MAPPING LOGIC TO RECONFIGURABLE FPGA ROUTING

Karel Heyse[*] , *Karel Bruneel, and Dirk Stroobandt*

Department of Electronics and Information Systems
Ghent University
Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium
{Karel.Heyse,Karel.Bruneel,Dirk.Stroobandt}@UGent.be

ABSTRACT

Parameterised configurations for FPGAs are configuration bitstreams of which part of the bits are defined as Boolean functions of parameters. By evaluating these Boolean functions using different parameter values, it is possible to quickly and efficiently derive specialised configuration bitstreams with different properties. An important application of parameterised configurations is the generation of specialised configuration bitstreams for Dynamic Circuit Specialisation.

Generating and using parameterised configurations requires a new FPGA tool flow. In this paper we present an algorithm for technology mapping of parameterised designs that can exploit the reconfigurability of the logic blocks and routing of the FPGA. This algorithm, called TCONMAP, is based on "Cut enumeration, cut ranking, node selection". As part of it, a new method to calculate the feasibility of cuts based on the Binary Decision Diagrams (BDD) of their local function is proposed.

1. INTRODUCTION

Parameterised configurations [1] for FPGAs are configuration bitstreams of which part of the bits are defined as Boolean functions of parameters. By evaluating these Boolean functions using different parameter values, it is possible to quickly and efficiently derive specialised configuration bitstreams with different properties and/or functionality from a single parameterised configuration.

The advantage of generating specialised configurations from a parameterised configuration, instead of directly running the conventional FPGA tool flow, is the much lower time cost per configuration. Specialising a parameterised configuration requires only the evaluation of some Boolean functions, instead of solving the computationally hard problems contained in the full conventional tool flow, such as placement and routing.

[*]Karel Heyse is supported by a Ph.D. grant of the Flemish Fund for Scientific Research (FWO – Vlaanderen).

Parameterised configurations have many potential applications. Some examples are the "hard coding" of device specific information in a production process: e.g. a MAC address, an encryption key or calibration data. Or speeding up development by allowing the developer to tune coefficients, e.g. the address range of a bus peripheral or the coefficients of a DSP filter, without having to rerun the complete tool flow.

Another important application of parameterised configurations is the efficient generation of configuration bitstreams for Dynamic Circuit Specialisation (DCS). With DCS, Run-Time Reconfiguration of (parts of) an FPGA is used to dynamically optimise the circuit for the current situation in order to improve its performance, increase the functional density of the FPGA and consequently to reduce the cost of the design [2].

Using the conventional tool flow to create the specialised configurations for DCS at run-time would in many cases take too much time (seconds to minutes), while computing all specialised configurations in advance and storing them in a database quickly becomes infeasible because of the immense number of possible configurations. E.g. 2^{32} or ca. 4 billion configurations for a 32-bit multiplier with a semi-constant coefficient. With a parameterised configuration on the other hand, only one configuration needs to be stored while the different specialised configurations can be generated in milliseconds [1].

To use parameterised configurations and generate them from HDL a new FPGA tool flow is needed. This new tool flow contains, similar to the conventional tool flow, a synthesis, technology mapping, placement and routing step. An overview of this tool flow is given in Section 2.

In this paper we focus on the technology mapping step. At this time, a mapping algorithm, called TLUTMAP [1], exists which can be used to create a parameterised configuration in which the truth tables of the LookUp Tables (LUT) can be expressed as Boolean functions of parameters. The TLUTMAP method is, however, limited because it can't be used to parameterise the interconnect network of the FPGA. A new algorithm for technology mapping, called TCON-

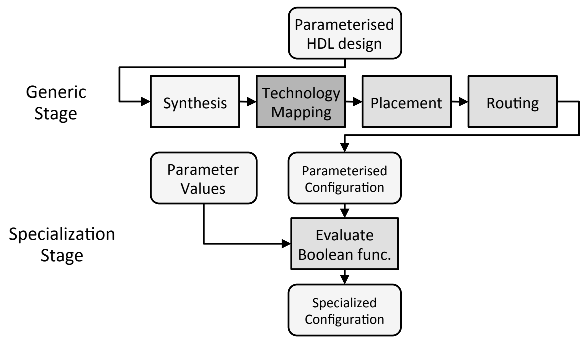

Fig. 1. Tool flow for parameterised configurations

MAP, is presented in Section 4 that is aware of the parameterisability of the LUTs and the interconnect network of the FPGA.

TCONMAP is based on the widely used "Cut enumeration, cut ranking, node selection" algorithm [3, 4], but uses a new method to check the feasibility of cuts based on the Binary Decision Diagrams (BDD) of their local function. It is shown that the compressed form of the BDDs can be exploited to efficiently compute the feasibility of cuts for mapping parameterised designs.

To our knowledge, TCONMAP is the first mapper that can exploit the reconfigurable properties of the interconnect network of FPGAs.

2. TOOL FLOW

To generate and use parameterised configurations the conventional FPGA tool flow has to be upgraded. The new tool flow, depicted in Figure 1, consists of two stages. The generic stage is performed off-line at design time to create a parameterised configuration from an HDL description in which a number of inputs are denoted as parameters. The specialisation stage is performed every time a specialised configuration is needed to (re-)configure an FPGA.

During the specialisation stage, the Boolean functions contained in the parameterised configuration are evaluated to create a fully defined configuration bitstream. This can be done efficiently using a microprocessor.

The generic stage of our new flow consists of similar, but adapted, parts as the conventional tool flow. We give a description of the changes made in each part.

2.1. Generic Stage

2.1.1. Synthesis

The synthesis step converts a HDL description in which some inputs are annotated as parameters into a Boolean network of logic gates (AND, NOT, flip-flop, ...). No significant changes need to be made to this step because parameters can be synthesised just like regular inputs. Though parameter inputs in the HDL description do have to be annotated as parameter inputs in the resulting Boolean network as well.

2.1.2. Technology Mapping

During technology mapping, the Boolean network generated by the synthesis step is mapped onto the resource primitives available in the target FPGA architecture while trying to optimise the delay and area of the circuit.

A conventional mapper will map to static nets and LookUp Tables (LUT), thus resulting in a static configuration bitstream after placement and routing. To generate a parameterised configuration, however, we want to map to primitives that can be parameterised:

- Tuneable LUT (TLUT): A TLUT is a LookUp Table of which the entries are defined as Boolean functions of parameters instead of ones and zeros.
- Tuneable Connection (TCON): A TCON is a point-to-point connection which can be made or broken depending on the value of a Boolean function of parameters. It is implemented using the FPGA's routing network and reconfigurable switch blocks.

Previously, a mapping algorithm, called TLUTMAP [1], has been presented that can map a Boolean network with parameter inputs onto TLUTs. Several designs (FIR filter, TCAM [1], AES coder [5], ...) showed reductions in the number of (T)LUTs of up to 66% and clock speed improvements of up to 38% compared to generic designs when using a parameterised tool flow including the TLUTMAP mapper.

The TLUTMAP method, however, is limited because it can't be used to create a configuration where the interconnect network of the FPGA is also parameterised. In this paper we present TCONMAP, a new algorithm for technology mapping onto both TLUTs and TCONs, which can thus be used to create fully parameterised configurations.

2.1.3. Placement & Routing

During the placement step a physical LUT on the FPGA is chosen to implement every instance of the LUT and TLUT primitives, while during the subsequent routing step routing resources are chosen to implement the nets and TCONs.

The placement step makes extensive use of wirelength estimates of the connections between LUTs to optimise the routability and interconnect delay of the design. Since the TCON is very different from a static connection, new ways to estimate the length of connections had to be developed. A method to solve this new problem is being worked on and will be presented in the future.

In the routing step, special care has to be taken to find the best routing resources to implement the TCONs. Recent research on this subject has been presented in [6].

3. TECHNOLOGY MAPPING BACKGROUND

3.1. Definitions

3.1.1. Boolean network

A Boolean network is a directed acyclic graph of which the nodes represent logic gates and the directed edges the wires that connect them. In this paper we use And-Inverter Graphs (AIG) to represent combinational circuits. AIGs are Boolean networks containing only 2-input AND gates and inverted or non-inverted edges. AIGs can be used to represent any combinational circuit. Sequential circuits are transformed into combinational circuits by turning the registers into additional inputs and outputs of the circuit.

3.1.2. Primary inputs and outputs

Primary inputs (PI) and primary outputs (PO) are the inputs and outputs of the combinational circuit. A primary input can either be a parameter input or a regular input.

3.1.3. Cut

A cut of a node, n, is a set of nodes, called leaves, so that every path from the PIs to the node n passes trough at least one of the leaves. The node n is called the root of the cut. The trivial cut is the set, $\{n\}$, containing only the root itself.

The local function of a cut, $c = \{n_1, n_2, \dots\}$ with root n, is the value of the root of the cut in function of the leaves of the cut: $\psi_{n,c}(n_1, n_2, \dots)$.

3.2. Conventional Technology Mapping

TCONMAP has been built based on the widely used conventional mapping algorithm [3, 4]: "Cut enumeration, cut ranking, node selection". Many variations of the algorithm exist but the basic, main idea of the algorithm can be described as follows:

3.2.1. Cut enumeration

In the first step all K-feasible cuts, $\Phi(n)$, are computed for every node, n, in the AIG using Equations (1), (2) and (3). A cut is K-feasible if its local function can be implemented using a K input LUT.

$$\Phi(n) = \begin{cases} \{\{n\}\}, & \text{if } n \in PI \\ \{\{n\}\} \cup M(n), & \text{otherwise} \end{cases} \tag{1}$$

$$M(n) = \{c = c_1 \cup c_2 \mid c_i \in \Phi(n_{in_i}), \theta(n, c)\} \tag{2}$$

$$\theta(n, c) = |c| \leq K \tag{3}$$

The function $M(n)$ generates all possibly feasible cuts of n by combining a feasible cut of both its inputs n_{in_1} and n_{in_2}. The function $\theta(n, c)$ evaluates if a cut can be mapped to the K-LUT primitive. Cut enumeration is done for every node in topological order, from the PIs to the POs.

3.2.2. Cut ranking

For each node in topological order, the cut with the lowest depth is selected as the best cut. This ensures an end result with minimal number of LUTs on the longest path from any output to the inputs. The depth of a mapped circuit is the maximum depth of all outputs.

The delay for a cut is conventionally constant 1.

$$\begin{aligned} depth(n, c) = {} & delay(n, c) \\ & + \max_{m \in c} depth(bestCut(m)) \end{aligned} \tag{4}$$

$$bestCut(n) = \begin{cases} \{n\}, & \text{if } n \in PI \\ \arg\min\limits_{c \in \Phi(n)} depth(n, c), & \text{otherwise} \end{cases} \tag{5}$$

3.2.3. Node selection

This step starts by selecting the POs and then recursively selecting all nodes in the best cuts of the previously selected nodes until the PIs are reached (Equation 6). The local function of the best cut of each selected node will be implemented using a LUT. After this step a depth-optimal mapping has been defined.

$$S = PO \cup \bigcup_{n \in S} bestCut(n) \tag{6}$$

4. TCONMAP

In this section we describe the algorithm TCONMAP for technology mapping onto TLUTs and TCONs.

4.1. Mapping of AIGs with Parameters

When mapping an AIG with parameters, parts of the previously described algorithm "Cut enumeration, cut ranking, node selection" can be reused, but a number of things do change. Most importantly, the primitives mapped to are different, so a new definition of feasibility and evaluation function $\theta(n, c)$ are needed.

4.1.1. TLUT

The parameter inputs in a cut don't have to be implemented in hardware, this means that we don't have to reserve an input on a K-LUT for them. Instead, the parameter inputs will become the variables of the Boolean functions that define the truth table of the Tuneable LUT.

Consequently, every cut with at most K regular nodes, and any number of parameter inputs, can be mapped to a TLUT and is TLUT-feasible.

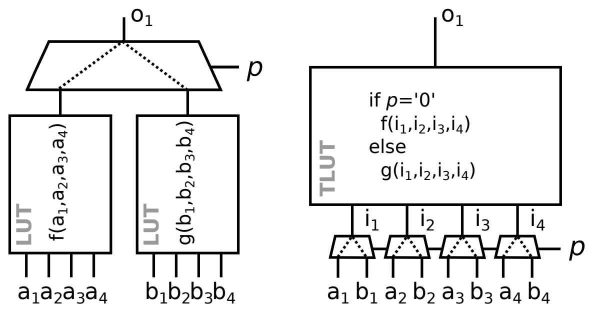

Fig. 2. The same circuit mapped to 1 TCON and 2 LUTs (left) vs. 1 TLC (right) (parameter p)

4.1.2. TCON

In a conventional mapping algorithm, nets are not actively mapped to because their function is passive: connecting the LUTs that the mapping algorithm finds. When using TCONs, connections become active elements which can change the output of a circuit in function of the parameters. To exploit this capability we will actively map to TCONs as well.

Because TCONs should not cause shorts for any value of the parameters, a group of TCONs with the same sink can be thought of as a multiplexer with a Boolean function of parameters at the select inputs. A cut whose local function is such a multiplexer can thus be mapped to TCONs. Those cuts have to be recognised as TCON-feasible.

The implementation of a cut using TCONs uses only routing resources and no LUTs, so a TCON-feasible cut does not contribute to the depth and area of the circuit.

4.1.3. TLC supergate

The TLC is a virtual supergate composed of a TLUT with TCONs, or multiplexers controlled by parameters, attached to its inputs. Supergates are used to reduce the influence of the structure of the Boolean network on the quality of the resulting mapping [7]. An illustration of this is given in Figure 2 which contains an example of a circuit that will be mapped more efficiently using a TLC than using TLUTs and TCONs separately.

A cut is TLC-feasible if for every combination of parameter values the local function of the cut is defined by at most K inputs (leaves), while the other inputs are don't cares. This ensures that in every specialised configuration the physical LUT implementing the TLC will need at most K input signals.

4.2. Feasibility Calculation Using BDD of Local Function

A method to compute the feasibility of a cut according to the previously defined criteria is needed for the cut enumeration step of the mapping algorithm. We propose an efficient way to evaluate the feasibility of a cut using the Binary Decision Diagram (BDD) of its local function. A BDD is a Boolean network representation of a Boolean function using only 2-to-1 multiplexers with the variables of the Boolean function as select inputs, and the constant nodes, '0' and '1'.

To calculate the feasibility of a cut, the variables in the BDD of the local function are reordered so that the multiplexers controlled by parameter inputs are closest to the root. The BDD is then cut in two so that all multiplexers controlled by parameter inputs lie at one side of the boundary while all other multiplexers lie at the other side. It is easy to see that every connection that crosses this boundary is the root of the (sub-)BDD of a partially evaluated or specialised local function for some values of the parameters.

Based on the criteria in Section 4.1 the cuts can then be classified as follows:

- If every sub-BDD contains at most 1 variable, the cut is recognised as a multiplexer and is TCON-feasible.
- If the full local function depends on at most K different non-parameter variables in total, the cut is TLUT-feasible.
- If every sub-BDD contains at most K different variables, then the cut is TLC-feasible.
- If any of the sub-BDDs contains more than K different variables, the cut is infeasible.

4.2.1. Example

Figure 3 contains the BDD of the local function, ψ, of a cut. The roots of the specialised local functions' sub-BDDs are marked with $\psi_{p=-,q=-}$. The specialised local functions $\psi_{p=0,q=0}$, $\psi_{p=0,q=1}$ and $\psi_{p=1,q=0}$ depend on variables a and b. Function $\psi_{p=1,q=1}$ is defined by the variables a and c. The cut can thus be implemented using a Tuneable 2-LUT with a at its first input and two TCONs connecting either b or c to its second input depending on the values of p and q.

Because a BDD is a compressed representation of the local function, the number of sub-BDDs that has to be checked is usually far less than the number of possible configurations (2^m with m the number of parameters of the local function). Moreover, classifying a cut can even be done in approximately linear time in the size of the BDD of the cut by further exploiting the compressed form of the sub-BDDs and the occurrence of subgraphs shared between sub-BDDs. This makes the Binary Decision Diagrams very efficient for feasibility calculation with the newly defined criteria.

4.3. Computational Complexity

The new definition of feasibility allows for cuts with more than K nodes, depending on the local function of the cuts. It is known that enumerating all cuts with at most x nodes

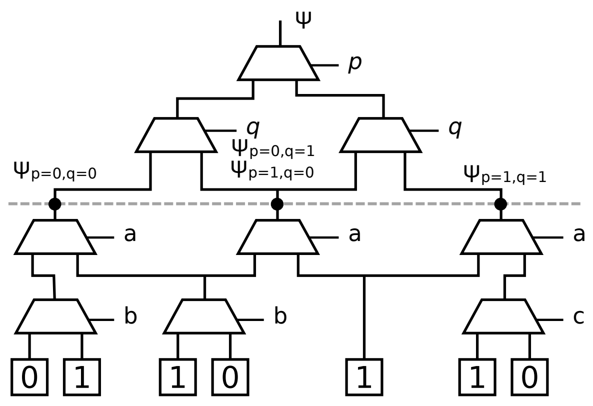

Fig. 3. BDD for (specialised) local function of a cut (parameters p, q)

is slow for large x, because of the large amount of possible cuts.

Experimentally, we found that many designs can be mapped in reasonable time using the proposed algorithm. For the parts of the design that do not depend on parameters the algorithm reduces to the conventional algorithm (cuts of at most K nodes) which is fast. For the parts that do depend on parameters, we found that the number of cuts per node is in most cases less than 2000. If the number of feasible cuts for a node reaches this number, it was chosen to halt the enumeration process for that node.

The number of cuts that has to be enumerated can be significantly reduced by excluding redundant and quasi redundant cuts. For instance, any cut with a leaf node that depends only on parameter inputs will not contribute to the quality of the mapping [1]. A node that has a TCON-feasible cut should also not be a leaf of another cut because the TCON-feasible cut can easily be merged with any cut of which it is an input.

The worst-case size of a BDD, and thus the worst-case execution time of the operations on the BDD, is exponential in the number of variables. State-of-the-art BDD packages do however perform optimisations, such as variable reordering, so that in most cases a better than worst case solution is found. The number of parameters per local function and activation function is also in most cases less than a few tens, so they can be represented efficiently using BDDs. Large cuts (e.g. containing more than 20 regular nodes) can be ignored as a heuristic to speed up the algorithm.

5. EXPERIMENTAL RESULTS

In this section we will illustrate, using a number of experiments, that TCONMAP can be used to create parameterised configurations that take into account and exploit the reconfigurable properties of the interconnect network of the FPGA. We show that the resulting mapped designs have a lower depth and require less resources than their generic counterparts.

Table 1. Number of 4-(T)LUTs used for various mapping methods (ratio relative to LUTMAP in %)

	LUTMAP	TLUTMAP	TCMAP lim.	TCMAP full	TCMAP fast
mux	13	5 (38)	0 (0)	0 (0)	0 (0)
barrelshift	64	32 (50)	0 (0)	0 (0)	0 (0)
crossbar	208	80 (38)	0 (0)	0 (0)	0 (0)
regex 1x	40	21 (53)	17 (43)	16 (40)	16 (40)
regex 2x	82	40 (49)	34 (41)	30 (37)	30 (37)
regex 4x	160	73 (46)	65 (41)	60 (38)	60 (38)
tlc1	94	50 (53)	49 (52)	32 (34)	32 (34)
tlc2	31	19 (61)	17 (55)	14 (45)	14 (45)

Table 2. Number of 4-(T)LUTs in longest path for various mapping methods (ratio relative to LUTMAP in %)

	LUTMAP	TLUTMAP	TCMAP lim.	TCMAP full	TCMAP fast
mux	4	2 (50)	0 (0)	0 (0)	0 (0)
barrelshift	4	2 (50)	0 (0)	0 (0)	0 (0)
crossbar	4	2 (50)	0 (0)	0 (0)	0 (0)
regex 1x	8	4 (50)	4 (50)	3 (38)	3 (38)
regex 2x	10	5 (50)	5 (50)	4 (40)	4 (40)
regex 4x	14	7 (50)	7 (50)	6 (43)	6 (43)
tlc1	7	7 (100)	7 (100)	5 (71)	5 (71)
tlc2	3	2 (67)	2 (67)	1 (33)	1 (33)

5.1. Implementation and Verification

An implementation of the TCONMAP algorithm has been made as an extension of the tool *fpga* in ABC [8] release 70930. For the Binary Decision Diagrams the CUDD library [9] was used.

For every experiment the equivalence of the mapped design to the unmapped design was verified using ABC's *cec* and the feasibility of the chosen mapping primitives was checked.

5.2. Measurements for Various Mapping Methods

Table 1 contains the area and Table 2 the depth of the designs when mapped using each of several mapping algorithms.

Table 3. Execution time for various mapping methods

(sec)	LUTMAP	TLUTMAP	TCMAP lim.	TCMAP full	TCMAP fast
mux	0.000	0.001	0.026	0.004	0.004
barrelshift	0.001	0.016	0.363	0.005	0.005
crossbar	0.001	0.027	0.071	0.011	0.011
regex 1x	0.001	0.003	0.024	0.016	0.013
regex 2x	0.001	0.005	0.054	0.043	0.028
regex 4x	0.001	0.016	0.149	32.587	0.261
tlc1	0.002	0.007	0.031	1.160	1.100
tlc2	0.000	0.001	0.008	0.007	0.007

Table 3 contains the respective execution times.

We compared the mapping results of the following algorithms:

- LUTMAP: Conventional mapping to LUTs. The tool *fpga* in ABC. All other algorithms are implemented as extensions of this tool
- TLUTMAP: Mapping to TLUTs only [1]
- TCONMAP limited: Mapping to TLUTs and TCONs but not to TLCs (shortened: TCMAP lim.)
- TCONMAP full: Mapping to TLUTs, TCONs and TLCs. All cuts are enumerated up to 2000 cuts per node
- TCONMAP fast: Mapping to TLUTs, TCONs and TLCs. All cuts containing more than 20 regular nodes are removed

5.3. Area and Depth

In this experiment we show that a design that uses parameterised configurations is considerably faster and smaller than its generic counterpart, found by LUTMAP.

Three designs were used that can be implemented using only TCONs: A 16-to-1 multiplexer, a 16-bit barrel shifter and a 16-to-16 crossbar . The control signals of the designs were chosen as parameters.

For these designs TCONMAP is able to find the optimal mapping using no (T)LUTs at all and zero depth. Although this mapping is obvious, it is important to note that TCONMAP is, to our knowledge, the first mapper that can map functionality to the reconfigurable interconnect network of the FPGA.

Next to these examples a more complex real-world design was used. This design is a basic block of a regular expression matching engine [10]. One basic block can match a character and perform operations such as the union, negation, Kleene closure. . . A regular expression matching engine would contain large numbers of these basic blocks which can be configured and connected to implement a specific regular expression. The 39 configuration signals of each basic block were chosen as parameters. We have looked at one basic block separately and a small part of the macro-architecture containing 2 and 4 blocks connected to each other.

The regular expression blocks can be implemented using ca. 50% less hardware resources when using TLUTs instead of only the conventional LUTs. Using TCONs further reduces this number by approximately 10 percentage points. This means that a parameterised configuration that is aware of the reconfigurable interconnect network can implement about 30% more basic *regex* blocks on a given FPGA than when only the LUTs can be parameterised (Table 1).

The depth of the *regex* designs is reduced by 50% for TLUTMAP and 60% for full TCONMAP (Table 2).

5.4. TLC Supergate

The limited TCONMAP algorithm does map to TLUTs and TCONs only in order to allow us to evaluate the influence of the TLC supergate on the quality of the mapping.

For the previous designs the supergate causes a modest improvement of up to 8% in area and 20% in depth (Table 2).

To highlight the value of TLC supergates two more multimode datapaths have been created. Although these examples have been designed with TCONMAP in mind, we believe that similar problems can and will occur in real-world applications as should be clear from the simple nature of the designs, which are presented hereafter.

The first design, *tlc1*, performs a multiplication of an 8-bit input and adds an integer to this. The multiplication coefficient and added integer are determined by the mode of the datapath. Three parameters define which of the 8 modes is active. The second design, *tlc2*, takes two 16-bit inputs and masks them using a bit pattern. The resulting values are combined using an adder. The bit patterns are defined by the mode of the datapath. The active mode is selected from 4 possible modes using two parameters.

For these multimode datapaths the use of the TLC supergate does improve the mapping by 35% and 18% in area (Table 1) and 29% and 50% in depth for *tlc1* and *tlc2* (Table 2) respectively.

We believe that the potential improvement of the mapping warrants the extra complexity and execution time that comes with the TLC supergate.

5.5. Execution Time & Scaling Behaviour

Full TCONMAP is considerably slower than the conventional LUTMAP algorithm. This is the cost of TCONMAP's new capabilities. The TCONMAP implementation is relatively new and could likely still be optimised for speed.

For the circuit combining 4 *regex* blocks we see a large increase in execution time compared to *regex 2x*. This is caused by a long combinatorial path that traverses all 4 basic blocks in combination with the high density of parameter inputs; there are 25 regular inputs vs. 39 parameter inputs per basic block. As a result of this a very large number of cuts has to be enumerated which takes a long time.

As mentioned before, if a node has more than 2000 feasible cuts not all cuts will be enumerated which can have a negative influence on the quality of the mapping but limits the execution time. Next to *regex 4x*, *tlc1* is the only other test case where some nodes have more than 2000 cuts and thus not all feasible cuts have been enumerated.

TCONMAP fast removes all large cuts containing more than 20 regular nodes to reduce the time enumerating cuts that probably won't be used in the final mapping, and to avoid cuts with very large BDDs, which are slow to analyse. This considerably speeds up the algorithm for some of the slowest designs (129x for *regex 4x*), but can cause a deterioration of the resulting mapping if the optimal mapping does contain cuts with more than 20 regular nodes. For the presented designs this was not the case, so TCONMAP fast is likely a good speedup heuristic.

5.6. Direct Synthesis

In some cases TCONs can easily be inferred from, for example, "if" or "case" statements in the HDL description of the design. It is important to note, however, that this is often not the case. The possibility to map parts of a design's functionality to TCONs can then only be found by analysing the functionality of the design, such as is done by TCONMAP. E.g.: for the design *tlc2* the TCONs are the result of the bit masking of the input signals.

6. CONCLUSION AND FUTURE WORK

TCONMAP is a new technology mapping algorithm that is able to take automatically into account and exploit the reconfigurable properties of the interconnect network of the FPGA and map functionality of a parameterised design to it. The proposed TCONMAP algorithm maps a design to Tuneable LUTs, Tuneable Connections and TLC supergates, creating a parameterised mapping that uses considerably less resources than its generic counterpart. It can do this for designs in which the function of the parameters is complex and it is fully automatic and can thus be used in the new tool flow for parameterised configurations.

The feasibility of the use of BDDs in the cut enumeration process and the advantage of the TLC supergate have been shown in the experiments.

Among others the following improvement could still be made: Currently the cost of TCONs is assumed to be zero because TCONs do not contribute to the area and depth of the circuit. However, for the purpose of Dynamic Circuit Specialisation the (time) cost of reconfiguring a TCON is larger than zero. A more advanced mapping algorithm could take this into account.

Acknowledgment

This work was supported by the European Commission in the context of the FP7 FASTER project (#287804).

7. REFERENCES

[1] K. Bruneel, W. Heirman, and D. Stroobandt, "Dynamic Data Folding with Parameterizable FPGA Configurations," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 16, no. 4, pp. 43:1–43:29, 2011.

[2] M. J. Wirthlin and B. Hutchings, "Improving Functional Density Through Run-Time Constant Propagation," *Proceedings of the 1997 ACM Fifth International Symposium on Field-Programmable Gate Arrays*, pp. 86–92, 1997.

[3] D. Chen and J. Cong, "DAOmap: A Depth-Optimal Area Optimization Mapping Algorithm for FPGA Designs," *IEEE/ACM International Conference on Computer Aided Design,*, pp. 752–759, 2004.

[4] J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 1, pp. 1–12, 1994.

[5] T. Davidson, F. Abouelella, K. Bruneel, and D. Stroobandt, "Dynamic Circuit Specialisation for Key-Based Encryption Algorithms and DNA Alignment," *International Journal of Reconfigurable Computing*, vol. 2012, 2011.

[6] K. Bruneel and D. Stroobandt, "TROUTE: a reconfigurability-aware FPGA router," *LECTURE NOTES IN COMPUTER SCIENCE,*, vol. 5992, pp. 207–218, 2010.

[7] A. Mishchenko, S. Chatterjee, R. Brayton, X. Wang, and T. Kam, "Technology Mapping with Boolean Matching, Supergates and Choices," *ERL Technical Report: EECS Dept., UC Berkeley*, pp. 1–7, 2005.

[8] ABC: A System for Sequential Synthesis and Verification, Release 70930. Berkeley Logic Synthesis and Verification Group. [Online]. Available: http://www.eecs.berkeley.edu/~alanmi/abc

[9] F. Somenzi. CUDD: CU Decision Diagram Package, Release 2.3.1. [Online]. Available: http://vlsi.colorado.edu/~fabio/CUDD

[10] T. Davidson, M. Merlier, K. Bruneel, and D. Stroobandt, "Dynamically Reconfigurable Pattern Matcher for Regular Expressions on FPGA," *ParaFPGA*, 2011.