Fast Ripple-Carry Adders in Standard-Cell CMOS VLSI

Neil Burgess
ARM Inc.
Austin, TX, USA

Abstract— **This paper presents a number of new high-radix ripple-carry adder designs based on Ling's addition technique and a recently-published expansion thereof. The proposed adders all have one inverting CMOS cell per stage along the carry-in to carry-out critical path and, at 16-b wordlengths, the fastest of them matches the speed of a 16-b prefix adder for only 63% of the area. These adders will be of use in VLSI circuits implementing modern wireless DSP algorithms and in Floating-Point Unit exponent logic, both of which typically use short wordlength arithmetic.**

Keywords-Adders, CMOS VLSI, Ling

I. INTRODUCTION

Modern wireless DSP algorithms often operate on data wordlengths of less than 16 bits, but at very high speed due to the high bit-rates required and the complexity of underlying algorithms (e.g. turbo decoding, MIMO processing etc). The highest speed adders attain logarithmic time complexity (e.g. parallel prefix adder structures due to Kogge-Stone, Sklansky and Knowles [1]) but at the shorter wordlengths found in wireless DSP are not significantly faster than ripple-carry adders for the amount of extra hardware – and extra power – deployed. So-called (4:2) compressors ("redundant binary adders") afford constant-time addition but are relatively large, comprising the equivalent of two full adders per bit. Moreover, many wireless DSP algorithms require comparisons of quite short operands, which are not performable using (4:2) compressors. Finally, redundant binary results require two bits' storage per digit, implying an even greater area overhead relative to ripple-carry adders once the associated increase in flop area is taken into account.

Little work has been published on accelerating ripple-carry adders by manipulating the underlying logic equations, as opposed to implementing the full adder in novel logic families, typically based on pass transistors [e.g. 2,3]. On this latter topic, Zimmermann [4] concluded in a review, "Complementary CMOS proves to be superior to all pass-transistor logic styles in performance for all logic gates… The advantages of efficient circuit and layout implementation of simple gates, the absence of swing restoration circuitry, and the single-rail logic property are predominant in most circuit applications. CMOS also shows the highest robustness and smallest sensitivity to transistor and voltage scaling." Therefore, in very deep submicron CMOS VLSI technology, novel circuit designs of logic gates are generally unsuccessful and acceleration of ripple-carry adders must be achieved by other means: namely, new logic designs.

In previous work, Grad and Stine [5] have shown how to design ripple-carry adders with the carry-in to carry-out delay reduced to one 2-input NAND gate per full adder. This paper proposes multi-bit full adder design, in which the carry-in to carry-out delay is also reduced to one CMOS cell but across more than one bit per higher-radix full adder. Also, Knowles [6] has proposed a radix-4 "2-bits-at-a-time" full adder design, incorporated into a long-wordlength carry-skip adder, with a 2-to-1 multiplexer per radix-4 full adder. The adders presented in this paper employ carry propagation cells which are faster than 2-to-1 multiplexers and radices which are greater than two. The fastest of these adders achieve the same delay as parallel prefix adders at wordlengths of 16 bits but for approaching only half the area and lower power consumption, due to their having fewer cells with smaller sized transistors and lower fan-outs.

This paper is organized as follows: the next Section introduces the three underlying topics of this paper: Ling's theory of addition [7]; CMOS logic cell characteristics; and ripple-carry adder organization and layout. Then, these three topics are combined by presenting two radix-2 ripple-carry adder reference designs, one using Ling's technique. The succeeding Section derives radix-4 and radix-8 Ling full adders, which provide new routes to accelerated short wordlength addition at low hardware cost. Finally, Jackson and Talwar's expansion of Ling's addition theory [8] is introduced in order to motivate two higher-radix NAND-based full adders, which are subsequently presented and compared with the other adders. All the adders were validated by a logical equivalence tool and assessed by being implemented in a contemporary 65nm CMOS VLSI technology. The paper concludes with a brief discussion of further possible avenues of work

II. FOUNDATIONS

A. Ling Addition

In 1981, Ling [7] observed that the i+1th carry output, c_{i+1}, of an adder operating on two binary words denoted a and b, may be simplified as follows:

1063-6889/11 $26.00 © 2011 IEEE
DOI 10.1109/ARITH.2011.23

$$
\begin{aligned}
c_{i+1} = G_{i:0} &= g_i + nk_i.g_{i-1} + nk_i.nk_{i-1}.g_{i-2} + \ldots \\
&= nk_i.g_i + nk_i.g_{i-1} + nk_i.nk_{i-1}.g_{i-2} + \ldots \\
&= nk_i.(g_i + g_{i-1} + nk_{i-1}.g_{i-2} + \ldots) \\
&= nk_i.H_{i:0} \qquad (1)
\end{aligned}
$$

where nk denotes "not kill", defined as $nk_i = a_i + b_i$, g_i denotes "generate", defined as $a_i.b_i$, and $G_{i:0}$ denotes "group generate" as a function of all the significances from bit i down to bit 0. $H_{i:0}$, which is $G_{i:0}$ with one nk bit factored out, is popularly known as a "Ling carry". Ling also showed that the corresponding sum bit, s_{i+1}, is derived as:

$$
\begin{aligned}
s_{i+1} &= p_{i+1} \oplus c_{i+1} \\
&= p_{i+1} \oplus G_{i:0} \\
&= p_{i+1} \oplus (nk_i.H_{i:0}) \\
&= H_{i:0}.(p_{i+1} \oplus nk_i) + !H_{i:0}.p_{i+1} \qquad (2)
\end{aligned}
$$

where p denotes "propagate", defined as $p_i = a_i \oplus b_i$. That is, the simplified carry signal, $H_{i:0}$, is connected to the select input of a 2-to-1 multiplexer so that the p⊕nk XOR gate is taken off the critical path of the adder. Vazquez and Antelo [9] have demonstrated that applying Ling's technique to logarithmic-depth parallel prefix adders in CMOS VLSI leads to an acceleration of approximately 1 FO4 ("fan-out of 4") inverter delay irrespective of adder wordlength or topology by replacing the first two levels of logic cells of an adder by one more complex gate.

B. CMOS VLSI Standard Cells

Contemporary deep-submicron CMOS VLSI designs deploy carefully-characterized standard cell libraries so as to conceal low-level physical micro-electronic phenomena from logic designers. Standard cells are typically laid out following a grid system in which polysilicon gates are arranged as regularly-spaced vertical strips with regions of doped silicon, contacts and metal layers placed so as to implement required logic functions [10]. All logic cells have the same height, and have widths of n+1 grid units, where n is the number of inputs to the logic gate being implemented by the cell. Thus, for example, an inverter has a width of 2 grids, a 2-input NOR gate a width of 3 grids, and a cell implementing the complex gate z = !(a.b + c.d) a width of 5 grids.

An inverter measuring 2 grids across with its n- and p-transistors made as large as possible within the cell height while maintaining a particular n:p FET width ratio (typically somewhere between 1:1.5 and 1:2) is deemed to have unit drive strength. Other cells will have more than one transistor in series in at least one of the p-FET or n-FET networks reducing the available output current from the cell, *as a function of the logic function being implemented*. For example, a 2-input NAND gate has two n-FETs in series and 2 p-FET's in parallel; the n-FET's will have the same size as those in the inverter cell, but the p-FET's will be made a little smaller so as to preserve the output rise and fall times relative to the inverter cell. This in turn means that the input FET gate terminal capacitance is reduced relative to the unit drive strength inverter cell.

Cells with larger drive strengths are made by increasing the number of polysilicon fingers per input. In these cells, the width measured in grid units is n.f+1, where f is the number of fingers per input of the logic gate being implemented by the cell and n as before is the number of inputs to the logic gate. The input capacitance increases moreorless linearly with the number of fingers per input.

Table 1 gives representative values of drive, slowest transition delay, and input capacitance, c_{in}, for the small number of cells to be utilized in this paper, normalized to a single-finger inverter cell, with the cells' widths in grids.

Table 1 Representative parameters of selected CMOS VLSI standard logic cells

Cell	Delay (FO4)	drive	width	c_{in}
NOT	1.0	1.00	2	1
NAND2	1.2	0.65	3	0.85
NOR2	1.3	0.55	3	0.90
AOI21	1.5	0.55	4	0.90
OAI21	1.5	0.55	4	0.90
AOI22	1.8	0.55	5	0.95
OAI22	1.8	0.55	5	0.95
XOR2	2.0	0.55	7	1.70
MUX2	1.6, 2.0(sel)	0.55	7	1.0, 1.95(sel)

Two composite cells, XOR2 and MUX2, are included in the Table. As will be seen, both these cells are only needed in single-finger realizations because they are employed solely in deriving output sum signals. Fractionally-fingered cells are also available, comprising smaller transistors than the integer-fingered cells, but these occupy the same area as the next largest integer-fingered cell.

C. Ripple-carry Adder Layout Methodology

All the adders discussed in this paper were designed using the cells described in the previous section, manually placed (using a proprietary layout tool) and then auto-routed, back-annotated and simulated at the worst-case process corner in a standard 65nm design flow. Also, all the adders were checked using a logical equivalence tool for correctness. The adders' inputs were driven by flip-flops with an output drive strength of 2, and the adders' sum outputs loaded by 2-finger inverters so as to match the drive strength of the flops driving the adders with the adders' output loads. The flip-flops had twice the height of the logic cells so that the adders were organized as shown in Figure 1.

The load capacitance due to a piece of interconnect traversing 1 bit of the adder (i.e. the height of a flop) was found to be the approximately same as the input capacitance of a single-finger unit drive inverter. Cell drive strengths were selected so that the ratio of load capacitance to drive strength was around 4, in accordance with theoretical good practice [10].

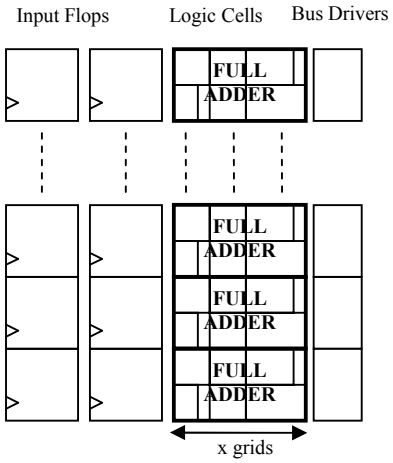

Figure 1 Layout of CMOS VLSI ripple-carry adders

III. BASELINE ADDERS

A. Radix-2 Ripple-Carry Adder

Figure 2 illustrates a radix-2 non-Ling ripple-carry adder, comprising two complementary full adder designs.

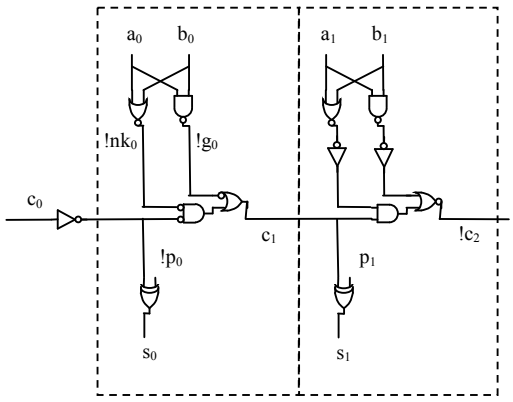

Figure 2 Simple radix-2 full adder designs

This is the simplest ripple-carry adder available using the cells listed in Table 1. There is an AOI21 or OAI21 gate per bit along the spine of the adder, each of which has a fan-out of just two other logic gates. The bit-level propagate signals, p and !p, are derived from g and nk according to p = nk.!g (or its complement !p = !nk + g), saving 2 grids relative to using an xor cell, with no adverse effect on speed. The sum xor gates are single-finger cells with a drive strength of 0.55, as this gives a value for load/drive of these output gates of 2.0/0.55 = 3.64. The drive strengths of the input NAND2, NOR2 and NOT cells are selected according to their total loads. The number of fingers needed in the c_n AOI21 (or OAI21) gates is calculated by determining the load capacitance on these gates and setting the load/drive ratio to 4, as follows:

load/drive = [1.0(wire) + 1.7(xor) + 0.85.f(aoi)]/0.55.f = 4 $\rightarrow$ f = 2.0

Hence, the AOI21/OAI21 gates should be 2-finger cells. Alternatively, if a half-strength inverter were to be inserted between c_n and the xor gate input, so as to reduce the load capacitance on the AOI21 (or OAI21 gates), the required number of fingers reduces:

load/drive = [1.0 + 0.5 + 0.85.f]/0.55.f = 4 $\rightarrow$ f = 1.1

However, this increases the delay of a ripple-carry adder made from these cells by one inverter delay for an area saving of one grid, which is not a worthwhile trade-off.

B. Radix-2 Ling Ripple-Carry Adder

Grad and Stine [5] have shown how a radix-2 ripple-carry adder can be designed with a 2-input NAND gate per full adder on the carry spine by using Ling's technique. Consider a carry propagating across two bits of an addition:

$$c_{i+1} = G_{i:0} = g_i + nk_i.g_{i-1} + nK_{i:i-1}G_{i-2:0} \quad (3)$$

Then, using Ling's factorization, the following expression is obtained:

$$\begin{aligned} G_{i:0} &= nk_i.g_i + nk_i.g_{i-1} + nK_{i:i-1}.nk_{i-2}.G_{i-2:0} \\ &= nk_i.(H_{i:i-1} + nK_{i-1:i-2}.G_{i-2:0}) \end{aligned} \quad (4)$$

The resulting ripple-carry adder design is shown in Figure 3, where $H^*_{x:c_0} = nk_x.G_{x-1:c_0}$ denotes a carry with an extra not kill bit, nk_x, AND-ed in. Note how the adder's carry input, c_0, has been combined with a_0 and b_0 in parallel with the AOI22 and OAI22 cells producing $!H_{i:i-1}$ and $!nK_{i+1:1}$ respectively, effectively reducing the adder's critical path by one bit.

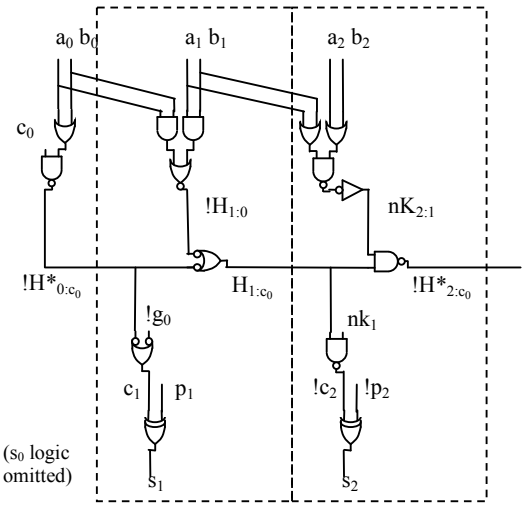

Figure 3 Radix-2 Ling full adder designs

The drive strength and thus physical size of the c_n NAND2 gate is determined as:

$[1.0(\text{wire}) + 0.85 + 0.85.f(\text{NAND2})]/0.67.f = 4 \rightarrow f = 1$

As noted in Section 2, Ling also identified a method for removing the extra logic gate needed to AND the missing nk bit back into the expression for the sum bit from the adder's critical path. The output XOR gate is replaced by a multiplexer whose select input is driven by the "reduced" carry signal, and whose two data inputs are speculatively-calculated sum bits:

$$\begin{aligned} s_n &= (nk_{n-1}.H_{n-1:0}) \oplus p_n \\ &= !H_{n-1:0}.(p_n) + H_{n-1:0}.(nk_{n-1} \oplus p_n) \end{aligned} \quad (5)$$

This technique is directly applicable in this adder to alternate sum bits. The same technique can also be applied to the intervening bits, as follows:

$$\begin{aligned} s_{n+1} &= (g_n + H^*_{n-1:0}) \oplus p_{n+1} \\ &= (g_n + nk_{n:n-1}.H_{n-1:0}) \oplus p_{n+1} \\ &= !H_{n-1:0}.(g_n \oplus p_{n+1}) + H_{n-1:0}.\{(g_n + nk_{n:n-1}) \oplus p_{n+1}\} \end{aligned} \quad (6)$$

In the context of a ripple-carry adder, this technique only needs to be applied to the msb full adder, accelerating the speed of the ripple-carry adder by a 2-input NAND delay for an increase in logic in the adder of two 2-to-1 multiplexers. Moreover, this extra logic can be placed above the msb full adder in the layout without affecting the width of the design (see Figure 1). This adder is faster than the first radix-2 adder because a 2-input NAND gate is inherently faster than a 3-input AND-OR-INVERT gate.

Logical Effort is a well-established technique for estimating the delays of digital CMOS VLSI circuits [10]. Using Logical Effort to estimate the delays of the two adders presented in this paper as a function of w, the adders' wordlength, yields a Path Effort, F, and a total Parasitic Delay, P, of:

$$F = 3.65 \times 3.95 \times (6.25)^{w-1} = 2.31 \times (6.25)^w$$
$$P = 7 + (w-1) \cdot 5/2$$

(For this analysis, the logical effort of an individual cell, g, has been taken from Table 1 as $g = c_{in}/\text{drive}$; see [11] for details of the method used.) Then, the delay of a w-bit non-Ling adder is:

$$N = \log_4 F = 1.32 \cdot w + 0.60; \alpha \equiv 4.0$$
$$D = (N\alpha + P)/5 = (4 \times (1.32 \cdot w + 0.6) + 4.5 + 2.5 \cdot w)/5$$
$$= 1.56 \cdot w + 1.38 \text{ FO4 delays}$$

Performing the same analysis for the radix-2 Ling adder yields:

$$F = \Pi g \cdot b = 2.3 \times 5.9 \times (4.1)^{w-2} = 0.81 \times (4.1)^w$$
$$P = 9 + 2w$$
$$N = \log_4 F = 1.02 \cdot w - 0.15; \alpha \equiv 4.0$$
$$D = (N\alpha + P)/5 = (4 \times (1.02 \cdot w + 0.77) + 7 + 2w)/5$$
$$= 1.22 \cdot w + 2.02 \text{ FO4 delays}$$

The Logical Effort analysis confirms that the Ling ripple-carry adder based on NAND2 cells is 1.56/1.22 or 28% per bit faster. Further acceleration is available by applying Ling's technique to ripple-carry adders with higher-radix full adders.

IV. HIGHER-RADIX LING ADDERS

A. Radix-4 Ling Ripple-Carry Adder

Figure 5 illustrates a radix-4 Ling ripple-carry adder, again comprising a pair of complementary full adder designs. As with the radix-2 baseline design, there is an AOI21/OAI21 gate per full adder along the critical carry path; in common with the radix-2 Ling adder, the inputs to the carry propagation gates are driven by AOI22/OAI22 cells and additional logic is required to derive the even-numbered carry bits in the sum logic. Note that a half-strength inverter has been inserted to drive H (or !H) to the sum logic, reducing the load capacitance on the AOI21/OAI21 carry gates.

The number of fingers in the c_n AOI21/OAI21 gates is calculated by determining the load capacitance and setting the load/drive ratio to 4, as before:

$$[2.0(\text{wire}) + 0.5(\text{inv}) + 0.85 \cdot f]/0.55 \cdot f = 4 \rightarrow f = 1.85$$

(Repeating this calculation but without the inverter separating the carry from the two bits of sum logic yields $f \approx 3$ due to the extra cell and the increased wiring load.)

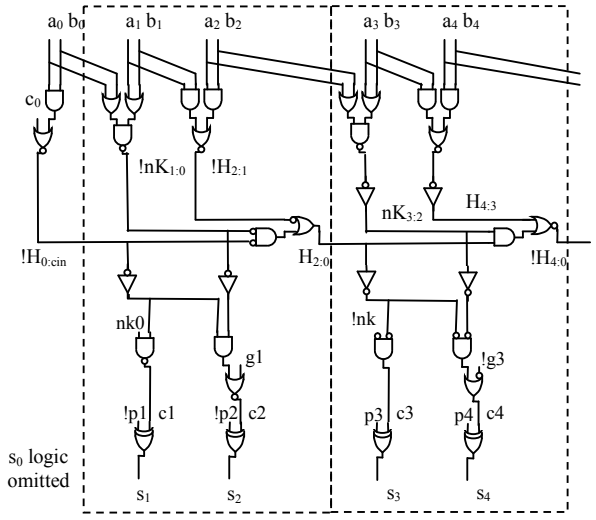

Figure 5 Radix-4 Ling ripple-carry adder

Logical Effort analysis of this adder gives:

$$F = 174.4 \times (5.15)^{\lfloor w/2 \rfloor - 2} = 6.57 \times (5.15)^{\lfloor w/2 \rfloor}$$
$$P = 10 + 5/2 \cdot (\lfloor w/2 \rfloor - 2)$$
$$N = \log_4 F = 1.18 \cdot \lfloor w/2 \rfloor + 1.36; \alpha \equiv 4.0$$
$$D = (4 \times (1.18 \cdot \lfloor w/2 \rfloor + 1.36) + 5 + 5/2 \cdot \lfloor w/2 \rfloor)/5$$

$= 1.44 \cdot \lfloor w/2 \rfloor + 2.09$ FO4 delays

Thus, changing the radix from 2 to 4 provides a further significant acceleration over either of the radix-2 adders for only a relatively small increase in logic.

B. Radix-8 Ling ripple-carry adder

Ling's factorization can also be applied to a radix-8 ripple-carry adder: Figure 6 illustrates one of a pair of complementary "3-bits-at-a-time" full adders. Note how the input logic has increased in depth by a logic stage.

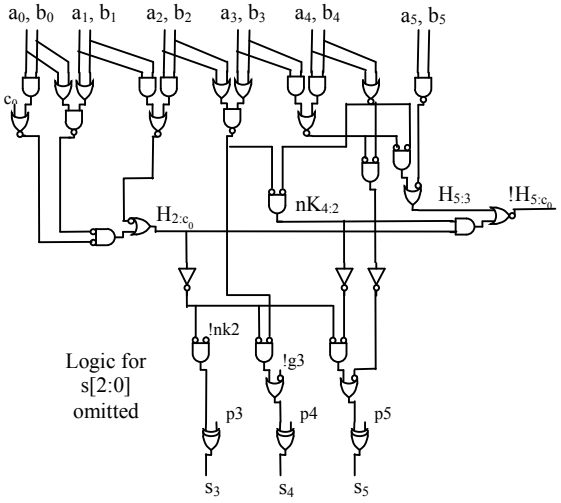

Figure 6 Radix-8 Ling full adder design

The number of fingers in the c_n AOI21/OAI21 gates is calculated by determining the load capacitance and setting the load/drive ratio to 4, as before:

$$[3.0(\text{wire}) + 1.0(\text{inv}) + 0.85.f]/0.55.f = 4 \rightarrow f = 3$$

This is a large cell, measuring 3×3+1 = 10 grids across. Logical Effort analysis yields the following expression for the adder's delay:

$$F = 335.8 \times (6.15)^{\lfloor w/3 \rfloor - 2}$$
$$P = 12.7 + 5/2 \cdot (\lfloor w/3 \rfloor - 2)$$
$$N = \log_4 F = 1.31 \cdot \lfloor w/3 \rfloor + 1.83;\ \alpha \equiv 4.0$$
$$D = (4 \times (1.31 \cdot \lfloor w/3 \rfloor + 1.83) + 7.7 + 5/2 \cdot \lfloor w/3 \rfloor)/5$$
$$= 1.55 \cdot \lfloor w/3 \rfloor + 3.0 \text{ FO4 delays}$$

This is in turn significantly faster than the radix-4 adder despite the increase in both the complexity and the amount of logic required to implement the adder.

V. HIGHER-RADIX JACKSON ADDERS

Recently, Jackson and Talwar [2] have shown that more than one not kill bit can be removed from the basic carry expression as:

$$G_{i:0} = D_{i:i-m+1}.R^{(m)}_{i:0} \quad (7)$$

where $R^{(m)}_{i:0}$ denotes a "reduced" carry signal that has had m not kill bits removed, and which Jackson and Talwar call "sub-generate". D is an m-bit correction term that restores the full carry signal, $G_{i:0}$ from the reduced one, and Jackson and Talwar have derived the following expression for $D_{i:i-m}$:

$$D_{i:i-m+1} = G_{i:i-m+1} + nK_{i:i-m+1} = G_{i:i-m+2} + nK_{i:i-m+1} \quad (8)$$

where $nK_{i:i-m+1}$ denotes "group not kill" as a function of all the significances from bit i down to bit i-m+1, defined as:

$$nK_{i:i-m+1} = nk_i.nk_{i-1}.nk_{i-2}\ldots nk_{i-m+1} \quad (9)$$

In a similar manner to Ling addition, the sum bit is obtained using a 2-to-1 multiplexer:

$$\begin{aligned} s_{i+1} &= p_{i+1} \oplus c_{i+1} = p_{i+1} \oplus G_{i:0} \\ &= p_{i+1} \oplus \{D_{i:i-m+1}.R^{(m)}_{i:0}\} \\ &= R^{(m)}_{i:0}.\{p_{i+1} \oplus D_{i:i-m+1}\} + !R^{(m)}_{i:0}.p_{i+1} \end{aligned} \quad (10)$$

Ling addition can now be seen as a special case of Jackson and Talwar's technique with m = 1. Many of Jackson and Talwar's factorizations contain terms of the form nK.D. This product is denoted Q and defined as a "hyper-propagate", as follows:

$$Q^{(m)}_{i:j} = nK_{i:i-m+1}.D_{i-m:j} \quad (11)$$

where the superscript, m, denotes the number of bits in the leading group not kill term, nK. This leads directly to the following expressions for R (all but one containing Q):

Valency-2

$$R^{(i-j+m)}_{i:0} = R^{(m)}_{i:j+1} + R^{(m)}_{j:0}$$
$$R^{(m)}_{i:0} = R^{(m)}_{i:j+1} + Q^{(i-m-j)}_{i-m:j+1-m}.R^{(m)}_{j:0} \quad (i \geq j)$$

Valency-3

$$R^{(i-k+m)}_{i:0} = R^{(m)}_{i:j+1} + R^{(m)}_{j:k+1} + R^{(m)}_{k:0}$$
$$R^{(i-j+m)}_{i:0} = R^{(m)}_{i:j+1} + R^{(m)}_{j:k+1} + Q^{(j-m-k)}_{j-m:k+1-m}.R^{(m)}_{k:0}$$
$$R^{(m)}_{i:0} = R^{(m)}_{i:j+1} + Q^{(i-m-j)}_{i-m:j+1-m}.R^{(m)}_{j:k+1} + Q_{i-m:j+1-m}.Q^{(j-m-k)}_{j-m:k+1-m}.R^{(m)}_{k:0} \quad (i \geq j \geq k)$$

A. Radix-4 Jackson ripple-carry adder

Jackson and Talwar's factorization yields a radix-4 ripple-carry adder design in which the carry propagation delay is reduced to a 2-input NAND gate per *two* bits of the adder, faster than the radix-4 adder presented in the previous Section and approximately the same speed as the Ling radix-8 adder. The pair of complementary designs is presented in Figure 7 and both make use of the following factorization:

$$\begin{aligned} R^{(2)}_{n+4:0} &= R^{(2)}_{n+4:n+1} + Q^{(2)}_{n+2:n-1}.R^{(2)}_{n:0} \\ &= R^{(2)}_{n+4:n+1} + R^{(-2)}_{n+2:0} \end{aligned} \quad (12)$$

where:

$$R^{(2)}_{n+4:n+1} = R^{(1)}_{n+4:n+3} + nk_{n+2}.R^{(1)}_{n+2:n+1} \quad (13a)$$

$$R^{(-2)}_{n+4:n+1} = nK_{n+4:n+3}.nk_{n+2}.R^{(1)}_{n+2:n+1} \quad (13b)$$

and

$$Q^{(2)}_{n+2:n-1} = Q^{(1)}_{n+2:n+1}.(g_n + Q^{(1)}_{n:n-1}) \quad (13c)$$

Then, alternating NAND2 cells along the carry spine implement and drive $R^{(2)}$ and $R^{(-2)}$ signals across the radix-4 full adders. The sum bits are more difficult to derive from the partial carries than before:

$$\begin{aligned} s_{n+1} &= p_{n+1} \oplus (D_{n:n-1}.R^{(2)}_{n:0}), \qquad D_{n:n-1} = g_n + nK_{n:n-1} \\ s_{n+2} &= p_{n+2} \oplus (g_{n+1} + nk_{n+1}.D_{n:n-1}.R^{(2)}_{n:0}) \\ s_{n+3} &= p_{n+3} \oplus (G_{n+2:n+1} + R^{(-2)}_{n+2:0}) \\ s_{n+4} &= p_{n+4} \oplus (g_{n+3} + nk_{n+3}.G_{n+2:n+1} + nk_{n+3}.R^{(-2)}_{n+2:0}) \end{aligned} \quad (14)$$

The drive strength and thus physical size of the NAND2 gates is determined in the usual manner:

$$[2.0(\text{wire}) + 1.0(\text{NOT}) + 0.85.f]/0.67.f = 4 \rightarrow f = 1.6$$

This value of f shows that a faster adder (with f = 2 for the NAND2 gates) is available for no increase in area relative to an adder with f = 1.5 NAND2 gates. However, the logic gates implementing $Q^{(2)}_{n+2:n-1}$ and $R^{(2)}_{n+4:n+1}$ driving the NAND2 gates will be a little slower due to the increased gate capacitance of the NAND2 gates.

According to simulation results, using the NAND gates with the larger value of f reduces the delay per NAND2 gate by 0.1 FO4, but at the expense of increasing the delay of the Q and R gates by 0.2 FO4. This latter increase in delay is more than compensated however by a reduction in the sum logic delay by 0.3 F04, due to the faster edge rates of the NAND2 gates' output transitions.

If Ling's sum logic optimization is applied to the two most significant bits of a radix-4 adder, the critical path goes through the third most significant sum bit of the adder. This is because the AOI21 gate in the sum logic is slower than the NAND2 gate on the carry spine. This could be fixed by applying Ling's optimization to this bit as well, but would be awkward to lay out as there is no "white space" available in the ripple-carry adder floorplan where the extra logic can fit.

Logical Effort analysis yields the following expression for the adder's delay:

$$\begin{aligned} F &= 933.4 \times (4.8)^{\lfloor w/2 \rfloor - 3} = 8.44 \times (4.8)^{\lfloor w/2 \rfloor} \\ P &= 15 + 2 \cdot (\lfloor w/2 \rfloor - 3) \\ N &= \log_4 F = 1.13 \cdot \lfloor w/2 \rfloor + 1.54; \ \alpha \equiv 4.0 \\ D &= (4 \times (1.13 \cdot \lfloor w/2 \rfloor + 1.54) + 9 + 2 \cdot \lfloor w/2 \rfloor)/5 \\ &= 1.30 \cdot \lfloor w/2 \rfloor + 3.03 \text{ FO4 delays} \end{aligned}$$

This is significantly faster than the radix-4 Ling adder for little extra logic complexity due mainly to the use of NAND2 gates along the ripple-carry chain.

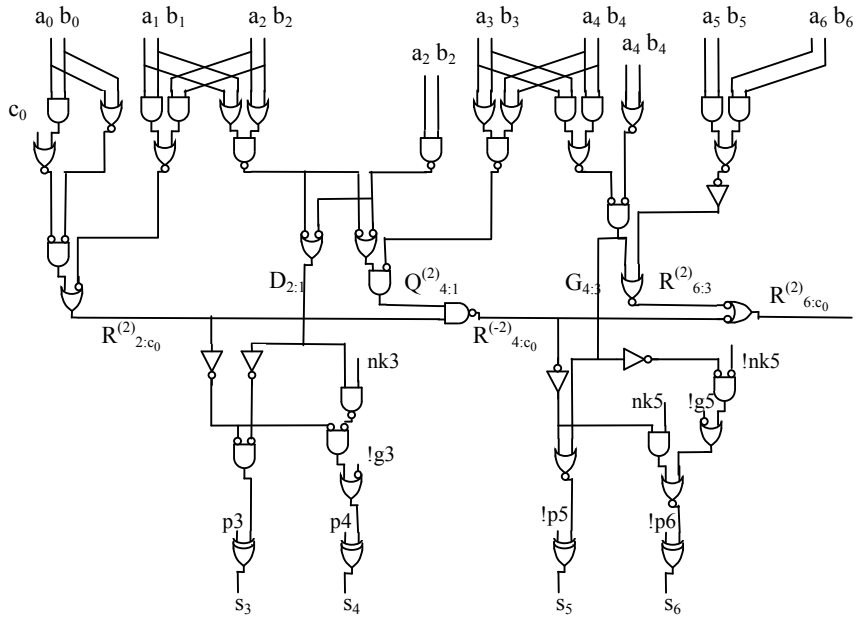

Figure 7 Radix-4 Jackson full adder design

B. Radix-8 Jackson ripple-carry adder

Having seen that Jackson and Talwar's factorization leads to a very fast radix-4 ripple-carry adder with a delay of one 2-input NAND gate per two bits, it seems worthwhile to investigate a radix-8 full adder design with a 2-input NAND gate as the carry propagation gate. In a similar manner to before, the following expressions are obtained:

$$R^{(3)}_{n+6:0} = R^{(3)}_{n+6:n+1} + Q^{(3)}_{n+3:n-2}.R^{(3)}_{n:0} = R^{(3)}_{n+6:n+1} + R^{(-3)}_{n+3:0} \quad (15)$$

where:

$$R^{(3)}_{n+6:n+1} = R^{(1)}_{n+6:n+5} + R^{(1)}_{n+4:n+3} + Q^{(1)}_{n+3:n+2}.R^{(1)}_{n+2:n+1} \quad (16a)$$

$$R^{(-3)}_{n+3:n-2} = nK_{n+3:n+2}.nK_{n+1:n}.(R^{(1)}_{n:n-1} + nK_{n-1:n-2})R^{(1)}_{n+2:n+1} \quad (16b)$$

and

$$Q^{(3)}_{n+3:n-2} = Q^{(1)}_{n+3:n+2}.Q^{(1)}_{n+1:n}.(R^{(1)}_{n:n-1} + Q^{(1)}_{n-1:n-2}) \quad (16c)$$

Then, alternating NAND2 cells along the carry spine implement and drive $R^{(3)}$ and $R^{(-3)}$ signals across the radix-8 full adders. The expressions for $R^{(3)}_{n+6:n+1}$ and $Q^{(3)}_{n+3:n-2}$ are more complicated than those for any other adder's input logic but both are directly implementable using 4-input CMOS cells not previously considered.

Table 3 gives the parameters of these CMOS cells from which it is clear the input logic is slower than for previous adders because OAI/AOI211 gates are slower than other cells. Moreover, OAI/AOI211 gates have a low drive strength (40% that of a single-finger inverter) due to their having three transistors in series in either their p-FET or n-FET networks.

Table 3 Representative parameters of CMOS VLSI OAI/AOI211 logic cells

Gate	Delay (FO4)	Drive	width	c_{in}
NOT	1.0	1.00	2	1
AOI211	2.1	0.40	5	0.90
OAI211	2.0	0.40	5	0.85

The drive strength and thus physical size of the NAND2 gates is determined in the usual fashion:

$$[3.0(\text{wire}) + 1.5(\text{NOT}) + 0.85.f]/0.67.f = 4 \rightarrow f = 2.5$$

Again it looks worthwhile to use f=3 NAND2 gates, for no area increase, in order to enhance the adder's performance. But, in order to drive f=3 NAND2 gates, which have input gate capacitances of 3×0.85 = 2.55, either f=2 OAI/AOI211 gates are required, with a significant area overhead (4 grids), or f=1 inverters need to be inserted between the OAI/AOI211 gates and the NAND2 gates, adding nearly 1 FO4 delay to the adders' critical path. Either it takes a long time to get the carry launched along this adder, or the weakly-driven NAND gates along the critical path will have extended rise and fall times due to relatively shallow slopes of the carry signals. A straightforward alternative is to replace the "carry launch" logic up to the NAND gate producing $R^{(-3)}_{7:c_0}$ by the equivalent logic from the radix-4 adder. Then, the first carry produced by a NAND2 at the bottom end of the adder is $R^{(-2)}_{4:c_0}$. The succeeding NAND2 gate can OR in the 6-bit partial carry $R^{(3)}_{7:2}$ according to:

$$R^{(3)}_{7:c_0} = R^{(3)}_{7:2} + R^{(-2)}_{4:c_0} = H_{7:6} + H_{5:4} + nK_{4:3}H_{3:2} + Q_{4:1}R^{(2)}_{2:c_0} \quad (17)$$

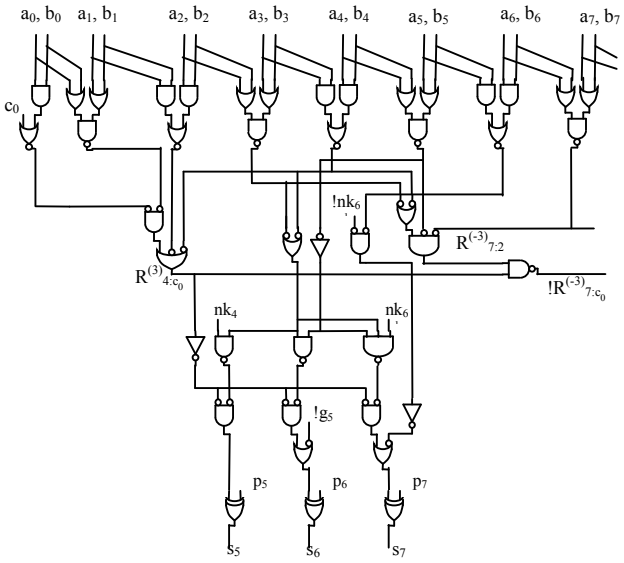

Figure 8 Radix-8 Jackson full adder design

The sum bits are derived as follows, and also require ever more complex logic than the other ripple-carry adders considered in this paper:

$$s_{n+1} = p_{n+1} \oplus (D_{n:n-2}.R^{(3)}_{n:0})$$
$$s_{n+2} = p_{n+2} \oplus (g_{n+1} + nk_{n+1}.D_{n:n-2}.R^{(3)}_{n:0})$$
$$s_{n+3} = p_{n+3} \oplus (G_{n+2:n+1} + nK_{n+2:n+1}.D_{n:n-2}.R^{(3)}_{n:0})$$
$$s_{n+4} = p_{n+4} \oplus (G_{n+3:n+1} + R^{(-3)}_{n+3:0})$$
$$s_{n+5} = p_{n+5} \oplus (G_{n+4:n+1} + nk_{n+4}.R^{(-2)}_{5:0})$$
$$s_{n+6} = p_{n+6} \oplus (G_{n+5:n+1} + nK_{n+5:n+4}.R^{(-2)}_{5:0}) \quad (18)$$

However, closer inspection of these sum equations shows that, in every case, the critical path from $R_{n:0}$ to s_{n+1} can be reduced to an AOI21/OAI21 cell followed by an XOR (see Figure 8).

VI. Simulation Results and Discussion

All the adders presented in this paper were assessed by being placed semi-manually using a 65nm CMOS cell library, auto-routed, and timing analyzed. Table 4 presents the width of each adder (in grids, excluding the width of the flops – see Figure 1) along with the delay of the logic to the inputs of the first gate on the carry chain, the delay of the carry gate itself, and the delay through the output sum logic.

Figure 9 presents the data from Table 4 as a graph. The radix-4 and -8 adders are markedly faster than the radix-2 adders with both the Jackson adders proving faster than the Ling adders by virtue of their employing NAND2 gates.

By way of comparison, the delay and area of a 16-bit Ladner-Fisher adder designed using the same design flow had a delay of 11.0 FO4, the same as a 16-b radix-8 Jackson ripple adder, but a maximum width of 38 grids across the msb half of the adder.

Table 4 Constituent delays of radix-4 and radix-8 ripple-carry adders

	Delays (FO4 units)				
Adder	Input logic	Carry gate	Sum logic	w-bit adder	Width (grids)
R2	1.19	1.45	2.05	1.45w + 1.79	14
R2-L	1.54	1.16	2.05*	1.16w + 1.27	14
R4-L	1.81	1.46	3.06*	$1.46\lfloor w/2 \rfloor + 3.41$	17
R8-L	3.27	1.56	3.04*	$1.56\lceil w/3 \rceil + 4.75$	22
R4-J	3.38	1.03	4.24**	$1.03\lfloor w/2 \rfloor + 4.53$	21
R8-J	3.38	1.09	4.27**	$1.09\lfloor (w-1)/3 \rfloor + 5.55$	24

* with Ling output logic at msb
** critical path does not traverse msb sum bit

The prefix adder is larger not only because it uses more cells than a ripple-carry adder but also because many of its cells are multi-fingered so as to drive larger fanouts (i.e. extra cells, longer wires, and larger multi-finger FET gate capacitance loads) than in a ripple-carry adder. Also, a 16-b Kogge-Stone adder using the same design flow had a delay of 10.5 FO4 and measured 46 grids across. At radix-16 and above, however, these high-radix ripple-carry-adders adders are unlikely to be as successful due to the ever-increasing wire load along the carry spine, necessitating buffer insertion on the critical path, and the increasing complexity of the input and output logic, with reduced performance. (see Figure 8).

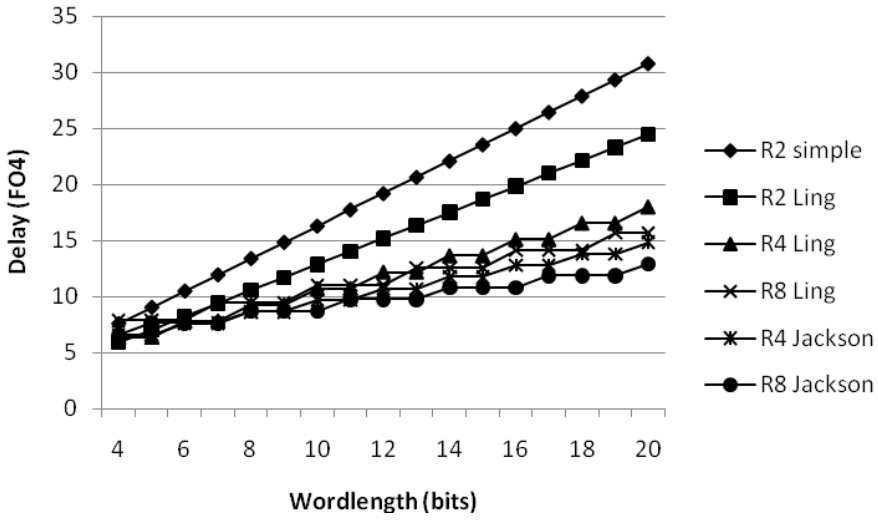

Figure 9 Simulated adder delays

VII. SUMMARY

This paper has introduced a number of new higher-radix ripple-carry adder designs based on Ling's factorization and its expansion [8] implementable in standard cell CMOS VLSI. The fastest adders are competitive in speed with 16-b parallel prefix adders for significantly less area and consume less power by virtue of comprising fewer cells with smaller transistors than those used in prefix adders. It would be interesting to see how well these adders synthesize using automated rather than structured placement techniques: because there are only a small number of critical paths to be optimized, an EDA tool might well find layouts which minimize wire capacitance along the critical path as well as logic cell delay. Finally, these adders might also prove advantageous in building hybrid sparse-tree parallel prefix adders, or higher-radix carry-skip adders with fewer "skip" stages.

ACKNOWLEDGMENT

The author is grateful to S.K.Mathew for valuable feedback during the preparation of the final version of this paper .

REFERENCES

[1] S. Knowles, "A Family of Adders", Proc. 14th IEEE Computer Arithmetic Symposium, Adelaide, 1999, pp. 277 – 281

[2] J-F. Lin, M-H. Sheu and Y-T Hwang, "Low-Power and Low-Complexity Full Adder Design for Wireless Base Band Application", Proc. IEEE Int. Conf. on Communications, Circuits and Systems, Guilin, China, 2006, Vol. 4, pp. 2337 - 2341

[3] X. Zeng and P. Wang, "Design of low-power Complementary Pass-Transistor and ternary adder based on multi-valued switch-signal theory", Proc. 8th IEEE International Conference on ASIC, (ASICON '09), Changsha, China, 2009, pp. 851 - 854

[4] R. Zimmermann and W. Fichtner, "Low-power logic styles: CMOS versus pass-transistor logic", *IEEE J. Solid-State Circuits*, Vol. 32, (July 1997), pp. 1079 - 1090

[5] J. Grad and J.E. Stine, "New algorithms for carry propagation", Proc. 15th ACM Great Lakes symposium on VLSI, Chicago, 2005, pp. 396 - 399

[6] S. Knowles, "Arithmetic processor design for the T9000 transputer", Proc. SPIE, Vol. 1566, San Diego, 1991, pp. 230 - 243.

[7] H. Ling, "High-Speed Binary Adder", *IBM J. Research & Dev*., vol 25, p.156-166 (May 1981)

[8] R. Jackson and S. Talwar, "High-Speed Binary Addition", Proc. 38th IEEE Asilomar Conference on Systems, Signals, and Computers, Asilomar, CA, 2005, 1350 – 1353

[9] A. Vazquez and E. Antelo, "New insights on Ling adders", Proc. IEEE Int. Conf. on Application-Specific Systems, Architectures and Processors, Leuven, 2008 pp. 227 – 232

[10] N. Weste and D. Harris, *CMOS VLSI Design*, 4/e, Addison-Wesley, 2010

[11] N. Burgess, "New Models of Prefix Adder Topologies", *J. VLSI Signal Processing Systems*, Vol. 40, (June 2004), pp. 125 – 141.