[19] S. Novack, A. Nicolau, and N. Dutt, "A unified code generation approach using mutation scheduling," in *Code Generation for Embedded Processors*. Boston, MA: Kluwer, 1995.
[20] T. A. Proebsting and C. W. Fraser, "Detecting pipeline hazards quickly," *PPL*, Jan. 1994.
[21] B. R. Rau, "Iterative modulo scheduling: An algorithm for software pipelining loops," in *Proc. 27th Microarchitecture*, Nov. 1994, pp. 63–74.
[22] C. Siska, "A processor description language supporting retargetable multi-pipeline DSP program development tools," in *Proc. Int. System Synthesis Symp.*, Dec. 1998, pp. 31–36.
[23] *TMS320C62x/C67x CPU and Instruction Set Reference Guide*, Texas Instruments Incorporated, Dallas, TX, 1998.
[24] (1997) *The MDES User Manual, Trimaran Release*. [Online]. Available: http://www.trimaran.org.

Design of a Cycle-Efficient 64-b/32-b Integer Divisor Using a Table-Sharing Algorithm

Chua-Chin Wang, Po-Ming Lee, Jun-Jie Wang, and Chenn-Jung Huang

***Abstract*—In new generations of microprocessors, the superscalar architecture is widely adopted to increase the number of instructions executed in one cycle. The division instruction among all of the instructions needs more cycles than the rest, e.g., addition and multiplication. It then makes division instruction an important cycles-per-instruction figure for modern microprocessors. In this paper, a radix-16/8/4/2 divisor is proposed, which uses a variety of techniques, including operand scaling, table partitioning, and, particularly, table sharing, to increase performance without the cost of increasing complexity. A physical chip using the proposed method is implemented by 0.35-μm single poly four metal (1P4M) CMOS technology. The testing measurement shows that the chip can execute signed 64-b/32-b integer division between 3–13 cycles with a 80-MHz operating clock.**

***Index Terms*—Integer division, mixed radixes, on-the-fly conversion, operand scaling, table folding, table sharing.**

I. Introduction

Integer division is a critical operation in CPU design since the number of clock cycles to complete an integer division is probably very long and unpredictable. The role of division is becoming more and more critical owing to the requirement of signed computer arithmetics, the modulus computation, the calculation of encryption keys, and so on. Division algorithms can be roughly classified into two categories, namely, digit-recurrence methods [1] and functional iteration techniques [1], while the former is commonly used. Regarding the digit-recurrence method, traditionally there are two types of division schemes, i.e., restoring and nonrestoring schemes. However, they both require multiple operation steps to derive a quotient bit. Not only is the efficiency drastically poor, but also a long adder/subtracter is needed to execute the remainder bit adjustment.

In this paper, we employ a modified high-radix, i.e., radix-16/8/4/2, digit-recurrence division method and on-the-fly conversion method to reduce the required cycles for the 64-b/32-b signed integer division, while keeping the hardware complexity in control.

Manuscript received March 30, 2001; revised April 30, 2002. This work was supported by the Academic Foundation of the Taiwan Ericsson Company Ltd. This work was supported in part by the National Science Council under Grant NSC 89-2218-E-110-014 and Grant NSC 89-2218-E-110-015.

C.-C. Wang and P.-M. Lee are with the Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan 80424, R.O.C. (e-mail: ccwang@ee.nsysu.edu.tw).

J.-J. Wang is with VIA Technologies Inc., Taipei, Taiwan, 231 R.O.C.

C.-J. Huang is with the Department of Computer Science and Information Education, National Taitung Teachers College, Taitung, Taiwan 95004, R.O.C. (e-mail: cjh@cc.nttc.edu.tw).

Digital Object Identifier 10.1109/TVLSI.2003.816143

II. High-Radix 64-b/32-b Signed Integer Divisor

A. Digit-Recurrence Theory

Assume x, d, q, and rem to be the dividend, divider, quotient, and remainder in the division operation. We also denote the radix of the division as being r. The division is then defined as $x = q \cdot d + \mathrm{rem}$. In the digit-recurrence division algorithm [2], 1–b bits of quotient digit can be obtained every iteration in a radix-2^b digit-recurrence division. In other words, b bits of quotient can be obtained every iteration. In [3], the digit-recurrence algorithm is defined as

$$w[j+1] = r \cdot w[j] - d \cdot q_{j+1} \tag{1}$$

where $w[j+1]$ is the residual of the $(j+1)$th iteration, r is the radix, and q_{j+1} is the quotient digit generated in the $(j+1)$th iteration. In a radix-r, $r = 2^b$, division, the quotient digit set is defined as $q_j \in D_a = \{-a, \ldots, -1, 0, 1, \ldots, a\}$. Since $\|D_a\| > r$, it uses more than r numbers to present the quotient digits, which make this quotient representation form to be a redundant form. Besides, the restriction of a is $a \geq \lceil r/2 \rceil$. In (1), the quotient digits are generated in every iteration. Hence, we can define the quotient-digit selection function as $q_{j+1} = \mathrm{SEL}(w[j], d)$, where the $\mathrm{SEL}()$ function can be simplified as a table lookup function.

Although the digit-recurrence algorithm has been well written in [2], there are many unsolved difficulties when it comes to hardwaredly realizing such a divisor, including the following.

1) A long adder is needed at the adjustment of the remainder.
2) Extra adjustment actions are required when the last cycle of the division contains nonmultiple digits of the radix. (For instance, the radix is 16, but there is only 1 b left in the dividend to be processed.)
3) The adjustment of the remainder is missing when the signed division is executed.
4) A data flow control unit is required, which provides correct timing control such that the results of the division can be correctly placed on the output ports.
5) The size of the quotient selection table will grow exponentially with the radix. Besides, it is likely that one radix needs one table. These two factors lead to a huge chip area consumption if the divisor is implemented on silicon.

In short, the above problems will occur during the realization of a long signed divisor. If these problems are not resolved efficiently, the hardware divisor will be large and slow.

B. Mixed Radix-16/8/4/2 64-b/32-b Integer Divisor

In [4], a mixed radix-8/4/2 integer divisor was proposed, of which performance is better than that of a normal radix-4/2 integer divisor [5]. However, it paid the price of increasing the complexity of hardware, and then nearly doubled the total area of the divisor owing to the sizes of tables. In this study, despite that the radix will be raised up to 16 to retire more bits of the quotient per cycle, the complexity of the hardware will be retained to a similar degree by using several methods, including operand prescaling, table partitioning, [6], and table folding.

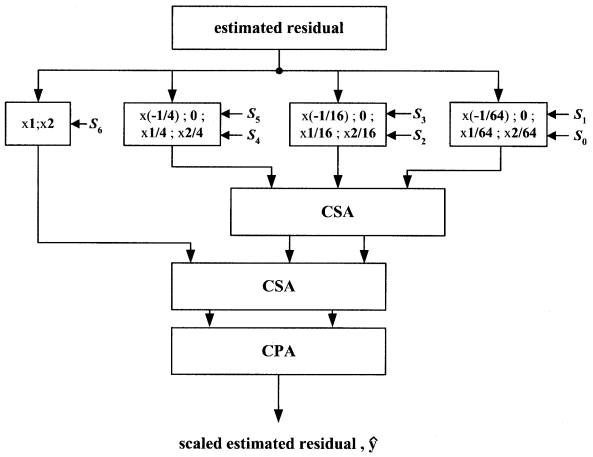

Fig. 1. Generation of scaled estimated residual $\hat{y}$.

1) Operand Scaling: In high-radix divisors, the cycle time is generally determined by the quotient-digit selection operation, which is basically a table lookup operation. The complexity of the quotient selection function increases exponentially if the radix increases linearly. Consequently, it results in a long table lookup time. Operand scaling is a better alternative to avoid the long table lookup time.

The maximal overlap between quotient digits appears when the divider is the maximum. That is, the maximal amount of overlap occurs when a divider d is normalized and it approaches to one. This observation leads to the concept of divider prescaling. In the first step of the scaling method, the divisor is prescaled by a factor M so that the scaled divider z is $1-\alpha \le z = M \cdot d \le 1+\beta$, where α and β are chosen such that the scaling factor S_i, $i \in \{0,\ldots,6\}$ is identical in all divider intervals and the quotient-digit selection is independent of the divider. Besides, the value of M should be chosen to minimize $(\alpha+\beta)$, which produces the smallest achievable range of z. In order to preserve the value of the quotient, three alternative ways of performing the scaling were proposed [2]. Operand scaling process produces a scaled estimated residual $\hat{y}$, which is generated as shown in Fig. 1. In Fig. 1, the "estimated residual" is chosen from the first seven bits from $w[j]$ in (1).

2) Table-Sharing Algorithm: The quotient-digit selection function in the radix-8 division has been proven to be the bottleneck in each iteration [4]. Hence, the radix-16/8/4/2 integer division will be infeasible unless a simplified quotient-digit selection function is developed. A modified version of the table-partitioning algorithm [6], called "table sharing," is proposed to simplify the digit selection process. The table-partition method in [6] arranged the entries in a nonmergable manner. We simply reorder the rows to place the negative entries of *Lo.* on the top half of the table and the positive ones on the bottom. Hence, the modified table possesses a feature that is mergable for different radixes. Moreover, the merge of a total of the radix-2, radix-4, radix-8, and the radix-16 quotient selection tables reduces the number of the required tables, which, in turn, reduces the area of the chip.

3) Quotient Digit Decomposition and Table Sharing: In the proposed scheme, the maximally redundant quotient digit set is chosen for radix-16 division and decomposed into four components as follows:

$$\begin{aligned} q_{j+1} = q_{h,h} + q_{h,l} + q_{l,h} + q_{l,l}, \qquad & q_{h,h} \in \{-8,0,8\}; \\ q_{h,l} \in \{-4,0,4\}; \quad & q_{l,h} \in \{-2,0,2\}; \\ q_{l,l} \in \{-1,0,1\} \end{aligned} \tag{2}$$

where $q_{j+1} \in \{-15,-14,\ldots,0,\ldots,14,15\}$, and $q_{h,h}$, $q_{h,l}$, $q_{l,h}$, and $q_{l,l}$ are tabulated as shown in Table I.

TABLE I
DECOMPOSITION OF q_{j+1} FOR RADIX-16/8/4/2 DIVISOR

				Lo.	*Hi.*	q_{j+1}	$q_{h,h}$	$q_{h,l}$	$q_{l,h}$	$q_{l,l}$
16	8	4	2	-16.5	-15	-15	-8	-4	-2	-1
				-14.5	-14	-14	-8	-4	-2	0
				-13.5	-13	-13	-8	-4	0	-1
				-12.5	-12	-12	-8	-4	0	0
				-11.5	-11	-11	-8	0	-2	-1
				-10.5	-10	-10	-8	0	-2	0
				-9.5	-9	-9	-8	0	0	-1
				-8.5	-8	-8	-8	0	0	0
				-7.5	-7	-7	0	-4	-2	-1
				-6.5	-6	-6	0	-4	-2	0
				-5.5	-5	-5	0	-4	0	-1
				-4.5	-4	-4	0	-4	0	0
				-3.5	-3	-3	0	0	-2	-1
				-2.5	-2	-2	0	0	-2	0
				-1.5	-1	-1	0	0	0	-1
				-0.5	-0	-0	0	0	0	0
				0.5	1	1	0	0	0	1
				1.5	2	2	0	0	2	0
				2.5	3	3	0	0	2	1
				3.5	4	4	0	4	0	0
				4.5	5	5	0	4	0	1
				5.5	6	6	0	4	2	0
				6.5	7	7	0	4	2	1
				7.5	8	8	8	0	0	0
				8.5	9	9	8	0	0	1
				9.5	10	10	8	0	2	0
				10.5	11	11	8	0	2	1
				11.5	12	12	8	4	0	0
				12.5	13	13	8	4	0	1
				13.5	14	14	8	4	2	0
				14.5	16	15	8	4	2	1

TABLE II
GENERATION OF q_h AND q_l FOR DIFFERENT RADICES

radix		bit 2 (MSB)	bit 1	bit 0 (LSB)
16	q_h	$q^h_{l,l}$	$q^l_{h,h}$	$q^l_{h,l}$
	q_l	$q^h_{l,l}$	$q^l_{l,h}$	$q^l_{l,l}$
8	q_h	$q^h_{l,l}$	$q^l_{h,h} + q^l_{h,l}$	$q^l_{l,h}$
	q_l	$q^h_{l,l}$	$q^l_{l,l}$	0
4	q_h	$q^h_{l,l}$	$q^l_{h,h} + q^l_{h,l} + q^l_{l,h}$	$q^l_{h,h} + q^l_{h,l} + q^l_{l,l}$
	q_l	$q^h_{l,l}$	0	0
2	q_h	$q^h_{l,l}$	$q^l_{h,h} + q^l_{h,l} + q^l_{l,h} + q^l_{l,l}$	0
	q_l	$q^h_{l,l}$	0	0

According to Table I, the selection intervals for q_{j+1} in radix-16, radix-8, radix-4, and radix-2 divisions are included and tabulated as indicated by the four braces, respectively. Besides, *Lo.* and *Hi.* in Table I denote the lower and upper bounds of the shifted estimated residual, respectively. Notably, the bounds of the scaled shifted residual $\hat{y}$ are derived from $|w[j]| \le (8288/8192)$ and the corresponding 2-b truncation error. Namely, $\lfloor -r \cdot (8288/8192) - 2^{-2} \rfloor \le \hat{y} \le \lfloor r \cdot (8288/8192) \rfloor$. Nevertheless, since the quotient-digit table is shared by different radixes, the highest order digit will be incorrectly enabled if the value of $\hat{y}$ is close to the bounds, as illustrated in the first and last rows and as indicated by the braces in Table I. Fortunately, this can be fixed easily later at the quotient-digit assimilation stage where $q_{h,h}$, $q_{h,l}$, $q_{l,h}$, and $q_{l,l}$ re-compose the quotient digits.

4) Table Folding: By inspecting Table I, the entries of the top half are identical to the opposite ones in the bottom half. It allows us to simply implement only the positive half. Accordingly, the proposed

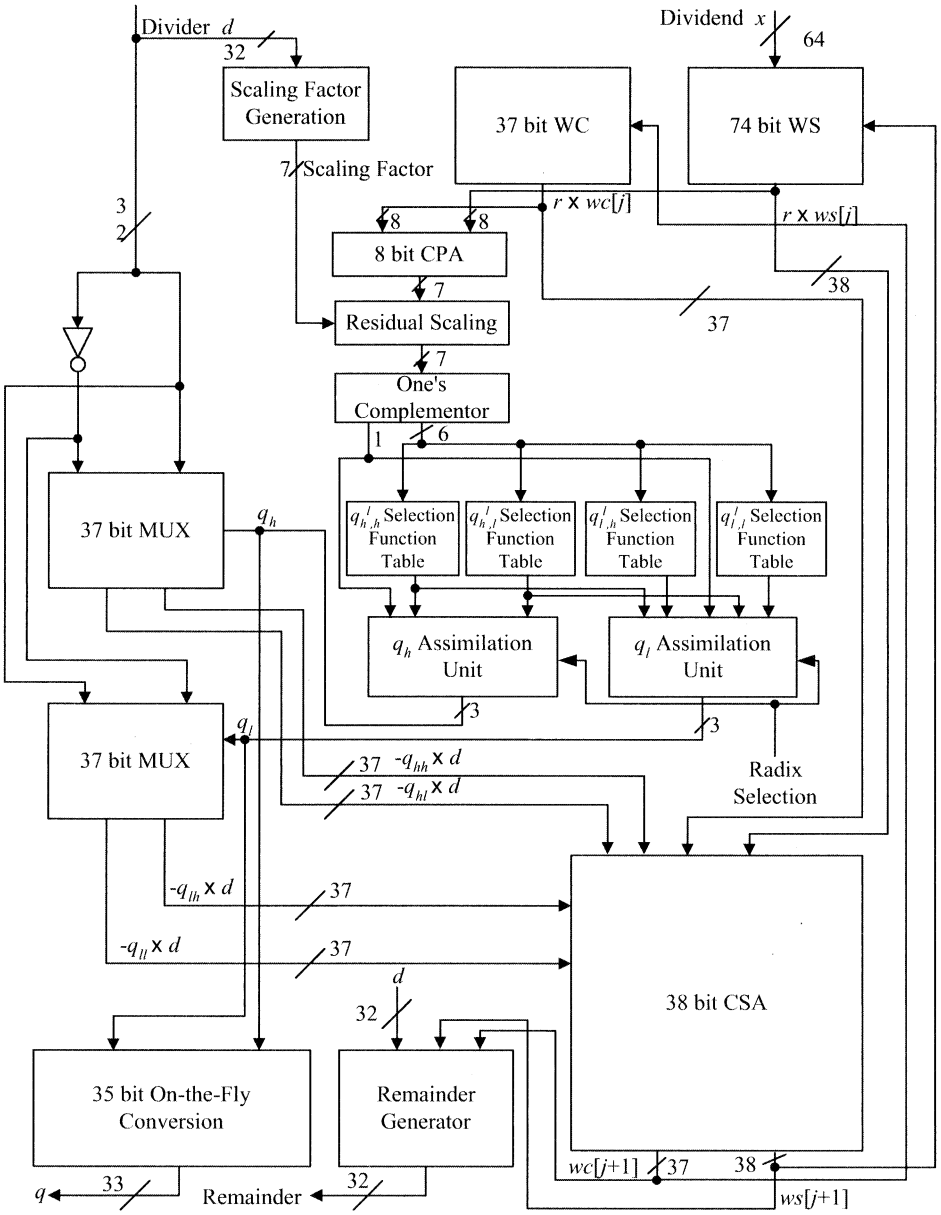

Fig. 2. Architecture of the radix 16/8/4/2 64-b/32-b integer divisor.

scheme needs only six bits, besides the common sign bit, as the input to the quotient-digit selection table, including five integer bits and one fractional bit of $\hat{y}$ in contrast to 11 bits required in the radix-8 division presented [4]. Thus, a total of seven bits, $\hat{y} = y_5y_4y_3y_2y_1y_0y_{-1}$ are used to derive $q^h_{h,h}$, $q^h_{h,l}$, $q^h_{l,h}$, $q^h_{l,l}$, $q^l_{h,h}$, $q^l_{h,l}$, $q^l_{l,h}$, and $q^l_{l,l}$ as follows:

$$\begin{aligned}
q^h_{h,h} &= q^h_{h,l} = q^h_{l,h} = q^h_{l,l} = y_5\\
q^l_{h,h} &= \bar{y}_4 y_3 + \bar{y}_4 y_2 y_1 y_0 y_{-1} + y_4 \bar{y}_3 \bar{y}_2 \bar{y}_1 \bar{y}_0 \bar{y}_{-1}\\
q^l_{h,l} &= \bar{y}_4 y_3 y_2 + \bar{y}_4 y_2 \bar{y}_1 + \bar{y}_4 y_2 \bar{y}_0 + \bar{y}_4 y_2 \bar{y}_{-1} + \bar{y}_4 \bar{y}_2 y_1 y_0 y_{-1}\\
&\quad + y_4 \bar{y}_3 \bar{y}_2 \bar{y}_1 \bar{y}_0 \bar{y}_{-1}\\
q^l_{l,h} &= \bar{y}_4 y_1 \bar{y}_0 + \bar{y}_4 y_1 \bar{y}_{-1} + \bar{y}_4 y_3 y_2 y_1 + \bar{y}_4 \bar{y}_1 y_0 \bar{y}_{-1}\\
&\quad + y_4 \bar{y}_3 \bar{y}_2 \bar{y}_1 \bar{y}_0 \bar{y}_{-1}\\
q^l_{l,l} &= \bar{y}_4 \bar{y}_0 y_1 + \bar{y}_4 y_0 \bar{y}_{-1} + \bar{y}_4 y_3 y_2 y_1 y_0 y_{-1} + y_4 \bar{y}_3 \bar{y}_2 \bar{y}_1 \bar{y}_0 \bar{y}_{-1}.
\end{aligned} \tag{3}$$

Notably, all four radixes can share the expressions given in (3) without any changes. The scenario requiring many quotient selection tables for different radixes appearing in the prior designs no longer exists.

C. Quotient Digit Assimilation Unit

The quotient digit assimilation unit performs the assimilation of the selections $q^l_{h,h}$, $q^l_{h,l}$, $q^l_{l,h}$, and $q^l_{l,l}$ into the single digit $q_{j+1} = q_h \cdot 4 + q_l$, where q_h is formed by assimilating $q^l_{h,h}$ and $q^l_{h,l}$, and q_l is created by assimilating $q^l_{l,h}$ and $q^l_{l,l}$. Meanwhile, the assimilation unit can also be shared by different radixes. q_h and q_l are functions of $q^h_{h,h}$, $q^h_{h,l}$, $q^h_{l,h}$, $q^h_{l,l}$, $q^l_{h,h}$, $q^l_{h,l}$, $q^l_{l,h}$, and $q^l_{l,l}$ for radix-16–radix-2 divisions , as shown in Table II.

The 38-bit carry-save adder (CSA) in Fig. 2 is fed with the shifted residual $-q_{hh} \cdot d$, $-q_{hl} \cdot d$, $-q_{lh} \cdot d$, $-q_{ll} \cdot d$, $r \times wc[j]$, and $r \times ws[j]$ to generate $wc[j+1]$ and $ws[j+1]$, where $q_{hh} + q_{hl} = q_h$ and $q_{lh} + q_{ll} = q_l$. Notably, q_{hh}, q_{lh}, q_{hl}, and q_{ll} must be numbers of two's power. For instance, a pre-computation of $3 \cdot d$ can be decomposed into $3 \cdot d = (2^0 + 2^1)d$ in one pass through the CSA and saved in registers

TABLE III
COMPARISON OF MIXED RADIX-4/2, MIXED RADIX-8/4/2, AOKI'S, AND PROPOSED MIXED RADIX-16/8/4/2 DIVISOR

	Mixed Radix-4/2 [5]	Mixed Radix-8/4/2 [4]	Aoki's [8]	Ours
CMOS Process	0.6 μm	0.6 μm	0.35 μm	0.35 μm
Chip Size	2.291 × 2.836 mm^2 (with PADs)	4.063 × 4.104 mm^2 (with PADs)	3.12 × 2.02 mm^2 (core only)	1.409 × 1.392 mm^2 (core only)
Area by Synopsys	9984.75	17468.76	N/A	19996.42
Number of cycles	23-3	18-3	17	13-3
Clock rate	50 MHz	66 MHz	300 MHz (32 stages pipelined)	80 MHz

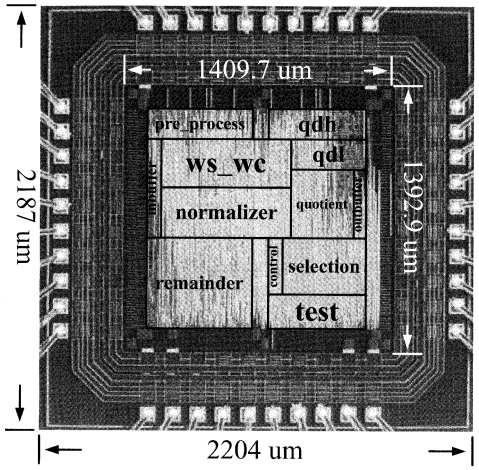

Fig. 3. Die photograph of the proposed integer divisor.

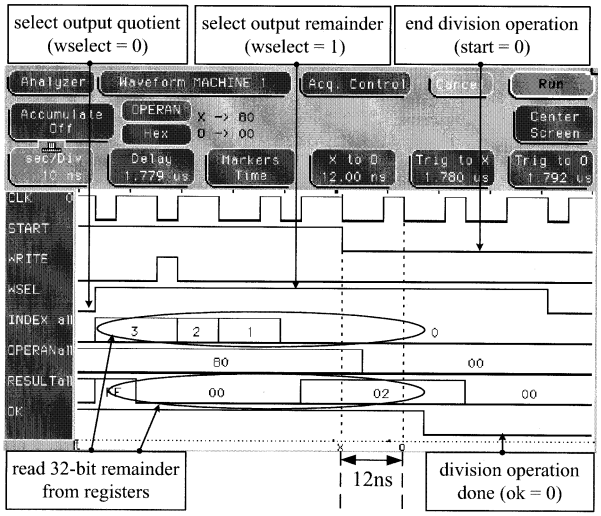

Fig. 4. Chip testing: output waveforms given a 80-MHz clock.

right after the divider d is scaled in every operation cycle. Multipliers will then not be required in our high-radix division implementations.

Note that the expressions of q_l and q_h for different radixes are identical, except for the cases that the higher order quotient digit is incorrectly set to one when the value of $\hat{y}$ is close to the bounds, as illustrated in the boundary rows of each radix in Table I. The complete scheme for the mixed radix-16/8/4/2 64-b/32-b integer divisor is presented in Fig. 2.

III. IMPLEMENTATION AND MEASUREMENT

The chip is implemented by synthesizable Verilog register transfer level (RTL) code and synthesized by Synopsys. Taiwan Semiconductor Manufacturing Company (TSMC) 0.35-μm 1P4M CMOS technology is employed to carry out the design, while CADENCE standard delay format (SDF) simulation tools are used to execute both the pre- and post-layout simulations. The highest working clock of this radix-16/8/4/2 64-b/32-b divisor is 80 MHz. Table III is the comparison of mixed radix-4/2, mixed radix-8/4/2, Aoki's high-radix divider [8], and our design. Fig. 3 shows a die photograph of the physical chip fabricated by TSMC. The area of the die is 2187 × 2204 μm^2. Fig. 4 is the measured results given by the HP 1660CP logic analyzer. Hence, this outcome addresses that our proposed design is silicon proven.

IV. CONCLUSION

In this paper, we have proposed a novel scheme to meliorate the performance of integer division. The methods that we propose include operand scaling, table folding, and table sharing to realize the mixed radix-16/8/4/2 quotient selection tables. A physical chip is implemented to prove our method on silicon. The results verify that our design saves operating cycles at a obscure increase of gate count.

ACKNOWLEDGMENT

The authors would like to express the highest appreciation to Taiwan Ericsson Company Ltd., Taiwan, R.O.C.

REFERENCES

[1] A. E. Bashagha and M. K. Ibeahim, "Two's complement high radix division," in *Int. Circuits Systems Symp.*, Hong Kong, June 1997, pp. 2088–2091.
[2] M. D. Ercegovac and T. Lang, *Division and Square Root—Digit-Recurrence Algorithms and Implementations*. Reading, MA: Kluwer, 1994.
[3] K. Hwang, *Computer Arithmetic: Principles, Architectures, and Design*. Reading, MA: Wiley, 1979.
[4] C.-C. Wang, C.-J. Huang, and I.-Y. Chang, "Design and analysis of radix-8/4/2 64b/32b integer divider using COMPASS cell library," *VLSI Syst. Des.*, vol. 11, no. 4, pp. 331–338, Dec. 2000.
[5] C.-C. Wang, C.-J. Huang, and G.-C. Lin, "Cell-based implementation of radix-4/2 64b/32b signed integer divider using COMPASS cell library," *Proc. Inst. Elect. Eng.*, vol. 147, pp. 109–115, Mar. 2000.
[6] P. Montuschi and L. Ciminiera, "Design of a radix 4 division unit with simple selection table," *IEEE Trans. Comput.*, vol. 41, pp. 1606–1611, Dec. 1992.
[7] A. Edelman, "The mathematics of the pentium division bug," *SIAM J.*, vol. 39, no. 1, pp. 54–67, Mar. 1997.
[8] T. Aoki, K. Nakazawa, and T. Higuchi, "High-radix parallel VLSI dividers without using quotient digit selection tables," in *Proc. 30th IEEE Int. Multiple-Valued Logic Symp.*, 2000, pp. 345–352.