# Designing Universal Logic Module FPGA Architectures for Use With Ambipolar Transistor Technology

Johannes Pfau*, Maximilian Reuter†, Klaus Hofmann† and Jürgen Becker*
*Institut für Technik der Informationsverarbeitung, Karlsruher Institut für Technologie
†Integrierte Elektronische Systeme, Technische Universität Darmstadt
Email: pfau@kit.edu, maximilian.reuter@ies.tu-darmstadt.de,
klaus.hofmann@ies.tu-darmstadt.de, becker@kit.edu

***Abstract*—Recent publications show a rise of ambipolar transistor technology research and associated implementations of multi-function logic cells in these technologies. Special properties of these technologies enable implementations of Universal Logic Modules (ULMs) using few transistors, which draws renewed interest to use such ULMs as basic logic blocks for FPGA architectures. Unlike $N$-input Lookup Tables (LUTs), most ULMs only implement a fixed subset of the possible Boolean functions.**

**In this work, we first adapt the Verilog-to-Routing (VTR) 8.0 toolflow to target such reduced-function ULM primitives. We then modify VTR's flagship 40nm architecture to use an ULM primitive instead of LUTs, modeling the double-gate carbon nanotube FET 8-function logic gate CNT-DR8F published by Liu et al. Using VTR's extensive benchmark framework, we analyze effects caused by the limited set of function offered by these primitives. To counter some of the observed effects, we present various clustered architectures, where multiple ULM cells are combined in a logic block. We conclude with an analysis of various parameters which affect performance of the different implementations.**

***Index Terms*—Reconfigurable logic, CNTFETs, Design tools**

## I. Introduction

With the increasing effort required for further shrinking CMOS processes, it is likely that Moore's Law cannot be kept up by downscaling of conventional MOSFETs [1]. A different approach to increase logic density in digital integrated circuits, which departs from increasing transistor count per area, is to extend the functionality of a single transistor and then reduce the transistor count per logic function.

Higher expressiveness on device level, due to switching either an electron (NMOS) or hole current (PMOS), can be enabled by ambipolar transistors: These transistors are capable of conducting electron and hole current in the same device, often by introduction of additional gates to program the polarity (p- or n-type). Whereas in CMOS circuits, the polarity is predefined in the design phase and established during processing for each transistor, this determination can now be postponed to runtime. The additional degree of freedom which comes with the selectable polarity can be exploited to reconfigure the transistor dynamically during runtime or statically with a fixed voltage.

These Reconfigurable Field Effect Transistors (RFETs) break with the practice of chemically doped p-n junctions and current modulation by channel inversion. Schottky junctions adjacent to the channel are electrostatically influenced (*doped*) instead, to conduct either electrons or holes, and therefore determine the polarity of the device. Different approaches for implementations of RFETs have been published, featuring silicon nanowires [2], carbon nanotubes [3], 2D materials [4] and planar structures [5].

To the best of our knowledge, research on Reconfigurable Field Effect Transistors (RFETs) is focused mainly on device level and logic cells with low transistor count like NAND, NOR, XOR and composed functions like AOI (AND-OR-INV) or OAI. [6] shows how a circuit incorporating three-independent-gate RFETs can represent different basic logic functions like NAND or NOR depending on a program signal. The device reconfigurability enables the reconfiguration of entire cells regarding their logic function. While in an ASIC, reconfigurable cells can be used to save chip area e.g. in non simultaneously executing logic paths, an ideal application for RFETs seems to be FPGAs, as they provide highly homogeneous, but individually reconfigurable structures. Complex manufacturing processes have so far prevented large scale adaption of RFET technologies. Although newer technologies focus on a simpler manufacturing process [5], these processes are still in a pre-industrial research phase, which limits the size of manufacturable circuits. Because of this, we will focus on the system level aspects affecting the FPGA design, rather than on exact area and timing results.

## II. Related Work

There have been two mostly independent discussions of ULMs in different fields in literature: The first discussion of ULMs took place in the Field Programmable Gate Array (FPGA) architecture community, when it was not yet clear whether commercial FPGAs would settle on LUT based or fixed function, ULM based architectures. Early research in this regard was carried out by Chen and Hurst [7]. They introduce the notion of *ULM.2* cells, which can implement all functions of 2 input variables, observing that there are only six basic realizations of such a cell. They also introduce

This work was supported by the German Research Foundation (DFG) within the PARFAIT project (DFG 326384402).

and discuss circuit implementations of the cell. These early discussions where not focused on usage of the cells in FPGAs, which was addressed over ten years later by Thakur and Wong [8]. Takur and Wong note that due to the interconnect in FPGAs, it is not necessary to use ULM.$m$ modules, which implement all functions of $m$ inputs. They propose that ULMs for FPGAs only have to provide one function per equivalence class instead to still allow for all functions to be implementable. Additionally, they describe an approach to derive implementations of ULM.$m$ and "almost ULM.$m$" and show that a larger set of available functions leads to better area results. This stream of work on ULM.$m$ modules culminated in Zilic's and Vranesic's work describing 3-input LUT replacements with 3 configuration bits and 4-input LUT replacements with 13 configuration bits [9]. Zilic and Vranesic also provide a generic methodology to design such cells. After this peak, some publications extended the ULM design to modules with more input variables, see e.g. Hutter in [10]. In general, however, work on ULMs calmed down since then, as FPGAs research focused on LUT technology.

Another, independent stream of ULM research has arisen in the transistor technology community: Unlike the previously described works, these works do not focus on generic design of ULMs. Instead, they are motivated by specific features of the ambipolar transistor technologies and the opportunities those offer for the design of RFET based reconfigurable logic gates. RFET reconfigurable gates can be realized with fewer transistors than their equivalent cells in standard Complementary Metal–Oxide–Semiconductor (CMOS) technology, but unlike CMOS based designs, it is not readily possible to choose the set of functions implemented by such a reconfigurable RFET cell: The set of functions is directly tied to the topology of the cell. Examples of publications on such cells include Liu et al.'s work on Doublegate Carbon Nanotube Field Effect Transistor (DG-CNTFET) cells [11], which will be used to build on in this publication. In later work, Jabeur et al. extended the 2-input function generator to be a true ULM.2, implementing all 16 possible functions [12], showing a 2x improvement in time-delay and power consumption compared to 2 input LUTs. Whereas these publications have not addressed use of the cells in FPGAs, Park et al.'s work focuses on this application [13]. They also show the arguably largest benefit of using RFET based reconfigurable cells: The configuration memory can be realized as charge storage on the transistors making up the reconfigurable cell. In that case, there is no need for external configuration storage for the logic elements.

Whereas the first set of publications focused on complete ULMs, RFET based reconfigurable cells are not complete. The focus of these earlier publications therefore lay mostly on the design of efficient ULM function generators, whereas this publication takes a certain implementation as dictated by available RFET implementations, and analyzes efficient use of these in the FPGA architecture. The second set of cited publications does largely not consider use of the reconfigurable cells in FPGAs and is primarily focused on the design of the reconfigurable cell itself. Park et al. do describe the application of the cell in FPGAs, but they do not provide information on the FPGA system architecture and associated effects of the cells on system level. This publication therefore brings RFET based ULMs into modern FPGA architectures: We take an existing RFET reconfigurable cell and uses it to replace the LUTs in an FPGA architecture which includes clustered logic elements, the Versatile Place and Route (VPR) 40 nm reference architecture. We describe how the state-of-the-art academic Verilog-to-Routing (VTR) tool flow has been adapted to allow synthesis of hardware description code to such ULM based FPGAs. In addition, we provide information on how to quantify the expressiveness of such an FPGA architecture, describe how various parameters influence it and compare the results to the LUT based reference architecture.

TABLE I
CNT-DR8F CELL FUNCTION SET. ADAPTED FROM [11].

| $V_{bA}$ | $V_{bB}$ | $V_{bC}$ | Y | Name |
|---|---|---|---|---|
| +V | +V | +V | $\overline{A+B}$ | ulm_nor2 |
| +V | +V | -V | $A+B$ | ulm_or2 |
| -V | -V | +V | $A \cdot B$ | ulm_and2 |
| -V | -V | -V | $\overline{A \cdot B}$ | ulm_nand2 |
| +V | -V | +V | $\overline{A}B$ | ulm_and2n |
| +V | -V | -V | $A+\overline{B}$ | ulm_or2n |
| -V | +V | +V | $A\overline{B}$ | ulm_and2n |
| -V | +V | -V | $B+\overline{A}$ | ulm_or2n |

## III. Designing a Modern ULM-Based Logic Cell

In order to design a FPGA architecture which properly maps to the opportunities offered by ambipolar transistor technology, we reviewed existing ambipolar technology based reconfigurable cells. We chose the CNT-DR8F cell presented by Liu et al. in [11] as a base for our work: As a two-input, eight-function cell, it offers more choices for the mapping and packing algorithms compared to very simple, e.g. two-function base cells. At the same time, unlike some larger proposed cells, this cell doe not provide the complete function set of two variables: A cell which allows to represent all possible functions of $N$ input variable can be treated as a LUT in most of the EDA tool flow. It is therefore not as restrictive as using a cell with a reduced function set. As it is often not easily possible to influence the realized function set in RFET cells, it is important to evaluate the EDA tool flow and the architecture with regards to a real ambipolar base cell, taking into account restrictions which may arise because of the cells used. The CNT-DR8F cell offers the function set shown in Table I, which also lists the names of the functions as used in the EDA tools and FPGA architecture. The function pairs ($\overline{A}B$, $A\overline{B}$) and ($A+\overline{B}$, $\overline{A}+B$) have been combined into single ulm_and2n and ulm_or2n functions.

Whereas Table I also shows the configuration values to obtain a certain output function, for the following discussions, the exact ($V_{bA}$, $V_{bB}$, $V_{bC}$) combination used is not relevant. If the physical view (which considers the voltages) is mapped to a logical view (which only considers configuration bits of

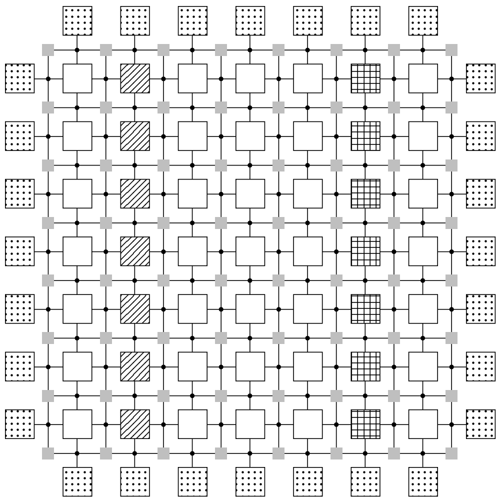
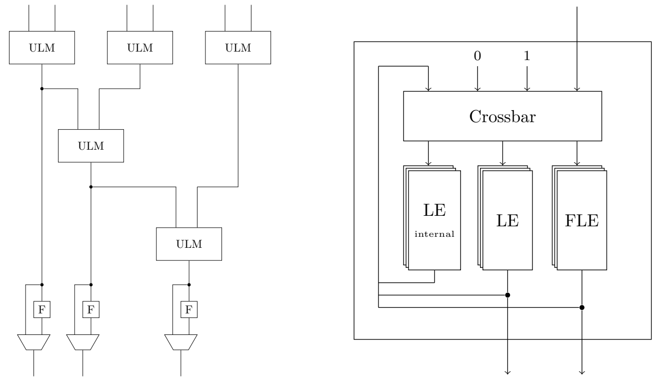


Fig. 1. Top Level FPGA Architecture. Dotted: IO blocks; white: Configurable logic blocks; diagonal: Memory blocks; grid: Multiplier blocks. Multiplier and memory blocks repeat every 8th column. Wire length of the global interconnect is 4.



(a) ULM Based 6-Input FLEs.

(b) Configurable Logic Block Cluster of Logic Elements.

Fig. 2. Basic Structure of the ULM Based Logic Elements and Logic Cluster.

SRAM cells), the combination determines essentially only the bitstream encoding used. A relevant point is that such a cell requires three configurable inputs of two input values each, so the configuration storage for such an element can be kept in three SRAM cells. To make full use of ambipolar technology benefits, some technologies allow to store the configuration as charges of the respective configuration gates [13]. In this case, logic and memory are combined and the extra transistors for configuration memory storage are not necessary. The total transistor count for the 8 function cell, including configuration storage, then reduces to seven transistors [11].

As the CNT-DR8F is a basic cell of only two inputs, it has less expressiveness than the LUT-6 often used in FPGA architectures as logic elements. Using it directly as a configurable logic block in the FPGA grid (see Fig. 1) leads to excessive global routing. Such routing leads to increased wire delay, negatively affecting the critical path and severely limiting the maximum frequency achievable for user designs in the FPGA architecture. It may also lead to routing congestion, and considering that the interconnect in modern FPGAs already makes up most of the total area, using higher interconnect channel widths is not an option.

Regarding the global interconnect, the architecture we evaluate is identical to the reference architecture. As the Configurable Logic Block (CLB) replacement is designed targeting a similar expressiveness as the reference architecture CLB, the global interconnect is not directly affected and can be kept unchanged. Other factors such as cell area, however do have an effect on the interconnect, especially considering the important share of interconnect area on the total FPGA area. Further investigation requires a more detailed analysis of logic cell and interconnect area in the target RFET technology, which is deferred to later work.

Aiming to stay close to the design of the VPR k6_frac_N10_mem32K_40nm reference architecture, the 6-input cell structure shown in Fig. 2a has been derived. The cell combines 5 ULM base cells in a tree-like structure, providing three outputs. The outputs are not independent and can only be used in parallel if a intermediate output is required. Analysis of the packed netlist will show whether such situations occur often enough in the netlists of real applications to be beneficial. The maximum depth of the cell is three ULM cells deep, but it is not a fully populated tree. Depths of one ULM or two ULMs (full tree) are available at the other outputs.

Such a cell does not allow to implement all possible input functions on an FPGA: As an example, the case where an IO input is directly connected to a register cannot be represented in this architecture, as all signals have to be routed through a complex logic block before being fed into a register. LUT based architectures are not affected by this problem, as they always allow to implement the identity function, a simple passthrough. To make the ULM based architecture universal, a passthrough mode has to ben added to the ULM element, as will be explained and analyzed in further sections.

As the expressiveness of this cell is still limited, and to be consistent with the VPR reference architecture (which uses 10 6-input LUTs in a logic cluster), the ULM based Fracturable Logic Element (FLE) cell has also been embedded in a cluster, as shown in Fig. 2b. In the same way as in the reference architecture, the inputs of the cluster are connected to the inputs of the FLEs using a full crossbar. Furthermore, also as in the reference architecture, the outputs of the cell are fed back into the crossbar. This allows to route connections between multiple of these cells locally, without using global interconnect. In order to design a logic element with similar expressiveness than the reference architecture (i.e. same device width / height of the FPGA device for benchmark circuits and and same global interconnect usage), the exact amount and type of FLEs in a complex cluster will be determined in the following sections.

## IV. EDA Toolflow and Methodology

Evaluation and benchmarking of the different architecture variants was performed using the VTR toolkit in its most recent release, version 8.0 [14]. As the VTR tool suite has been designed originally for LUT based FPGAs, various changes are necessary to use it with fixed function logic cells. The primary change is the adaption of the technology mapping step, which is performed in the ABC tool and guided by a synthesis script. The VTR tool flow provides such a synthesis script to map the user designs to LUTs. This script was adapted to perform a standard cell mapping, using a custom genlib technology file specifying the different ULM modes as individual gates. As a slight complication, VTR sometimes performs multiple iterations of reading, optimizing and writing the mapped netlist. This is supported by ABC when using LUTs as the target technology, as the representation of logic in unmapped netlists happens to be the same as for LUTs. For standard cell synthesis, this is not the case and care must be taken to perform initial ABC steps using LUT mapping, only switching to standard cell mapping in the final iteration.

The mapped netlist generated by ABC in standard cell synthesis mode cannot be directly used in the pack and place phase of VPR: The generated Berkeley Logic Interchange Format (BLIF) file uses *.gate* directives, which are not supported in VPR. Therefore a small custom script is used to transform the *.gate* directives to *.subckt* directives, which are supported by VPR. These directives are usually used to specify black boxes and more complex predefined blocks in the FPGA architecture, such as Digital Signal Processing (DSP) blocks or similar. As the VPR packer was designed to be flexible, it is used here as a quick way to pack the ULMs into the FLEs. The drawback of this simple approach is largely extended tool runtime, as an unusually large amount of cells has to be processed by the packer and as many logic functions cannot be legally packed onto some ULMs in a FLEs due to routing restrictions.

## V. Statistics to Guide Cell Architecture Design

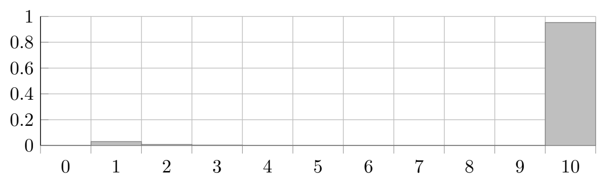
The actual benchmarking of the architecture is performed on VTR's benchmark set, whereas for evaluation of intermediate architecture results, only a reduced set of benchmarks is used due to the tool runtime issue mentioned above. To evaluate the results and guide architecture design, the statistics offered by VPR are useful, but not sufficient: These statistics are mostly centered on the top-level view of the FPGA architecture, including the FPGA dimensions in blocks of the top level grid, the block type which dominates the device size, channel width and congestion for global routing, and similar data points. The main conclusion that can be drawn from these measurements is whether the objective of performing similarly to a LUT based architecture, with respect to the global architecture, has been fulfilled. For this, a comparison between the VTR reference architecture and the same architecture with the LUT based logic block replaced by ULM logic clusters, will be carried out.

To actually design a replacement logic cell which integrates well with the global architecture and does not require changes of this global architecture, re-evaluating these global aspects is of limited use. Bottlenecks such as reduced logic expressiveness because of too few logic elements in a cluster, because of restricted local routing or because of underutilized logic elements can not be deduced from these top-level statistics. To remedy this, a custom tool for analysis and statistics collection for the logic cluster blocks has been designed. By parsing VPR's structured, packed netlist output, it is possible to gain interesting information about the sub-blocks in the hierarchy. Quantities analyzed include the utilization of input and output ports of the logic clusters, the utilization of the available FLE cells in a cluster, utilization of inputs and outputs of the FLE cells, ULM configurations used in the FLE cells and similar. Statistics are generated for aggregated quantities, such as the average and median number of inputs used, and similar values. To make informed decisions on the architecture, histograms are generated in addition. As an example, those can be used to gauge how likely all outputs, all inputs or all Flip-Flops (FFs) in a FLE are used at the same time. In addition to aggregated statistics, statistics for individual outputs and cells are generated. This allows to answer questions such as how often a specific output or cell has been used.

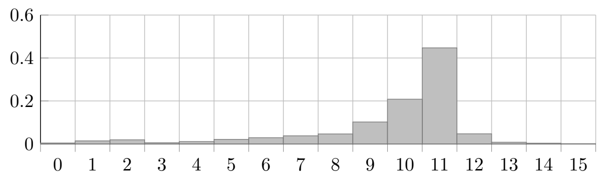
Changing certain parameters in the FLE element can largely affect the the expressiveness of the cell, which can be detected and measured in various ways. We derived the following quantities to describe these results:

1) Total FPGA size: The width and height of the FPGA in complex blocks. This measurement is only relevant if the logic blocks (LUT- or ULM-Cluster) determine the device size. Both points can be determined from the statistics file generated by VPR.
2) Percentage of FLEs used in logic clusters: This measurement gives an overview of how well device logic resources are utilized and allows to draw indirect conclusions on congestion issues of the global and local interconnect. Non-fully utilized FLEs (if the device size limiting resource are logic cells), hints that the cells cannot be connected accordingly. This may be caused by congestion on global or local interconnects, as well as by not enough available inputs or outputs.
3) Logic cluster input and output utilization: A high utilization of inputs or outputs at little FLE utilization suggests that excessive amounts of signals have to be routed using the global interconnect. Improving connectivity of the local interconnect can unburden the global interconnect and allow for better FLE utilization.
4) FF utilization: The amount of FFs which are actually used. This measurement is particularly sensitive to the set of benchmarks used. If the ULM based logic cluster is designed to have the same expressiveness as a LUT based cluster, we also expect to see the same amount of FFs in such a cluster as in the reference architecture.

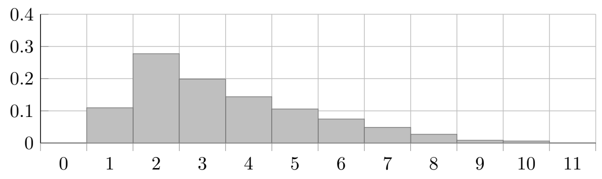
In addition to those quantities which mainly guide logic



(a) FLE Utilization.

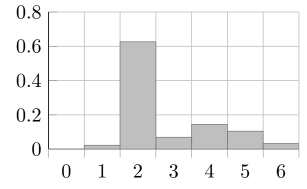


(b) Input Utilization (Abridged, 40 Inputs Available).

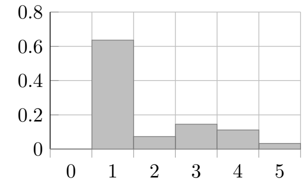


(c) Output Utilization (Abridged, 20 Outputs Available).

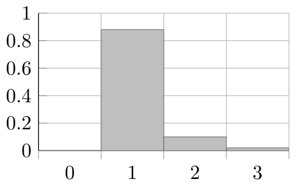
Fig. 3. Logic Cluster Utilization for Simple FLE Consisting of One ULM.



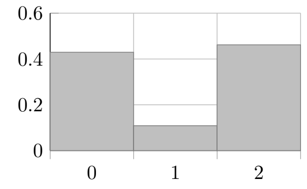
(a) Input Utilization.



(b) ULM Utilization.



(c) Output Utilization.



(d) Output Distribution.

Fig. 4. Utilization of Fracturable 6-Input ULM.

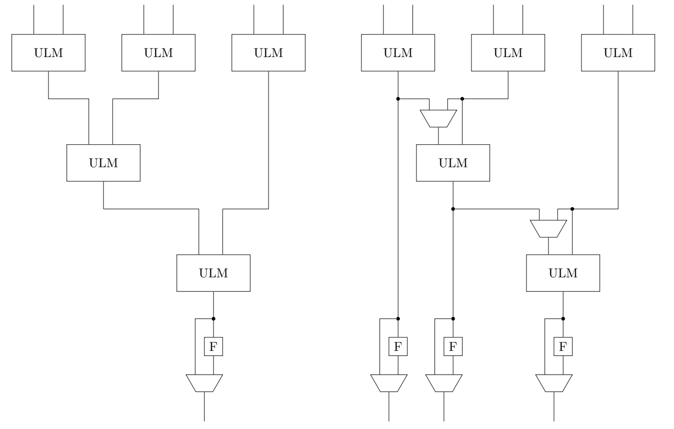
cluster design, the following quantities were used to gain insights for FLE design:

1) FLE input usage and distribution: Information about the average number of inputs used can suggest whether a FLE with more or less inputs may lead to a better utilization of ULM cells. A low number of used inputs suggests that the input circuits do not map well to the ULM topology, wheras a low number of distinct inputs suggests that some ULM inputs may be combined into a single input, to reduce size of the logic cluster crossbar. The distribution of inputs can also be used to gain certain insights: It allows to draw conclusions which part of the fracturable logic is most often used.
2) ULM usage and distribution: This is another measurement to determine which part of the fracturable logic is most used. It further provides direct feedback whether the FLE cell successfully matches the input logic functions, or whether the topology of ULMs does not allow nets to be mapped efficiently.
3) FLE output usage and distribution: This is the primary way to determine whether a FLE design is working efficiently. A low number in the average amount of used outputs suggests that the cell can not drive multiple outputs at the same time, likely caused by the cell topology. A low utilization of a certain, specific output can hint that this sub-part of the FLE topology is not frequently used and that the output may be removed from the architecture without reduction of expressiveness.

## VI. Logic Architecture Parameters

*1) Number of Logic Inputs:* Fig. 3 shows the FLE utilization and input and output utilization for a very simple, naïve FLE which consists of only one ULM. This FLE is instantiated ten times in the cluster, according to Fig. 2b. The results motivate the need for a more complex cell, as it makes use of only 9.64 inputs and 3.5 outputs in average at a ULM utilization of 96.1 %. On the other hand's side, it increases the amount of complex logic blocks used in the benchmarks by 241.4 % compared to the reference architecture. As expected, because of fewer utilized inputs and outputs, the minimum channel width is also reduced to 77.0 %.

*2) Fracturable Cells:* As an initial improvement, the combined cell in Fig. 2a is presented: Like the 6-LUT building block in the reference architecture, it can take up to six different inputs. This decision has been made to initially have similar consequences for the local interconnect in the logic cluster as in the reference architecture, especially with regards to the input crossbar. In addition, the cells have been arranged in a way to allow the system to be fracturable: If ULM 4 in the tree is configured to pass through its right-hand input, the 6-input FLE decomposes into a two-input and a four-input logic element. Similarly, the cell can be decomposed into three two-input logic elements by further putting ULM 3 into bypass mode, forwarding only its right input. In addition, multiple outputs can be derived from all six inputs, if the outputs are related and their combinational logic functions can be mapped to the topology shown in Fig. 2a. Results for these adjustments can be seen in Fig. 4. It can be seen that only one output is heavily used. Nevertheless, a fraction of the functions mapped to this output actually was two-input functions, which forced other ULM cells into bypass mode and essentially implemented simple functions in overly complex cells. We assume that the main problem here lies within our simple technology mapping and packing approach: The simple mapping by the synthesis tool onto ULMs and the combination



(a) Non-Fracturable Logic Element. (b) Introducing Bypass Mutliplexers.

Fig. 5. Modified ULM Logic Element.

of multiple such mapped functions into more complex logic blocks in the packing stage, prevents the synthesis tool from transforming functions to explicitly match those larger cells, consisting of multiple ULMs. In addition, modeling FLEs as a simple set of ULMs requires quite some effort in the packing stage, which will result in increased tool runtime.

We conclude that fracturable cells require a slightly more sophisticated approach in the EDA tools to be used effectively. Instead of modeling the base primitives for technology mapping, a more exhaustive technology library needs to be derived to reproduce every complex N-input logic function which can be represented by the complex cluster cells. Mapping to more complex functions makes less use of the local and global interconnect than mapping to multiple, simple functions. This should be reflected in the generated library to incentivize the synthesis tool to use the larger cells. An initial approach to realize this in readily available EDA tools may be to adjust the area of these *virtual* cells accordingly. For packing, the complex cells then are simply described as different configurations, depending on the number of inputs required. As an example, the cell mentioned above, which can be configured as 6-input, 2-input and 4-input or three times 2-input, may be represented using three different configurations. The task of the packing tool is then reduced to its original task, as it does not have to map multiple cells into one larger cluster. For our evaluation using the simple EDA tool approach, we decided to remove the additional outputs, as shown in Fig. 5a.

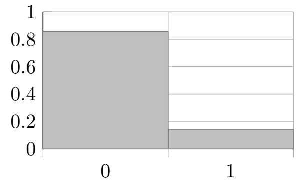
*3) Cluster-Internal Cells:* In order to reduce utilization of the global interconnect, logic clusters usually provide a local interconnect with direct feedback paths from the logic output to the logic input. If an output is fed back in the local interconnect, unless it is also required as an input for another logic function, it is not routed on the global interconnect. This means that larger utilization of the local interconnect will lead to less utilization of the logic cluster outputs. In order to avoid wasting resources, there are two possible solutions: One is to reduce the number of outputs. This however has consequences for the global interconnect and the overall FPGA architecture. As we want to keep close to the reference architecture, we decided to keep the same number of outputs in a logic cluster. The alternative we chose to influence output utilization is the introduction of internal-only cells: The output of these cells are only connected via the local interconnect to other logic cells' inputs in the same cluster, but not to outputs of the logic cluster and therefore not to the global interconnect. The logic cluster architecture including these internal cells is shown in Fig. 2b.

For internal cells, questions regarding number of inputs, the amount of basic ULMs chained and the overall topology of the Logic Element (LE) arise in the same way they do for the non-internal cells. If internal cells do not contain FFs, they need to be part of a deeper logic function. In this case, the depth of the internal cell itself should be reduced to make sure that total depth of one internal-only cell chained with one non-internal-only cell is not too large, which would prevent mapping of logic functions. If FFs are used in internal cells, the internal cells can be used to terminate a combinational net. The output must however still be mapped to a non-internal cell and if the FF in a cell is optional and bypassed, the considerations regarding path depth still hold.

These considerations and analysis of the experimental benchmark statistics suggest that simple logic cells consisting of one ULM and one FF that can be bypassed, are effective. In the final architecture, we implemented 5 of these logic elements and 15 FLEs. This adds up to 20 outputs in total, which matches the logic cluster output count. We therefore exposed the outputs of all cells and do not use internal-only cells.

*4) Additional Cell Modes:* Apart from the functional modes of Table I which are provided by the ambipolar base cell, the Electronic Design Automation (EDA) tools used require certain other, artificial modes: The ABC tool for synthesis expects a buffer cell, an inverter cell and constant zero and one generator cells. The buffer cell is unnecessary (or implicit) in an FPGA architecture and can simply be removed from the netlist. The inverter cell can be represented in the CNT-DR8F cell, when both inputs are connected to the same input variable and a configuration such as *ulm_nor2* is chosen. Connecting inputs in this way is possible for those ULM cells which are directly connected to the input crossbar, i.e. simple logic element cells and the first level of cells in the FLEs. For other ULMs, an inverter mode can not be realized without additional hardware. As mentioned earlier, a more sophisticated representation of the logic cluster in the synthesis tools should allow these to better fit logic functions to the cells. This also reduces use of the inverter mode, which should be largely absorbed into complex modes with inverted inputs.

For constant inputs, LUT based FPGA architectures can simply adjust the lookup tables to adjust for the constant inputs. An ULM based architecture on the other hand's side, actually has to provide these constant values as possible inputs to the logic cells. The constant zero and one generator cells have therefore been implemented as two additional inputs to


(a) FLE FF Utilization.

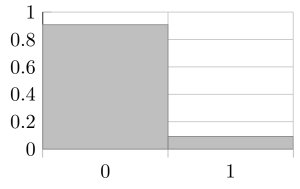

(b) Simple Cell FF Utilization.

Fig. 6. Utilization of FF in Logic Cluster.

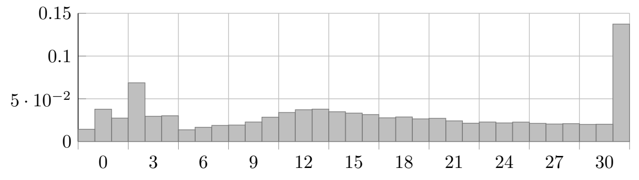
the logic cluster input crossbar, as shown in Fig. 2b. This avoids having to actually route these constant nets, avoiding increasing congestion on the interconnects, while still creating minimal resource usage in a logic cluster.

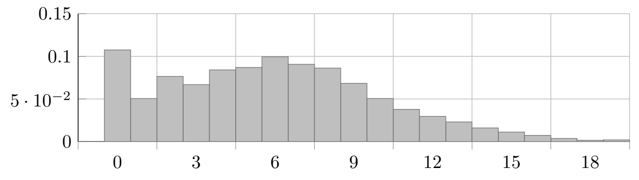
*5) Cell Bypass Mode:* In some cases, ULMs need a bypass mode which simply forwards one of the inputs to the output. This mode is functionally equivalent to the buffer mode which is required by synthesis tools, but actually removed from the netlist before packing. The main difference is the reason such a mode is required: Whereas the buffer mode is required because of limitations of EDA tools not adapted completely to an ULM workflow, the bypass mode is actually required to increase flexibility of the fracturable logic cells: As mentioned before, to realize simple two-input functions in the proposed FLE, some ULMs need to be bypassed. In the same ways as inverter cells, bypass modes can be implemented with no overhead if both inputs of the bypassed ULM can be connected to the same input. This is again the case for the ULMs directly connected to the input crossbar, as the ULMs can be configured in *ulm_and2* mode to forward its input to its output. For other ULMs, additional multiplexers are required to either bypass the cell, or connect both inputs to the same value. Connecting the inputs has the benefit of the bypassed ULM still serving as an electrical buffer and has been implemented in Fig. 5b. This approach has the drawback of requiring additional transistors for the multiplexers, as well as an additional bit of configuration storage.

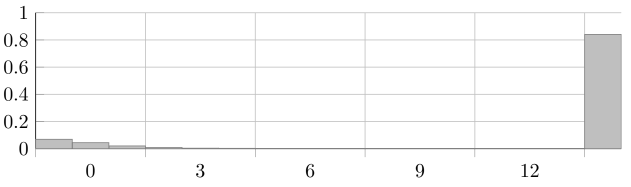
*6) FF Usage:* As there are more FLEs in our ULM based logic cluster than in the reference architecture LUT cluster, it can be questioned whether each FLE should contain one optional FF. Gathering the FF usage statistics from the VPR benchmarks shown in Fig. 6 suggests that only 12.2 % of the FFs in FLEs and 9.2 % of the FFs in internal-only logic elements are used. Adjustments to the architecture and further benchmark statistics suggest that a total of 10 FFs per logic cluster enables almost complete utilization of all FFs. At the same time, it does not increase the total FPGA area, an effect that would occur if there are too few FF in a logic cluster. To reduce the amount of FFs in the cluster to 10, we use 5 simple logic elements with FFs, 5 FLEs with FFs and 10 FLEs without FFs for the final architecture.

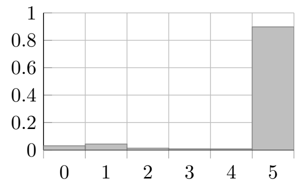
## VII. ULM Logic Cluster Cell Results

For the final architecture, all parameters were tuned as previously explained: The logic cluster has 40 inputs that


(a) Input Utilization, Mean = 16.8.


(b) Output Utilization, Mean = 6.9.


(c) FLE Utilization, Mean = 12.8.


(d) Simple LE Utilization.

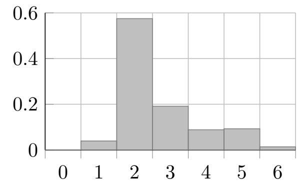
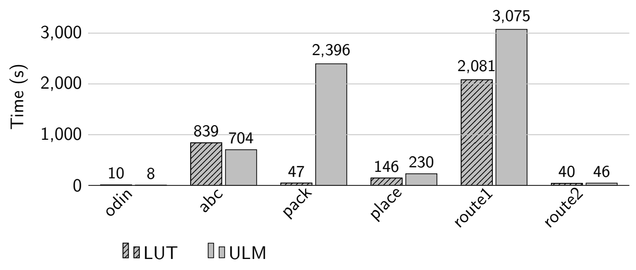

(e) FLE Input Utilization.

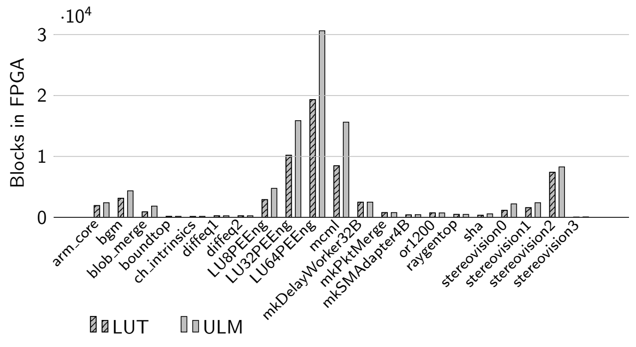
Fig. 7. Utilization Statistics for Proposed Logic Cluster.

feed an input crossbar. It combines 5 simple, two-input logic elements with one FF each, 5 FLEs with one FF each and 10 FLE without FF. Each of the previously mentioned cells provides one output of the 20 outputs of the logic cluster and there are no internal-only cells. The FLE consists of the ULM topology of Fig. 5a. Fig. 7a and Fig. 7b show the input and output utilization of the logic cluster averaged over the VPR benchmarks. It can be seen that the architecture uses up to 32 inputs, which is the maximum channel width in the routed design. Fig. 7c and Fig. 7d show the utilization of FLE and simple LE in the benchmarks set. As expected, utilization is close to 100 %. Fig. 7e shows the utilization of FLE inputs. It can be seen that sometimes more complex functions are mapped to the FLE, making use of more inputs, sometimes simpler functions are mapped. Because of this, not all inputs are always fully utilized. As there is only one output per cell, the outputs are always fully utilized.

This publication does not include timing related results for the FPGA architecture, as a reliable comparison requires timing characterization of all components in the target technology. For the CNTFET technology, not all required data is available, but transferring the results to our own custom



(a) EDA Tool Runtime, Averaged Over Circuits.



(b) FPGA Device Size in Various Circuits.

Fig. 8. Comparison of Routed Circuit Results to LUT Reference Architecture Results.

technology [5] will enable this analysis in the future. Timing delay also influences FPGA size due timing driven routing algorithms. Because of this, timing driven routing and packing was disabled in VPR.

Fig. 8a shows the average tool runtime compared to the runtime in the LUT based FPGA case. As can be seen, runtime is mostly similar, except for the packing phase, which is 100 % more expensive in the ULM approach presented. The slight increase in the routing phase can be explained by the in average slightly larger circuits. Fig. 8b shows the FPGA sizes for the circuits in the VPR benchmark set. The ULM based device is less than 10 % larger in number of blocks in most cases. For the LU*PEEng and for the mcml circuits, the ULM based FPGA can however be up to 50 % larger, which suggests that further adjustments may be necessary. We found this difference to be acceptable, but it can be further reduced by including more FLEs in the logic cluster.

## VIII. Conclusion

We first presented a simple EDA workflow to support synthesis and mapping for ULM based FPGA architectures. Then we introduced a statistics gathering process and important variables which can be measured to evaluate performance of the ULM FPGA architecture and the EDA workflow. We continued by describing how various parameters of the ULM FPGA architecture affect its performance and derived one architecture which provides similar expressiveness than the VPR 40nm reference. In the last section, we then presented a performance evaluation using the statistic variables derived earlier and compared results to the VPR reference architecture.

The presented logic cluster fits within the existing VPR reference FPGA architecture and can efficiently implement the VPR benchmark circuits. Whereas a 6-LUT requires 126 transistors for the multiplexer network and 64 bits of SRAM storage, a single ULM requires 7 transistors and 3 bits of storage. Due to the limited expressiveness of the two-input ULMs used, multiple ULM need to be combined in the FLE. This results in an amount of 35 transistors and 15 bits of memory per FLE in the ULM case, where approximately two FLEs are necessary to reach the expressiveness of a LUT-6. The input crossbar is larger than in the reference architecture, as more FLE are needed for the same expressiveness. This effect can be countered by partial depopulation of the crossbar switches, where further research is necessary to determine the most effective way of depopulation. That considered, the use of ULM cells by itself does not largely reduce FPGA size, reinforcing the decision of commercial FPGA vendors to use LUTs instead of other function generator cells. The main profits of the ULM based architecture can be reaped, when external configuration storage is replaced by on-gate charge storage as shown by Park et al. in [13]. In that case, configuration storage for ULMs is embedded in a logic-in-memory style and does not require additional area. Whereas the CNT technology used for the CNT-DR8F implementation in [11] has not seen widespread adaption, the device design is transferable to other ambipolar technologies. We intend intend to transfer the cell design to recent planar RFETs [5], [15], which will enable a simpler manufacturing process.

Further work on system level may analyze which set of complex functions is most beneficial in a FLE and reduce the set of available configurations. In that case, the configuration writing circuit required to configure the FLE can be reduced. As discussed, a more complex technology mapping approach may be implemented to allow the EDA tools to more efficiently map logic to the available complex cells. As the focus of this publication was on the FPGA architecture itself, this research is left for further work.

## References

[1] M. Lundstrom, "Moore's law forever?" *Science*, vol. 299, no. 5604, pp. 210–211, 2003. [Online]. Available: https://science.sciencemag.org/content/299/5604/210

[2] W. Weber, A. Heinzig, J. Trommer, D. Martin, M. Grube, and T. Mikolajick, "Reconfigurable nanowire electronics–a review," *Solid-State Electronics*, vol. 102, pp. 12–24, 2014.

[3] Yu-Ming Lin, J. Appenzeller, and P. Avouris, "Novel carbon nanotube fet design with tunable polarity," in *IEDM Technical Digest. IEEE International Electron Devices Meeting, 2004.*, 2004, pp. 687–690.

[4] B. Radisavljevic, A. Radenovic, J. Brivio, V. Giacometti, and A. Kis, "Single-layer mos 2 transistors," *Nature nanotechnology*, vol. 6, no. 3, pp. 147–150, 2011.

[5] T. A. Krauss, "Planar electrostatically doped reconfigurable schottky barrier fdsoi field-effect transistor structures," Ph.D. dissertation, Technische Universität, Darmstadt, Juni 2019. [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/8771/

[6] S. Rai, J. Trommer, M. Raitza, T. Mikolajick, W. M. Weber, and A. Kumar, "Designing efficient circuits based on runtime-reconfigurable field-effect transistors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 3, pp. 560–572, 2019.

[7] Chen and Hurst, "A comparison of universal-logic-module realizations and their application in the synthesis of combinatorial and sequential logic networks," *IEEE Transactions on Computers*, vol. C-31, no. 2, pp. 140–147, 1982.

[8] S. Thakur and D. F. Wong, "On designing ulm-based fpga logic modules," in *Third International ACM Symposium on Field-Programmable Gate Arrays*, 1995, pp. 3–9.

[9] Z. Zilic and Z. G. Vranesic, "Using bdds to design ulms for fpgas," in *Fourth International ACM Symposium on Field-Programmable Gate Arrays*, 1996, pp. 24–30.

[10] M. Hutter, "Designing universal logic modules," in *9th International Conference on Electronics, Circuits and Systems*, vol. 2, 2002, pp. 709–712 vol.2.

[11] J. Liu, I. O'Connor, D. Navarro, and F. Gaffiot, "Design of a novel cntfet-based reconfigurable logic gate," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI '07)*. IEEE, 09.03.2007 - 11.03.2007, pp. 285–290.

[12] K. Jabeur, N. Yakymets, I. O'Connor, and S. Le-Beux, "Fine-grain reconfigurable logic cells based on double-gate cntfets," in *Proceedings of the 21st Edition of the Great Lakes Symposium on Great Lakes Symposium on VLSI*, ser. GLSVLSI '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 19–24.

[13] J. Park, J. Bae, J. Eum, S. H. Jin, B. Park, and J. Lee, "High-density reconfigurable devices with programmable bottom-gate array," *IEEE Electron Device Letters*, vol. 38, no. 5, pp. 564–567, 2017.

[14] 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," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 7, no. 2, pp. 1–30, 2014.

[15] M. Reuter, J. Pfau, T. A. Krauss, M. Moradinasab, U. Schwalke, J. Becker, and K. Hofmann, "Towards ambipolar planar devices: The defet device in area constrained xor applications," in *2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS)*, 2020, pp. 1–4.