Asynchronous Circuit Placement by Lagrangian Relaxation

Gang Wu*, Tao Lin*, Hsin-Ho Huang†, Chris Chu* and Peter A. Beerel†
*Department of Electrical and Computer Engineering, Iowa State University, IA
†Ming Hsieh Department of Electrical Engineering, University of Southern California, CA
Email: {gangwu, tlin, cnchu}@iastate.edu, {hsinhohu, pabeerel}@usc.edu

***Abstract*—Recent asynchronous VLSI circuit placement approach tries to leverage synchronous placement tools as much as possible by manual loop-breaking and creation of virtual clocks. However, this approach produces an exponential number of explicit timing constraints which is beyond the ability of synchronous placement tools to handle. Thus, synchronous placer can only produce suboptimal results. Also, it can be very costly in terms of runtime. This paper proposed a new placement approach for asynchronous VLSI circuits. We formulated the asynchronous timing-driven placement problem and transform this problem into a weighted wirelength minimization problem based on a Lagrangian relaxation framework. The problem can then be efficiently solved using any standard wirelength-driven placement engine that can handle net weights. We demonstrate our approach on QDI PCHB asynchronous circuit with a state-of-art quadratic placer. The experimental results show that our algorithm can effectively improve the asynchronous circuits performance at placement stage. In addition, the runtime of our algorithm is shown to be more scalable to large-scale circuits compared with the loop-breaking approach.**

I. Introduction

With the continual diminishing of feature size, integrated circuit (IC) design is progressively more difficult. Synchronous design is facing particularly severe challenges due to the increasing variations in process parameters and demand in low power consumption. Asynchronous design provides a promising solution to the emerging challenges in advanced technology. Its potential advantages over synchronous design include robustness towards process-voltage-temperature (PVT) variations, lower power consumption, avoidance of the difficult problem of clock distribution, higher operating speed, less emission of electromagnetic noise, lower stress on power distribution network, improved security, and better composability and modularity [1] [2] [3]. However, even asynchronous design has all the potential advantages, synchronous circuits still predominate. A main reason is that EDA tool support for asynchronous design is grossly inadequate.

In current aggressive technologies, placement for asynchronous circuit becomes a more important issue, as wire delays are becoming more critical than gate delays. Most works on the timing-driven placement of asynchronous circuits directly leverage synchronous placement tools [4] [5] [6]. For synchronous circuit, the circuit performance is bounded by the most critical path and synchronous timing-driven placer will minimize the maximum path delay between flip-flops. However, in the case of asynchronous circuit, the performance is bounded by the most critical cycle [2]. This creates issues when applying synchronous placement tools for asynchronous timing optimization, since the optimization objective is different and timing-loops are not supported. In [4], a minimal set of cut-points is identified to break these timing loops. Then the handshaking cycles of the design are explicitly expressed as a set of set_max_delay timing constraints for each segment. The resulting number of constraints turns out to be far greater than in a typical synchronous placement flow and exponential to the circuit size. For large scale designs, it becomes impossible for placement engine to satisfy all these constraints within a reasonable amount of runtime. Also, these timing constraints are too conservative to achieve a good optimization result, as time borrowing is not allowed for segments along the same cycle.

Another issue to leverage synchronous placement tool is to enforce the timing constraints necessary for the functional correctness of asynchronous circuits. Instead of setup and hold time constraints for typical synchronous design, certain asynchronous design style requires relative timing constraints which constrains the relative delay between two paths [7]. Some other design styles require minimum and maximum bounded delay values on gates and wires [8]. This difference of timing assumptions between asynchronous design and synchronous design creates extra difficulties when applying synchronous placement tools to asynchronous design.

Few works have been done for timing aware placement algorithms targeting at optimizing critical cycles. In [9], sequential timing analysis based placement approach has been proposed to optimize the cycle delay for synchronous design under the assumption that retiming and clock skew scheduling can be applied. Unfortunately, this technique is practical only for the last stages of physical design and only a small amount of useful skew is allowed. In [10], performance and relative timing constraints for QDI circuits have been incorporated into a constructive placer to handle asynchronous designs. However, given their framework of construction based placement approach, this algorithm can easily be trapped into local minimum and produce suboptimal results. Also, this approach will lead to high density placement hot spots which can cause routability problems. In [11], a floorplan method

This work is supported in part by NSF award CCF-1219100.

Peter A. Beerel is also Chief Scientist, Technology Development in the Communications, Storage, and Infrastructure Group, Intel, Calabasas, CA.

for asynchronous circuits based on simulated annealing and sequence-pair has been proposed. However, they still need to leverage synchronous place-and-route tools at the placement stage.

In this paper, we proposed a new placement flow based on a Lagrangian relaxation framework. We formulated the asynchronous timing-driven placement problem considering both performance and timing constraints. Instead of adding explicit cycle constraints whose numbers can grow exponentially with circuit size, we incorporated the cycle metric calculation linear program into our problem formulation and the number of constraints we have is polynomial in circuit size. In addition, the special structure of the formulated timing-driven placement problem allows us to simplify the Lagrangian dual problem using Karush-Kuhn-Tucker (KKT) conditions and the original problem is transformed into a weighted wirelength minimization problem which can be solved effectively with existing placement approaches. The general modeling of performance and timing constraints also makes our approach applicable to a wide variety of asynchronous design styles, including Micropipeline [8], QDI [12], GasP [13] and Mousetraps [14] pipeline templates.

We explored our approach on quasi-delay-insensitive (QDI) Pre-Charged Half Buffer (PCHB) asynchronous designs synthesized using the Proteus RTL flow [4]. The Lagrangian relaxation frame work is incorporated into state-of-art quadratic placer POLAR [15]. We compared our results after detailed placement and routing with both an industrial placement tool and the Proteus placement flow.

The rest of this paper organized as follows. Section II introduces timing issues faced by asynchronous design. Section III elaborates our general Lagrangian relaxation framework. Section IV proposed our asynchronous placement flow on QDI PCHB asynchronous design. Section V shows the experimental results compared with other approaches. Finally, Section VI concludes the paper.

II. Timing for Asynchronous Circuits

First we define some notations that we use in this paper. An asynchronous circuit can be represented by a hypergraph $G = (V, E)$. Let $V = \{v_1, v_2, \ldots, v_{|V|}\}$ be the set of cells. Let $E = \{e_1, e_2, \ldots, e_{|E|}\}$ be the set of hyperedges. Let $AT = \{a_1, a_2, \ldots, a_{|V|}\}$ be the arrival time associated with each cell.

A. Performance for Asynchronous Circuits

Here we introduce a Petri net [16], which is a commonly used tool for modeling concurrent systems. A Petri net consists of places, transitions, and arcs. Places in a Petri net can have one or more tokens, or no token at all. The distribution of tokens over the places will represent an initial marking of the system. Transitions in a Petri net can fire if all its input places contain at least one token. When a transition fires, it consumes one token per input place and generates one token per output place. Specifically, a Petri net is called marked graph if all places have only one input and one output transition.

The performance of unconditional asynchronous circuits can be modeled using timed marked graphs. For conditional asynchronous circuits, we treat them as unconditional and the circuit performance can be guaranteed conservatively as proved in [17].

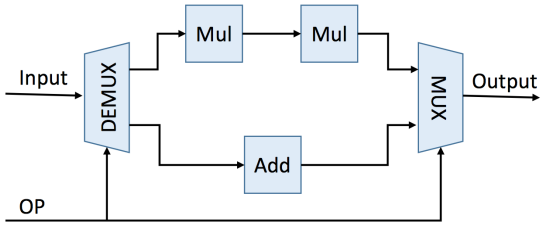

Fig. 1: Asynchronous ALU.

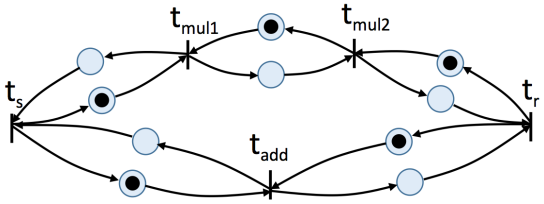

Fig. 2: Marked graph representation for Asynchronous ALU.

As an example, an asynchronous ALU design with two operation modes, addition and multiplication, is shown in Fig. 1. We use the Full Buffer Channel Net model proposed in [18] to obtain timed marked graph. The extracted marked graph is shown in Fig. 2. Each cell is modeled using a transition (t) and asynchronous channels between cells are modeled with a pair of places (shown as circles), a forward place and a backward place, which are labeled with forward and backward delays of the corresponding channel. Black dots inside the circle denotes the initial marking of the marked graph.

For any marked graph, let C_p be the set of neighboring transition pairs which have a place between them. The cycle time τ can be obtained by solving the following linear program [19].

$$\begin{aligned} &\text{Minimize} \quad \tau \\ &\text{Subject to} \quad a_i + D_{ij} - m_{ij}\tau \leq a_j \quad \forall (i,j) \in C_p \end{aligned}$$

where a_i and a_j are the arrival time associated with transitions t_i and t_j, which correspond to nodes v_i and v_j in the graph. D_{ij} is the delay associated with place p between transition t_i and t_j, which corresponds to forward or backward path delay of an asynchronous channel. m_{ij} is the number of tokens in the place p. $m_{ij} = 0$ if the corresponding place p does not contain token, which is quite often.

B. Timing Constraints

Timing assumptions made for different logic implementation style have a direct influence in the timing constraints necessary to ensure hazard-free operation of asynchronous circuits. Except for delay-insensitive (DI) design [20] which are

premised on the fact that they will function correctly regardless of the delays on the gates and the wires, timing constraints for other asynchronous designs fall into two categories [2].

First is explicit timing constraints in the form of minimum and maximum bounded delay values for gates and wires in the circuit. An example is the bounded-delay asynchronous circuits [8].

Let U_{ij} be the maximum bounded delay and L_{ij} be the minimum bounded delay between nodes v_i and v_j. Let C_e be the set of node pairs which we need to enforce explicit delay bounds. The explicit timing constraints can be written as:

$$L_{ij} \leq a_j - a_i \leq U_{ij} \quad \forall (i,j) \in C_e \tag{1}$$

Second is relative timing constraints, referred to as $relative\ timing$ [7], which dictate the relative delay of two paths that stem from a common point of divergence. Examples design styles that have relative timing constraints include the quasi-delay-insensitive (QDI) design style, such as WCHB, PCHB and the Multi-Level Domino (MLD) template [2].

For a relative timing constraint from a node v_k and forking into two nodes v_i and v_j constraints can be written as:

$$|(a_i - a_k) - (a_j - a_k)| \leq I_{ij} \quad \forall (i,j) \in C_r \tag{2}$$

which bound the maximum difference in time that the signal arrives at the two end-points of the fork. This type of constraint captures the notion of an $isochronic\ fork$ [2], a common type of constraint in quasi-delay-insensitive designs. Here I_{ij} is the delay bound for $isochronic\ fork$. C_r is the set of node pairs which have relative timing constraints.

III. Asynchronous Placement with Lagrangian Relaxation

Given an asynchronous circuit, we are interested in the minimum total wirelength and cycletime achievable with respect to the timing constraints necessary to guarantee functional correctness. In Section III-A, we first show how to formulate this problem as a constrained optimization problem. Then, we apply Lagrangian relaxation in Section III-B which is a general technique for converting constrained optimization problem into unconstrained problem. In Section III-C, we explore the special structure of the primal problem which allows us to extensively simplify the Lagrangian subproblem. In Section III-D, we show how to solve the simplified Lagrangian subproblem as weighted wirelength minimization problem. In Section III-E, we describe how to solve Lagrangian dual problem using a direction finding approach.

A. Problem Formulation

We denote the x-coordinates of cells by a vector $\mathbf{x} = \big(x_1, x_2, \cdots, x_{|V|}\big)$, and y-coordinates by a vector $\mathbf{y} = \big(y_1, y_2, \cdots, y_{|V|}\big)$. For pure wirelength-driven placement, the objective is to minimize the sum of the half-perimeter bounding box (HPWL) for all hyperedge e:

$$\text{HPWL}(\mathbf{x}, \mathbf{y}) = \sum_{e \in E} [\max_{i \in e} x_i - \min_{i \in e} x_i + \max_{i \in e} y_i - \min_{i \in e} y_i]$$

Then the problem of minimizing both total HPWL and cycletime subject to timing constraints (1), (2) in section II can be formulated directly as:

$$\begin{aligned}
\text{Minimize} \quad & \text{HPWL}(\mathbf{x}, \mathbf{y}) + \alpha\tau \\
\text{Subject to} \quad & a_i + D_{ij} - m_{ij}\tau \leq a_j \quad \forall (i,j) \in C_p && (3) \\
& L_{ij} \leq a_j - a_i \leq U_{ij} \quad \forall (i,j) \in C_e && (4) \\
& |(a_i - a_k) - (a_j - a_k)| \leq I_{ij} \quad \forall (i,j) \in C_r && (5)
\end{aligned}$$

where the constant α in the objective function can be chosen to adjust the tradeoff between minimizing wirelength and cycletime.

Note that Equations (4), (5) can be rewritten into the same form as:

$$(a_i + L_{ij} \leq a_j) \wedge (a_j - U_{ij} \leq a_i) \tag{6}$$

$$(a_j - I_{ij} \leq a_i) \wedge (a_i - I_{ij} \leq a_j) \tag{7}$$

To make equations in Section III-B, Section III-C more concise, we combine Equations (3), (6), (7) and the primal problem can be rewritten as:

$$\begin{aligned}
\text{Minimize} \quad & \text{HPWL}(\mathbf{x}, \mathbf{y}) + \alpha\tau \\
\text{Subject to} \quad & a_i + W_{ij} - \widetilde{m}_{ij}\tau \leq a_j \quad \forall (i,j)
\end{aligned}$$

where $W_{ij} = L_{ij}$ or $-U_{ij}$ $\forall (i,j) \in C_e$, $W_{ij} = -I_{ij}$ $\forall (i,j) \in C_r$ and $W_{ij} = D_{ij}$ $\forall (i,j) \in C_p$. Similarly, $\widetilde{m}_{ij} = m_{ij}$ $\forall (i,j) \in C_p$ and $\widetilde{m}_{ij} = 0$ $\forall (i,j) \in C_e$ or C_r.

B. Lagrangian Relaxation

We relax all the constraints following the Lagrangian relaxation procedure. Nonnegative Lagrangian multiplier λ_{ij} is introduced for each constraint. Let $\boldsymbol{\lambda}$ be a vector of all the Lagrange multipliers.

$$\begin{aligned}
\text{Let} \quad \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{a}, \tau) &= \text{HPWL}(\mathbf{x}, \mathbf{y}) + \alpha\tau \\
&+ \sum_{\forall (i,j)} \lambda_{ij}(a_i + W_{ij} - \widetilde{m}_{ij}\tau - a_j)
\end{aligned}$$

Then the Lagrangian subproblem, which gives a lower bound for the primal problem for any $\boldsymbol{\lambda} \geq 0$ [22], can be formulated as:

$$\mathcal{LRS}: \quad \text{Mimimize} \quad \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{a}, \tau)$$

Let the function $q(\boldsymbol{\lambda})$ be the optimal value of the problem $\mathcal{LRS}$. We are interested in finding the values for the Lagrangian multipliers $\boldsymbol{\lambda}$ to give the maximum lower bound, which is labeled as the Lagrangian dual problem and defined as follows:

$$\begin{aligned}
\mathcal{LDP}: \quad & \text{Maximize} \quad q(\boldsymbol{\lambda}) \\
& \text{Subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0}
\end{aligned}$$

Solving $\mathcal{LDP}$ will provide a solution to the primal problem.

C. Simplification of $\mathcal{LRS}$

Inspired by [21], we rearrange terms here and the Lagrangian function $\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{a}, \tau)$ can be rewritten as:

$$\begin{aligned}\mathcal{L} = \text{HPWL}(\mathbf{x}, \mathbf{y}) + (\alpha - \sum_{\forall(i,j)} \lambda_{ij}\widetilde{m}_{ij})\tau \\ + \sum_{k \in V}(\sum_{\forall(k,j)} \lambda_{kj} - \sum_{\forall(i,k)} \lambda_{ik})a_k \\ + \sum_{\forall(i,j)} \lambda_{ij} W_{ij}\end{aligned}$$

The KKT conditions imply $\partial\mathcal{L}/\partial a_i = 0$ for $1 \leq i \leq |v|$ and $\partial\mathcal{L}/\partial\tau = 0$ at the optimal solution of the primal problem. Then the optimality conditions $\mathcal{K}$ on $\boldsymbol{\lambda}$ can be obtained as:

$$\alpha = \sum_{\forall(i,j)} \lambda_{ij}\widetilde{m}_{ij}$$

$$\sum_{\forall(k,j)} \lambda_{kj} = \sum_{\forall(i,k)} \lambda_{ik} \quad \forall k \in V$$

Apply the optimality conditions into $\mathcal{LRS}$, we can obtain a simplified Lagrangian subproblem $\mathcal{LRS}^*$:

$$\text{Minimize} \quad \mathcal{L}^*(\mathbf{x}, \mathbf{y}) = \text{HPWL}(\mathbf{x}, \mathbf{y}) + \sum_{\forall(i,j)} \lambda_{ij} W_{ij}$$

It can easily be seen that solving $\mathcal{LRS}$ is equivalent of solving $\mathcal{LRS}^*$.

D. Solving $\mathcal{LRS}^*$

Since it is impossible to get accurate timing without detailed placement and routing, as an approximation, we take the wire delay as being proportional to the HPWL of the hyperedge e associated with node i and j, which can be written as:

$$W_{ij} = d_i + HPWL_e \cdot \gamma_e$$

where d_i is the intrinsic gate delay and $HPWL_e \cdot \gamma_e$ is the total wire load delay. γ_e is a constant value associated with each edge and depends on the driver cell, load cells and electrical characterization for the wires.

Note that for constraints (4) and (5), the corresponding W_{ij} is a constant value. For simplicity, we don't write them here explicitly. Then $\mathcal{LRS}^*$ can be written as:

$$\begin{aligned}&\text{Minimize} \quad \mathcal{L}^*(\mathbf{x}, \mathbf{y}) \\ &= \text{HPWL}(\mathbf{x}, \mathbf{y}) + \sum_{\forall(i,j)} \lambda_{ij}(d_i + HPWL_e \cdot \gamma_e) \\ &\quad + \text{terms independent of } \mathbf{x}, \mathbf{y} \\ &= \text{HPWL}(\mathbf{x}, \mathbf{y}) + \sum_{\forall(i,j)} HPWL_e \cdot \lambda_{ij}\gamma_e \\ &\quad + \text{terms independent of } \mathbf{x}, \mathbf{y}\end{aligned}$$

Here the objective function only contains $\mathbf{x}$, $\mathbf{y}$ as variables. $\mathcal{LRS}^*$ becomes a weighted wirelength minimization problem for a set of hyperedges, which can be solved well by existing standard synchronous placement engine with the ability to weight nets.

E. Solving $\mathcal{LDP}$

Traditional approach of solving Lagrangian dual problem is to apply the subgradient optimization method [22]. However, this approach requires projection after updating $\boldsymbol{\lambda}$ in order to maintain $\boldsymbol{\lambda}$ within the dual feasible region. In addition, practical convergence of the subgradient optimization is difficult and usually requires a good choice of initial solution and step size. Here we apply a direction finding approach inspired by [23] to solve $\mathcal{LDP}$, which is shown to have better convergence compared with the traditional approach. Combining $\mathcal{LDP}$ defined in Section III-B with optimality condition $\mathcal{K}$ derived in Section III-C, we rewrite $\mathcal{LDP}$ here as:

$$\begin{aligned}\mathcal{LDP}: \quad &\text{Maximize} \quad q(\boldsymbol{\lambda}) \\ &\text{Subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\lambda} \in \mathcal{K}\end{aligned}$$

For any feasible $\boldsymbol{\lambda}$, we want to find an improving feasible direction $\Delta\boldsymbol{\lambda}$ and a step size β such that:

$$q(\boldsymbol{\lambda} + \beta\Delta\boldsymbol{\lambda}) > q(\boldsymbol{\lambda})$$

Note that $\nabla\mathcal{L}^*_{\lambda_{ij}}(\mathbf{x}, \mathbf{y}) = W_{ij} \ \forall(i,j)$. Then an increasing feasible direction $\Delta\boldsymbol{\lambda}$ can be found by solving the following linear program $\mathcal{D}$:

$$\begin{aligned}&\text{Maximize} \quad \sum_{\forall(i,j)} \Delta\lambda_{ij} W_{ij} \\ &\text{Subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\lambda} \in \mathcal{K} \\ &\qquad \max(-u, -\lambda_{ij}) \leq \Delta\lambda_{ij} \leq u\end{aligned}$$

where u is a constraint we introduced to bound the objective function, similar to [23].

Our algorithm to solve $\mathcal{LDP}$ is shown in Algorithm 1. It starts from an initial dual feasible $\boldsymbol{\lambda}$, then the method iteratively improves $q(\boldsymbol{\lambda})$ by finding an improving direction and performing a line search. The algorithm terminates when change of $q(\boldsymbol{\lambda})$ is small enough or the duality gap $\text{HPWL}(\mathbf{x}, \mathbf{y}) + \alpha\tau - q(\boldsymbol{\lambda})$ is less than $Error\ bound$.

Algorithm 1 Solve Lagrangian Dual Problem

Ensure: $\boldsymbol{\lambda}$ which maximizes $\mathcal{LRS}^*$
1: $n =: 1$; /* step counter */
2: $\boldsymbol{\lambda} =:$ initial positive value satisfy optimality condition $\mathcal{K}$;
3: Solve linear program $\mathcal{D}$ to obtain optimal increasing direction $\Delta\boldsymbol{\lambda}$;
4: Perform line search on $q(\boldsymbol{\lambda})$. Then a step size β which improves function value $q(\boldsymbol{\lambda} + \beta\Delta\boldsymbol{\lambda}) > q(\boldsymbol{\lambda})$ can be found. Terminate the algorithm if the change of $q(\boldsymbol{\lambda})$ is small enough;
5: Moving one step further by updating $\boldsymbol{\lambda} = \boldsymbol{\lambda} + \beta\Delta\boldsymbol{\lambda}$;
6: $n =: n + 1$;
7: Repeat Step 3-6 until $(\text{HPWL}(\mathbf{x}, \mathbf{y}) + \alpha\tau - Q(\boldsymbol{\lambda})) \leq Error\ bound$;

IV. Asynchronous Placement Flow for QDI PCHB Pipeline Templates

A. Asynchronous Design with Pre-Charged Half Buffer (PCHB) Templates

PCHB is a QDI template developed at Caltech [12], which designed with dual-rail asynchronous channels and 1-of-N

handshaking protocol [2]. Fig. 3 shows a three stage PCHB pipeline structure with control circuit (CTRL), C-elements (C) and domino logic (FU) for computation.

Marked lines in Fig. 3 show an example of timing assumptions made by PCHB. In particular, it requires the input to the domino block go low before a rising transition on the control signal 'en' occurs. This timing assumption is a relaxed interpretation of the $isochronic\ fork$ assumption [24] and can easily be met without special care. We ignore this timing constraint at global placement stage and leave it to be checked after detailed placement and routing, similar to [4] and [6].

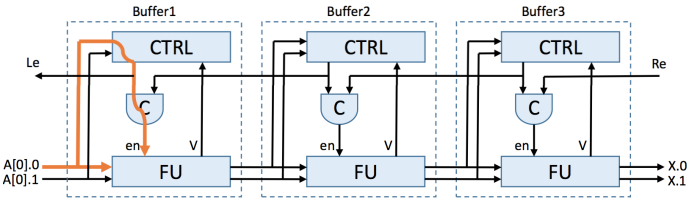

Fig. 3: PCHB pipeline template.

B. Asynchronous Placement Flow

Fig. 4 illustrates our placement flow for QDI PCHB pipeline templates. We use a state-of-art quadratic placer POLAR [15] as the placement engine. Initially, pure wirelength-driven placement is performed on the asynchronous circuit, as a good starting placement with minimized wirelength is necessary in order to achieve better cycletime in later stage.

After initial placement, we calculate the total wirelength and initial cycletime to set parameter $\alpha = k \cdot \text{HPWL}(\mathbf{x}, \mathbf{y})/\tau$. k is normally set to 1 in order to achieve a good balance between total wirelength and cycletime. We first extract the performance constraints based on our marked graph modeling of the original circuit. Then, a linear program similar to $\mathcal{D}$ is solved using Gurobi [25] to obtain an initial non-negative $\boldsymbol{\lambda}$ satisfy $\mathcal{K}$. Next, net weights as derived in Section II-C are added into the hypergraph which is used to guide POLAR solving $\mathcal{LRS}^*$. At this point, we enter the timing optimization stage of our flow. We start the loop of solving $\mathcal{LDP}$ by iteratively solving the direction finding problem with Gurobi and weighted wirelength minimization problem with POLAR to find a direction and step size which increase the objective value $q(\boldsymbol{\lambda})$, until we achieve a small duality gap or the improvement on the objective value is tiny. The solution to $\mathcal{LDP}$ is a placement with optimized timing which will be the output of our flow. Note that the constraints we have for PCHB will not introduce negative edge weights. Thus, it can be handled well by quadratic placer POLAR. For other asynchronous design style which has different timing constraints and negative weight edge exists, we need to choose a non-linear type placement engine to solve $\mathcal{LRS}^*$.

Finally, we export our design into Encounter to perform legalization and routing.

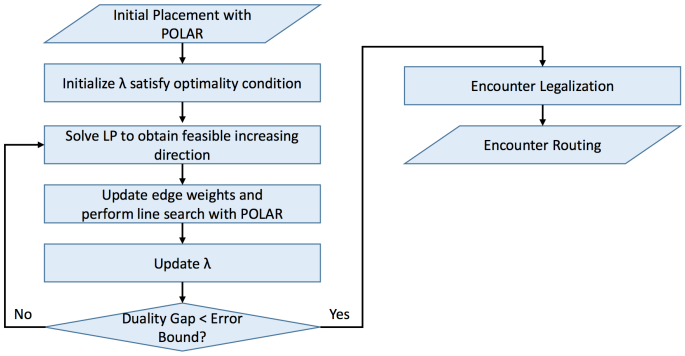

Fig. 4: QDI PCHB placement flow.

V. Experimental Results

The proposed approach is implemented using C++. All experiments were run on a Liunx PC with 47GB of memory and Intel Core-i3 3.3GHz CPU.

We run our flow on two sets of benchmarks. First is a set of ISCAS89 benchmark circuits which are converted to unconditional asynchronous circuits using Proteus's legacy RTL design flow [4]. In particular, flip-flops from ISCAS89 are mapped as token buffers and combinational gates are mapped as logic cells in PCHB cell library. In addition, we developed several benchmarks in RTL level and synthesized it using Proteus. For ALU and Accumulator design, we choose different bit width for the datapath to create a set of benchmarks with different sizes.

We compared our flow against both Encounter and the Proteus placement flow. The die size and I/O pin locations are set to be the same for all three flows. In our case, Encounter is used as a pure wirelength-driven placer without any input timing constraints. For Proteus flow, we disabled the gate resizing at its placement stage to avoid the changing of input netlists. Finally, all placement results are exported into Encounter and routed with same configuration.

Our experimental results are shown in Table 1. The size column shows the number of cells in each design. The wirelength column shows the detailed routed wirelength reported by Encounter. All the designs are shown to be routable. The cycletime column shows the cycletime calculated using the linear program introduced in Section II-A for the final routed design using our delay model.

We show significant improvement in cycletime compared with the non-timing driven Encounter placement and Proteus placement flow. For all the benchmarks, we achieved an average improvement in cycletime of 12% over the Proteus placement flow and 32% over the results of non-timing driven placement by Encounter. The wirelength of our approach is 3% worse compared with Encounter, which is expected considering the extensive timing optimization that has been performed and 6% better than Proteus placement flow. In addition, our runtime is also shown to be much more scalable in comparison with the Proteus approach.

Table I. Comparison on asynchronous benchmarks

Design	Size	Routed Wirelength x 10^6 (nm)			Cycletime (ns)			Runtime (s)		
		POLAR	Proteus	Encounter	POLAR	Proteus	Encounter	POLAR	Proteus	Encounter
s444	256	10.58	11.32	10.46	4.67	5.72	6.06	36	245	8
s510	519	33.10	35.38	32.84	6.87	7.01	7.25	37	330	12
s526	307	13.30	14.85	14.24	3.96	4.83	4.20	34	257	8
s526a	297	13.34	13.79	12.22	4.14	3.75	4.89	33	249	8
s641	636	22.83	26.96	22.02	4.41	4.88	5.00	36	313	10
s713	584	21.42	24.38	21.06	5.72	5.71	6.15	40	228	9
s820	681	44.30	48.53	43.39	7.10	8.34	10.40	41	431	13
s832	706	46.15	53.04	45.84	6.19	7.32	6.82	40	465	15
s838	707	38.65	37.49	33.32	4.88	5.58	5.91	37	337	13
s953	931	64.01	71.85	61.84	6.90	7.10	6.12	41	576	18
s1488	1314	124.88	137.06	130.13	11.80	11.35	12.79	49	771	27
s1423	1119	63.21	71.25	64.48	8.47	8.37	14.85	39	692	20
s9234	2108	120.19	134.48	119.40	6.76	8.19	9.83	51	517	31
s13207	5658	381.02	386.13	338.18	11.17	11.86	13.72	82	1202	67
s38417	15447	1310.20	1208.16	1253.42	42.44	68.30	80.43	298	1050	283
ALU4	413	16.92	18.78	18.01	5.22	5.99	5.68	44	261	10
ALU8	916	50.31	53.89	55.03	4.44	5.11	8.43	41	470	16
acc32	1187	65.50	59.87	49.09	5.46	5.57	6.45	41	528	17
acc64	3355	161.67	145.88	138.06	5.15	7.37	10.01	64	757	39
GCD	1505	23.55	24.17	23.32	18.63	20.68	20.05	45	604	21
FetchingUnit	5304	435.78	453.41	396.16	9.31	7.81	16.06	104	958	62
Average		1.00	1.06	0.97	1.00	1.12	1.32	1.00	10.34	0.43

VI. Conclusion

In this paper, we have proposed a timing-driven placement approach targeting asynchronous circuits. Our problem formulation only introduce polynomial number of performance constraints which is more efficient and effective than the approaches using loop-breaking techniques or enforcing explicit cycle constraints. The flexibility of our Lagrangian relaxation framework also makes our framework applicable to a wide range of asynchronous design styles. In addition, we simplified the timing-driven placement problem into a weighted wirelength minimization problem which can be solved by standard placement algorithms with the ability to handle net weights. We implemented a placement flow with a quadratic placer to demonstrate our idea. The experimental results shows our approach can greatly improve the performance for a given asynchronous circuits at the placement stage. The runtime and placement quality is also shown to be much better than the previous state-of-the-art.

References

[1] Semiconductor Industry Association. The International Technology Roadmap for Semiconductors, 2011.

[2] P. A. Beerel, R. O. Ozdag, and M. Ferretti, *A Designer's Guide to Asynchronous VLSI*. Cambridge University Press, 2010.

[3] C. J. Myers, *Asynchronous Circuit Design*. John Wiley & Sons, 2004.

[4] P. A. Beerel, G. Dimou, and A. Lines, "Proteus: An ASIC Flow for GHz Asynchronous Designs," *Design Test of Computers, IEEE*, vol. 28, pp. 36–51, Sept 2011.

[5] A. Yakovlev, P. Vivet, and M. Renaudin, "Advances in Asynchronous Logic: From Principles to GALS & NoC, Recent Industry Applications, and Commercial CAD Tools," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pp. 1715–1724, March 2013.

[6] Y. Thonnart, E. Beigne, and P. Vivet, "A Pseudo-synchronous Implementation Flow for WCHB QDI Asynchronous Circuits," in *Asynchronous Circuits and Systems (ASYNC), 2012*, pp. 73–80, May 2012.

[7] K. S. Stevens, R. Ginosar, and S. Rotem, "Relative Timing," *Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 1, pp. 129–140, 2003.

[8] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.

[9] A. P. Hurst, P. Chong, and A. Kuehlmann, "Physical Placement Driven by Sequential Timing Analysis," in *ICCAD 2004*, pp. 379–386, Nov 2004.

[10] E. Kounalakis and C. Sotiriou, "CPlace: A Constructive Placer for Synchronous and Asynchronous Circuits," in *Asynchronous Circuits and Systems (ASYNC), 2011*, pp. 22–29, April 2011.

[11] M. Iizuka and H. Saito, "A Floorplan Method for ASIC Designs of Asynchronous Circuits with Bundled-data Implementation," in *New Circuits and Systems Conference (NEWCAS), 2013*, pp. 1–4, June 2013.

[12] A. M. Lines, *Pipelined Asynchronous Circuits*. Master's thesis, California Institute of Technology, 1998.

[13] I. Sutherland and S. Fairbanks, "GasP: A Minimal FIFO Control," in *Asynchronous Circuits and Systems (ASYNC), 2001*, pp. 46–53, 2001.

[14] M. Singh and S. Nowick, "MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines," *Very Large Scale Integration (VLSI) Systems*, vol. 15, pp. 684–698, June 2007.

[15] T. Lin, C. Chu, J. Shinnerl, I. Bustany, and I. Nedelchev, "POLAR: Placement Based on Novel Rough Legalization and Refinement," in *ICCAD 2013*, pp. 357–362, Nov 2013.

[16] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Prentice Hall PTR, 1981.

[17] M. Najibi and P. A. Beerel, "Performance Bounds of Asynchronous Circuits with Mode-Based Conditional Behavior," in *Asynchronous Circuits and Systems (ASYNC)*, pp. 9–16, May 2012.

[18] P. A. Beerel, A. Lines, M. Davies, and N.-H. Kim, "Slack Matching Asynchronous Designs," in *Asynchronous Circuits and Systems (ASYNC), 2006*, pp. 11–pp, IEEE, 2006.

[19] J. Magott, "Performance Evaluation of Concurrent Systems using Petri Nets," *Information Processing Letters*, vol. 18, no. 1, pp. 7–13, 1984.

[20] J. T. Udding, "A Formal Model for Defining and Classifying Delay-insensitive Circuits and Systems," *Distributed Computing*, vol. 1, no. 4, pp. 197–204, 1986.

[21] C.-P. Chen, C. Chu, and D. F. Wong, "Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation," in *ICCAD 1998*, pp. 617–624, Nov 1998.

[22] C. R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Inc., 1993.

[23] J. Wang, D. Das, and H. Zhou, "Gate Sizing by Lagrangian Relaxation Revisited," *Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, pp. 1071–1084, July 2009.

[24] M. Prakash, *Library Characterization and Static Timing Analysis of Asynchronous Circuits*. ProQuest, 2007.

[25] Gurobi Optimizer: http://www.gurobi.com.