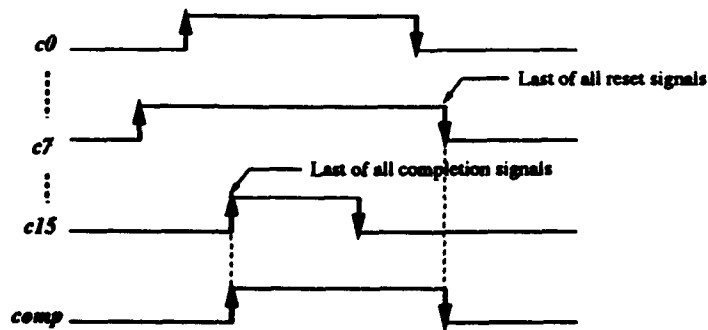
Figure 1: Timing diagrams for 2-input Muller C-elements



(a) A typical self-timed computational block



(b) Timing diagram of the completion/reset signals

Figure 2: Join of completion/reset signals

2

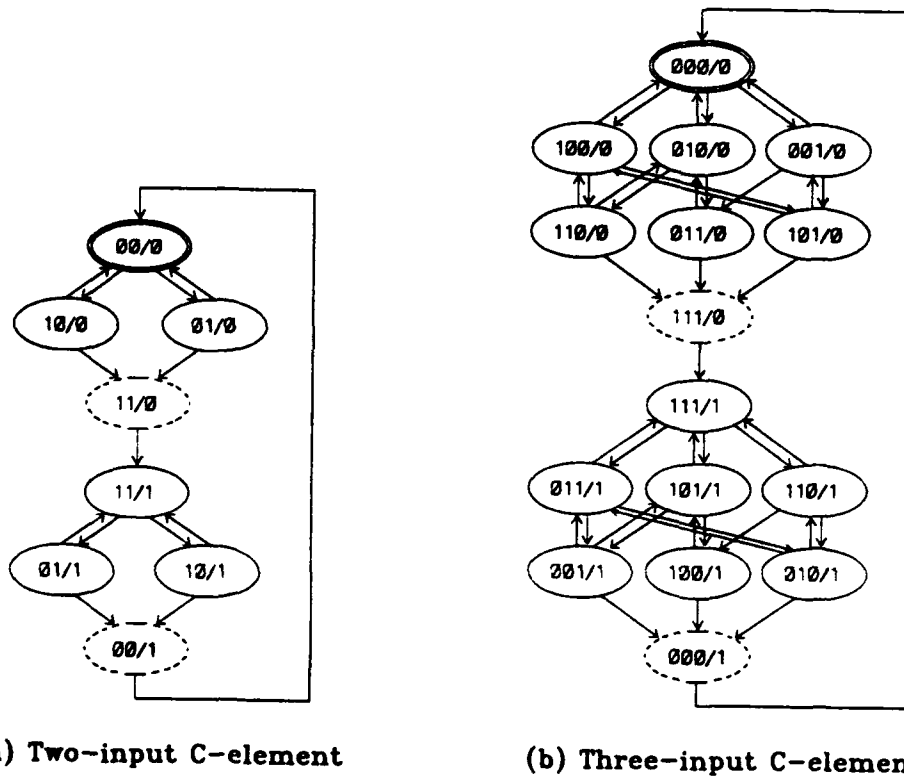(a) Two-input C-element



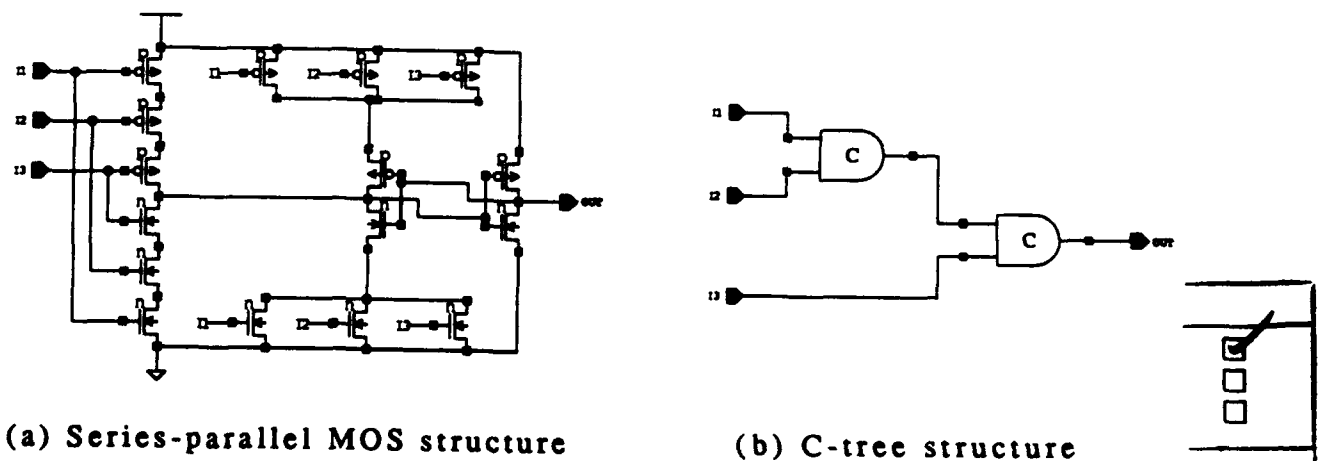(b) Three-input C-element

Figure 3: State diagrams for Muller C-elements



(a) Series-parallel MOS structure



(b) C-tree structure

Figure 4: Three-input series-parallel C-element designs

DTIC QUALITY INSPECTED 3

3

C-elements with large numbers of inputs are very useful and an efficient implementation in terms of area and speed is needed. Two designs of multi-input C-elements have been used: a series-parallel MOS structure [6] shown in Figure 4(a) and the C-element tree [4, 6] shown in Figure 4(b). Because the C-element is associative, the tree implementation uses $(n-1)$ two-input C-elements to form an $n$-input C-element. The input-output delay of the series-parallel MOS structure is less than that of the tree structure. However, the series-parallel implementation is not feasible for large $n$. For this reason, most existing designs employ the tree structure. The main disadvantages of the tree implementation are that it is very slow and the variance in the delays over different input-output paths is very large.

In this paper we present an efficient design of a multi-input C-element. Our design is symmetric and the variance in delay over different input-output paths is very small. In Section 2 we derive a symmetric form of an n-input C-element and provide estimates of delay and area. In Section 3 we demonstrate the advantages of our design by presenting experimental results for C-elements with inputs ranging from 2 to 128.

# 2   Design of a Multi-input C-element

The target technology of our design is CMOS. Since inverted logic is faster than non-inverted logic in CMOS, we will use *and-or-invert* (AOI) logic and inverters instead of *and-or* logic. The defining equation of an n-input C-element with a *reset* is given by

$$OUT = ((I_1 \bullet I_2 \bullet \ldots \bullet I_n) + (I_1 + I_2 + \ldots + I_n) \bullet OUT) \bullet RESET \tag{1}$$

where $I_i$, for $i = 1, \ldots, n$, are inputs of the C-element, and $OUT$ is the output of the C-element. Using DeMorgan's law we transform Equation (1) as follows:

$$
\begin{aligned}
OUT &= ((I_1 \bullet I_2 \bullet \ldots \bullet I_n) + (I_1 + I_2 + \ldots + I_n) \bullet OUT) \bullet RESET \tag{2} \\
&= \overline{\overline{((I_1 \bullet I_2 \bullet \ldots \bullet I_n) + (I_1 + I_2 + \ldots + I_n) \bullet OUT) \bullet RESET}} \\
&= \overline{\overline{((I_1 \bullet I_2 \bullet \ldots \bullet I_n) + (I_1 + I_2 + \ldots + I_n) \bullet OUT)} + \overline{RESET}} \\
&= \overline{\overline{(I_1 \bullet I_2 \bullet \ldots \bullet I_n)} \bullet \overline{(I_1 + I_2 + \ldots + I_n) \bullet OUT} + \overline{RESET}} \\
&= \overline{\overline{(I_1 \bullet I_2 \bullet \ldots \bullet I_n)} \bullet (\overline{(I_1 + I_2 + \ldots + I_n)} + \overline{OUT}) + \overline{RESET}} \tag{3}
\end{aligned}
$$

Equation (3) can be further decomposed to following equations.

$$
\begin{aligned}
NAND\_TREE &= \overline{(I_1 \bullet I_2 \bullet \ldots \bullet I_n)} \tag{4} \\
NOR\_TREE &= \overline{(I_1 + I_2 + \ldots + I_n)} \tag{5} \\
OUT &= \overline{NAND\_TREE \bullet (NOR\_TREE + \overline{OUT}) + \overline{RESET}} \tag{6}
\end{aligned}
$$

4

Figure 5: C-elements design



(a) Two-level NAND_TREE implementation



(b) Three-level NAND_TREE implementation

Figure 6: Multi-level NAND_TREE implementation

5

The above decomposition is very useful when it is mapped to a CMOS cell-based implementation. In Figure 5 we show a C-element design consisting of 3 parts: a NAND_TREE, a NOR_TREE, and a OAO_PART (*or-and-or*), each implemented separately. The OAO_PART shown in Figure 5 remains the same for all values of $n$.

Therefore, we only need to find a proper NAND_TREE/NOR_TREE implementation. In the HP C34000 library [8], there are $n$-input NAND and NOR gates for $2 \leq n \leq 8$. For larger $n$, NAND_TREE (NOR_TREE) can be further decomposed into two-level NAND-OR tree (NOR-AND tree). When a two-level structure is not sufficient, we can use more levels, i.e., use a *nand-nor-nand* three level structure to implement a NAND_TREE or a *nor-nand-nor* structure to implement a NOR_TREE. Figures 6 (a) and (b) show the two-level and the three-level implementations for NAND_TREE. In order to minimize the variance of the input-output delays, the structure of the NAND_TREE implementation is identical to the structure of the NOR TREE implementation.

We now provide estimates of delay and area for our design and compare them with the tree implementation of a C-element. A comparison with a series-parallel MOS structure is unnecessary since such an implementation is not feasible for large numbers of inputs.

**Delay:** For an $n$-input C-element, the input-output delay of a C-element tree implementation is equal to the number of levels in the structure multiplied by the input-output delay of a 2-input C-element, where the number of levels is $\lceil log_2 n \rceil$, i.e.,

$$D_{tree}(n) = \lceil log_2 n \rceil * ( \text{ 2-input Muller C-element delay } ) \tag{7}$$
$$\approx \lceil log_2 n \rceil * ( \text{ 2-input NAND/NOR delay } + \text{OAOI delay } ) \tag{8}$$
$$\approx \lceil log_2 n \rceil * \text{OAOI delay} + \lceil log_2 n \rceil * \text{2-input NAND/NOR delay}$$

The input-output of our design is

$$D_{multi}(n) = \text{OAOI delay} + \text{n-input NAND\_TREE/NOR\_TREE delay} \tag{9}$$

The n-input NAND_TREE/NOR_TREE can be implemented by $\lceil log_2 n \rceil$ stages of a NAND2-NOR2 tree, and can be made faster by using $\lceil log_m n \rceil$ stages of NAND$m$-NOR$m$ tree. Therefore,

$$D_{multi}(n) \leq \text{OAOI delay} + \lceil log_2 n \rceil * \text{2-input NAND/NOR delay} \tag{10}$$

Obviously, our design is much faster than C-element tree implementation for $n > 2$, although both $D_{tree}(n)$ and $D_{multi}(n)$ are $O(log(n))$.

**Delay Variance:** In order to have less delay variance among the input-output paths in our design, two sufficient conditions need to be met.

1. Transistors in the OAOI element of the OAO_PART must be sized so that the delay from one OR gate input to the output and the delay from one AND gate input to the output is the same.

6

2. the structure of NAND_TREE should be symmetric and identical to the structure of the NOR_TREE.

These two conditions are easily satisfied for cell-based designs, and they are a property of the HP standard cell library. However, the tree implementation of an n-input C-element is balanced only if $n = 2^{\lceil log_2 n \rceil}$. For this reason, the variation in delays among the different input-output paths is very large.

**Area:** In comparing our design with the C-element tree implementation, the routing area is not considered. Similar to the delay comparison above, the C-element tree has a big overhead in terms of size due to the repeated use of the OAO_PART in every two-input C-element in the tree. An estimate of the area of a C-element tree implementation is

$$A_{tree}(n) \approx (n-1) * (A_{NAND2} + A_{NOR2} + A_{OAOI} + A_{INV}) \tag{11}$$
$$\approx (n-1) * (A_{OAOI} + A_{INV}) + (n-1) * A_{NAND2} + (n-1) * A_{NOR2}$$

The estimate of area for our design is

$$A_{multi}(n) \approx A_{OAOI} + A_{INV} + A_{\text{n-input NAND\_TREE}} + A_{\text{n-input NOR\_TREE}} \tag{12}$$

The n-input NAND_TREE/NOR_TREE can be implemented by $(n-1)$ number of NAND2-NOR2 elements, and it may have a similar sized implementation by using $\lceil log_m n \rceil$ stages of NAND$m$-NOR$m$ tree. Therefore,

$$A_{multi}(n) \approx A_{OAOI} + A_{INV} + (n-1) * A_{NAND2} + (n-1) * A_{NOR2}$$

Obviously, our design is much smaller than C-element tree implementation for $n > 2$, although both $A_{tree}(n)$ and $A_{multi}(n)$ are $O(n)$.

# 3  Experimental results

We present experimental results using the HP C34000 standard cell library [8]. The input-output path delay is obtained by simulating a Verilog$^{TM}$ [1] model distributed with this cell library. Wiring capacitances are included in the model. For each $n$-input C-element, both the rise and fall delays for all input-output paths are simulated, i.e., $2n$ path delays are collected. The path delay of an $n$-input C-element is taken to be the average of these $2n$ delays. The delay variance among input-output paths is calculated by the difference between the maximum input-output path delay and the minimum input-output path delay of all input-output paths. The area of a design is computed as the sum of the area of all cells in the design. The results are shown in the Table 1. In this table, the name of Muller C-element suffixed with "A" is the series-parallel structure implementation, the name suffixed with "B" is our design, and the name suffixed with "AT" is the C-element tree implementation using "mullerC2A", and the name suffixed with "BT" is the C-element tree implementation using "mullerC2B".

---

[1]Verilog is a hardware description language, and it is a trademark of Cadence Design Systems, Inc.

| | min (nsec) | max (nsec) | avg (nsec) | max - min (nsec) | area ($\mu^2$) |
|---|---|---|---|---|---|
| mullerC2A | 2.8 | 3.6 | 3.2 | 0.8 | 3319 |
| mullerC2B | 2.9 | 3.2 | 3.0 | 0.3 | 4085 |
| mullerC3B | 3.0 | 3.1 | 3.1 | 0.1 | 4596 |
| mullerC4B | 3.0 | 3.3 | 3.1 | 0.3 | 5107 |
| mullerC5B | 3.0 | 3.5 | 3.3 | 0.5 | 5874 |
| mullerC6B | 3.9 | 4.5 | 4.1 | 0.6 | 6384 |
| mullerC7B | 3.9 | 4.8 | 4.2 | 0.9 | 6894 |
| mullerC8B | 3.9 | 4.6 | 4.2 | 0.7 | 7661 |
| mullerC9B | 3.9 | 4.4 | 4.1 | 0.5 | 6384 |
| mullerC16B | 3.9 | 4.4 | 4.2 | 0.5 | 16343 |
| mullerC32B | 4.2 | 4.8 | 4.6 | 0.6 | 32175 |
| mullerC64B | 4.4 | 4.9 | 4.6 | 0.5 | 61287 |
| mullerC128B | 5.3 | 6.0 | 5.6 | 0.7 | 122063 |
| mullerC2AT | 2.8 | 3.6 | 3.2 | 0.8 | 3319 |
| mullerC3AT | 2.8 | 7.9 | 5.9 | 5.1 | 6638 |
| mullerC4AT | 6.0 | 7.9 | 6.9 | 1.9 | 9957 |
| mullerC5AT | 6.4 | 12.1 | 8.6 | 5.7 | 13276 |
| mullerC6AT | 6.5 | 12.1 | 9.6 | 5.6 | 16595 |
| mullerC7AT | 6.0 | 12.1 | 10.2 | 6.1 | 19914 |
| mullerC8AT | 9.3 | 12.1 | 10.6 | 2.8 | 23233 |
| mullerC9AT | 9.7 | 16.3 | 11.7 | 6.6 | 26552 |
| mullerC16AT | 12.5 | 16.3 | 14.4 | 3.8 | 49785 |
| mullerC32AT | 14.5 | 20.0 | 17.9 | 5.5 | 102889 |
| mullerC64AT | 19.0 | 24.7 | 21.7 | 5.7 | 209097 |
| mullerC128AT | 22.3 | 28.9 | 25.4 | 6.6 | 421513 |
| mullerC2BT | 2.9 | 3.2 | 3.0 | 0.3 | 4085 |
| mullerC3BT | 3.2 | 7.4 | 5.9 | 4.2 | 8170 |
| mullerC4BT | 7.2 | 7.6 | 7.4 | 0.4 | 12255 |
| mullerC5BT | 7.1 | 11.6 | 9.0 | 4.5 | 16340 |
| mullerC6BT | 7.5 | 11.9 | 10.3 | 4.4 | 20425 |
| mullerC7BT | 7.5 | 11.8 | 11.1 | 4.3 | 24510 |
| mullerC8BT | 11.4 | 12.0 | 11.7 | 0.6 | 28595 |
| mullerC9BT | 11.5 | 15.8 | 12.6 | 4.3 | 32680 |
| mullerC16BT | 15.7 | 16.5 | 16.1 | 0.8 | 61275 |
| mullerC32BT | 19.9 | 21.1 | 20.4 | 1.2 | 122635 |
| mullerC64BT | 24.1 | 25.6 | 24.7 | 1.5 | 257355 |
| mullerC128BT | 28.3 | 30.1 | 29.1 | 1.8 | 518795 |

Table 1: Delay and delay variance and area comparisons of Muller C-element

Figure 7 shows the path delays in terms of $log_2 n$. As expected, the path delay of the C-element tree is approximately equal to $\lceil log_2 n \rceil * K$, where $K$ is the delay of a 2-input C-element including wiring delay. Our design is much faster than the C-element tree implementation, and path delay grows very slowly as $n$ increases. For example, our 32-input C-element design is 3.89 to 4.43 times faster than C-element tree design. Figure 8 shows the delay variance among input-output paths in terms of $n$. The delay variance of our design is small for any number of inputs, but the delay variance of the tree implementation is only small for a balanced tree, i.e., where $n = 2^{\lceil log_2 n \rceil}$. Finally, the area comparison between our design and the C-element tree implementation is shown in Figure 9. In both our design and C-tree implementation, the cell area grows linearly with $n$. However, the increasing area in our design is due to its NAND_TREE and NOR_TREE whereas the increasing area for the C-element tree implementation is due to the number of 2-input C-elements. Therefore, the rate of increase of area for our design is much smaller than for the tree implementation. For example, our 32-input design is 3.20 to 3.81 times smaller than the C-element tree implementation.
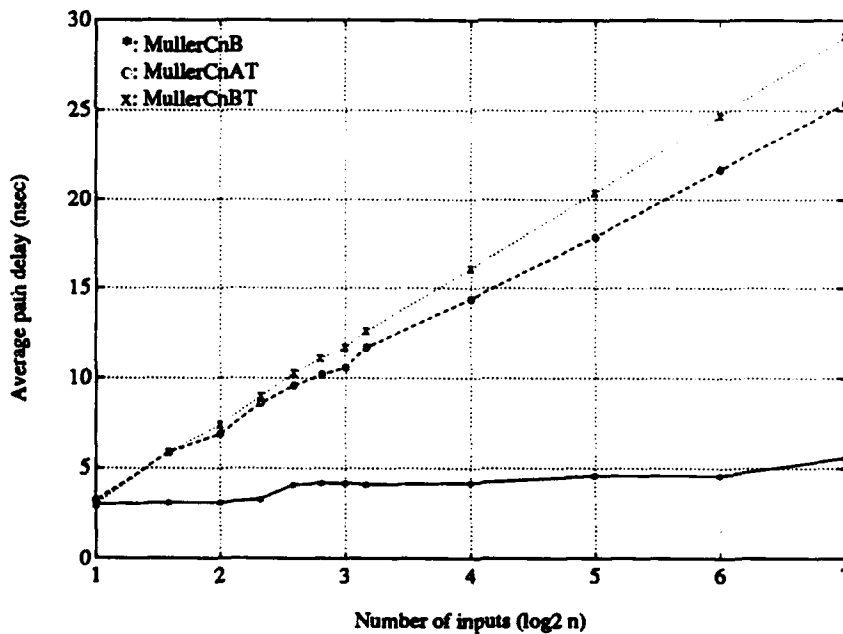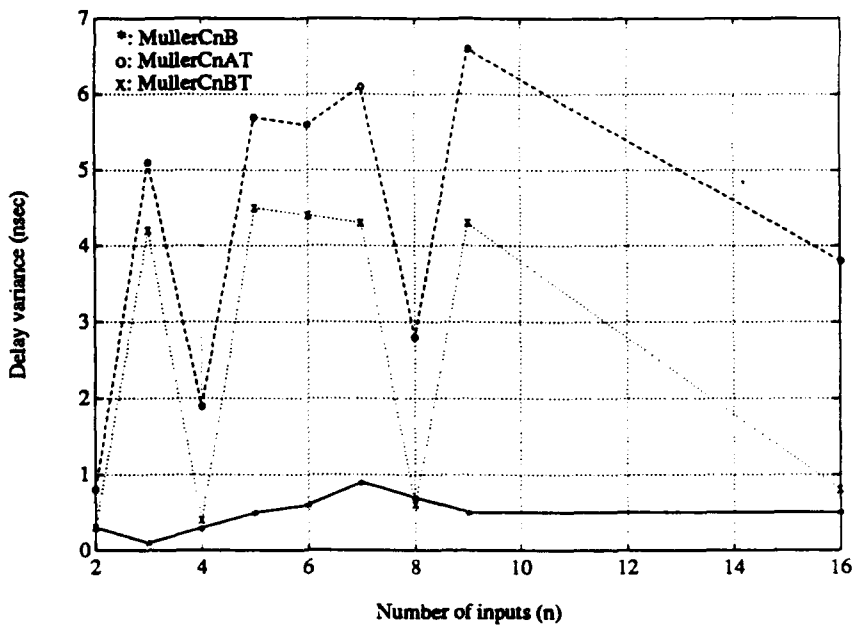


Figure 7: Comparison of path delay
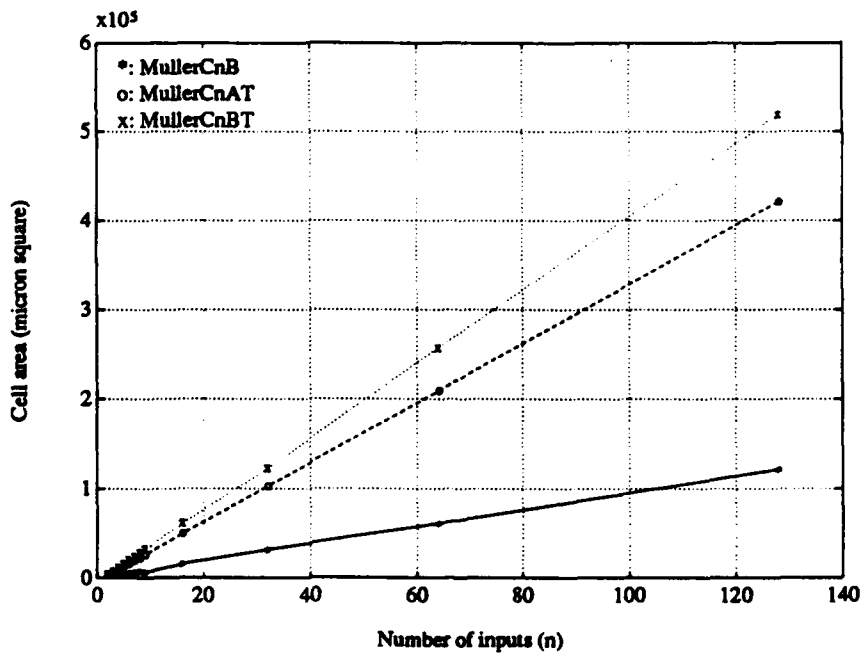
Figure 8: Comparison of delay variance



Figure 9: Comparison of cell area

10

# 4   Conclusions

We have presented a new design for an $n$-input C-element. which is faster. has less delay variance among input-output paths, and is smaller than the C-element tree implementation. We have also demonstrated the advantages of our design using an industrial standard cell library.

**Acknowledgement**

# References

[1] E. Brunvand and R. F. Sproull. Translating Concurrent Communicating Programs into Delay-Insensitive Circuits. Technical Report CMU-CS-89-126. School of Computer Science, Carnegie Mellon University, April 1989.

[2] I. David, R. Ginosar, and M. Yoeli. "An Efficient Implementation of Boolean Functions and Finite State Machines as Self-Timed Circuits". *ACM Computer Architecture News.* 17(6):91–104, 1989.

[3] A. J. Martin et al. The Design of an Asynchronous Microprocessor. *Proceedings of the Decennial Caltech Conference on VLSI*, pages 351–373, March 1989.

[4] T. H.-Y. Meng. *Synchronization Design for Digital Systems.* Kluwer Academic Publishers, 1991.

[5] D. E. Muller and W. S. Bartky. "A Theory of Asynchronous Circuits". *Proceedings of an International Symposium on the Theory of Switching*, 29:204–243, 1959.

[6] C. L. Seitz. "System Timing". In *Introduction to VLSI Systems.* by C. Mead and L. Conway, Addison Wesley, pages 128–262, 1980.

[7] I. E. Sutherland. "MICROPIPELINES", The 1988 Turing Award Lecture. *Communications of the ACM*, 32(6):720–738, 1989.

[8] The HP C34000 Standard Cell Library Data Manual (Draft). Hewlett-Packard Company, Integrated Circuit Business Division, June 1992.