Communication-aware Module Placement for Power Reduction in Large FPGA Designs

Kalindu Herath, Alok Prakash, Udaree Kanewala, Thambipillai Srikanthan
Nanyang Technological University, Singapore
{kalindub001@e, alok@, n1702552l@e, astsrikan@}ntu.edu.sg

***Abstract*—Modern multi-million logic FPGAs allow hardware designers to map increasingly large designs into FPGAs. However, traditional FPGA CAD flows scale poorly for large designs, often producing low quality solutions in terms of performance and power in such cases. To improve design productivity, modular design methodology partitions a large design into subsystems, compiles them individually and finally collates the individual solutions to complete the mapping process. Existing work has attempted to partition large designs into smaller subsystems, based on the intra-subsystem communication, to reduce routing power dissipation. However, inter-subsystem communication has not been considered, especially, during the placement stage. In this work, we first show the adverse effect of ignoring the inter-subsystem communication during the placement stage. Next, we propose an inter-subsystem communication-aware placement technique using a Simulated Annealing based approach to achieve significant power savings. Experimental results show over 7% reduction in routing power when compared to the existing state-of-the-art partitioning flow that ignores inter-subsystem communication, while the routing power reduction is over 11% when compared to a commercial CAD tool such as Altera Quartus.**

***Keywords*-FPGA, CAD, modular design, placement**

I. Introduction

Rapid scaling of transistors has enabled commercial Field Programmable Gate Arrays (FPGA) manufacturers like Altera [1] and Xilinx [2] to produce FPGAs with millions of logic gates alongside diverse hard Intellectual Property (IP) cores like Digital Signal Processors (DSPs) and Block-RAMs (BRAMs) with varying capacities. This resource richness encourages designers to port immensely large designs into modern FPGAs. Apart from the resource abundance, low non-recurring engineering (NRE) cost and the time-to-market (TTM) with respect to Application Specific Integrated Circuits (ASICs) [3] have also made FPGAs popular.

Commercial FPGA Computer-Aided Design (CAD) tools are well known to produce high quality mappings for small to medium-scale designs. However, compared to the rapid development in FPGAs, the available FPGA CAD tools do not show sufficient maturity yet to efficiently map large designs [4]. This typically results in inferior quality-of-result (QoR) in terms of performance and power consumption for such designs. Longer compilation time [5] is also commonly observed with existing CAD flows, which eventually lowers design productivity and adversely affects the TTM. Among the major steps in CAD flow, the placement step is known to consume a significant amount of time, often taking close to 50% of the total CAD runtime [6]. Hence, commercial FPGA vendors as well as the research community have focused on improving the tools used for the placement step.

Existing placement techniques can be categorized into three types, namely, Simulated Annealing (SA)-based, partition-based and analytical. Altera uses parallel SA techniques [7] in their Q2P [6] placer to improve their Quartus CAD tool, while Xilinx incorporates analytical placement techniques [8] to improve theirs. Despite the current endeavors, significant effort is required to achieve better runtime and performance while mapping large FPGA designs.

Recently, Modular Design Methodology (MDM) has been gaining traction in both the research as well as commercial communities, especially in order to reduce the runtime of the existing CAD flows for large designs. The MDM technique divides a large design into a number of smaller modules. This technique is inspired by the fact that some sections in a design, for instance, board support package or external memory controllers, are not frequently changed during design time. Such sections can be compiled and mapped into an FPGA once, while continuing to iterate the rest of the sections as required. As a result, MDM enables code reuse and is effective in reducing the compile time as well as in improving design productivity. Many approaches which consider the reuse of pre-compiled modules (macros) and IP cores along with library-based design can be seen in literature. Dividing a large design into smaller modules in the MDM technique also allows designers to leverage the existing CAD tools that already produce high quality mappings for small to medium scale designs. However, MDM still suffers from performance degradation and does not explicitly focus on developing power-efficient solutions [9].

A significant contributor to the dynamic power consumption in FPGAs is the charging and discharging of capacitive loads that occur during the flow or *communication* of data via the interconnect fabric [10] [11], especially at long distances. This issue is further exacerbated by frequent communication over long distance links [9]. Hence, to reduce the overall power consumption in FPGAs, it is necessary to reduce the long-distance links, especially for high frequency communications. Authors in [9] proposed to divide a large FPGA design into modules such that frequently communicating entities fall within a single module, thereby maximizing intra-module communication frequency. In the subsequent compilation step, all the design entities from these modules were placed in closed proximity by the CAD flow, resulting in reduced interconnect power dissipation. However, the authors failed to consider the inter-module communication during the placement step that can potentially result in further power savings.

In this paper, we propose an inter-module communication-aware module placement strategy for placing modules that builds on the work in [9] to further lower interconnect power dissipation of large designs on FPGAs.

The rest of the paper is organized as follows. In Section II we briefly explain related workflows followed by a motivational example in Section III. In Section IV, we explain the proposed methodology. Next, we show the effectiveness of the technique in Section V and we conclude the paper in Section VI.

II. Related Work

MDM includes two phases; module creation and module assembling. During the first phase, a large design is divided into modules. Each module is treated as an isolated design hence it can go through CAD flow independently. During the next phase, the pre-compiled modules are assembled and inter-module links are routed to get the programmable bitcode of the entire design. Placement techniques used in the module assembly phase are discussed in this section.

Frontier [12] is a placement technique based on pre-placed IP cores, also known as macro-blocks. These macros are combined to form fixed sized and shaped clusters and then assigned to FPGA regions while minimizing the timing and wiring costs. Pre-compiled macro-block based design flow has been discussed in BPR [5] and HMFlow [13]. BPR introduces a new FPGA CAD tool for fast compilation of FPGA circuits where placement for each macro-block is selected from a database such that distance and expected routing congestion between macro-blocks are minimized. A modified version of VPR's SA placement algorithm [14] is used here to resolve the overlaps of placements that occur due to non-atomic type and size of blocks. HMFlow focuses on accelerating the compile time of designs having a 50% or less FPGA resource utilization. Three module placement algorithms are discussed in their work; recursive bi-partitioning placer, random placer and heuristic placer.The heuristic placer which is the fastest among them, considers the amount of connectivity between hard macros and prioritizes the large macros and macros with BRAMs and DSPs.

Qflow [15] separates the design into invariant and evolving sets. Placement of evolving modules is handled by an SA placer specialized for bigger (coarse-grained) modules. In [16], cluster placement methodology provides an estimation of the cluster's wire-length and criticality to the annealer. A floor planner based on Mixed Integer Linear Programming (MILP) [17] allows the designer to customize the objective function such that metrics like total wire length, area occupancy and aspect ratio can be weighted and incorporated as a linear combination. On the other hand, an analytical placer which optimize wire-length, is used in [18] instead of a SA based placer. Power-aware CAD techniques, for instance, [19] and [20], have not discussed their suitability for large designs. On the other hand, scalable techniques mainly focus on improving performance rather than power consumption. Although MDM is a scalable proposition, its module placement phase aims on improving compile time while optimizing wire-length or performance. Previous work lacks a module placement technique which considers data communication and power consumption.

III. Motivation

There are two major modes of power dissipation in FPGAs; (i) Static power and (ii) Dynamic power. The former is due to leakage current and subthreshold current whereas the latter is due to the charging and discharging of capacitive loads. Typical island-style FPGA consists of columns of Configurable Logic Blocks (CLBs) interleaved with BRAM and DSP columns, and input/output (IO) pins surrounding the chip area. The interconnect fabric, which connects CLBs, BRAMs, DSPs, and input/output(IO) pins to each other, consumes 80% of its total area. As a result, the dynamic power consumption of the interconnect fabric dominates the total dynamic power dissipation. Dynamic power dissipation (P) of an FPGA is modeled as follows [19];

$$P = \frac{1}{2} \sum_{i \in nets} C_i . \alpha_i . v_{dd}^2 \tag{1}$$

where $nets$ are all the connections and respective components of the FPGA, C is the capacitance of each net, α is the signal toggle rate (analogous to communication in this paper) and v_{dd} is the operating voltage.

In [9], the authors have explained that in a typical application, some code sections execute more frequently than the rest. Hence, the data communication of these code sections is higher as well. Consequently, on the FPGA, nets corresponding to such code segments show a higher toggle rate, α_i, due to data communication. In [9], authors have proposed to break down a large design into a set of modules based on the communication between nodes. We refer to these modules as *subsystems* throughout this paper. Nodes with high communication activity are included into the same subsystem and the CAD is instructed to place them in a close proximity. Having shorter routing wires for such connections reduces both C_i and α_i, reducing overall dynamic power dissipation considerably. Ideally, the subsystem creation should not introduce large inter-subsystem communication. While the approach manages to have shorter routing wires for the connections *within* the subsystems, it does not guarantee that the communication between the subsystems will also be minimized at the same time. Hence, wrong placement of such subsystems on an FPGA can allocate longer interconnect wires to the links with high inter-subsystem communication, thereby increasing dynamic power dissipation. [9] maps subsystems into FPGAs using existing CAD placement, which does not consider communication during the placement stage.

We elaborate this aspect using the following example. Consider a design with 7 design units illustrated as a graph shown in the figure 1(a). Each node shows its area requirement on FPGA space and intra-communication as $[l, m, d][\alpha]$ where l,m and d represent required number of CLBs, BRAMs and DSPs and α represents communication within the node/subsystem respectively. An edge is a connection between two design units where data flow occurs. It is annotated with communication between nodes/subsystems.

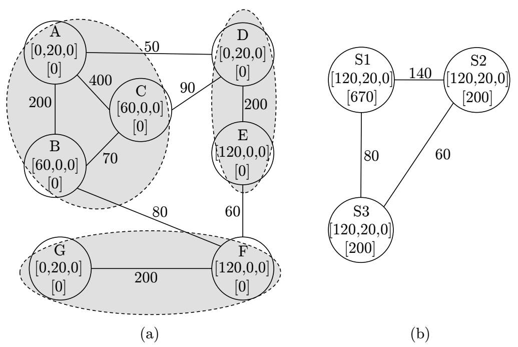

Figure 1: Subsystem creation based on communication

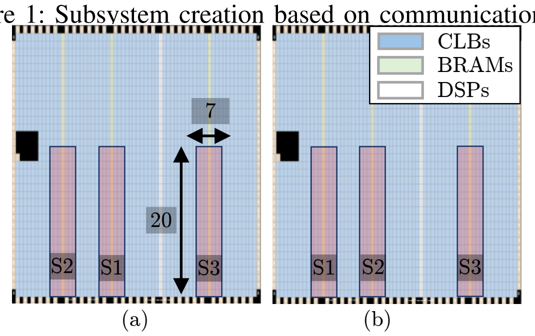

Figure 2: Mapping subsystems to FPGA without placement information

Performing a partitioning technique could result a network of subsystems as given in the figure 1(b). At this point, previous work has guided the CAD flow to group the relevant design units as subsystems. But the location of these subsystems on FPGA space is decided by the CAD flow. However, we observe that CAD flow could map subsystems with higher communication between them far apart. For instance, subsystem in figure 1(b) could be mapped on FPGA as shown in figure 2. However, the mapping in figure 2(a) would cause lower wire length for higher inter-subsystem communication links than the mapping in figure 2(b). This leads to higher power consumption in option a than in option b. In this paper, we strive to explicitly minimize the inter-subsystem communication to achieve further power savings.

IV. Methodology

In this section, we describe communication-aware subsystem placement methodology in detail. Similar to some workflows on subsystem placement in MDM, we use an SA based algorithm for our methodology. The inputs to our approach are (i) a network of subsystems and (ii) target FPGA architecture model, and the output is the placement details for each subsystem on the target FPGA space.

A. Problem Formulation

A network of subsystems can be represented as an undirected graph $G(V, E)$, where V is the set of vertices that represents each subsystem and E is the set of edges. An edge between two vertices indicates that there is one or more inter-subsystems connection between the components of both subsystems. Each edge is associated with a cost value, α to represent inter-subsystem communication frequency between associated subsystems. Each subsystem is annotated with area requirement $A(l, m, d)$ where l, m, and d represents the required number of CLBs, BRAMs and DSPs respectively.

Given a network of subsystem $G(V, E)$, area requirement of each subsystem $v \in V$ as A_v, communication frequency between subsystems of each inter-subsystem connection $e \in E$ as α_e, the placement problem is to find a set of non-overlapping rectangular region L_v on FPGA space for each subsystem v, where L_v satisfies the area requirement of A_v.

B. Algorithm

1) Simulated Annealing Algorithm: We explain our SA based placement methodology in algorithm 1. A typical SA begins with a random initial placement solution s (line 2). Placement in CAD flow considers allocating atomic components such as CLBs and BRAMs on FPGA space. But in subsystem placement, a number of such atomic components need to be placed. Instead of deciding the location of each atomic component, subsystem placement decides a region where all the components that belong to a subsystem can be placed on FPGA. $COST(s)$ (line 3) refers to the placement quality of the solution s. Temperature (T) is analogous to the current iteration of annealing. There can be n iterations (line 5) before it converges to a solution. In each iteration of the algorithm, the solution of the previous iteration is altered by moving one or more subsystems within FPGA space (line 8), which is called a neighbor solution s_N. If s_N is better than reported best s_{best}, s_N is assigned to be the best solution so far (line 11-12). However, in case if the new solution s_N is not the reported best solution, it is still considered as a valid solution if it satisfies a probability equation P (line 13-14).

2) Neighbor generation: In each iteration, a subsystem is selected randomly to move withing FPGA space to create neighbor solution. Maximum allowed movement along horizontal and vertical direction on FPGA space (δx_{max} , δy_{max}) of the selected subsystem is a function of T, where δx_{max} and δy_{max} are gradually decreased when T reduces. Once the maximum movement for current T is obtained, the selected subsystem is moved in the range [$-\delta x_{max}$,δx_{max}] horizontally and [$-\delta y_{max}$,δy_{max}] vertically.

3) Shape of the subsystems: Subsystems with requirement for multiple resource types (CLBs and BRAMs) are split into sub-modules in a way such that sub-modules contains only one resource type. For instance, as shown in figure 3(a), subsystem $S3$ with area requirement $A_3 = (120, 20, 0)$ is divided into two whose area requirements are $A_3 = (120, 0, 0)$ and $A_{3.1} = (0, 20, 0)$ respectively. We set the communication between subsystem $S3$ and $S3.1$ to a very large number (N). However, the original edges in the graph G are preserved. This subdivision of subsystems is done to cater for the area requirement A_v during neighbor generation. For instance, moving a subsystem having both CLBs and BRAMs from location L_1 to L_2 might not find

Algorithm 1 Communication-aware subsystem placement

```
1: procedure SIMULATED_ANNEALING
2:     $s \leftarrow$ INIT_PLACEMENT()
3:     $\phi \leftarrow$ COST($s$)
4:     $s_{best} \leftarrow s, \phi_{best} \leftarrow \phi$
5:     $T_0 \leftarrow n$
6:     for $T \leftarrow T_0 ... 0$ do
7:         do
8:             $s_N \leftarrow$ NEIGHBOR($s, T$)
9:         while MAX_OVERLAP($T$) $<$ OVERLAP($s_N$)
10:        $\phi_N \leftarrow$ COST($s_N$)
11:        if $\phi_N < \phi_{best}$ then
12:            $s_{best} \leftarrow s_N, \phi_{best} \leftarrow \phi_N$
13:        if P($e, e_N, T$) $>$ RAND() then
14:            $s \leftarrow s_N, \phi \leftarrow \phi_N$
15:    return $s_{best}$
16: procedure NEIGHBOR($s, T$)
17:    $c \leftarrow$ SELECT_SUBSYSTEM($s$)
18:    $\delta x_{max} =$ F($T$), $\delta y_{max} =$ F($T$)
19:    $\delta x \leftarrow$ RAND($-\delta x_{max}, \delta x_{max}$)
20:    $\delta y \leftarrow$ RAND($-\delta y_{max}, \delta y_{max}$)
21:    $c(x, y) \leftarrow c(x + \delta x, y + \delta y)$
22:    return $s$
```

BRAMs at L_2. However, in this work, we do not consider the importance of the shape of a subsystem on FPGA space. Therefore, in each iteration, we set the shape of subsystems with CLBs to have a height to width ratio closer to 1 as shown in figure 3(b). It has been shown in [21] that shapes with aspect ratio closer to 1, help in reducing the wire length within a subsystem.

4) Overlap of subsystems: Typical SA placement algorithms do not allow overlaps of atomic components during each iteration. However, since subsystems are non-atomic components, it is difficult to produce a non-overlapping solution in each iteration. The search space is also limited if overlapping of subsystems is not allowed, which might cause the algorithm to converge to a local minimum. To avoid such situations, our approach allows overlapping of subsystems. We use an exponential function of T to decide maximum allowable overlaps of subsystems, which reaches zero as T decreases. Number of overlaps are counted as the area of the overlapped region. Since we divide the subsystems into sub-modules with single resource type, intersection of $S2$ and $S3.1$ in figure 3 is not considered as an overlap.

5) Congestion model: Routability evaluation is an important step in a placement algorithm. For instance, placing a large number of components in a smaller area could lead to excessive requirement of routing resources. Routing process avoids highly congested areas and therefore may use longer wires to connect two components. Hence, quality of result (performance and power) is degraded. In our approach, we incorporate a bounding box based congestion analysis to estimate the routing congestion similar to [22]. In this model, we consider the FPGA space as a 2D array, which we call as

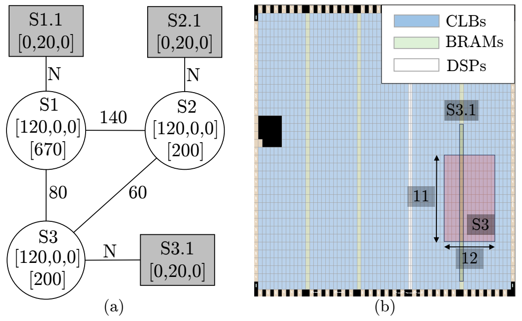

Figure 3: a) Split subsystems b) Shape of subsystems

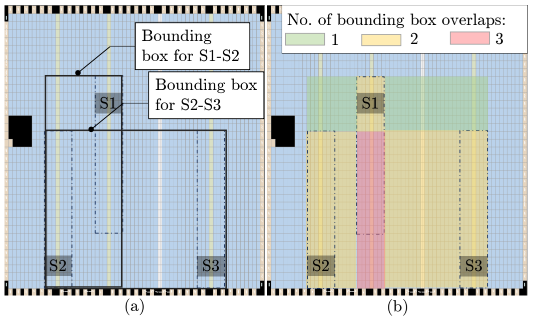

Figure 4: Congestion Model

congestion map. All congestion map elements are initialized to zero at start. For each inter-subsystem connection in figure 1(b), a bounding box can be formed as figure 4(a). Note that the bounding box for the edge between $S1$ and $S3$ is not shown. Next, congestion map elements relevant to each bounding box is incremented. Overlapping bounding boxes, therefore, reflect high congestion regions. Congestion map relevant to the subsystem placement shown in figure 4(a) is shown in 4(b). We update the congestion map in each iteration of the algorithm.

6) Cost Function and constraints: The main purpose of our work is to avoid longer routing wires, especially for high inter-subsystem communication links. Therefore, our objective cost function is to minimize

$$COST(s) = \sum_{e \in E} l_e . \alpha_e \quad (2)$$

where l_e is half perimeter wire length for the bounding box enclosing the subsystems relevant to edge e, whereas α_e is inter-subsystem communication. While minimizing the cost function, we also consider a routing congestion parameter C. For a valid placement, its maximum congestion derived above must be less than empirically evaluated C.

C. Implementation on FPGA

It is important to note that our approach of communication-aware subsystem placement does not depend on a specific CAD tool. For implementation and evaluation of our methodology, we use Altera's Quartus CAD flow [23]. But, an equivalent approach could be followed with Xilinx CAD flow as well [24]. The first phase of our subsystem placement approach partitions a

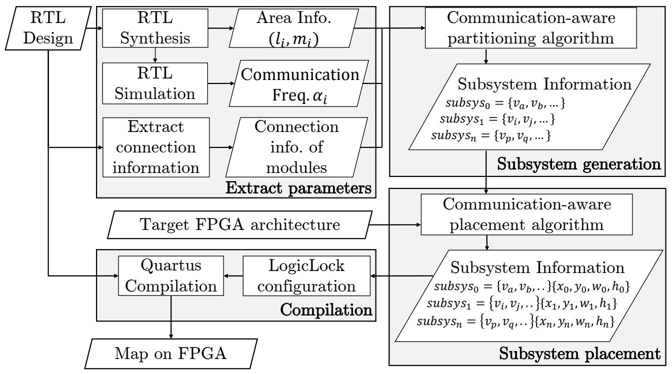
RTL Design
RTL Synthesis
Area Info. (l_i, m_i)
RTL Simulation
Communication Freq. α_i
Extract connection information
Connection info. of modules
Extract parameters
Communication-aware partitioning algorithm
Subsystem Information
$subsys_0 = \{v_a, v_b, \dots\}$
$subsys_1 = \{v_i, v_j, \dots\}$
$subsys_n = \{v_p, v_q, \dots\}$
Subsystem generation
Target FPGA architecture
Communication-aware placement algorithm
Subsystem Information
$subsys_0 = \{v_a, v_b, ..\}\{x_0, y_0, w_0, h_0\}$
$subsys_1 = \{v_i, v_j, ..\}\{x_1, y_1, w_1, h_1\}$
$subsys_n = \{v_p, v_q, ..\}\{x_n, y_n, w_n, h_n\}$
Subsystem placement
Quartus Compilation
LogicLock configuration
Compilation
Map on FPGA

Figure 5: Communication-aware mapping methodology

large design into subsystems as done in [9]. This partitioning technique works at entity/module level of an RTL code. Entities which have high communication between them are grouped into subsystems. For that, a given RTL code is processed to extract the following details. This step is faster than the full CAD compilation.

1) Area of the RTL entities: RTL synthesis step in Quartus CAD flow estimates the resource requirement for each entity. This step is faster than the full CAD compilation.
2) Communication frequency between entities: An RTL simulation can dump all the signal toggles of desired connections between entities.
3) Connectivity of entities: Entity-level RTL connection information is extracted by doing text processing on code

Once these parameters are extracted, the communication-aware partitioning algorithm forms subsystems such that intra-subsystem communication is much higher than inter-subsystems communication. The output of the partitioning framework is a list of subsystems and a list of design entities that belongs to each subsystem. We modified the output slightly in order to obtain (i) a graph representation of subsystems (ii) the inter-subsystem communication frequency α_e of each connection e between subsystems (iii) area requirement A_v of each subsystem v. Additionally, for our placement algorithm, target FPGA architecture floorplan information is needed. It should show the position of CLB, BRAM and DSP columns and the number of resource columns and rows in the floorplan. The placement algorithm produces the placement for each subsystem v with (i) left bottom coordinate (x_v, y_v) of the rectangular region in FPGA space and (ii) width w_v and the height h_v of the rectangular region. We can use these parameters to invoke Quartus LogicLock feature during the compilation to define subsystems on FPGA space. Note that, in [9], placement information to the LogicLock feature is not given allowing CAD flow to decide the placement of each subsystem, whereas this work overrides the tool's placement decision with our placement information.

V. Results and Discussion

Next, we discuss the performance of the proposed communication-aware placement strategy. We select application set coded in RTL for the evaluation. Compilation reports produced by Quartus CAD flow are used for the comparison.

A. Comparison strategy

We compare our methodology with two existing compilation flows: (i) Original Quartus compilation without subsystems (ii) Quartus compilation with subsystems without placement information as proposed in [9]. Full Quartus compilation is done for these two cases and for proposed solution followed by gate-level simulation to get the most accurate power estimations from Quartus PowerPlay. Since, we mainly target to reduce the power dissipation in the FPGA interconnect, we compare the routing power dissipation using the three approaches. Since, the operating frequency affects the power dissipation, the routing power is measured at a common operating frequency for all tests.

B. Benchmark Applications and Target FPGAs

Selected benchmark applications are similar to the applications in [9]. We use three handwritten applications as presented in [9]. In addition, two applications have been developed by modifying applications from the popular polybench [25] benchmark suite. However, we extend the benchmark application set that are inspired by polybench kernels by introducing three more applications by following the same concept for Gesummv, Cholesky and Symm kernels to develop gesummv′, cholesky′ and symm′ respectively. In addition, for a comprehensive evaluation, the benchmark set is mapped into three newer devices in Altera Cyclone IV and Stratix IV series. Although the methodology is compatible with any device series including the latest Cyclone V and Cyclone 10 series (or latest Xilinx devices), the gate level simulation is not supported in the latest series which is required to get a precise power measurement. It should be noted that this is not the limitation of the proposed work. Therefore, Cyclone IV and Stratix IV were the latest series we could use to show the effectiveness of the methodology.

C. Results and Discussion

Three hand-coded applications are mapped to Cyclone IV EP4CGX50 device, gesummv′ is mapped to slightly bigger Stratix IV EPSGX70 device and the rest of the applications are mapped to Cyclone IV EP4CGX110 device. Each application is compiled in three different test versions as stated above. Table I reports percentage routing power reduction of the new methodology over (i) default Quartus compilation and (ii) over [9]. As observed from these results, the proposed communication-aware placement technique for subsystem placement during CAD flow helps to further reduce routing power. In particular, the proposed approach helps to reduce routing power of the benchmark applications ranging from 3.55% to 10.75% (average of 7.08%) when compared to [9]. It is also shown that the average routing power reduction, when compared to the default Quartus compilation, ranges from 3.69% to 18.21% for the application set with an average routing power reduction of 11.51%.

Table I: Routing Power Saving

Application	Target Device	$P_{routing}$ reduction Over default	$P_{routing}$ reduction Over [9]
Synth1	EP4CGX50	18.21%	8.10%
Synth2	EP4CGX50	16.40%	10.49%
Synth3	EP4CGX50	15.91%	6.19%
atax′	EP4CGX110	12.69%	10.75%
bicg′	EP4CGX110	3.69%	3.55%
gesummv′	EP4SGX70	6.68%	5.48%
cholesky′	EP4CGX110	8.19%	6.05%
symm′	EP4CGX110	10.31%	6.05%

Communication-aware subsystem placement actually depends on how the application is partitioned into subsystems. Some application partitioning might not create connections with high inter-subsystems communication. For instance, inter-subsystems links in bicg′ are not as significant as that of applications like Synth1 or atax′. As a result, the effectiveness of this methodology depends on the inter-subsystem communication. One can argue that placement technique would not be necessary if all the high communication nodes are included into subsystems during partition generation. However, it should be remembered that making larger subsystems reduces the effectiveness of creating subsystems in the first place, since it leads to the original problem of mapping large designs on the FPGA. Instead, the current subsystems are generated to strike a balance between the size and intra-susbsystem communication as described in [9]. In future, we will explore the subsystem generation step with both intra- as well as inter- subsystem communication frequencies in order to achieve even better results.

VI. Conclusion

In this paper, we presented a inter-subsystem communication aware placement technique that produces high quality solutions with lower power consumption, especially for large FPGA designs. The proposed Simulated Annealing based approach reduces the distance between subsystems with higher inter-subsystem communication. This approach results in reducing routing power consumption by over 7% when compared to an existing methodology that does not consider the inter-subsystem communication frequency and by over 11% when compared to a commercial CAD tool such as Quartus. In future, we proposed to also evaluate the footprints of subsytems for further power savings.

Acknowledgment

This project was partially funded by the National Research Foundation Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme with the Technical University of Munich at TUMCREATE.

References

[1] Altera. [Online]. Available: https://www.altera.com/

[2] Xilinx. [Online]. Available: https://www.xilinx.com/

[3] S. M. Trimberger, "Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology," *Proceedings of the IEEE*, 2015.

[4] H. Bian *et al.*, "Towards scalable placement for FPGAs," in *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, 2010.

[5] J. Coole *et al.*, "BPR: fast FPGA placement and routing using macroblocks," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2012.

[6] M. An *et al.*, "Speeding up FPGA placement: Parallel algorithms and methods," in *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*, 2014.

[7] A. Ludwin *et al.*, "Efficient and deterministic parallel placement for FPGAs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2011.

[8] T.-H. Lin *et al.*, "An efficient and effective analytical placer for FPGAs," in *Proceedings of the 50th Annual Design Automation Conference*, 2013.

[9] K. Herath *et al.*, "Communication-aware Partitioning for Energy Optimization of Large FPGA Designs," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, 2017.

[10] L. Shang *et al.*, "Dynamic power consumption in Virtex-II FPGA family," in *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*.

[11] T. Tuan *et al.*, "A 90nm low-power FPGA for battery-powered applications," in *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*.

[12] R. Tessier, "Fast placement approaches for FPGAs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2002.

[13] C. Lavin *et al.*, "HMFlow: Accelerating FPGA compilation with hard macros for rapid prototyping," in *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, 2011.

[14] V. Betz *et al.*, "VPR: A new packing, placement and routing tool for FPGA research," in *International Workshop on Field Programmable Logic and Applications*, 1997.

[15] T. Frangieh *et al.*, "A design assembly framework for FPGA back-end acceleration," in *Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on*, 2012.

[16] F. Gharibian *et al.*, "Identifying and placing heterogeneously-sized cluster groupings based on FPGA placement data," in *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*, 2014.

[17] M. Rabozzi *et al.*, "Floorplanning for partially-reconfigurable fpga systems via mixed-integer linear programming," in *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*, 2014.

[18] M. Gort *et al.*, "Design re-use for compile time reduction in FPGA high-level synthesis flows," in *Field-Programmable Technology (FPT), 2014 International Conference on*, 2014.

[19] S. Gupta *et al.*, "CAD techniques for power optimization in Virtex-5 FPGAs," in *Custom Integrated Circuits Conference, 2007. CICC'07. IEEE*, 2007.

[20] J. Lamoureux *et al.*, "On the interaction between power-aware FPGA CAD algorithms," in *2003 IEEE/ACM international conference on Computer-aided design*.

[21] M. Wang *et al.*, "Multi-million gate FPGA physical design challenges," in *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, 2003.

[22] Y. Zhuo *et al.*, "A congestion driven placement algorithm for FPGA synthesis," in *Field Programmable Logic and Applications, 2006. FPL'06. International Conference on*.

[23] "Increasing Productivity with Quartus II Incremental Compilation," https://goo.gl/uy225f.

[24] "Vivado Design Suite User Guide-Hierarchical Design," https://goo.gl/6bUqqD.

[25] L.-N. Pouchet, "Polybench: The Polyhedral Benchmark Suite," http://web.cs.ucla.edu/˜pouchet/software/polybench/.