Effects of I/O Routing through Column Interfaces in Embedded FPGA Fabrics

Christophe Huriaux, Olivier Sentieys, Russell Tessier

Effects of I/O Routing through Column Interfaces in Embedded FPGA Fabrics

Christophe Huriaux
Inria
IRISA
Lannion, France
christophe.huriaux@irisa.fr

Olivier Sentieys
Inria
IRISA
Lannion, France
olivier.sentieys@inria.fr

Russell Tessier
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003
tessier@umass.edu

***Abstract*—The emergence of 2.5D and 3D packaging technologies enables the integration of FPGA dice into more complex systems. Both heterogeneous manycore designs, which include an FPGA layer, and interposer-based multi-FPGA systems support the inclusion of reconfigurable hardware in 3D-stacked integrated circuits. In these architectures, the communication between FPGA dice or between FPGA and fixed-function layers often takes place through dedicated communication interfaces spread over the FPGA logic fabric, as opposed to an I/O ring around the fabric. In this paper, we investigate the effect of organizing FPGA fabric I/O into coarse-grained interface blocks distributed throughout the FPGA fabric. Specifically, we consider the quality of results for the placement and routing phases of the FPGA physical design flow. We evaluate the routing of I/O signals of large applications through dedicated interface blocks at various granularities in the logic fabric, and study its implications on the critical path delay of routed designs. We show that the impact of such I/O routing is limited and can improve chip routability and circuit delay in many cases.**

***Index Terms*—FPGA; 3D ICs; 2.5D ICs; Dedicated I/O routing**

I. Introduction

FPGAs are quickly becoming important integrated components in a variety of systems. For example, the availability of FPGAs and one or more hard microprocessors is widespread and likely to increase in popularity in coming years. The integration of hard microprocessors, FPGA fabric, DRAM, and other large fixed-function components can drastically change the interfaces used to exchange data with the fabric. Effectively, data transfer between the fabric and these coarse-grained components can be considered at the level of data packets rather than collections of discrete I/O signals distributed around the perimeter of the FPGA array.

Several recent FPGA offerings strongly suggest this trend of I/O aggregation. A clear trend is the emergence of 2.5D architectures which include multiple FPGA dice in a single package to optimize both power consumption and yield. Xilinx high-density devices make use of Stacked Silicon Interconnect (SSI) to group multiple FPGA dice into a single package [1] to enhance the amount of available logic and reduce the latency penalty for signals crossing multiple dice. Xilinx Ultrascale architectures now include well-defined coarse-grained interfaces which allow for data communication across the interposer [2]. A similar trend can be seen in the FlexTiles architecture which explored a number of 3D-stacked heterogeneous architectures [3]. Such heterogeneous computing machines may include a multicore layer comprised of a collection of processors, caches, network-on-chip (NoC) routers, and an FPGA fabric stacked on a second layer. Communications between processors and the FPGA fabric, which host hardware acceleration blocks, take place via a network-on-chip extended to the third dimension. This type of interconnection requires an FPGA-processor interface which focuses more on data packets than on individual wires.

To support this time-multiplexed, packet-based approach, new architectural insights into FPGA fabric I/O can be considered. To assist interconnection to other die layers, I/O can be transferred through fixed hard interface blocks within the FPGA fabric. The blocks are distributed throughout the FPGA fabric in a manner that is similar to the distribution of memory and multiplier blocks. In our evaluation, all communication between an FPGA fabric and compute resources on other layers takes place via these interface blocks. Depending on the application space of the multi-layer system, the architecture of the interface block may be arbitrarily complex. For example, complexity may range from a series of synchronizing flip flops to network-on-chip interfaces. To explore the effects of these new blocks, we consider the portion of an interface which directly connects to the FPGA fabric to be two FIFOs, one for input data and one for output data. The use of FIFOs emphasizes that this architectural exploration is focused on cross-layer interconnect between functions with well-defined streaming interfaces implemented in FPGA logic or in fixed blocks, such as hard microprocessors.

In this work we examine several issues related to migrating FPGA I/O to interface blocks. The impact on required FPGA channel width, routability, and design delay is considered for a collection of large FPGA designs. The number of I/Os per block and the sparsity of block deployment in the device are also considered. These issues are addressed in the context of interface area relative to the size of the FPGA fabric itself. During experimentation, we consider two different types of interface block architectures. One *full* interface exposes just the data and control signals to access data in a FIFO. The second *I/O-only* interface implements the FIFO in FPGA block memory and sends the FIFO output to external resources

through the interface block.

The remainder of this paper is organized as follows. Section II discusses related work in FPGA I/O organization and its impact on FPGA architecture. Section III reviews our architectural assumptions for the interface blocks. The experimental approach we use for our work is described in Section IV. Section V provides experimental results and Section VI offers conclusions and directions for future work.

II. Background

Although FPGA fabrics have been integrated with a variety of different computing elements, analyses of I/O organization on FPGA routing architecture have been limited. Several papers from the late 1990s [4] [5] considered the use of *area I/Os* which provide external FPGA I/O from selected interior FPGA switch-boxes. The studies generally showed a small FPGA performance improvement (a few percent) for designs, primarily due to reduced wire length. This benefit came at the expense of increased packaging cost. For designs limited by available perimeter I/O, the extra area I/O pins provided for higher FPGA logic utilization. Both Altera Arria 10 [6] and Xilinx Ultrascale [7] architectures support I/O columns embedded within the FPGA array for general-purpose interconnect. Coarse-grained interfacing with blocks on other die layers is not explicitly addressed by these I/O architectures.

Several recent papers have examined the effects of limited 2.5D interconnect using interposers. It was found that limited I/O interconnect can make design placement more difficult, impacting design performance [8]. More recent 2.5D FPGAs [2] address this issue by including clusters of flip flops in interface blocks on the FPGA die close to the interposer boundary (Laguna tiles). Other work has studied the integration of hard I/O blocks within FPGAs in the form of Network-On-Chip (NoC) routers [9] and showed that hardening such interfaces is beneficial both in terms of area footprint and maximum achievable frequency.

3D integrated circuit technologies have been found to reduce the power consumption of power-hungry ultra-high bandwidth memory interfaces [10]. The applicability of such 3D technologies to FPGAs is currently limited to academic research on the subject [11], although previously-mentioned 2.5D circuits in the FPGA market [2] [12] achieve high logic integration and improve fabrication yield. These factors point to the promise of future 3D architectures with FPGAs. As mentioned in the previous section, the FlexTiles platform [3] implements hardware tasks in a reconfigurable logic fabric stacked over the multicore layer and accesses them via an NoC.

III. Overview of approach

The model for our approach considers an FPGA logic fabric on which hardware tasks are loaded. These tasks are logic modules of variable geometries which completely fit in a single logic fabric and interact through dedicated embedded interfaces. These interfaces realize connections between logic modules in the logic fabric and other types of circuitry (e.g. processors). This model fits streaming applications partitioned over multiple general-purpose or specialized processors and logic modules on different die layers which may operate at different clock speeds.

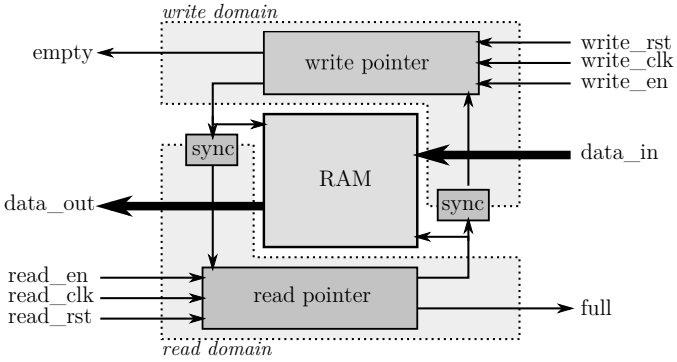


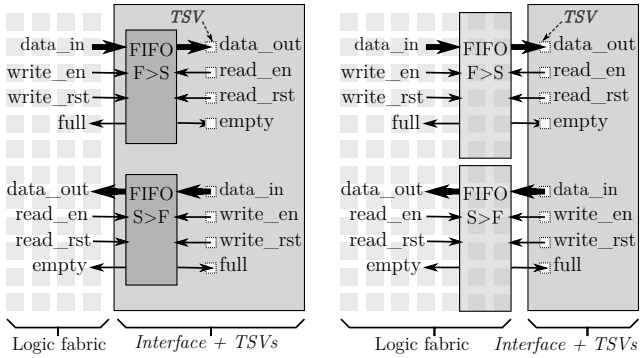
Fig. 1: FIFO interface model

A. I/O interface

In our considered design, all communications between the FPGA logic fabric and other parts of the system in which the FPGA layer is located are performed through one or more embedded interfaces, as opposed to I/O blocks spread around the logic fabric perimeter usually found in monolithic devices. The functional model of such an architecture thus relies on dedicated hard blocks realizing this interface within the logic fabric.

The side of the interface not facing the FPGA logic fabric could be connected in a variety of ways. For example, it could connect to a bus routed on an interposer (in the case of a 2.5D architecture) or to a complex NoC. Logic in the interface itself makes the translation between this external-facing side and the connections to the FPGA logic. As the range of possible applications for FPGAs is very wide, we focus our implementation on a generic first-in-first-out (FIFO) interface between the FPGA on one side and an external *system* on the other (i.e. anything connected to our interface: a NoC or bus). The model of our interface is a bi-directional asynchronous FIFO. The asynchronous nature of the interface is a strong requirement with regards to the target architectures. Indeed, single global clock distribution over multiple 3D-stacked dice can be challenging and error prone to design. Therefore, we assume that the interface should be the link between the system and FPGA clock domains, hence the use of asynchronous FIFOs.

Figure 1 depicts an overview of one half of the FIFO interface. The write domain, on the right, is separated from the read domain, on the left. Both read and write pointers to the FIFO memory are synchronized together to prevent any glitch or clocking problem between the two domains. Two control signals, `empty` and `full`, respectively, signal that the FIFO is empty to the read domain, and that it is full to the write domain. Read and write reset signals (`read_rst`, `write_rst`) to reset the FIFO pointers are also provided along with port enable signals (`read_en`, `write_en`). Each



(a) Full hard block (b) I/O-only hard block

F→S: FPGA→System FIFO, S→F: System→FPGA FIFO

Fig. 2: Two different implementations of the I/O interface

FIFO[1]	Signal	Width	Dir.[2]
-	clk	1	I
F→S	full	1	O
	write_en	1	I
	write_rst	1	I
	data_in	W	I
S→F	empty	1	O
	read_en	1	I
	read_rst	1	I
	data_out	W	O

(a) Full interface

FIFO[1]	Signal	Width	Dir.[2]
-	clk	1	I
F→S	data_out	W	I
	read_en	1	O
	read_rst	1	O
	empty	1	I
S→F	full	1	I
	write_en	1	O
	write_rst	1	O
	data_in	W	O

(b) I/O-only interface

1. F→S: FPGA→System, S→F: System→FPGA
2. I: Input, O: Output

TABLE I: Interface block signals routed on the fabric

interface block contains two of the FIFOs shown Figure 1, each FIFO can be accessed by the FPGA fabric. One is used for reading and the other for writing.

Each FIFO is parameterized by its data width W and its depth D. Multiple data widths were explored in the context of our experimentation: parameter W has a direct impact on the placement and routing performance of a design on the FPGA layer as the size and number of pins tied to the interface block in the logic fabric increase the number of signals routed to this block. We fixed the depth D of both FIFO interfaces to 16 elements: the depth depends on the maximum clock speed and throughput achievable both on the system side and on the FPGA layer. These parameters were not investigated in our generic implementation. Two hard interface block implementations of different complexities were evaluated in our study, as shown in Figure 2.

1) Full interface: In the first implementation, the *full* interface, shown in Figure 2a, all the interface logic is contained within the hard interface block: the memories, synchronizers, and read and write pointer logic. In this scheme, the only signals effectively routed through the logic fabric correspond to the data and control signals of the read domain of one of the interface FIFOs, and those of the write domain of the other FIFO, as depicted in Figure 2a. With this implementation, the area footprint of the interface block is high since the interface is contained within the interface block itself and only signals useful for bi-directional communication with the system are exposed. The utilization of logic resources in the FPGA logic fabric is kept to a minimum as the only additional interface logic needed is the control logic to read from and write to the respective FIFO.

2) I/O-only interface: The second considered implementation of the interface, the *I/O-only* interface, shown in Figure 2b, only contains the inputs and outputs routed vertically from another layer or interposer into the interface block. It is the lightest organization in terms of fixed-function silicon area, as its footprint only depends on the number of data and control signals and on the area constraints tied to the TSVs or micro-bumps realizing the 3-D links. Using the I/O-only interface implies that the logic and memory resources originally included in the full interfaces are now synthesized for implementation in FPGA fabric logic along with the considered design logic.

B. Implementation in VTR

To properly model the FPGA fabric architectures with interface-based interior I/Os, the Verilog-To-Routing (VTR) framework [13], commonly used for academic FPGA architecture explorations, was used. The VTR framework first performs FPGA Verilog HDL design elaboration and logic synthesis to an equivalent netlist in Berkeley Logic Interchange Format (BLIF). Design placement and routing onto a user-defined architecture described in an XML configuration file is subsequently performed.

The architecture file used in our experiments relies on the architectural model shipped with VTR, `k6_frac_N10_mem32K_40nm`. This FPGA model is based on the Altera Stratix IV heterogeneous architecture and features clusters of 10 fracturable 6-LUTs as its logic elements, 32Kb single- or dual-port memories with configurable aspect ratios from 512×64 to 32768×1, and fracturable 36×36 multipliers. This architecture file was modified to include embedded communication interfaces with varying bus widths and locations, as detailed in Section III-B3.

In the VTR framework, and specifically in the Versatile Place and Route (VPR) tool, the specification of an interface block can be easily supported without modifications to the tool source code. We added an additional hard block, the `io_interface`, to the Stratix IV-like architecture file which handles all communications between the FPGA layer and the rest of the system.

1) Interface architectures: As detailed earlier, the signals routed to the interface on the logic fabric greatly differ for the two considered implementations. Although the overall number of I/O signals is identical for both interface organizations, control signals which were routed from the hard interface on the FPGA to other layers of the system in the full interface must be generated or sinked by FPGA logic and routed back to the I/O-only interface, as seen on Figure 2.

2) Interface inputs and outputs: Full and I/O-only architectures differ in I/O interface signal directions. The signals routed on the logic fabric in the case of the full and I/O-only interfaces are respectively detailed in Tables Ia and Ib. In both cases, the total number of routed signals amounts to $6+2\times W$, as the clock clk is considered to be globally routed in the fabric.

We let VPR handle the location assignment of the I/O pins around the hard block in the logic fabric, using the *spread* parameter for the pin location specification in the XML architecture file. Using this parameter, VPR spreads the I/O pins on each accessible horizontal and vertical routing channel of the FPGA interconnect architecture around the hard block. The interconnection parameter f_c of the I/O pins to their adjacent routing channels has been set to $f_c = 0.15$ for the input pins (i.e. each input pin is connected to 15% of the adjacent routing wires), and $f_c = 0.10$ for the output pins. These values were chosen in accordance with the interconnection parameters used for the other hard blocks in this architecture.

3) Floorplanning: In order to study the effects of various interface complexities and sparsities across the logic fabric, each architecture is parameterized by two variables. The first parameter determines the data width W used for the data signals of the two interface FIFOs. In the case of the full interface, W bits are dedicated to the data received from the system and sent to the fabric, and W bits are reserved for the data sent from the fabric. The second parameter fixes the spacing R between two columns of interfaces on the logic fabric. The parameter R defines the sparsity of columns of interface blocks. Effectively, a column of interfaces appears in the FPGA once every R columns.

C. Interface Modeling Using Quartus

To further validate our interface approach, we also performed *full* interface experiments using Altera Quartus Prime. Hard interfaces were constructed by reserving the I/O pins on a vertical column of Stratix IV logic clusters. Qualitative comparisons between VPR and Quartus have previously been conducted [14] to evaluate the quality of results between the academic software for a Stratix IV architecture and the vendor equivalent in terms of logic utilization and tool run time.

1) Interface model: The first step in modeling our architecture using an existing Stratix IV architecture for Quartus is to define a placeholder for the interfaces. Stratix IV modeling capabilities in Quartus Prime are limited to features offered by the logic fabric. While VTR handles hard block instantiation using a hard block module prototype (i.e. the actual logic content of the hard block is not specified), the implementation of the interfaces in a Stratix IV device requires a model which uses the external pin interfaces of existing logic clusters. In order to realistically take into account the constraints of our I/O interfaces located within the logic fabric, the interfaces must be integrated so that interface I/O pins are spread over a surface equivalent to an actual physical interface implemented in non-programmable silicon.

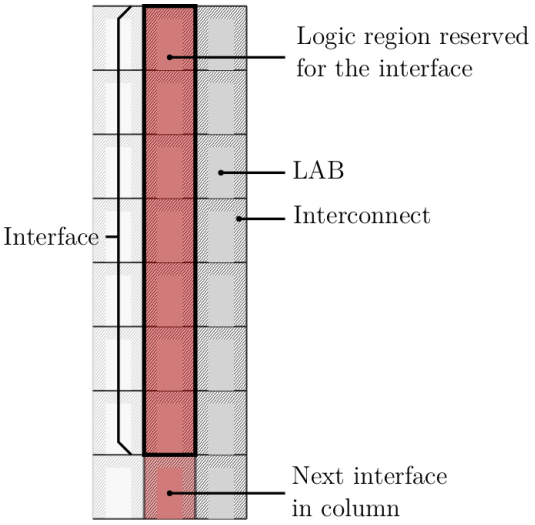


Fig. 3: Floorplan of the interface columns in Quartus

We determined the number of Stratix IV Logic Array Blocks (LABs) to include to model the area of a hard *full* interface block based on the synthesized interface area for an ASIC implementation of a *full* interface and from the area estimation of a Stratix IV LAB specified in VTR documentation. Each Stratix IV LAB contains 10 Adaptive Logic Modules (ALMs) [15], each of which can be configured to act as two full 4-input look-up tables (4-LUTs). For `io_interface` modeling, we inserted 20 4-LUTs per LAB in each of the LABs reserved to consume the area which would be required to implement an interface block. The LUTs serve no functional purpose in the experiments beyond serving as placeholders so that interface block I/O signals will be routed to their LAB I/O pins.

2) Optimization prevention: Since in our Quartus design implementations all design I/Os are assigned to dummy interface modules, effectively the design appears to Quartus to have no I/Os. As a result, care was required to avoid the removal of much of the design during synthesis. To prevent the optimization, the registers of the considered designs were annotated with the `noprune` and `preserve` synthesis attributes. The first annotation instructs the synthesizer to avoid pruning registers which do not directly or indirectly interact with an FPGA I/O. The latter attribute prevents a register from being removed when it is driven by a constant driver.

3) Interface placement: To simulate the placement of columns of interfaces within the FPGA logic fabric in Quartus, the placement of the dummy 4-LUTs modeling the I/O interfaces is restricted to specific portions of the target device. For our experimental purposes, we chose the largest available Stratix IV FPGA, the `EP4SGX530NF45C2` device, a 184 row $\times$128 column FPGA containing LABs, memories, arithmetic accelerators and transceivers. The interfaces are organized within regions spanning 1 LAB width and 7 LABs height, as shown in Figure 3.

We used the *LogicLock* feature of the Quartus placer and

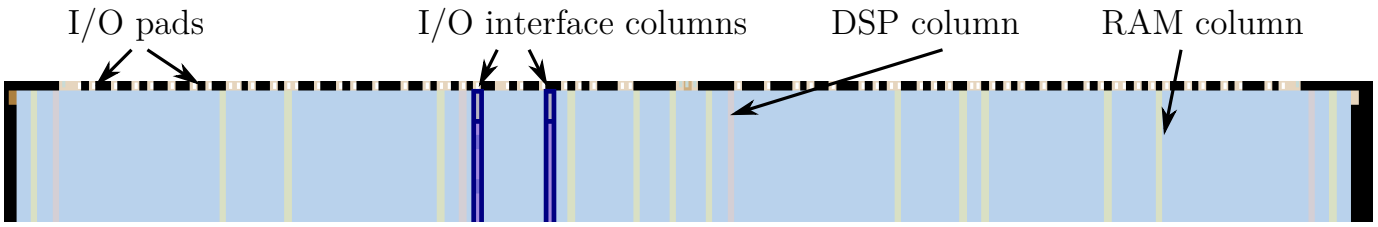


Fig. 4: Organization of the Stratix IV FPGA device as seen in the *Chip Planner* tool.

W	Logic area (40nm, μm^2)	TSV count	Total area (μm^2)	LABs
32	5,565	76	20,461	4
64	10,649	140	38,090	7
128	20,377	268	72,905	14

TABLE II: Full I/O interface area footprint and 3-D link count

router to manually delimit regions of logic dedicated to interfaces, organized in columns spaced horizontally by 25 LABs. In Figure 4, the interface columns are highlighted in dark blue, the green columns are the memories, while the red columns are the arithmetic accelerators (DSP blocks). The rest of the logic fabric, in light blue, is made of LABs. This model is similar to the modified architecture description we use for VPR to minimize differences and provide a basis for delay comparison.

IV. Experimental approach

In our initial experiments, we used the VPR framework with our modified architecture file to evaluate the effectiveness of using I/O interfaces spread across the logic fabric. These results were compared to those generated using an FPGA with standard perimeter I/O.

A. Interface area footprint

To collect accurate VTR results, the area footprint of the *full* interface was evaluated. We implemented the full `io_interface` module in Verilog and synthesized it using a 65nm technology library from STMicroelectronics for various FIFO data bus widths.

1) Comparison with the Stratix IV: The Stratix IV FPGA devices are fabricated with a 40nm technology half-node. The 65nm results obtained for the synthesis of the full interface module therefore have been scaled down by a factor $S = \left(\frac{65}{40}\right)^2$ to obtain an equivalent approximate area footprint in the 40nm half-node. The synthesis results are presented in Table II for FIFO data bus widths of 32, 64 and 128.

To properly estimate the height of the `io_interface` module with regards to the height of a single logic block (i.e. a LAB) of the architecture, we must first define the area of these logic blocks. We rely for our calculations on the LAB area approximation given in the Stratix IV-like architecture shipped with VTR, which reports this area as $53,000$ minimum width transistor area (MWTA). This area is defined as the minimum area occupied by a transistor and the space around it which is needed to pass the technology node design rules. In the 40nm technology half-node, 1 MWTA is $60L^2$, L being the node size (i.e. 0.04 μm). The area footprint of a LAB is thus equal to $53,000 \times 60 \times 0.04^2 = 5,088$ μm^2.

2) 3-D TSV area: To acknowledge the 3-D link nature of the interfaces, their area footprint must also account for the area and design rules of the through-silicon vias (TSVs), realizing the link between the FPGA die and the rest of the system. The TSV model we used to estimate the total area footprint of the I/O interface is a 6 μm TSV diameter associated with a keep-out zone of 4 μm. The minimum square enclosing the TSV is thus equal to $(6 + 2 \times 4)^2 = 196$ μm^2.

The number of TSVs associated with each interface is directly tied to the number of I/O signals between the interface and the system. It represents half of the total number of signals handled by the I/O interface, the other half being routed on the logic fabric.

Table II presents the number of TSV links that each interface width comprises, and sums the interface logic area and the TSV area footprint to report the total area of the interfaces. The *LABs* column includes the number of equivalent LABs for each interface total area. This number is used in the VTR and Quartus experiments to size the interfaces for the hard block height in the VPR architecture file, and for the number of dummy LUTs in Quartus, respectively.

B. Benchmark set

We performed the architecture evaluation for VPR using the benchmark set shipped with VTR [16]. Table III presents the benchmark details. This set of 19 benchmark circuits includes heterogeneous designs of various complexities, among which 8 make use of multiplier blocks, and 10 use hard memory blocks. This choice of benchmark circuits is particularly useful for our experiments as most of the designs have a high number of inputs and outputs, including 6 with over 400 I/Os.

For each benchmark circuit, we added a top wrapper module enclosing the original circuit which handles the instantiation of an adequate number of interface blocks. The number of interfaces is determined by taking the number of inputs or outputs, whichever is greater, and dividing it by the width W of the FIFO data buses. For the VTR experiments, the top-level module either instantiates the raw, full interface block and some logic to load and store values from the interface, or a combination of the I/O-only interface hard block and the associated FIFO logic and memories to be placed on the logic fabric.

Circuit	Inputs	Outputs	6-LUTs	Mult.	Mem.
bgm	257	32	30,089	11	0
blob_merge	36	100	6,016	0	0
boundtop	275	192	2,921	0	1
ch_intrinsics	99	130	413	0	1
diffeq1	162	96	434	5	0
diffeq2	66	96	277	5	0
LU8PEEng	114	102	21,954	8	9
LU32PEEng	114	102	75,530	32	9
mcml	36	33	99,700	30	10
mkDelayWorker32B	511	553	5,580	0	9
mkPktMerge	311	156	226	0	3
mkSMAdapter4B	195	205	1,977	0	3
or1200	385	394	2,963	1	2
raygentop	239	305	2,134	18	1
sha	38	36	2,212	0	0
stereovision0	157	197	11,462	0	0
stereovision1	133	145	10,366	0	0
stereovision2	149	182	29,849	0	0
stereovision3	10	30	174	0	0

TABLE III: VTR Benchmark set

V. Results

In this section we evaluate the performance of routing the circuit I/Os through interfaces spread across the logic fabric. Each wrapped benchmark circuit detailed in Section IV has been elaborated, placed and routed using the VTR framework on both full and I/O-only interface architectures, as explained in Section III. Experiments with Quartus Prime are used to evaluate full interfaces.

For VTR experiments with the two architectures, multiple data width W and column spacing R parameters were explored to quantify the effect of routing I/Os to interior FPGA interfaces rather than perimeter I/Os. In VTR experiments, the chosen interface area (i.e. its height in number of LABs in the architecture) is fixed to 7 LABs, corresponding to the $W = 64$ architecture (Table II). We kept the same interface size for both $W = 32$ and 128 architectures to evaluate the effect of sparser and denser I/O pin allocation around the I/O interface, as the interface area grows roughly linearly with FIFO data bus width.

A. Standard architecture

The comparison basis for all VTR experiments was generated using the 19 non-wrapped benchmark circuits and the `k6_frac_N10_mem32K_40nm.xml` base architecture (i.e. without interface columns). Timing driven place and route were used to optimize delay results. Table IV sums up the results of these experiments. The VPR placer and router tries to find the smallest possible square logic fabric which can host all of the logic and hard block resources used by the considered circuits. On this minimal logic fabric size, the timing-driven routing pass iteratively adapts the circuit interconnect to find the smallest possible channel width $minW$ able to route the design. The critical path delay results are thus obtained on both an area and routing constrained architecture.

Circuit	Min. chan. W (wires)	Crit. path delay (ns)
bgm	114	26.18
blob_merge	74	10.43
boundtop	58	6.65
ch_intrinsics	50	3.96
diffeq1	56	21.67
diffeq2	48	16.52
LU8PEEng	112	114.26
LU32PEEng	170	115.50
mcml	106	79.55
mkDElayWorker32B	82	7.24
mkPktMerge	50	4.37
mkSMAdapter4B	56	5.48
or1200	74	13.69
raygentop	72	5.28
sha	54	13.83
stereovision0	58	4.36
stereovision1	104	5.91
stereovision2	156	17.54
stereovision3	34	2.77

TABLE IV: Minimum channel width and delay results for the standard architecture

	R			
W	15	20	25	30
32	0.923	0.911	**0.908**	0.911
64	0.954	0.939	0.940	0.940
128	1.065	1.100	**1.104**	1.093

TABLE V: Average normalized channel width of the full interface architecture

B. Full interface architecture

With the full interface architecture, the circuit I/Os are routed to interfaces which includes all the necessary logic and memory. Each benchmark circuit was run on every combination of the parameters W and R, with $W \in \{32; 64; 128\}$ and $R \in \{15, 20, 25, 30\}$. Each benchmark run was evaluated in terms of the minimum channel width required to route the circuit, and on the delay of its critical path.

The placement and routing results of the minimum channel width and delay results using full interfaces are respectively detailed in Tables V and VI. For each couple $(W; R)$, the minimum channel width and critical path delay were normalized to those of a standard architecture shown in Table IV and averaged over the set of benchmarks to obtain the results presented in the two tables.

1) Quality of results: Overall, the quality of results of the full interface architecture is close to those of the standard architecture. The average channel width stays within $\sim 10\%$ of the standard I/O routing, while the critical path delay varies by at most 1.8% for the interface block case.

FIFO data width $W = 32$ and 64 both expose a reduction of the channel width, while the $W = 128$ interface is the only one to show average channel widths greater than the standard architecture. For $W = 128$, there are twice as many I/O pins to be routed to and from the full interface in comparison to the $W = 64$ interface. The greater number of wires routed to single interface blocks causes greater congestion

W	R 15	20	25	30
32	1.002	**1.008**	1.003	1.000
64	1.002	0.991	0.987	0.997
128	0.999	0.992	**0.982**	0.995

TABLE VI: Average normalized critical path delay of the full interface architecture

W	R 15	20	25	30
32	**0.979**	1.003	0.986	0.983
64	1.019	1.005	1.025	1.021
128	1.004	0.998	1.025	**1.034**

TABLE VII: Average normalized channel width of the I/O-only interface architecture

W	R 15	20	25	30
32	1.019	1.011	0.995	**0.994**
64	1.010	1.013	0.998	1.012
128	1.014	**1.024**	1.010	1.010

TABLE VIII: Average normalized critical path delay of the I/O-only interface architecture

of interconnect resources around the interfaces. In this case, VPR cannot successfully route the design on a smaller channel width, and this width increases.

2) Routing stress: It should be noted that the capacity of the I/O pads in the base VTR Stratix IV architecture file is set to 8, meaning that each I/O pad can host at most 8 I/O signals, and each I/O pad is tied to a single routing channel. In the case of our architectures with internal I/O interfaces, the input and output signals are spread over all the routing channels surrounding the interface block. Since the interface blocks are organized in columns, and since the interface block height has been set to 7 logic blocks (LABs) for $W = 64$ given its area footprint, each interface is surrounded by 2×7 vertical routing channels, and 2 more horizontal channels on the top and bottom sides of the hard block. The average number of I/O pins tied to each routing is thus, in the case of the $W = 64$ interface, $\frac{2\times64+6}{2\times7+2} = 8.375$, which is almost equal to the I/O pad capacity of the standard architecture. The routing stress to the I/O interface is thus similar between the $W = 64$ interface architecture and the standard I/O ring architecture.

3) Effects of small circuits: For the $W = 32$ and 64 interface architectures, the channel width reduction is mainly due to smaller benchmarks which have a very high $\frac{\text{I/O count}}{\text{logic resources}}$ ratio. For example, the second smallest circuit of the VTR benchmark suite, $mkPktMerge$, has 467 inputs and outputs for only 226 6-LUTs. Using I/O interfaces in columns within the logic fabric allows a spread in I/O wiring across the logic fabric rather than causing congestion in the outermost routing channels when the circuit is I/O limited. Removing the smaller circuits from the calculation of the average channel width and critical path delay reduces the gap between the standard architecture and the full interface architecture. For example, by putting aside the circuits `mkPktMerge`, `stereovision3`, `diffeq2`, `diffeq1` and `ch_intrinsics`, the average normalized minimum channel width over the 14 remaining benchmarks is halved and brought down to within $\sim 5\%$ of the standard architecture.

C. I/O-only interface architecture

The area for the I/O-only interfaces only includes the required area for the 3-D TSV links interfaced to the FPGA layer. The FIFO logic and memory are placed and routed in the FPGA logic fabric along with the original design logic. The 19 benchmark circuits of the VTR benchmark suite have been placed and routed using the I/O-only interface architecture with the same W and R parameters couples used for the full interface, as detailed in Section V-B.

Similar to the full interface experiments, each benchmark circuit is wrapped with a custom top-module. For the I/O-only experiments, the top-module not only instantiates the I/O-only interface blocks, but also the FIFO logic and memory.

1) Quality of results: Overall, the quality of results of the circuits placed and routed on architectures using the I/O-only interface is closer to the standard architecture than the full interface. The normalized channel width stays within $\sim 3\%$ of the standard architecture channel width on average, while the critical path delay has a variation of at most 2.4% in comparison to the standard I/O ring architecture.

With regards to the full interface architecture results, the I/O-only interface puts more stress on the routing architecture, as the average normalized channel width results indicate. With the increase of resources to be placed on the logic fabric (i.e. the FIFO logic and its associated memories), the place and route step has to deal with an increase of wire-length which constrains the circuit and increases the minimum achievable channel width able to successfully route the benchmark designs.

2) Additional FIFO resources: In addition to limited minimum channel width and critical path delay increases for $W = 64$ or 128, the I/O-only interface requires the placement of interface FIFO resources in the logic fabric. Table IX details the average number of additional resources required by the wrapped benchmark circuits with I/O-only interfaces. The additional resource numbers were averaged over the considered circuits. The two considered resources are logic blocks (the LABs, containing $\sim$ 20 4-LUTs) and memory blocks.

The number of additional memory and logic blocks required

W	Memories	LABs
32	11.87	33.33
64	12.80	25.67
128	15.47	26.07

TABLE IX: Average amount of additional resources required by the I/O-only architectures

Circuit	Standard arch. F_{max} (MHz)	Full interface arch. F_{max} (MHz)
bgm	81.17	76.48
blob_merge	103.75	108.71
mcml	35.73	35.78
stereovision1	136.93	130.36
stereovision2	113.95	125.08

TABLE X: Performance comparison of the standard and full interface architectures using Quartus

Circuit	Standard arch. ALUTs	Full interface arch. ALUTs
bgm	14,719	15,415
blob_merge	5,430	5,848
mcml	49,288	52,838
stereovision1	1,351	1,680
stereovision2	23,984	24,132

TABLE XI: LUTs required by standard and full interface architectures using Quartus. The dummy LUTs for the I/O interfaces are included in the full interface results

by each circuit varies as a function of the FIFO data bus width W and of the number of inputs and outputs of the circuits, as more I/Os implies more instantiated interfaces. With bigger W parameters, the number of interfaces goes down as more signals can be routed into a single I/O-only interface, which reduces the amount of logic needed to handle the FIFOs, as shown in Table IX.

D. Quartus Prime F_{max} for a Stratix IV device using a Full Interface

To better evaluate the performance of the circuits reported by VPR, a subset of the largest VTR benchmarks has been wrapped, synthesized, placed, and routed using Quartus Prime software. The comparison is made on an architecture supporting full interfaces with $W = 64$ FIFO data bits. The methodology described in Section III-C is used in timing driven mode with an unreachable target 1 GHz clock frequency. The dummy interface height is fixed at 7 LABs and the number of low-level 4-LUT primitives in the full interface is defined as 140 (i.e. 2 full 4-LUTs/ALM, 10 ALMs/LAB) to spread the interface across an area equivalent to a required VLSI footprint in 40nm technology. The experiments were run with columns of interfaces separated by $R = 25$ logic block columns. In contrast with the VPR experiments, in which the benchmarks designs were placed and routed on the smallest possible logic fabric area, Quartus physical design was performed for the full Stratix IV device architecture.

The comparison is detailed in Table X. Overall, in comparison to the standard architecture (i.e. without interfaces), the maximum achievable clock frequency F_{max} reported by Quartus Prime varies compared to results from the architecture using full interfaces. Most of the evaluated circuits show a variation of $\pm 10\%$ of F_{max} when their I/Os are routed to the dummy full interfaces.

Table XI sums up the Adaptive LUTs (ALUTs) used in both architecture implementations in Quartus. All designs show an increase of the number of ALUTs used in the Stratix IV FPGA for the full interface architecture. This increase is caused by two factors. First, each instantiated full interface requires ~ 140 dummy 4-LUTs to occupy an area equivalent to its VLSI footprint and provide interface I/Os. Second, the FIFO mechanism used by the interfaces requires some additional logic to feed the circuit with the input FIFO data and to extract the circuit output value from the output FIFO.

VI. Conclusions and future work

In this paper we have investigated the effects of new interfaces for FPGA logic fabrics embedded in 3D or 2.5D packaging. For these types of multilayer systems, a traditional I/O ring around an FPGA logic core is of limited value because of wire bonding constraints. As a result. all inputs and outputs are routed to I/O interfaces spread over the logic fabric in columns like other FPGA hard blocks.

We analyzed the effect of two organizations of such interfaces on the delay and routability of circuits from the VTR benchmark suite. The first organization, the *full* interface, contains all the necessary logic and memory to asynchronously perform transfers between the FPGA layer to the rest of the system. For the second organization, the *I/O-only* interface, we considered a split design where the FIFO control logic and memory are placed and routed in the FPGA logic fabric along with the target design.

Overall, both full and I/O-only organizations proved to be implementable with little impact on overall architecture routability. With interface data widths of $W = 32$ and 64, the minimum channel width required to route the circuits can be decreased by as much as 10% on average using the full interface. The impact on the critical path delay is within 2% for our benchmarks for both interface organizations.

The problem of routing FPGA I/Os in the context of 3D integrated circuits should become of increasing interest as 3D and 2.5D integration technologies evolve. Heterogeneous multiprocessor systems may benefit from the flexibility brought by a reconfigurable layer in a 3D stacked system. Several open problems exist related to the integration of logic fabrics in 3D circuits. Notably, the integration of 3D TSV or microbump links must follow specific design rules regarding their placement and distribution over a silicon die. This issue, in turn, puts constraints on the distribution of I/O interfaces in the logic fabric.

References

[1] K. Saban, "Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency," Xilinx Corporation, Tech. Rep. WP180, 2012.

[2] *UltraScale Architecture Configurable Logic Block - User's Guide*, Xilinx Corporation, 2015.

[3] F. Lemonnier, P. Millet, G. M. Almeida, M. Hubner, J. Becker, S. Pillement, O. Sentieys, M. Koedam, S. Sinha, K. Goossens, C. Piguet, M. N. Morgan, and R. Lemaire, "Towards future adaptive multiprocessor systems-on-chip: An innovative approach for flexible architectures," in *International Conference on Embedded Computers*, 2012, pp. 228–235.

[4] H. V. Marck, J. Depreitere, D. Stroobandt, and J. V. Campenhout, "A quantitative study of the benefits of area-I/O in FPGAs," in *Proceedings of the Great Lakes Symposium on VLSI*, Feb. 1998, pp. 392–399.

[5] V. Maheshwari, J. Darnauer, J. Ramirez, and W. W.-M. Dai, "A quantitative study of the benefits of area-I/O in FPGAs," in *ACM/SIGDA International Symposium on FPGAs*, Feb. 1998, pp. 17–23.

[6] *Arria 10 Transceiver PHY User Guide*, Altera Corporation, 2016.

[7] *UltraScale Architecture and Product Overview*, Xilinx Corporation, 2016.

[8] A. H. Pereira and V. Betz, "CAD and routing architecture for interposer-based multi-FPGA systems," in *Proceedings of the ACM Symposium on Field-Programmable Gate Arrays*, Feb. 2014, pp. 75–84.

[9] M. S. Abdelfattah and V. Betz, "Design tradeoffs for hard and soft FPGA-based Networks-on-Chip," in *International Conference on Field-Programmable Technology*, 2012, pp. 95–103.

[10] D. Dutoit, E. Guthmuller, and I. Miro-Panades, "3D integration for power-efficient computing," in *Proceedings of the Design, Automation and Test in Europe Conference*, 2013, pp. 779–784.

[11] M. Lin, A. El Gamal, Y.-C. Lu, and S. Wong, "Performance benefits of monolithically stacked 3-D FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 216–229, 2007.

[12] R. Venkata, M. Won, and M. Deo, "Achieving the Highest Levels of Integration in Programmable Logic," Altera Corporation, Tech. Rep. WP-01258-1.0, 02 2016.

[13] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose, and V. Betz, "VTR 7.0: Next Generation Architecture and CAD System for FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 2, pp. 6:1–6:30, June 2014.

[14] K. E. Murray, S. Whitty, S. Liu, J. Luu, and V. Betz, "Titan: Enabling large and complex benchmarks in academic CAD," in *International Conference on Field-Programmable Logic and Applications*, 2013, pp. 1–8.

[15] *Stratix IV Device Handbook Volume 1*, Altera Corporation, 2011.

[16] J. S. Rose, J. Luu, C. W. Yu, O. Densmore, J. Goeders, A. Somerville, K. B. Kent, P. Jamieson, and J. Anderson, "The VTR project: architecture and CAD for FPGAs from Verilog To Routing," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Feb. 2012, pp. 77–86.