An Architecture Independent Packing Method for LUT-based Commercial FPGA

Meng Yang
State Key Lab of ASIC and Systems, Fudan University, Shanghai, China
Email: mengyang@fudan.edu.cn

Jinmei Lai
State Key Lab of ASIC and Systems, Fudan University, Shanghai, China
Email: jmlai@fudan.edu.cn

A.E.A. Almaini
School of Engineering and the Built Environment, Edinburgh Napier University, Edinburgh, UK
Email: a.almaini@napier.ac.uk

***Abstract*—This paper proposes an efficient architecture independent packing method for commercial FPGA. All specific logics of commercial FPGA such as carry chain arithmetic, x-LUT, are pre-designed into reference circuits according to its architecture. Due to complex architecture of contemporary FPGA, to enumerate all reference circuits in a fine-grain manner is impractical. To overcome this problem, coarse-grain manner is adapted in the approach. By using constraint satisfaction problem technique the proposed method matches pre-designed reference circuits from the given user logic circuit. Transformation from the reference circuit to the pre-packed cluster is simplified by using several specifically designed instructions. In the next stage, those directly connected FFs are absorbed into the pre-packed clusters. The Last stage packs LUTs and FFs into clusters in a delay-based manner. This method is architecture independent and can be applied for any other commercial FPGAs as long as the pre-designed reference circuits are modified accordingly. The results obtained and compared with commercial tool, ISE MAP, and academic tool, PAM MAP, have shown the effectiveness of the proposed method.**

***Index Terms*—Packing, Algorithm, Computer-aided design, FPGA**

I. INTRODUCTION

Contemporary commercial field-programmable gate arrays (FPGAs) consist of a cluster of configurable logic blocks (CLBs) formed by look-up tables (LUTs) and flip-flops (FFs) as well as arithmetic circuitry, configurable I/O blocks (IOBs) and specialised hard IP blocks. For example, a SLICE, a half of CLB, in the latest Xilinx Virtex-7 FPGA family device contains four six-input LUTs, eight FFs, carry chain arithmetic logic and other circuitry. It is widely acknowledged that FPGAs are slower, less area-efficient and less power efficient than custom ASICs [1]. However, the programmability of FPGAs, gives them the advantage of short time to market. As a result, they have been widely used in a variety of applications such as domestic communications and automotive electronics.

Packing, which falls between technology mapping and placement, is an extremely important step of the FPGA computer aided design (CAD) flow. This step is most commonly regarded as packing LUTs and FFs together to form clusters [2]. However, in commercial FPGAs, packing is the step that the various logic gates of technology mapped circuit including not only LUTs and FFs but also other logic gates are mapped to FPGA fabric according to the available hardware resources. Packing algorithms are well-studied in the literature for the academic FPGA model, which consists of several basic logic elements (BLEs). Each BLE has one LUT and one FF. The FF can be optionally bypassed for implementing combinational logic only. Local interconnect is available for realising fast paths within the cluster. The output of LUT/FF drives both local interconnect and general interconnect. Inputs to the cluster come from general interconnect [2].

The earliest work based on the academic FPGA model proposed an area-driven packing algorithm (VPack) in the earlier version of versatile placement and routing (VPR) CAD tool [3]. This used the simplest graph pattern match to pack LUTs and registers into BLEs in the first step and packs BLEs into clusters in the second step. Marquardt further extended the previous work carried out by Betz to perform timing-driven packing (T-VPack) [4] and improve speed and density. Recently, Verilog-to-routing (VTR) [5], the latest version of VPR was proposed, in which hardcore IPs are supported in the packing stage.

Tom et al [6] proposed a non-uniform depopulation technique, (Un/DoPack), which runs the FPGA CAD flow twice. First iteration is the regular CAD flow. In the second iteration, packing uses the layout result of the first iteration and depopulates the congested regions. While reducing the channel width, Un/DoPack, similar to the other depopulation-based packing approaches, observes an increase in total area and critical path delay.

T-NDPack [7] proposed an objective cost function with consideration of the criticality in terms of delay and

doi:10.4304/jcp.9.5.1131-1137

routability simultaneously, which consequently reduces the channel width requirements and the depth of the critical path. However, it incurs logic area overhead. It was claimed that minimum channel width and critical path delay were reduced by 11.07% and 2.89% respectively while increasing the number of CLBs by 13.28% compared to T-VPack.

Easwaran et al proposed a routability driven power-aware packing method (W-T-VPack) [8] with introduction of a new packing cost function based on predicted individual net length. It claimed that W-T-VPack outperforms T-RPack [9] and iRAC [10] in terms of energy by 11.23% and 9.07%, respectively.

Rajavel et al proposed a many-objective FPGA circuit packing strategy (MO-Pack) [11] that minimised the channel width and the energy of a circuit implementation without incurring any overhead on critical path delay.

Yang et al proposed a yet another many-objective FPGA packing method (YAMO-Pack) [12]. It claimed that YAMO-Pack outperforms iRAC and MO-Pack in terms of channel width by 38.8% and 42.2%, respectively and in terms of delay by 11.8% and 11.5%, respectively. However, it requires acceptably more CPU time.

All methods mentioned above target the academic FPGA model, which is significantly simpler than that used for commercial FPGAs. Ahmed et al [13] from Xilinx reported an architecture-specific packing for Virtex-5 FPGAs. However, it can only be used for Xilinx FPGA devices. Moreover, Shao, et al developed an area-driven architecture independent PAM MAP algorithm [14]. The architecture they used differs from the academic model, but it targets area reduction only. To our best knowledge, no timing-driven architecture independent packing method has ever been published for commercial FPGA. The remainder of the paper is organized as follows. Section II gives details of Virtex-7 FPGA architecture, which will be used in the experiment for demonstration. Constraint satisfaction packing techniques and specific designed instructions are given in Section III. Section IV discusses comparison results between the proposed method and other tools. Conclusion is then given in Section V.

II. Virtex-7 FPGA CLB Architecture

To show the complexity of the contemporary commercial FPGA architecture, a virtex-7 FPGA is reviewed in this section. This architecture will be used for evaluation experiment for demonstration purpose. A Virtex-7 logic block, which is referred to as a CLB, comprises two SLICEs (SLICEL and SLICEM) and a switch matrix. SLICEL and SLICEM are exactly identical, except that LUT in SLICEL is used for logic only and SLICEM can be used for implementing memory cells. The switch matrix allows for connections from a SLICE back to the same SLICE, between the two SLICEs, as well as into rows and columns of general interconnect. Each SLICE contains four 6-input LUTs and 8 flip-flops. The LUTs in Virtex-7 are implemented as what Xilinx called true 6-LUTs, rather than being constructed using smaller LUTs that can be optionally combined together via multiplexers. The output of two true 6-LUTs, either in top half of a SLICE or bottom half of a SLICE, can be constructed as one 7-LUT via multiplexer F7MUX. Two 7-LUTs can function in one SLICE at the same time. Besides, two 7-LUTs can be further combined together via multiplexer F8MUX to form an 8-LUT in one SLICE. Both outputs of 7-LUT and 8-LUT can be registered individually. Fig. 1 shows the architecture of a SLICE of Virtex-7 FPGAs.

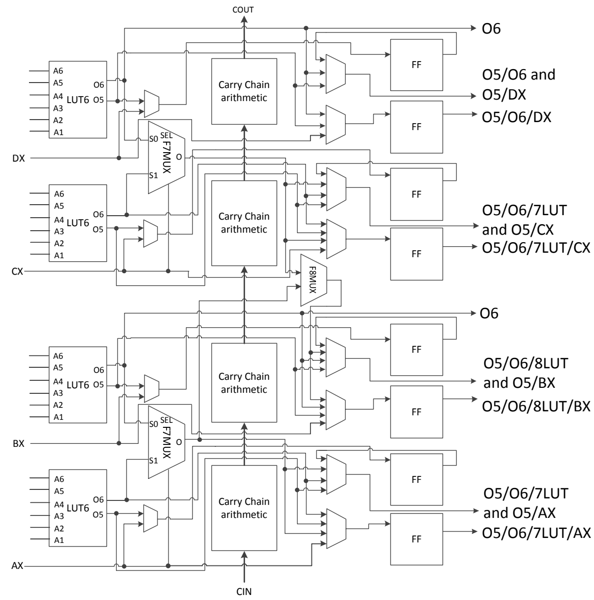

Fig. 1 Virtex-7 SLICE architecture.

III. Constraint Satisfaction Problem Technique for FPGA Packing

A constraint satisfaction problem [15] is defined by an ordered set of n variables $X=(1,2,\cdots,n)$, a finite domain D_i of possible values for each variable i, and a set of constraints among variables. A constraint $R_{j_1,j_2,\cdots,j_r}$ on the ordered set of variables $(j_1,j_2,\cdots,j_r)$ is a subset of $D_{j_1}\times D_{j_2}\cdots\times D_{j_r}$, which only contains the allowed combinations of values for variables $j_1,j_2,\cdots,j_r$.

An isomorphism of a graph $G_1=(V_1,E_1)$ with a sub-graph of a graph $G_2=(V_2,E_2)$ is equivalent to the constraint satisfaction problem [16]. A variable i is associated with each vertex $v_i\in V_1$, and all variables take values on domain V_2. Let n be the cardinality of V_1. Finding a sub-graph isomorphism is then equivalent to finding a complete assignment satisfying the following structure constraint:

$$R_{i,j}=\left\{\begin{matrix}(v_a,v_b)\in V_2\times V_2 \mid v_a\neq v_b \wedge edge(G_1,i,j)\\ \Rightarrow edge(G_2,v_a,v_b)\end{matrix}\right\} \quad (1)$$
$$\text{for all } i,j=1,2,\cdots,n \text{ with } i\neq j$$

Packing problem is similar to isomorphic match problem. A user circuit C can be described by a directed graph $G_1=(V_1,E_1)$, where each vertex $v_i\in V_1$ in G_1 corresponds to a component or a primary input or primary output in *C*, and each directed edge $e_i \in E_1$ corresponds to a wire connecting between two different vertexes in *C*. The set of given circuits is a set of configurable circuits implementing different types of logic functions, which is known as reference circuits from packing point of view and can also be described by directed graphs respectively. Each directed graph $G_2 = (V_2, E_2)$ corresponds to a reference circuit. These configurable circuits are pre-constructed manually according to available FPGA hardware logic resources. Packing algorithm identifies all isomorphic matches in a user design circuit according to a set of given reference circuits.

In order to match reference circuits in a user design circuit, several constraints should be applied. Type constraint should be satisfied for the purpose of matching exact type of vertex in the circuit such as LUT and FF. Start constraint is used for the outgoing edge from a vertex. Similarly, end constraint is for the incoming edge from a vertex. These two constraints are used for matching one particular edge of graph, i.e., from one type of logic gate to another. Input constraint and output constraint are used for primary input and primary output respectively. Shared input constraint identifies shared inputs which is used in the case of more than one sink net shared by two pins.

As long as the reference circuits represent all the functionalities that FPGA hardware resources can implement, it can always find a feasible solution for packing result. However, it is impossible to enumerate all reference circuits for a complex contemporary FPGA, which makes isomorphism packing impractical. Let us consider a case of two 6-LUTs and a 2to1 multiplexer F7MUX forming one 7-LUT in one SLICE. If ignoring sequential outputs, there are 4 cases already, as shown in Fig. 2. Hence, four reference circuits must be constructed in order to match all these patterns. If considering sequential outputs, the number of combination patterns can be increased significantly. It is therefore crucial to select the proper reference circuits, achieving not only less number of reference circuits but also covering all the functionalities a SLICE of contemporary FPGA can implement.

In order to reduce the number of reference circuits, the construction of reference circuits in the proposed method only considers combinational logic. Although the sequential logic is not included in the reference circuits, it will be dealt with after graph pattern match in the second step of packing method. By doing so, it can not only reduce the complexity of the individual reference circuit but the count number of the reference circuits as well. Those different logic functions that behave a similar function are categorized as one function type. For example, there are four different ways to form 7-LUT in one SLICE, as shown in Fig.2(a), Fig.2(b), Fig.2(c) and Fig.2(d), respectively. The graph, shown in Fig. 2(a), is the subset of the graph shown in Fig.2(d). The graphs, shown in Fig.2(b) and Fig.2(c), are also the subset of the graph shown in Fig.2(d). Therefore those four graphs are considered as one function type. One function type accordingly has only one reference circuit. The directed graph of reference circuit is modified by inserting a virtual primary input (VPI) at the input of the vertex and inserting a virtual primary output (VPO) at the output of the vertex, as shown in Fig. 3.

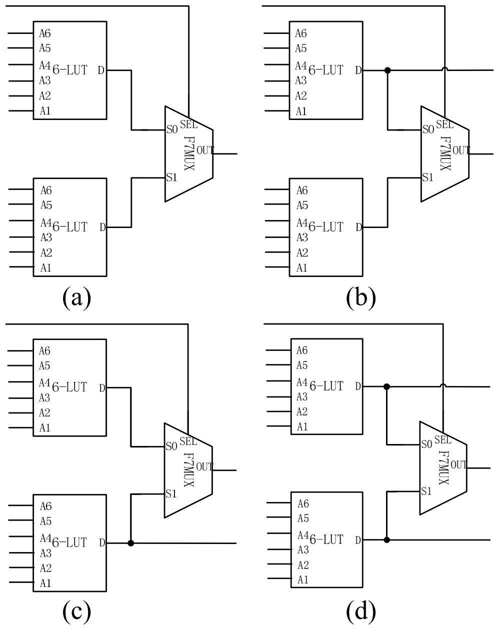

Fig. 2 (a) 7-LUT with single output, (b) 7-LUT with 2 outputs, in which one output is from 7-LUT and the other is from the top 6-LUT, (c) 7-LUT with 2 outputs, in which one output is from 7-LUT and the other is from the bottom 6-LUT, (d) 7-LUT with 3 outputs, in which two outputs are from 6-LUTs and one is from 7-LUT.

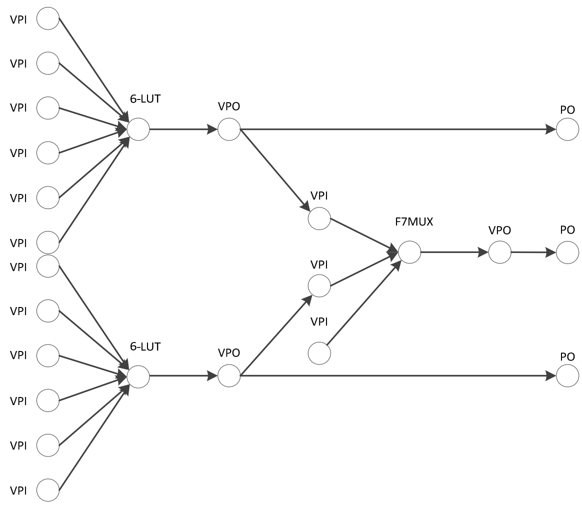

Fig. 3 Directed graph for 7-LUT reference circuit

After a reference circuit is matched from a given user design circuit by utilising graph constraint satisfaction technique, transformation from the reference circuit to the pre-packed cluster process is required. The process for the newly created cluster involves creating a new cluster, wires connection, wires disconnection and specifying configurations such as buffer, MUX, LUT and FF. A key observation is that for a given reference circuit wire connections for the newly created cluster and the configuration settings never alter. In addition, the transformation processes of different reference circuits are identical. The net connections and the configuration values for different created clusters are different. Therefore each step in the process can be used as an instruction. As a result, the whole process works as executing instruction one after another. For a different specific architecture, reference circuits are different and those reference circuits must be modified accordingly. However, the execution of the instruction is the same for a different architecture. The designed instructions are architecture independent, simple but effective, as shown in TABLE I.

TABLE I
SUMMARY OF INSTRUCTIONS

Instructions	Description
create_instance (inst, type)	Creates a new instance according to its type.
unhook (insta.p1)	Disconnects the pin with name p1 of instance from its net.
connect (inst1.p1, inst2.p1)	Connects the pin with name p1 of instance1 to the net which has pin with name p1 of instance2.
reconnect (inst1.p1, inst2.p1)	Disconnects pin with name p1 of instance1 from its net and connects it to the net which has pin with name p1 of instance2.
xconnect (inst.p1, inst.p2)	Exchanges the net connections of two different pins, p1 and p2, which is used for two pins swap.
set_configuration(inst, value)	Sets one configuration of the instance.
copy_property (inst1,value1, inst2,value1)	Copies value1, which is one property of instance1, to instance2.
set_ property (inst, value)	Sets one property of the instance.
create_instance (inst, type)	Creates a new instance according to its type.
unhook (inst.p1)	Disconnects the pin with name p1 of instance from its net.
connect (inst1.p1, inst2.p1)	Connects the pin with name p1 of instance1 to the net which has pin with name p1 of instance2.

Example 1: Use designed instructions to create a SLICE with functionality of 7-LUT by combining two 6-LUTs and F7MUX. Assume 6-LUT has six inputs A1, A2, A3, A4, A5 and A6 as well as two outputs O5 and O6. F7MUX has three inputs I0, I1 and S as well as one output O. The SLICE has the same architecture as Xilinx-7 FPGA.

1. Create a slice with name slice_a by using instruction create_slice (slice_a, SLICE)
2. Reconnect function generator A6LUT input connections to the newly created slice inputs by using following instructions.

   reconnect (A6LUT.A1, slice_a.A1)
   reconnect (A6LUT.A2, slice_a.A2)
   reconnect (A6LUT.A3, slice_a.A3)
   reconnect (A6LUT.A4, slice_a.A4)
   reconnect (A6LUT.A5, slice_a.A5)
   reconnect (A6LUT.A6, slice_a.A6)

3. Reconnect function generator B6LUT input connections to the newly created slice inputs by using following instructions.

   reconnect (B6LUT.A1, slice_a.B1)
   reconnect (B6LUT.A2, slice_a.B2)
   reconnect (B6LUT.A3, slice_a.B3)
   reconnect (B6LUT.A4, slice_a.B4)
   reconnect (B6LUT.A5, slice_a.B5)
   reconnect (B6LUT.A6, slice_a.B6)

4. Reconnect wires to the newly created slice outputs and internal connections by using following instructions, in which A, B, AX and AMUX are SLICE pin name of Xilinx Vertex 7 series family FPGA device.

   reconnect (A6LUT.O6, slice_a.A)
   reconnect (B6LUT.O6, slice_a.B)
   reconnect (F7MUX.S, slice_a.AX)
   reconnect (F7MUX.O, slice_a.AMUX)
   connect (A6LUT.O6,F7MUX.I0)
   connect (B6LUT.O6,F7MUX.I1)

5. Copy properties from 6-LUTs and set properties by using following instructions, in which “INIT” is the 6-LUT initial value and “NAME” is the 6-LUT instance name.

   copy_property (A6LUT,INIT, slice_a,A6#LUT)
   copy_property (B6LUT,INIT, slice_a,B6#LUT)
   copy_property (A6LUT,NAME, slice_a, ANAME)
   copy_property (B6LUT,NAME, slice_a, BNAME)
   set_property (slice_a, FXLUT::TRUE)

6. Set configurations by using following instructions, in which AOUTMUX, AUSED, BUSED are SLICE configurations of Xilinx Vertex 7 FPGA device.

   set_configuration (slice_a, AOUTMUX::F7)
   set_configuration (slice_a, A6#LUT::A6#LUT)
   set_configuration (slice_a, AUSED::0)
   set_configuration (slice_a, BUSED::0)

For the consideration of timing issue, the constraint satisfaction problem technique of graph matching mentioned in early sections is only used for the first stage of the proposed packing. In this stage, only combinational specific logics are matched and packed for a given user design. As a result, the input to the second stage is a netlist consisting of pre-packed combinational clusters, hard IP blocks, LUTs and FFs. In the second stage it packs selected FFs to pre-packed combinational clusters, in which the FF directly driven by the output of the cluster is selected. In other words, if the FF is driven by the output of other block such as a LUT or a FF, this FF is ignored. It is known as FF absorption stage. In the same way, it repeatedly packs the selected FFs into the cluster until no more FF can be selected for packing. After this stage completes, the netlist consists of pre-packed combinational clusters, pre-packed sequential clusters, hard IP blocks, LUTs and FFs. Final stage deals with LUTs and FFs in a delay-based manner, which is similar to MO-Pack [11] and YAMO-Pack [12], to pack them into clusters.

The pseudo code of proposed algorithm is outlined as follows.

```
Inputs: a netlist of the user design circuit after technology mapping C and a set of
pre-designed reference circuits S
Output: the packed circuit netlist Cp consisting of SLICEs and IP cores
    Convert C to graph Gc
    for all reference circuits r in R do {
      Convert r to graph Gr
      Using constraint satisfaction technique to match a pattern Gr from Gc
      If matched
           Map Gr to SLICE using the pre-designed corresponding instructions
           Update Gc
           update netlist Cp
      endif
    }
    for all SLICEs s in S do {
      for all outputs o in s {
        Trace the output o of s to the data input of FF dff
        If dff exists
             Pack dff to the SLICE s
           modify s configurations
           update netlist Cp
        endif
      }
    }
    unpackedBLEs = use simple pattern match to pack LUTs and FFs to BLEs
    for all BLEs ble in unpackedBLEs do
        calculate ble criticality as in T-VPack
    while (unpackedBLEs != NULL) {
      ble = selected the most critical BLE in unpackedBLEs
      create a new SLICE sn and pack ble to sn
      modify sn configurations
      update netlist Cp
      unpackedBLEs = unpackedBLEs – selectedBLE
      while (sn is not full) {
        selectedBLE = select max attraction BLE in unpackedBLEs
        pack selectedBLE to sn
        modify sn configurations
        update netlist Cp
        unpackedBLEs = unpackedBLEs – selectedBLE
      }
    }
    Output netlist Cp
```

IV. RESULTS

The proposed method is developed under Microsoft Visual Studio 2010 and implemented in C++. The results have been run on the PC with an INTEL CPU 2.4 GHz and 4 GB RAM.

To verify the effectiveness of the proposed method, design circuits in register transfer level Verilog format from the benchmark suite in Quartus II university interface program (QUIP) [17] are chosen. The selected designs are architecture independent and those circuits can be logically optimised by Xilinx commercial logic synthesis tool XST. Xilinx ISE MAP and the proposed method are then applied to the output of XST to pack logic into Xilinx Virtex-7 FPGA SLICE. The device xc7k160t-fbg676-3 is chosen for demonstration. TABLE II shows the comparison of the mapping results.

TABLE II
COMPARISON OF ISE MAP AND OURS IN TERMS OF DELAY

Benchmarks	ISE MAP (ns)	Ours (ns)	Improvement (%)
barrel16	2.243	2.164	3.52
barrel16a	2.829	2.706	4.35
barrel32	2.995	2.978	0.57
barrel64	3.495	3.420	2.15
fip_cordic_cla	4.762	4.555	4.35
fip_cordic_rca	4.017	3.841	4.38
fip_risc8	8.243	8.454	-2.56
mux32_16bit	2.268	2.205	2.78
mux64_16bit	3.114	3.038	2.44
mux8_128bit	2.595	2.685	-3.47
mux8_64bit	1.364	1.314	3.67
oc_aes_core	4.885	4.964	-1.62
oc_aes_core_inv	7.73	8.144	-5.36
oc_des_area_opt	2.389	2.262	5.32
oc_des_des3area	3.115	3.260	-4.65
oc_des_des3perf	3.392	3.305	2.56
oc_des_perf_opt	2.423	2.356	2.77
oc_minirisc	6.612	6.381	3.49
oc_miniuart	1.812	1.753	3.26
oc_mips	14.64	14.924	-1.94
oc_rtc	3.312	3.222	2.72
oc_ssram	1.609	1.500	6.77
oc_video_dec	3.223	3.207	0.50
oc_video_enc	2.089	2.164	-3.59
oc_video_jpeg	3.723	3.677	1.24
Average	3.96	3.94	0.51

TABLE III
COMPARISON OF PAM MAP AND OURS

Benchmarks	Area (No. of SLICE)			Delay (ns)		
	PAM MAP	Ours	Imp (%)	PAM MAP	Ours	Imp (%)
barrel16	24	20	-17	2.473	2.164	-12.5
barrel16a	49	41	-16	2.961	2.706	-8.6
barrel32	110	103	-6	3.384	2.978	-12.0
barrel64	140	133	-5	3.842	3.42	-11.0
fip_cordic_cla	120	118	-2	4.945	4.555	-7.9
fip_cordic_rca	74	68	-8	3.961	3.841	-3.0
fip_risc8	134	122	-9	8.454	8.454	0.0
mux32_16bit	136	135	-1	2.225	2.205	-0.9
mux64_16bit	268	259	-3	3.874	3.038	-21.6
mux8_128bit	279	270	-3	2.952	2.685	-9.0
mux8_64bit	140	136	-3	1.456	1.314	-9.8
oc_aes_core	170	158	-7	6.023	4.964	-17.6
oc_aes_core_inv	300	295	-2	8.465	8.144	-3.8
oc_des_area_opt	160	147	-8	2.845	2.262	-20.5
oc_des_des3area	297	285	-4	3.856	3.26	-15.5
oc_des_des3perf	300	280	-7	3.505	3.305	-5.7
oc_des_perf_opt	580	560	-3	2.969	2.356	-20.6
oc_minirisc	180	165	-8	7.023	6.381	-9.1
oc_miniuart	36	34	-6	1.965	1.753	-10.8
oc_mips	1100	1055	-4	16.4	14.92	-9.0
oc_rtc	130	118	-9	3.762	3.222	-14.4
oc_ssram	36	33	-8	1.865	1.5	-19.6
oc_video_dec	134	130	-3	3.723	3.207	-13.9
oc_video_enc	104	96	-8	2.476	2.164	-12.6
oc_video_jpeg	476	465	-2	3.937	3.677	-6.6
Average	219	209	-6	4.374	3.939	-11

It can be seen that the proposed method can achieve comparable results compared to Xilinx ISE MAP. It should be noted that since the proposed method is architecture independent it can be used for Altera FPGA architecture as well as long as the pre-designed reference circuits are modified accordingly to be suitable for Altera FPGA architecture.

Other published methods such as iRAC [9], MO-Pack [11], YAMO-Pack [12] etc are not comparable because they are targeting academic FPGA model. The method presented in [13] is not comparable either, because the test suite used is from industry and not available. Therefore, PAM MAP [14] is chosen for comparison, since PAM MAP is architecture independent and it can target Virtex-7 as well. The comparison results are shown in TABLE III. It can be seen that the proposed method can outperform PAM MAP in terms of area and delay in all tested cases, achieving, on average, 6% and 11% improvement, respectively.

V. Conclusions

The latest FPGAs contain composite logic blocks with LUTs, FFs, MUXs and other arithmetic circuitry. Packing design elements into the available logic resources is an extremely complex problem. In this paper, an architecture independent packing method for the commercial FPGA device is proposed. The proposed method has three stages. In the first stage, the constraint satisfaction problem technique of graph matching is utilised to implement specific logic such as 7-LUT, 8-LUT and carry chain arithmetic logic from the given user design circuit. Second stage packs the selected FFs to pre-packed combinational clusters. In the third stage, the delay-based method is carried out to deal with unclustered LUTs and FFs. The experimental results show that the proposed approach achieves similar performance in terms of speed compared with Xilinx commercial tool ISE MAP. The proposed algorithm also outperforms area-driven architecture independent PAM MAP, which can achieve on average, 6% and 11% in terms of area and speed, respectively.

Acknowledgment

This work was supported by a grant (No. 11MS011) from State Key Lab of ASIC and System, China and the National High Technology Research and Development (863) Thematic Program of China (No. 2012AA012001).

References

[1] I. Kuon, J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.26, pp. 203–215, 2007.

[2] V. Betz, J. Rose, A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publisher, 1999.

[3] V. Betz, J. Rose, VPR, "A new packing, placement and routing tool for FPGA research," *Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications*, pp. 213-222, 1997.

[4] A. Marquardt, V. Betz, J. Rose, "Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," *Proceedings of the ACM/SIGDA 7th International Symposium on Field Programmable Gate Arrays*, pp. 37-46, 1999.

[5] J. Rose, J. Luu, C. Yu, et al, "The VTR project: architecture and CAD for FPGAs from Verilog to routing," *Proceedings of the ACM/SIGDA 20th International Symposium on Field Programmable Gate Arrays*, pp. 77-86, 2012.

[6] M. Tom, D. Leong, G. Lemieux, "Un/DoPack: reclustering of large system-on-chip designs with interconnect variation for low-cost FPGAs," *Proceedings of the IEEE/ACM 2006 International Conference on Computer-Aided Design*, pp. 680-687, 2006.

[7] H. Liu and A. Akoglu, "Timing-driven nonuniform depopulation-based clustering," *International Journal of Reconfigurable Computing*, vol. 2010, pp. 1-11, 2010.

[8] L. Easwaran, and A. Akoglu, "Net-length-based routability-driven power-aware clustering," *ACM Transaction on Reconfigurable Technology and Systems*, vol. 4, pp. 38:1-16, 2011.

[9] A. Singh, G. Parthasarathy, and M. Marek-Sadowska, "Efficient circuit clustering for area and power reduction in FPGAs," *ACM Transaction on Design Automation Electronic Systems*, vol. 7, pp. 643-663, 2002.

[10] E. Bozorgzadeh, S. O. Memik, X. Yang, and M. Sarrafzadeh, Routability-driven packing: metrics and algorithms for cluster-based FPGAs, *Journal of Circuits, Systems and Computers*, vol. 13, pp. 77–100, 2004.

[11] S. Rajavel, and A. Akoglu, "MO-Pack: Many-objective clustering for FPGA CAD," *Proceedings of the 48th ACM/IEEE Design Automation Conference*, pp. 818-823, 2011.

[12] M. Yang, J.M. Lai and J.R. Tong, "Yet Another Many-Objective Clustering (YAMO-Pack) for FPGA CAD," *Proceeding of the 23rd International Conference on Field Programmable Logic and Applications*, pp. 1-4, 2013.

[13] T. Ahmed, P. D. Kundarewich, J. H. Anderson, at al, "Architecture-Specific Packing for Virtex-5 FPGAs," *Proceedings of the ACM/SIGDA 16th International Symposium on Field Programmable Gate Arrays*, pp. 5-13, 2008.

[14] Y. Shao, J.M. Lai, J. Wang and J.R. Tong, "PAM Map: an architecture-independent logic block mapping algorithm for sram-based FPGAs," *Proceedings of the 5th Southern Conference on Programmable Logic*, pp. 15-19, 2009.

[15] L.P. CordelIa, P. Foggia, C. Sansone and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1367-1372, 2004.

[16] B.N. Tran, T.D. Nguyen, "An Efficient Algorithm for Isomorphic Problem on Generic Simple Graphs," *Proceedings of the Second Asia International Conference on Modeling Simulation*, pp. 824-829, 2008.

[17] Quartus II University Interface Program. Available: http://www.altera.com.cn/education/univ/research/unvquip.html

Meng Yang received Bachelor of Engineering (Honor) degree in Electrical Engineering from Shanghai University, Shanghai, China, in 1999. He received Master of Science with distinction in Electronics and Communication Engineering and Ph.D. in Electronics from School of Engineering Edinburgh Napier University, Edinburgh, UK, in 2002 and 2006, respectively.

Currently he is a lecturer of State Key Lab of ASIC and System and Department of Microelectronics, School of Information Science and Technology, Fudan University, Shanghai, China. His research interests include algorithms in FPGA design automation, logic synthesis, and dynamic reconfigurable FPGA automation design. He has published more than 30 research papers.

Jinmei Lai received PhD degree in Shanghai Jiaotong University, Shanghai, China, in 1998. She was a Post-Doctor in Zhejiang University and Fudan University.

Currently she is a full professor of State Key Lab of ASIC and System, Fudan University, Shanghai, China. Her research interests include low power and reconfigurable architecture of FPGA and SOC, embedded IP core generation automation, logic synthesis and dynamic reconfigurable FPGA automation design, SOC testing automation. She has published more than 80 research papers and holds dozens of Chinese patents.

A.E.A. Almaini was born in Baghdad where he completed his school education. He received the B.Sc. (Eng), M.Sc., and Ph.D. degrees in electrical & electronic engineering from universities in England. He published a book, Electronic Logic Systems, and over 100 research papers.

Currently he is a Professor Emeritus at Edinburgh Napier University, Edinburgh, UK. His main research interests include the synthesis, optimization and automation in the field of digital electronics.