Optimum Prefix Adders in a Comprehensive Area, Timing and Power Design Space

Jianhua Liu, Yi Zhu, Haikun Zhu, Chung-Kuan Cheng
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
{jhliu, y2zhu, hazhu, kuan}@cs.ucsd.edu

John Lillis
Department of Computer Science
University of Illinois at Chicago
Chicago, IL 60607
jlillis@cs.ucsd.edu

***Abstract*— Parallel prefix adder is the most flexible and widely-used binary adder for ASIC designs. Many high-level synthesis techniques have been developed to find optimal prefix structures for specific applications. However, the gap between these techniques and back-end designs is increasingly large. In this paper, we propose an integer linear programming method to build minimal-power prefix adders within given timing and area constraints. It counts both gate and wire capacitances in the timing and power models, considers static and dynamic power consumptions, and can handle gate sizing and buffer insertion to improve the performance further. The proposed method is also adaptive for non-uniform arrival time and required time on each bit position. Therefore our method produces the optimum prefix adder for realistic constraints.**

I. Introduction

Binary addition is the most fundamental and frequently used arithmetic operation in datapath design. Among the adder structures that have been proposed, parallel prefix adder provides remarkable flexibility. Based on the formulation of prefix computation several regular prefix adders have been developed. These classic prefix networks include Sklansky [1], Kogge-Stone [2] and Brent-Kung [3] adders. They achieve three extreme cases: minimal logic levels and wire tracks, minimal max-fanout and logic levels, and minimal wire tracks and max-fanout, respectively. In addition, Ladner-Fischer [4], Han-Carlson [5] and Knowles [6] implemented the tradeoff between each pair of the extreme cases. Therefore, the design space of prefix adders is usually viewed as a triangle with three vertices: max-fanout, wire tracks, and logic levels, as shown in figure1(a) [7]. Besides the regular structures, a few irregular networks are also proposed to explore the solution space. Zimmermann developed a non-heuristic algorithm [8] with the capability to handle non-uniform input arrival times. Zero-deficiency adder [9] is defined and constructed to reach the lower-bound of number of components for given number of logic levels.

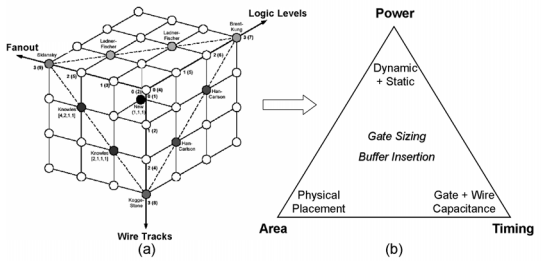

Fig. 1. Views of the Prefix Adder Design Space

Although the design space of prefix adder seems to have been well-studied, the common timing/area model in these previous works is still idealistic. The concept of logic levels is used to estimate timing, area, and even input arrival time. However, the real delay is not proportional to the number of logic levels. It highly depends on gate size and total load capacitance including both gate capacitance and wire capacitance. Furthermore, power consumption becomes an important design issue which is not studied in previous works. These changes demand a comprehensive timing/area/power model and a new methodology for prefix adder synthesis, to reduce the gap between high-level synthesis and back-end design. Therefore, the new model should incorporate physical placement to accurately estimate area and wire capacitance, count both gate and wire capacitance in timing and power estimation, and consider static power and dynamic power with activity probability. The revised prefix adder design space is shown in figure1(b).

In this paper, we formulate the prefix adder synthesis problem to an integer linear programming problem with prefix network and physical placement as decision variables. Prefix properties and gate/wire capacitance are represented in a set of linear formula of the decision variables. Based on the gate and wire capacitance, a linear timing model derived from logical effort [10] is applied, while the power model follows the activity analysis proposed in [11]. Finally, the feasibility of each variable assignment is checked by prefix property constraints, timing constraints and area constraints, and the objective is to minimize total power consumption. Therefore the ILP solution represents the minimal power prefix adder with placement information for given constraints. This method has been extended to handle gate sizing and buffer insertion. Compared with previous works, the proposed method concerns most of the essential factors on timing, area and power, and produces the most realistic candidate to back-end design. A drawback of this method is the unscalable computation load. For high bit-width applications, this method can be utilized as local optimization step in a divide-and-conquer strategy.

The rest of the paper is organized as follows. The prefix problem and area/timing/power model are defined in Section II. The ILP formulation is proposed in Section III. Section IV shows the experimental results on both numerical analysis and ASIC implementations, while Section V presents the conclusions.

II. Preliminaries

A. Prefix Addition & Prefix Adders

The binary addition problem can be defined as follows: given an n-bit augend A, an n-bit addend B, and a 1-bit carry-in c_0, generate the n-bit sum S and the 1-bit carry-out c_n. We denote $A = a_n \dots a_2 a_1$ and $B = b_n \dots b_2 b_1$. The sum bit s_i and carry bit c_i are computed as follows:

$$s_i = a_i \oplus b_i \oplus c_{i-1} \tag{1}$$

$$c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1} \tag{2}$$

In prefix addition, we need to use the *generate* bit g_i and *propagate* bit p_i, where $g_i = 1$ means that a carry is generated at bit i and $p_i = 1$ means that a carry is propagated through bit i. They are defined as:

$$g_i = \begin{cases} c_0 & if\ \text{i=0} \\ a_i b_i & otherwise \end{cases} \tag{3}$$

$$p_i = \begin{cases} 0 & if\ \text{i=0} \\ a_i \oplus b_i & otherwise \end{cases} \tag{4}$$

(Pre-processing)

1-4244-0630-7/07/$20.00 ©2007 IEEE.

The concept of generate and propagate can be extended to multiple bits. Let's define $G_{[i:k]}$ and $P_{[i:k]}$ $(i \geq k)$ as:

$$G_{[i:k]} = \begin{cases} g_i & if \text{ i=k} \\ G_{[i:j]} + P_{[i:j]} G_{[j-1:k]} & otherwise \end{cases} \quad (5)$$

$$P_{[i:k]} = \begin{cases} p_i & if \text{ i=k} \\ P_{[i:j]} P_{[j-1:k]} & otherwise \end{cases} \quad (6)$$

(Prefix computation)

Sum s_i and carry c_i can be calculated from G and P:

$$s_i = p_i \oplus c_{i-1} \quad (7)$$

$$c_i = G_{[i:0]} \quad (8)$$

(Post-processing)

To simplify the representation of G and P, an operator $\bullet$ is defined in (G, P) computation:

$$(G, P)_{[i:k]} = (G, P)_{[i:j]} \bullet (G, P)_{[j-1:k]} \quad (9)$$

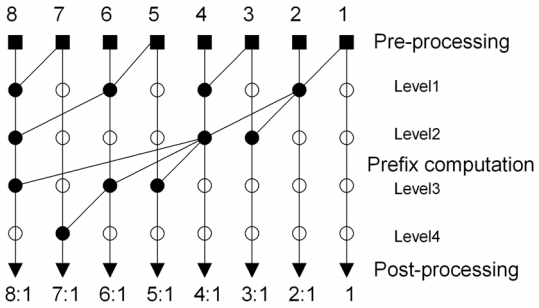

Fig. 2. Brent-Kung Prefix Adder

Figure2 shows the logical topology of an 8-bit Brent-Kung prefix adder with 4 logic levels. The symbols , $\bullet$ and $\blacktriangledown$ represent gp generators, GP adders and sum generators respectively. In the rest part, we denote the max number of logic levels as d.

B. Area Model

As a datapath component, prefix adder will keep the bit-slice structure in the placement, which is consistent with the logical structure. However, with each column each GP adder can take advantage of any empty space in the same bit-slice. When all the empty bubbles are below the GP adders in one bit-slice, it's called "compact placement". Figure3 shows the compact placement of the 8-bit Brent-Kung adder.

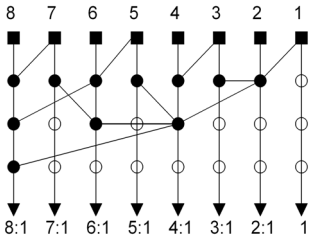

Fig. 3. Compact Placement of the 8-bit Brent-Kung Adder

It can be observed that the physical depth required to hold a prefix adder can be smaller than the logical depth. The lower bound of physical depth is the max number of GP adders in one bit-slice. Known the physical depth, denoted as m, the physical area is formulated to $n \times m$.

$$Area = n \times m \quad (10)$$

C. Timing Model

We adopt a linear timing model following the concept of logical effort, which decomposes the gate delay into two parts: a constant part and a linear part. The constant part is due to parasitic capacitance of the gate itself, so it is also called parasitic delay (PD). On the other hand, the linear part is due to the load capacitance. The coefficient is defined as Logical Effort (LE), and it depends on the complexity of the gate. Figure4 shows the structure and logical effort of an inverting CMOS GP adder in [12].

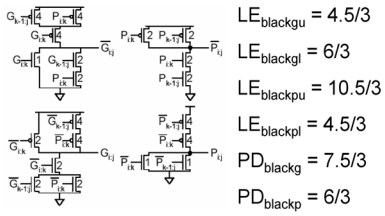

Fig. 4. Structure and Logical Effort of Inverting CMOS GP Adder

In figure4, it contains the logical efforts and parasitic delays for both G and P outputs. In practical, G network is always more complicated and slower than P network. Hence the previous model can be simplified to a two-inputs (left and right G inputs) one-output (G output) format. The logical effort formulation is written as:

$$LE_l = 4.5/3 = 1.5 \quad (11)$$

$$LE_r = 6/3 = 2 \quad (12)$$

$$PD = 7.5/3 = 2.5 \quad (13)$$

Given the logical effort and the parasitic delay, the gate delay depends on the ratio of output load and input capacitance. When gate size is uniform, the gate delay can be calculated from load capacitance. The result unit is $\frac{1}{5}$ FO4 delay, denoted as $\frac{1}{5} D_{FO4}$. Accordingly, the timing model of an GP adder is shown as follows:

$$Delay_l = 1.5 \cdot Cload + 2.5 \quad (14)$$

$$Delay_r = 2 \cdot Cload + 2.5 \quad (15)$$

In terms of load capacitance, it is composed of gate capacitance and wire capacitance. Gate capacitance can be derived directly from the number of fanouts, when every GP adder has unit input capacitance. Wire capacitance is linear to the wire length, which depends on the physical placement. For given physical placement, the total wire length driven by a GP adder is estimated by the number of columns and rows occupied by the bounding box around all the fanouts. It matches with the daisy-chain interconnect structure. Hence the wire capacitance can be calculated as the total wire length scaled by a scaling factor. We pick 0.5 as the scaling factor for current technology. Note that all the parameters can be various for different technology .

D. Power Model

Power consumption of CMOS circuit includes two parts: (a) dynamic power consumption (b) static power consumption. The dynamic power consumption is mainly due to the charge and discharge of capacitance. Hence the activity probability of each GP adder is an essential factor in dynamic power estimation. Vanichayobon [11] analyzes the activity of prefix adder and proposes a power consumption model, shown as follows, where d is the logical depth of the prefix network and w_i is the number GP adders in level i.

$$\sum_{i=1}^{d} i \cdot w_i \quad (16)$$

The equation demonstrates the switching probability of each node in level i. For the primary inputs of prefix network, the switching probability can be considered as 0.5 for general cased. Therefore, incorporating with load capacitance, the equation (16) is revised to equation (17), where C_i is the total load capacitance in level i.

$$P_{dyn} = (\sum_{i=1}^{d} i \cdot C_i) + 0.5 \cdot C_0 \tag{17}$$

Nowadays static power consumption grows fast with the technology development. It is independent from switching activities but proportional to the number of GP adders when gate size is uniform. We measure the static power consumption of one GP adder as half of the power unit, which is corresponding to $\frac{1}{4}$ FO4 switching power consumption, denoted as $\frac{1}{4}P_{FO4}$. Thus the total power consumption is calculated by:

$$Power = (\sum_{i=1}^{d} i \cdot C_i) + 0.5 \cdot C_0 + 0.5 \cdot \sum_{i=1}^{d} w_i \tag{18}$$

Note that the parameters introduced in this section can be various for different technology. For given technology, the parameters can be updated. It won't compromise the validity of the proposed methodology.

III. Basic ILP Formulation

In this section we present the ILP formulations to describe the prefix adder problem according to the area/timing/power models proposed in the previous section. The first subsection demonstrates the representations of a prefix adder and its physical placement, followed by GP property constraints described in subsection 2. Subsection 3 characterizes the calculation of load capacitance for each GP adder. Based on the load capacitance, timing constraints and power optimization objective are displayed in the following subsections.

A. Structural Constraints and Physical Placement

The ILP representation of a prefix adder is quite straightforward to match with the logic view of a prefix adder. It describes GP adders and interconnections in the $n \times d$ array, where n is the operand bit-width and d is the logical depth.

- **Bit-slice principle**: The bit-slice structure is kept for each column: the left fanin of each GP adder is always from some one above in the same column, and the right fanin is connected from the top-right quadrant. Also, at least one input is from the previous logical level. Finally each column has one primary output to the postprocessing stage.
- **GP principle**: To guarantee the described structure is a feasible prefix network, the GP property and GP operation principles must be satisfied. That is, each GP adder should cover a certain segment which is determined by the left and right inputs. In additional, the two inputs must cover two adjacent segments. The primary output at column i should cover the certain segment [i,1].
- **Overlap principle**: This step is to place a prefix network in a physical $n \times m$ array, where m is the physical depth. As mentioned in the area model, all GP adders in one logical column are placed in the same physical column. The only constraint is that no two GP adders are placed on the same physical position.

According to the previous discussion, we define the following variables:

- $x_{(i,j)} \in \{0,1\}$: 1 if and only if an GP adder is in the i^{th} bit and j^{th} level [1], for every (i, j) in the $n \times d$ array.
- $w^L_{(i,j,h)} \in \{0,1\}$: 1 if and only if there is a left fanin wire to position (i,j) from position (i,h), and $h < j$.
- $w^R_{(i,j,k,l)} \in \{0,1\}$: 1 if and only if there is a right fanin wire to position (i,j) from position (k,l), and $k < i, l < j$.
- $w^Z_{(i,j)} \in \{0,1\}$: 1 if and only if the output of the GP adder (i,j) connects to the primary output in column i.
- $y^L_{(i,j)}, y^R_{(i,j)} \in [1,n]$: the left and right segment bounds of GP adder (i,j). That means the GP adder in position (i,j) covers the segment [$y^L_{(i,j)}$:$y^R_{(i,j)}$].
- $p_{(i,j)} \in [0,m]$: the physical level of GP adder (i,j). Then its physical position will be (i,$p_{(i,j)}$).

The following constraints correspond with the **Bit-slice principle**, **GP principle** and **Overlap principle**:

Bit-slice principle:

$$\sum_h w^L_{(i,j,h)} = x_{(i,j)} \quad \forall\ j > h \tag{19}$$

$$\sum_{(k,l)} w^R_{(i,j,k,l)} = x_{(i,j)} \quad \forall\ i > k \& j > l \tag{20}$$

$$w^L_{(i,j,j-1)} + \sum_k w^R_{(i,j,k,j-1)} \geq x_{(i,j)} \tag{21}$$

$$\sum_j w^Z_{(i,j)} = 1 \tag{22}$$

GP principle:

$$y^L_{(i,j)} = y^L_{(i,h)} \quad if\ w^L_{(i,j,h)} = 1 \tag{23}$$

$$y^R_{(i,j)} = y^R_{(k,l)} \quad if\ w^R_{(i,j,k,l)} = 1 \tag{24}$$

$$y^R_{(i,h)} = y^L_{(k,l)} + 1 \quad if\ w^L_{(i,j,h)} = w^R_{(i,j,k,l)} = 1 \tag{25}$$

$$y^L_{(i,j)} = i \quad if\ w^Z_{(i,j)} = 1 \tag{26}$$

$$y^R_{(i,j)} = 1 \quad if\ w^Z_{(i,j)} = 1 \tag{27}$$

Overlap principle:

$$p_{(i,j)} \neq p_{(i,h)} \quad \forall\ j \neq h \tag{28}$$

A problem in the GP principle is that these constraints are conditional, which cannot be handled by ILP solver directly. They have to be transformed to strict linear constraints. Equation (29) and (30) shows the linear format of equation (23).

$$y^L_{(i,j)} \geq y^L_{(i,h)} - n \cdot (1 - w^L_{(i,j,h)}) \tag{29}$$

$$y^L_{(i,j)} \leq y^L_{(i,h)} + n \cdot (1 - w^L_{(i,j,h)}) \tag{30}$$

It can be observed that when $w^L_{(i,j,h)} = 1$, $y^L_{(i,j)}$ is restricted to $y^L_{(i,h)}$, otherwise these constraints are cancelled by the product term. It is important to notice that these constraints may not be effective until variable w^L is integerized. We call them "pseudo-linear" constraints. Pseudo-linear constraint is harmful to the performance of ILP solver, because it will reduce the chance to cut off infeasible solutions before every integer variable has been exactly determined.

B. Capacitance Constraints

Load capacitance is the essential parameter for both timing and power estimation. It includes gate load capacitance and wire load capacitance.

- **Gate principle**: Gate load capacitance depends on all the logical connections from a GP adder. Assume each GP adder has a unit input capacitance, the gate load capacitance can be calculated by the number of fanouts from a GP adder.
- **Wire principle**: The wire capacitance highly depends on the physical positions of each GP adders. As mentioned in timing model, we use the half perimeter of the bounding box covering all fanouts as the measure of wire length, as shown in figure5. The height/width of the bounding box is the max vertical/horizontal distance of each fanout. Based on the known wire length, wire capacitance is the product of a scaling factor and the wire length. We pick 0.5 as the scaling factor to represent the ratio of unit-length wire capacitance to unit gate capacitance.

Following variables are defined for capacitance calculation:

[1] We call this column i and row j, or simply position (i,j) later on

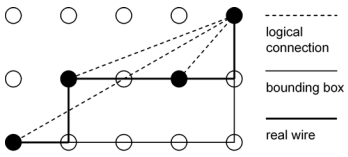

Fig. 5. Wire Length Estimation

- $c^G_{(i,j)} \in [0, Cmax]$: The gate load capacitance on GP adder (i,j). And $Cmax$ is a large constant representing the max load capacitance value.
- $c^{WV}_{(i,j)} \in [0, Cmax]$: The wire load capacitance from the vertical height of the fanout bounding box.
- $c^{WH}_{(i,j)} \in [0, Cmax]$: The wire load capacitance from the horizontal width of the fanout bounding box.
- $c_{(i,j)} \in [0, Cmax]$: The total load capacitance on GP adder (i,j).

The **Gate principle** and **Wire principle** in ILP formulation list as follows:

Gate principle:

$$c^G_{(i,j)} = \sum_h w^L_{(i,h,j)} + \sum_{(k,l)} w^R_{(k,l,i,j)} + w^Z_{(i,j)} \tag{31}$$

Wire principle:

$$c^{WV}_{(i,j)} \geq 0.5(p_{(i,h)} - p_{(i,j)}) \quad if\ w^L_{(i,h,j)} = 1 \tag{32}$$
$$c^{WV}_{(i,j)} \geq 0.5(p_{(k,l)} - p_{(i,j)}) \quad if\ w^R_{(k,l,i,j)} = 1 \tag{33}$$
$$c^{WH}_{(i,j)} \geq 0.5(k - i) \quad if\ w^R_{(k,l,i,j)} = 1 \tag{34}$$

Total load capacitance:

$$c_{(i,j)} = c^G_{(i,j)} + c^{WV}_{(i,j)} + c^{WH}_{(i,j)} \tag{35}$$

Note that the constraint of gate principle is linear, while the constraints in wire principle are all pseudo-linear. Hence the capacitance calculation is partial linear to the structure variables.

C. Timing Constraints

The timing analysis on the prefix structure is performed from top to bottom. The start points are the primary inputs with input arrival times, while the end points are the primary outputs, whose output times should be smaller than output required times.

- **Output principle**: The output time of each GP adder is the max path delay from input, which depends on the left and right input arrival times and the gate delay described in the previous section.
- **Input principle**: Input arrival times are obtained from the output times of two fanins.

We define following variables for input arrival times and output times:

- $t^L_{(i,j)}, t^R_{(i,j)} \in [0, Tmax]$: the left and right input arrival times of GP adder (i,j).
- $t_{(i,j)} \in [0, Tmax]$: The output time of GP adder (i,j). Tmax is a large constant.
- $t^Z_i \in [0, Tmax]$: The primary output arrival time at each bit-slice.

The **Input principle** and **Output principle** are formulated as:

Input principle:

$$t^L_{(i,j)} = t_{(i,h)} \quad if\ w^L_{(i,j,h)} = 1 \tag{36}$$
$$t^R_{(i,j)} = t_{(k,l)} \quad if\ w^R_{(i,j,k,l)} = 1 \tag{37}$$
$$t^Z_i = t_{(i,j)} \quad if\ w^Z_{(i,j)} = 1 \tag{38}$$

Output principle:

$$t_{(i,j)} \geq t^L_{(i,j)} + 1.5 \cdot c_{(i,j)} + 2.5 \tag{39}$$
$$t_{(i,j)} \geq t^R_{(i,j)} + 2.0 \cdot c_{(i,j)} + 2.5 \tag{40}$$
$$t^Z_i \leq Output_Requied_Time(i) \tag{41}$$

Among these timing constraints, the input arrival time constraints are pseudo-linear, and they are critical to the entire timing analysis. Conceptually the timing analysis cannot be finished until all structural variables are integerized.

D. Power Consumption Objective

Following equation (18), the total power consumption objective can be easily described as follow:

$$Minimize \quad (\sum_{(i,j)} j \cdot c_{(i,j)}) + 0.5 \cdot \sum_i c_{(i,0)} + 0.5 \cdot \sum_{(i,j)} x_{(i,j)} \tag{42}$$

The first term represents the dynamic power consumption and the second term corresponds to the static power consumption. Note that the static power is linear to the total number of GP adders, but the dynamic power is not entirely linear to all the structural variables. The wire load capacitance is pseudo-linear to connection variables. However, the gate load capacitance is linear to the structural variables. Overall, the linear components still dominate the total power consumption. With this property, ILP solver can efficiently search the solution space and find the optimal solutions quickly.

E. Extended ILP Formulations

While the basic ILP formulation already supports the comprehensive area, timing and power model, it can be further extended to support gate sizing and buffer insertion. This sub-section gives a brief introduction on the basic ideas to handle gate sizing and buffer insertion.

Gate sizing is a popular logical optimization technique to improve performance. For a given prefix structure, gate sizing has no effect on interconnect relations or GP property. So the structural constraints and GP property constraints keep unchanged. However it changes the input load capacitance and the driving strength of a GP adder.

- **Sizing-cap principle**: If the size of an GP adder is enlarged x times, the input load capacitance will increase by x times too.
- **Sizing-delay principle**: The logical effort delay will decrease x times while the parasitic delay keeps the same.
- Beside the effect on capacitance and timing analysis, gate sizing also increases the static power consumption linearly.

The extension to support buffer insertion, is more complicated than handling gate sizing. Because the buffer insertion will compromise the consistency between logical and physical connections, and consequently change the constraints on load capacitance, timing analysis and power estimation. In this case, it is not enough to present the logical structure only by structural variables. A complete explicit physical view of each solution is necessary. On the other hand, logical structure is essential to provide linear constraints on the objective function. So the extension supporting buffer insertion will operate on both logical and physical structural variables.

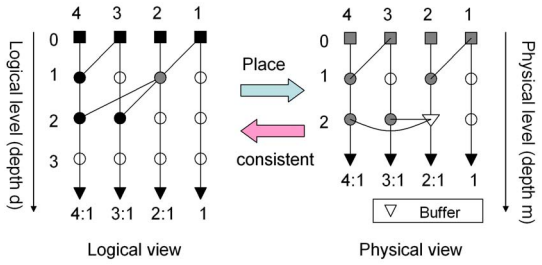

Fig. 6. Logical View and Physical View with Buffers

A prefix structure is completely represented in both logical view and physical view, as shown in figure6. There is no buffer in the logical view. It keeps the pure prefix network with logical interconnections. The logical network is placed in the physical view, and each logical interconnection can be physically implemented through one or multiple buffers. The physical view must be consistent with the logical view.

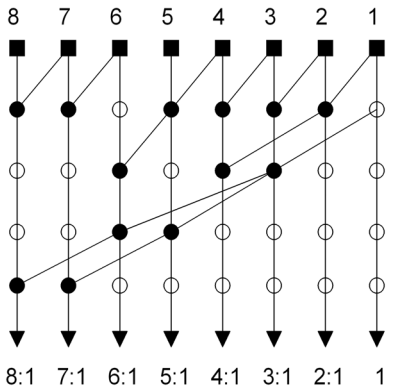

Fig. 7. Fastest Adder (Depth:2)

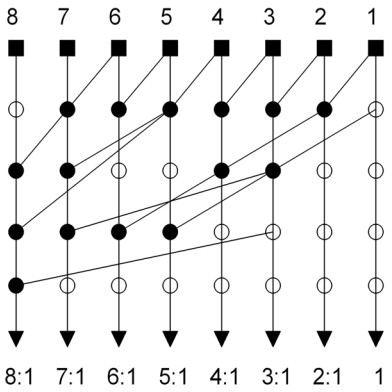

Fig. 8. Fastest Adder (Depth:3)

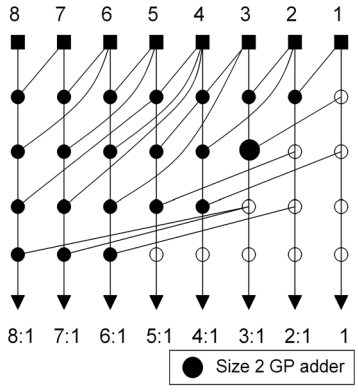

Fig. 9. Fastest Adder (Depth:4)

There are mainly two steps to maintain the consistency between the logical and physical views.

- **Matching principle**: The matching information must be recorded. That is, where each GP adder in logical view is placed in physical view, and for each GP adder or buffer in physical view, which GP adder in logical view it represents for.
  Note that due to the appearance of buffer, a GP adder in logical view may be represented by one GP adder and multiple buffers in physical view.
- **Interconnect principle**: Known the matching information, every physical interconnect can be checked if it represents a valid logical interconnect.

IV. Experimental Results

We implement the ILP formulation of the optimum prefix adders and solve the problem by CPLEX 9.1 with various timing and area configurations. For uniform signal arrival/required time profile, the entire 8-bit prefix adder design space is explored. We then apply this method to non-uniform signal arrival time applications. Also, for high-bit-width applications, the ILP method can be employed in a hierarchical design methodology. Finally we implement 64-bit ILP prefix adders with Synopsys datapath design flow, and compare the resulting designs with fast carry-look-ahead adders produced by Synopsys Module Compiler.

A. Uniform Input Arrival Time

To fully demonstrate the tradeoff between timing, power and area for 8-bit prefix adder design, we test every timing point with different area settings. The first starting point is the slowest but smallest structure: ripple carry structure. Then the timing requirement gradually becomes smaller, which implies tighter timing constraint, until no more feasible solution can be found. At each time point different physical depths are applied as area constraints. However, if they produce the same result, we only record the result associated with the smallest area constraint. In addition, all of the basic ILP, the extension supporting gate sizing and the extension supporting buffer insertion are tried. Again, if they produce the same solution, it is counted as no gate sizing. The timing and power results are normalized to FO4 delay (D_{FO4}) and FO4 switching power (P_{FO4}).

Note that the number of logical levels is not a parameter any more. Instead, it is adjusted by the ILP algorithm automatically. TableI shows the ILP results and three classic regular prefix adders, and figure10 demonstrates the optimum prefix adders in the design space corresponding to the data in the table. All the data in the table are based on the area/timing/power model in Section II.

According to the data, there are several observations:

- The ripple carry adder with the minimal area can be faster by gate sizing, but the power overhead is huge. On the other hand, physical depth 2 and 3 provide great flexibility for 8-bit prefix adders. When timing requirement is loose, gate sizing is not necessary. With the increase of timing requirement, either gate sizing or larger area will help to meet the timing constraint and reduce power. For extremely high performance adders, both gate sizing and large area are needed, while the power consumption increases sharply.
- None of the three classic prefix adders is optimal in terms of either area or power consumption for the given timing constraints. Figure7 to figure9 presents three fastest prefix adders with physical depth 2, 3 and 4 respectively. They all have 4 logical levels.
- Big gate size is not very helpful for 8-bit prefix adder. Although the max gate size allowed by the program is 3, only size 2 has appeared

TABLE I
Optimum Prefix Adders

Method	Timing (D_{FO4})	Depth	Power (P_{FO4})	CPU Time (s)
ILP	10.0	1	27.1	0.31
ILP	10.0	2	24.6	124
ILP (sizing)	9.0	1	32.3	2.83
ILP	9.0	2	24.6	83.4
ILP (sizing)	8.6	1	34.2	1.28
ILP	8.6	2	24.6	93.2
Brent-Kung	7.8	3	26.9	-
ILP	7.6	2	25.1	112
ILP	7.0	2	25.7	99.6
Sklansky	6.8	3	27.8	-
Kogge-Stone	6.2	3	35.6	-
ILP	6.0	2	34.5	259
ILP	5.6	2	29.7	45.7
ILP (sizing)	5.6	2	28.5	756
ILP	5.6	3	28.8	1237
ILP (sizing)	5.0	2	30.4	1208
ILP	5.0	3	32.3	4563
ILP	4.6	3	32.8	7439
ILP (sizing)	4.2	3	34.5	9654
ILP (sizing)	4.0	4	42.6	20211

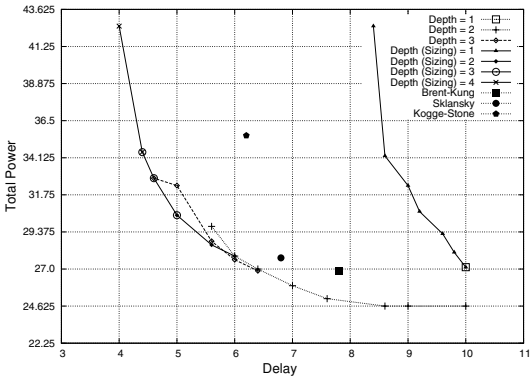

Fig. 10. Optimum Timing-Power Curves in the Design Space

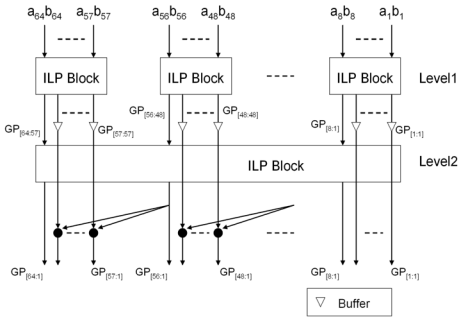

Fig. 11. 64-bit Hierarchy Prefix Adder

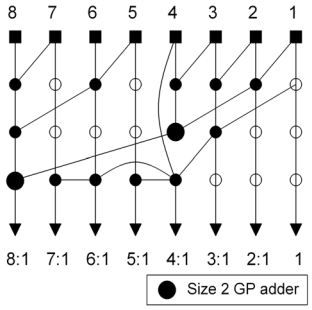

Fig. 12. Hierarchical ILP (level 1)

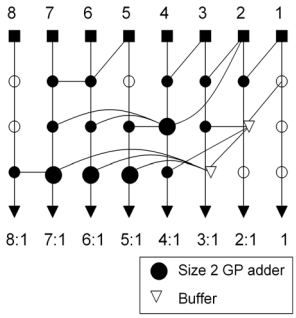

Fig. 13. Hierarchical ILP (level 2)

in the solutions. Also there is no buffer insertion in all 8-bit optimal prefix adders with uniform input arrival and output required time. One possible reason is that for 8-bit prefix addition, load capacitance is not big enough to take advantage of buffer insertion.

- The CPU time of the ILP solver is the main drawback of the proposed method. It raises quickly with the increasing timing requirement. The timing analysis is defined by pseudo-linear constraints. Therefore when timing constraint is too tight, ILP solver cannot efficiently verify the feasibility of each variable assignments. So the proposed method is more suitable for power optimization problem with moderate timing requirement.

B. Non-uniform Arrival and Required Times

Besides uniform signal arrival profile, some applications need non-uniform signal arrival/required times. Binary Multiplier is an example. A binary adder is used as final adder to sum up two partial product reduced from partial products reduction tree. The middle bits arrive later than most and least significant bits. We build the optimum prefix adders for three representative arrival time profiles: arrival times increase with bit significance (increasing), arrival times decrease with bit significance (decreasing) and the middle bits arrive later (convex). Table II lists the physical depth and power consumption of each case. These three cases have various structures, which shows the flexibility of the proposed method.

TABLE II
NON-UNIFORM ARRIVAL/REQUIRED TIME CASES

Case	Power	Depth
Increasing arrival time	27.8	3
Decreasing arrival time	31.8	3
Convex arrival time	28.5	2

C. Hierarchical Design

The previous experiments have shown the advantage of ILP method on 8-bit prefix addition applications. For high-bit-width applications, the ILP method can be applied in a hierarchical design methodology. There are two reasons to use hierarchical design methodology instead of pure ILP method. The first reason is that data-path design favors global regular structures. Global irregular structure increases the difficulty on detail routing and compromises the circuit reliability. The second reason is that ILP method is not scalable. The computation load of ILP solver increase exponentially with the operand bit-width.

ILP is applied to design a 64-bit two level hierarchical prefix adder. Sparse tree structure [13] is selected as global structure, and each 8-bit prefix block is generated by ILP method. Figure11 demonstrates the hierarchy structure.

In both hierarchy levels, each prefix block is 8-bit. The boundary timing requirement can be negotiated between the two levels. We build 64-bit hierarchical prefix adder for various timing requirement and compare them with 64-bit Kogge-Stone, Brent-Kung and Sklansky adders. TableIII shows the results in terms of delay and power. The Hierarchical ILP method not only achieves at least 16% power saving compared with 64-bit Brent-Kung and Sklansky adders, but also reach the same performance as 64-bit Kogge-Stone adder.

TABLE III
64-BIT PREFIX ADDERS

	Timing	Power
ILP Hierarchy	28	414
Brent-Kung	27	513
ILP Hierarchy	26	416
ILP Hierarchy	24	418
ILP Hierarchy	22	420
ILP Hierarchy	20	424
ILP Hierarchy	18	431
Sklansky	17	531
ILP Hierarchy	16	446
ILP Hierarchy	15	459
Kogge-Stone	15	2944
ILP Hierarchy	14	513

Figures 12 and 13 demonstrate the two level physical structures in the fastest 64-bit ILP adder. The level-1 structure has fast paths from inputs to the MSB output (critical path), but save power for other bits (non-critical path). The level-2 network utilizes gate-sizing to improve the drive strength for large fanouts. They all show the flexibility of the ILP method.

D. ASIC Implementation

To demonstrate the advantage of the proposed ILP methodology, we implement 64-bit prefix adders produced by hierarchical ILP method in Synopsys Data-path design flow. The flow starts from Module Compiler. The synthesized netlist with relative placement is then placed and routed by Astro. Based on all the necessary physical information including parasitic and coupling capacitance, the delay, area and power (both static and dynamic) are reported by Prime Power. The results are compared with 64-bit fast carry-look-ahead adders generated by Synopsys Module Compiler. The library is TSMC 90nm standard cell library.

Figure14 shows the ILP and Module compiler implementations after detail routing. They have the same physical boundary and pin locations. Two operands come from the left edge, while the sum result goes out at the right edge. It can be observed that the ILP structure consumes fewer cells and wires, especially long wires at right hand side.

TableIV summarizes the timing and power consumption of ILP and Module Compiler designs, and shows the percentage of ILP power to Module Compiler. These results shows more than 50% total power saving on ILP prefix adder for high-performance applications. Note that the power saving on net switching power is even larger. It is because the consideration of wire load in the ILP formulation.

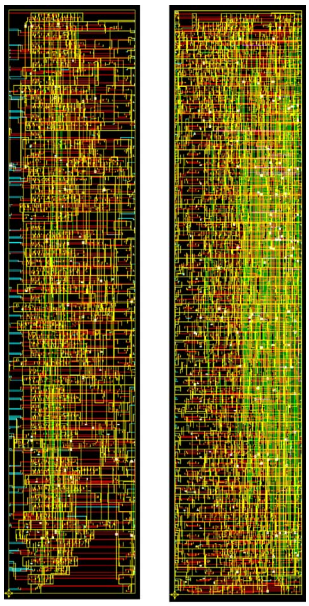

Fig. 14. ASIC Implementations (a) ILP (b) MC

TABLE IV
TIMING POWER COMPARISON

ILP: Timing (ns)	ILP: Total Power (mW) [Net Power]	MC: Timing (ns)	MC: Total Power (mW) [Net Power]	Power Saving (%) [Net]
0.74	1.9 [0.93]	0.75	4.9 [2.8]	61% [67%]
0.76	1.8 [0.90]	0.83	3.5 [2.1]	49% [57%]
1.13	1.15 [0.65]	1.24	2.3 [1.5]	50% [57%]

V. CONCLUSIONS

In this paper we first introduce a comprehensive area/power/timing model. Compared with the idealistic model used in previous works, the new model can capture the key characters of CMOS circuit, especially the effect of physical design. Based on the model, we propose an Integer Linear Programming method to build optimal prefix adder with minimal power consumption. By keeping the linear relation from decision variables to power objective, the ILP formulation can be solved efficiently. This method can handle non-uniform input arrival times and output required times for each application. The extension of the method can even support gate sizing and buffer insertion. The experiments demonstrate the complete 8-bit prefix adder solution space and the optimum area, timing and power tradeoff curves which outperform previous classic structures. For high-bit-width applications, the hierarchical ILP method shows a great flexibility and significant power saving comparing with several classical prefix adders.

VI. ACKNOWLEDGMENTS

The work presented in this paper is supported by NSF:CCF-0618163 and California MICRO program.

REFERENCES

[1] Sklansky J, "Conditional-sum addition logic", IRE Trans. Electronic Computers, vol. EC-9, pp. 226-231, June 1960.
[2] Kogge P, Stone H, "A parallel algorithm for the efficient solution of a general class of recurrence relations", IEEE Trans. Computers, vol. C- 22, no. 8, pp. 786-793, Aug. 1973.
[3] Brent R, Kung H, "A regular layout for parallel adders", IEEE Trans. Computers, vol. C-31, no. 3, pp. 260-264, March 1982.
[4] Ladner R, Fischer M, "Parallel prefix computation", J. ACM, vol. 27, no. 4, pp. 831-838, Oct. 1980.
[5] Han T, Carlson D, "Fast area-efficient VLSI adders", Proc. 8th Symp. Comp. Arith., pp. 49-56, Sept. 1987.
[6] Knowles S, "A family of adders", Proc. 15th IEEE Symp. Comp. Arith., pp. 277-281, June 2001.
[7] Harris D, "A Taxonomy of Parallel Prefix Networks", Proc.Asilomar Conference of Signals, Systems, and Computers,pages 2213C2217, Nov. 2003.
[8] Zimmermann R, "Non-heuristic optimization and synthesis of parallel-prefix adders", Int. Workshop on Logic and Architecture Synthesis. 123C132.
[9] Zhu H, Cheng C.K., Graham R, "Constructing Zero-deficiency Parallel Prefix Adder of Minimum Depth", Proceedings of the ASP-DAC 2005, pp.883 - 888.
[10] Sutherland I, Sproull B, Harris D, "Logical Effort: Designing Fast CMOS Circuits", Morgan Kaufmann Publishers, 1999.
[11] Vanichayobon S, Dhall S, Lakshmivarahan S, Antonio J, "Power-speed Trade-off in Parallel Prefix Circuits", Proceeding of SPIE Vol. 4863 2002, pp.109 - 120.
[12] Harris D, Sutherland I, "Logical Effort of Carry Propagate Adders", 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'04), pp. 269-279, 2004.
[13] Mathew S, Anders M, Krishnamurthy R, Borkar S, "A 4-GHz 130-nm Address Generation Unit With 32-bit Sparse-Tree Adder Core", IEEE Journal of Solid-State circuits, Vol38, No.5, May 2003.