An Efficient Physical Design of Fully-testable BDD-based Circuits

Andreas Rauchenecker
Institute for Integrated Circuits
Johannes Kepler University, Austria
Email: andreas.rauchenecker@jku.at

Robert Wille
Institute for Integrated Circuits
Johannes Kepler University, Austria
Email: robert.wille@jku.at

***Abstract*—For the manufacturing test of ASICs, it is important to reach a high test coverage in order to get the defect level as low as possible. However, complex digital circuits are usually not fully testable. In order to address that, previous work suggested to realize the circuits by means of *Binary Decisions Diagrams* (BDDs). Here, each node is implemented using *multiplexer gates* (MUX gates) which, with some minor additions, yield 100% testable circuits, with respect to stuck-at and path delay faults. Unfortunately, current physical implementations of MUX gates are rather expensive with respect to propagation delay, power consumption, or transistor count. Hence, despite the prospect of gaining 100% testability, BDD-based circuits did not find significant attention yet. In this work, we propose an alternative realization of MUX gates based on pass transistor logic which addresses these drawbacks. Experiments show that this allows for the realization of fully testable BDD-based circuits which are competitive to or, in many cases, even better than state-of-the-art realizations.**

I. Introduction

It has always been necessary to test chips during the manufacturing process. These tests are expensive and time consuming but are inevitable for mass market products. However, despite issues related to test pattern generation [1], compaction [2] [3], etc., the actual design of the circuit itself also significantly affects the test process. In fact, most circuit designs are not 100% testable, due to redundancies [4]. Motivated by that, researchers and engineers considered the question of how to improve the testability of circuits. A promising proposal in this direction has been made in [5]. Here, so-called *Binary Decision Diagrams* (BDDs, [6]) – a graph-based data-structure for the efficient representation and manipulation of Boolean functions to be synthesized – are employed. Each BDD can easily be mapped to a circuit composed of *multiplexer gates* (MUX gates). On top of that, the authors of [5] suggested (minor) extensions which eventually resulted in a MUX circuit with a testability of 100%, with respect to stuck-at and path delay faults. Due to these promises, BDD-based circuits received significant interest – however mostly in the domain of logic design. When it comes to their physical realization, not so much attention can be observed yet. This is mainly caused by the fact that hardly any efficient physical realization of BDD-based circuits is available thus far. More precisely, each BDD-based circuit basically represents a logic circuit composed of MUX gates. But most established methods for the physical design realize MUX gates by standard gates only (i.e. transform a MUX into a sub-circuit composed of AND, OR, and NOT gates), which are not optimized for MUX-realization and usually lead to rather large and expensive designs. This also applies to CMOS multiplexer cells which might be available in certain synthesis libraries.

Within other contexts, the realization of MUX gates has been considered using *Pass Transistor Logic* (PTL, [7]) – leading to the proposals e.g. in [8] [9] [10]. But also here, designs result which either have to be created manually (not feasible for large circuits), have some problems with inverters, or rely on, unbuffered signals chains and, because of that, do not allow for long signal chains, or require complex clock circuits due to domino logic structure [11]. Hence, also these approaches do not satisfy the needs for an efficient realization of BDD-based circuits.

Overall, realizing a given function employing BDD-based circuits yields significantly larger designs with respect to propagation delay, power consumption, or transistor count compared to state-of-the-art designs obtained e.g. by commercial tools such as the *Synopsys Design Compiler*. As a consequence, BDD-based circuits got not established despite their capability of guaranteeing 100% testability. In this work, we are addressing this problem. To this end, we are proposing alternative physical realizations of MUX gates which overcome the drawbacks of previous solutions. This eventually results in a design flow which allows for the automatic design of fully testable BDD-based circuits whose propagation delay, power consumption, or transistor count is competitive or, in many cases, even better than conventional realizations using state-of-the-art design tools. By this, we are paving the way for establishing BDD-based circuits as an efficient alternative to conventional design which guarantee 100% testability.

The remainder of this work is structured as follows: The next section provides the background on BDDs, BDD-based circuits and their testability, as well as related work on corresponding physical realizations. Afterwards, Section III discuss various MUX realizations and introduce the newly proposed solutions. This leads to a design flow for the efficient realization of BDD-based circuits which is described in Section IV. The resulting physical designs have intensely been simulated and evaluated – the respectively used setup as well as the obtained results are summarized in Section V. Finally, the paper is concluded in Section VI.

II. Background and Motivation

This section reviews the basics of BDDs as well as the synthesis scheme proposed in [5] which yields fully-testable circuits. Afterwards, we discuss the problems of this approach with respect to the physical realization of the resulting circuits and, by this, motivate the contribution of this work.

A. Binary Decision Diagrams

Every Boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B}$ can be represented by a graph-structure defined as follows:

Definition 1. *A* Binary Decision Diagram *(BDD, [6]) over Boolean variables X with terminals $T = \{0, 1\}$ is a directed acyclic graph $G = (V, E)$ with the following properties:*

1) *Each node $v \in V$ is either a terminal or a non-terminal.*

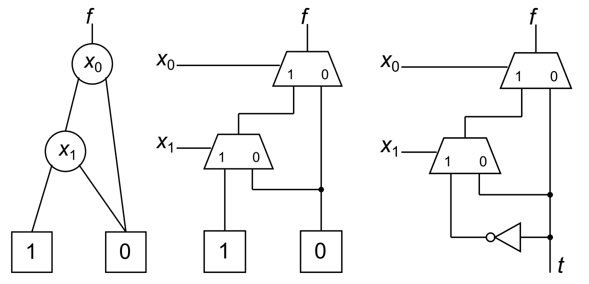
2) *Each terminal node $v \in V$ is labeled by a value $t \in T$ and has no outgoing edges.*
3) *Each non-terminal node $v \in V$ is labeled by a Boolean variable $x_i \in X$ and represents a Boolean function f.*
4) *In each non-terminal node (labeled by x_i), the Shannon decomposition*

$$f = \overline{x}_i f_{x_i=0} + x_i f_{x_i=1}$$

is carried out, leading to two outgoing edges $e \in E$ whose successors are denoted by $low(f)$ (for $f_{x_i=0}$) and $high(f)$ (for $f_{x_i=1}$), respectively.

The size *of a BDD is defined by the number of its (non-terminal) nodes.*

Example 1. *Fig.1(a) shows a BDD representing the function $f = x_0 \wedge x_1$. Edges to $low(v)$ ($high(v)$) are denoted by dashed (solid) lines. The BDD has a size of 2.*



(a) BDD (b) Resulting circuit (c) Fully testable circuit

Fig. 1: BDD and resulting circuits for $f = x_0 \wedge x_1$

A BDD is called *free* if each variable is encountered at most once on each path from the root to a terminal node. A BDD is called *ordered* if in addition all variables are encountered in the same order on all such paths. The respective *order* is defined by $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$. Finally, a BDD is called *reduced* if it does neither contain isomorphic sub-graphs nor redundant nodes. To achieve reduced BDDs, *reduction rules* are applied [6]. Applying the reduction rules leads to *shared nodes*, i.e. nodes that have more than one predecessor. In the following, reduced ordered binary decision diagrams are called BDDs for brevity. BDDs are canonical representations, i.e. for a given Boolean function and a fixed order, the BDD is unique [6].

B. Logic Synthesis

Given a BDD $G = (V, E)$, a logic circuit can be derived by traversing the decision diagram and substituting each node $v \in V$ with a MUX gate. More precisely, for each node $v \in V$, a MUX gate is created where the select input of each MUX gate is connected to the primary input x_i as given by v and the 0-input (1-input) is connected to the output of the MUX gates created for $low(v)$ ($high(v)$).

Example 2. *Consider again the BDD from Fig. 1(a). Applying the synthesis scheme described above yields the circuit as shown in Fig. 1(b).*

Circuits derived from BDDs have the advantageous property of being 100% testable for established fault models such as the stuck-at fault model or the path-delay fault model. To this end, only one additional input and one inverter has to be added. More precisely, the terminal node 0 is substituted by the new primary input (denoted by t for *test*) and t is additionally connected to the 1-terminal by an inverter.

Example 3. *Fig. 1(c) shows the resulting circuit obtained by adding t and the inverter as described above.*

Having the resulting circuit and setting t to 0, the original functionality results. Instead, setting t to 1, the complement is computed. This allows for executing the desired functionality while, at the same time, yields 100% testability e.g. for stuck-at faults. In fact, by changing the value of t, all internal signals of the circuits change their value. This guarantees full controllability, i.e. all possible fault locations can indeed be triggered. Observability, i.e. propagating the faulty effect to the primary outputs, is easily possible due to the select inputs of each MUX gate which always allow for the generation of a respective propagating path. Proofs showing this property have been provided in [5].

C. Physical Realization

Despite the promises of gaining 100% testability, how to efficiently realize BDD-based circuits has hardly been considered yet. A naive solution is to simply map each MUX gate into a corresponding sub-circuit composed of AND, OR, and NOT gates and physically realize the resulting structure. However, this yields rather expensive designs with respect to propagation delay, power consumption, or transistor count. Even if dedicated MUX cells are available in a synthesis library, they are still rather expensive (see also Section III-A). As an alternative, mapping the corresponding MUX gates to *Pass Transistor Logic* (PTL, [7]) seems promising. However, respective initial works such as proposed in [12] have the big disadvantage that these circuits are complex and are usually implemented by hand. Another way proposed in [8] suggested the introduction of custom library cells called *Lean Integration with Pass-Transistors* (LEAP). Here, three custom library cells have been considered which are capable of mapping one to three BDD nodes, requiring a complex algorithm for mapping the whole function. Since this PTL style additionally relies on a single rail, additional inverters have to be utilized for the needed inverted control signals of ascending pass transistors – again, way too much effort for an efficient design.

Finally, an approach proposed in [9] employs a similar strategy, but utilizes so-called Differential Cascode Voltage Switch with Pass Gate Logic (DCVSPG) cells rather than LEAP cells. This eliminates the need for inverters by a complementary rail. By this, and the fact that DCVSPG has no output buffer, the authors from [9] assume that DCVSPG is faster, but evaluations summarized in [13] showed that the resulting design scheme cannot be applied for longer chains of logic. This is caused by the fact that the pass signals have to traverse through the whole logic chain without being buffered. As a consequence, also this solution is unsuitable in order to realize BDD-based circuits. Overall, no efficient physical design for MUX gates exists yet which would allow for exploiting BDD-based circuits and their 100% testability.

III. Design Styles for Multiplexer

In this work, we aim to address the problem discussed above by providing an alternative physical design for BDD-based circuits. The main idea is to reduce the complexity of the technology mapping stage considering a single library cell only: a 2-to-1 MUX. Using a dual rail PTL, no inverters and no additional buffers have to be inserted. This allows for a mapping of a BDD to the corresponding physical design where each BDD node can independently be replaced by the proposed cell (no relations between the respective BDD-nodes have to be considered). Because of this, even different variations of cells can be considered – and, eventually, the one which fits best can be chosen (see design variants in section III-C).

In order to describe the details of the proposed solution, we continue with a review of the "standard" CMOS and PTL realizations of a MUX gate and show improvements to these designs to better suit our purposes. The resulting designs eventually allow for a physical realization of BDD-based circuits which is competitive against state-of-the-art solutions that do not guarantee 100% testability.

A. Standard CMOS Gate

The binary decision diagrams introduced in the previous chapter can be implemented in hardware with 2:1 MUX cells. Usually, these cells are available as standard cell in any ASIC technology and employ the basic CMOS structure of the logic function $Q = A * nS + B * S$.

Design variant 1 (STD). *The* standard *(STD) multiplexer gate realized as CMOS complex gate with NMOS and PMOS transistors is shown in Fig. 2.*

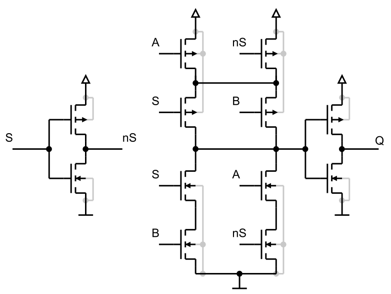


Fig. 2: Standard CMOS realization

B. Pass Transistor Logic Gates

Pass Transistor Logic (PTL, [7]) is predestined for realizing multiplexer structures. Here, the transistors are not used to establish a connection to the supply voltage or the ground. Instead, transistors are used as a pass gate which lets a signal pass or not depending on a control signal. This behavior directly describes a multiplexer.

Design variant 2 (PTL). *Fig. 3 shows the simplest way how a MUX can be implemented in PTL.*

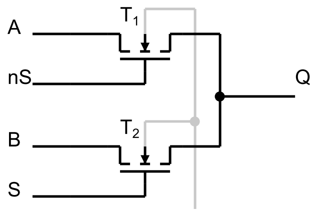


Fig. 3: Simplest PTL realization

The PTL design basically realizes a 2-to-1 MUX gate. A set of control signals is connected to the gates of the NMOS transistors and another set of pass signals is connected to the source pins of the transistor. Depending on the control signals, the pass signals are handed to the output [7]. In the circuit from Fig. 3, pass signal A is led to the output if control signal nS is set to "1" (switching the pass-transistor $T1$ on). Signal B is passed to the output if signal S is set to "1" and, therefore, $T2$ is activated. This behavior can be described with the boolean equation $Q = A * nS + B * S$. To avoid unwanted states at the output, only one of the transistors is allowed to be switched on. To guarantee this, a single control signal S is used which switches $T2$, while nS (controlling $T1$) is the inverted signal of S.

The problem of this design variant is that the corresponding signals suffer from a significant degradation (i.e. their "1"-voltage level will not be fully reached). Moreover, the driver strength of a corresponding MUX cell depends on the cell from which the pass signal originated. Hence, chaining multiple of these cells leads to an substantial drop of signal strength (in case of level "1") and driver strength – until the signal value can no longer be determined or the following cells cannot be driven anymore.

C. Optimized Pass Transistor Logic Gates

A first approach in getting a better output signal is by using PMOS transistors as level restorer. The PMOS transistors establish a conductive path to supply voltage in case an output signal is set to "1". By this, the voltage loss from the pass transistor is eliminated.

Design variant 3 (DCVSPG). *Fig. 4 shows an alternative design called* Differential Cascode Voltage Switch Pass Gate *(DCVSPG, [14]) which utilizes a level restorer at the output. The restorers are controlled by the complementary output and are only active when necessary. In addition to the level restorer, a complementary circuit is necessary to be able to control the level restorer. This leads to a differential structure. Since for MUX cells the inverted control signal is required anyway, this is actually advantageous in a MUX based design.*

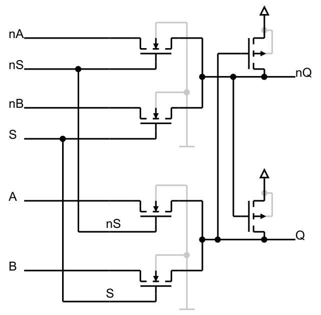


Fig. 4: DCVSPG realization

In order to improve the driver strength, a CMOS inverter buffering the output can be attached to the DCVSPG cell. Now the driver strength of the cell is independent of the preceding cell and can be adjusted by varying the W/L ratio of the inverter transistors.

Design variant 4 (CPL). *Adding the CMOS inverter yields the* Complementary Pass Transistor Logic *(CPL, [15], [16]) cell which is shown in Fig. 5.*

Using a CPL circuit, i.e. using an inverter to buffer the output, allows to skip the level restorer transistor. This is motivated by the fact that a single pass transistor at the voltage level is still sufficient to switch the inverter transistors on and off.

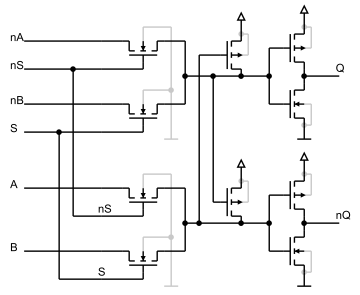


Fig. 5: CPL realization

Design variant 5 (BCPL). *Skipping the level restorer transistor, yields a structure called* Buffered Complementary Pass Transistor Logic *(BCPL, [17]) and is shown in Fig. 6.*

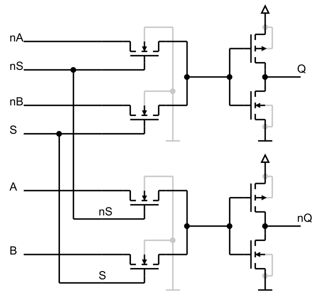


Fig. 6: BCPL realization

Besides that, another variant is proposed where the level restorers are not connected to supply voltage but to the differential output. This yields a solution which is optimized for power consumption.

Design variant 6 (EEPL). *Connecting the level restorers to the differential output yields a structure called* Energy Economized Pass Transistor Logic *(EEPL, [18]) as shown in Fig. 7 .*

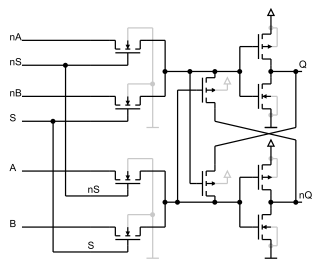


Fig. 7: EEPL realization

Overall, the considerations from above lead to the proposal of the following three physical designs for the realization of single MUX gates and, hence, BDD-based circuits:

1) **BCPL**, which is the fastest design variant; the reduced capacitance due to the lack of level restoring PMOS transistors leads to a minimum propagation delay
2) **CPL**, which is nearly as fast as BCPL but additionally employs a reduced power dissipation; the level restorer in front of the inverter ensures that the transistors are completely switched on or off which reduces the leakage current.
3) **EEPL**, which is the slowest design variant, but constitutes the most energy efficient PTL solution available.

Note that, the DCVSPG variant is not further considered as this solution is obviously outperformed by the other variants. Furthermore, recall that all solutions proposed in this subsection allow for a mapping of single MUX gates without the need to consider relations to other MUX gates. In contrast to previously proposed solutions, this allows for a simple automatic mapping of a BDD-based circuit to a physical design without any adaption needed for changing the PTL style. Because of that, all variants summarized above can be easily realized and simulated and, afterwards, the one with the best performance according to the designer's needs can be chosen. Considering that the effectiveness of a PTL style strongly depends on the design and function to be realized [7], this leads to great potential with respect to design exploration.

IV. Resulting Design Flow

Based on the discussions and proposals from the previous section, an automatic design flow can be defined which realizes an efficient and 100% testable circuit for a given function to be synthesized. This section describes the respective steps of the resulting flow.

Assuming the design to be implemented is available in Verilog on a higher abstraction level, the first step is to transform the Verilog code into a BDD representation. This task can be accomplished using tools like *CUDD* [19] or ABC [20]. Next, the resulting BDD is to be converted into a MUX circuit and dumped into a Verilog netlist, representing every MUX gate with an if-statement. Again, this can be conducted using *ABC*. Afterwards, the actual technology mapping is conducted. Here the respective cell variants presented above are applied and, if necessary, the respective inverters for control signals are added. To gain 100% testability in the next phase arrangements are made as described in section II-B. These steps have been automated using a Python script. Finally, the resulting netlist is handed over to a layout tool such as Cadence Encounter or Virtuoso which conducts the placement and routing and, eventually, completes the design for manufacturing. Besides that, corresponding automatic solutions for the simulation of the performance of the resulting designs have been created. For measuring the performance of the circuit, one has to simulate the maximum propagation delay. The most trivial way would be to perform a static timing analysis on the netlist, revealing the maximum delay. But this can only be done if a fully functional synthesis description of the MUX cell is available. This is not necessarily the case for custom made MUX cells like the pass transistor logic cells. To be able to simulate circuits without a full library, one could perform a transistor-level simulation. Simulating every possible input case is highly inefficient and, hence, infeasible. Therefore, it is inevitable to identify the critical path and the corresponding input case beforehand. Determining the critical path can be done by analyzing the STD gate representation of this circuit with tools like *Synopsys Primetime* or *Nanotime*. The critical path detected for this variant should also apply for the PTL design styles (at least for the PTL styles considered here). Hence, the resulting critical path with its corresponding input can be simulated on the transistor level with Simulators such as *Cadence Spectre* or *Synopsys HSpice*.

Overall this leads to the following flow for implementing and simulating a BDD-based circuit using the proposed PTL design styles.

Physical Design:

- Create a BDD representation
- Convert the BDD to a MUX representation
- Dump the circuit as a Verilog netlist
- Map the MUX netlist to the proposed PTL styles
- Modify the netlist to gain 100% testability
- Conduct placement and routing

Simulation:

- Identify the critical path
- Determine the input for the critical path
- Simulate the critical path
- Determine the results

As all these steps can be conducted automatically, all design alternatives proposed in Section III-C can efficiently be realized and simulated. Then, the one which fit's the design needs best can be chosen.

V. Simulation and Evaluation

The proposed physical designs and the resulting design flow have intensely been evaluated and compared to standard and state-of-the-art solutions. To this end, the design flow proposed in the previous section has been implemented using the mentioned tools as well as Python scripts. The design styles proposed in Section III-C, namely BCPL, CPL, and EEPL, have been realized in 350nm technology using *Cadence Virtuoso*. Then, functions provided in [5] have been taken as benchmarks and realized using the proposed flow. Finally, the resulting physical designs have been compared to:

- designs obtained by a solution which realizes the BDD-based circuit using standard gates (denoted by *STD* in the following), i.e. a solution which relies on the standard cells MUX2X2 and INVX2, and
- designs obtained by synthesizing the desired functions using the *Synopsys Design Compiler* (denoted by *SYN* in the following).

In the following the obtained results are summarized and discussed. Before that, we briefly review the considered quality criteria.

A. Considered Metrics

In our evaluations, the following metrics have been considered:

1) Propagation delay: For simulating the propagation delay, the critical path with its input wiring had to be found. This has been achieved by analyzing the gate netlists with the Synopsys tool *Nanotime*. Assuming the critical path is the same for all logic styles, analyses have been carried out using this path. Since synthesis with the *Synopsis Design Compiler* yields a completely different structure, the SYN design has also been analyzed with Nanotime for its own critical path.

2) Power Consumption: For an estimation of the power consumption, the current drawn from the supply voltage source has been measured and used for calculating the power using

$$P = V_{dd} \int_0^{8ns} |i(t)| dt. \tag{1}$$

The resulting value is the power consumed for one switching event. This value has been multiplied by 10^6 – assuming the circuit is running at 1 MHz.

3) Transistor count: The number of transistors in the designs is being used as a measure for the area of the resulting designs. It should be mentioned that pass transistor logic utilizes more NMOS transistors than PMOS transistors. Since PMOS transistors are usually larger than NMOS transistors, pass transistor logic results in a smaller design even if the transistor count is the same or slightly above the transistor count of the standard CMOS style.

Furthermore, note that the transistor count of SYN is only an estimation since the *Synopsis Design Compiler* only provides the cell area as a result. To estimate the transistor count, the obtained cell area has been divided by the cell area of the standard NAND gate (providing us with the corresponding number of NAND gates) and, afterwards, multiplied by four (since one NAND gate is realized using four transistors).

B. Obtained Results

The obtained results are summarized in Tables I to III for propagation delay, power consumption and transistor count, respectively. The first column respectively provides the name of the benchmark, while the remaining columns list the results for each considered design style.

1) Propagation delay: Results are shown in Table I. As clearly can be seen, BCPL performs best in terms of propagation delay. An exception can only be observed when the BDD results in a much more complex structure than standard synthesis approach (see e.g. benchmarks *i5* and *b9*). CPL is almost as fast as BCPL (as expected; see discussions in Section III-C) – only the additional capacitance of the level restorer transistors slows the design down. EEPL is rather slow, even slower than STD version. Moreover, it is interesting that in 5 out of 12 cases, the BDD-based realization is faster than the one automatically synthesized using the commercial tool (i.e. SYN).

TABLE I: Propagation delay in ns

	Proposed				S-o-t-a	
	BCPL	CPL	EEPL	Best PTL	STD	SYN
9symml	**1.321**	1.395	2.158	**1.321**	1.988	2.525
alu2	**1.886**	2.268	3.772	**1.886**	2.699	5.354
b9	1.646	1.878	2.957	1.646	2.732	**1.488**
C17	**0.573**	0.605	0.812	**0.573**	0.898	0.606
count	**2.805**	3.176	5.044	**2.805**	4.641	4.583
f51m	**1.210**	1.371	1.953	**1.210**	1.977	1.815
i1	1.528	1.683	2.600	1.528	2.633	**1.243**
i5	2.780	2.950	4.779	2.780	4.332	**0.571**
t481	1.057	**0.799**	1.673	**0.799**	1.445	3.796
tcon	**0.263**	0.288	0.305	**0.263**	0.575	0.599
x2	**1.008**	1.185	1.748	**1.008**	1.798	1.138
z4ml	**1.030**	1.111	1.673	**1.030**	1.549	1.675

2) Power consumption: Results are shown in Table II. As can be seen, power consumption strongly depends on the considered benchmark. It can be said that BCPL is more "power hungry" – especially if the design gets larger. The reason for this is the missing level restorer causing more cross-current through the buffers/inverters since the transistors are not completely turned off. CPL draws a little less current except for smaller designs, where the additional capacitances results in more consumed energy than the cross current in the BCPL buffer. Overall, EEPL is the most power efficient design, as it has been designed for energy efficiency. In most cases, the SYN design draws less current than the PTL designs. This is due to the CMOS structure drawing only current while switching and has in static case nearly no current consumption.

Power consumption has also a direct relation to the transistor count as one can see when comparing Table II with Table III.

TABLE II: Power consumption in uW

	Proposed				S-o-t-a	
	BCPL	CPL	EEPL	Best PTL	STD	SYN
9symml	2.777	1.961	1.851	1.851	**1.199**	27.880
alu2	26.199	19.203	**16.942**	**16.942**	38.664	95.084
b9	88.647	125.836	18.941	18.941	26.813	**9.506**
C17	1.902	2.297	1.497	1.497	1.484	**0.533**
count	114.755	48.157	42.213	42.213	35.477	**30.655**
f51m	9.092	6.284	**5.844**	**5.844**	11.834	21.065
i1	12.433	28.488	5.804	5.804	**4.694**	10.467
i5	496.311	326.188	69.418	69.418	4.929	**0.522**
t481	33.613	28.870	**6.806**	**6.806**	20.380	42.209
tcon	**0.814**	0.853	16.012	**0.814**	1.538	2.291
x2	6.640	11.182	4.741	4.741	11.255	**4.224**
z4ml	5.726	3.968	3.698	3.698	3.330	**1.041**

3) Transistor count: Results are shown in Table III. In 3 out of the 12 cases, SYN requires the smallest amount of transistors. As expected, the STD realization consists always of more transistors than the PTL designs. BCPL has the smallest transistor count of all PTL styles, as expected since it has the smallest number of transistors per cell. Overall one can say that the difference to SYN is limited and that the PTL designs are competitive to the SYN design. The benchmark *i5* is a extreme case since there the BDD results in a significantly larger and more complex design compared to the standard approach.

TABLE III: Transistor count

	Proposed				S-o-t-a	
	BCPL	CPL	EEPL	Best PTL	STD	SYN
9symml	**184**	230	230	**184**	290	204
alu2	**1368**	1710	1710	**1368**	2250	1408
b9	1536	1920	1920	1536	2462	**454**
C17	56	70	70	56	92	**30**
count	1520	1900	1900	1520	2312	**636**
f51m	**424**	530	530	**424**	700	566
i1	384	480	480	384	600	**222**
i5	4856	6070	6070	4856	7596	**660**
t481	**224**	280	280	**224**	376	942
tcon	**64**	80	80	**64**	96	160
x2	304	380	380	304	500	**200**
z4ml	192	240	240	192	312	**114**

4) Summary: Overall, the results clearly show that efficient physical realizations for BDD-based circuits can be obtained which are competitive or, in many cases, even better than conventional realizations using state-of-the-art design tools. By this, the benefit of 100% testability in BDD-based circuits can be exploited without the need to accept the drawback of costly physical realizations anymore. The results also show that of the three proposed PTL styles each has its own benefits. And these benefits can be exploited efficiently by deciding per design from which PTL style the function benefits the most.

VI. Conclusion

In this work, we proposed several designs for the realization of BDD-based circuits. By this, we addressed the problem that, although BDD-based circuits come with the promise of 100% testability, they hardly made it into practice yet due to their costly physical implementation. Here, three different design styles have been presented – each with their respective characteristics. As corresponding physical realizations can automatically and efficiently been derived for each logic style, the designer can eventually decide which fits the required specifications best. Simulations and evaluations confirmed that the proposed designs are competitive or, in many cases, even better than conventional state-of-the-art realizations. This is a substantial step towards establishing BDD-based circuits and their 100% testability as an efficient alternative to conventional designs.

References

[1] C. Wang, S. M. Reddy, I. Pomeranz, X. Lin, and J. Rajski, "Conflict driven techniques for improving deterministic test pattern generation," in *International conference on Computer-aided design*, 2002, pp. 87–93.

[2] S. Eggersglüss and R. Drechsler, "Improving test pattern compactness in sat-based ATPG," in *16th Asian Test Symposium (ATS 2007)*, Oct 2007, pp. 445–452.

[3] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 776–792, May 2004.

[4] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures*. Elsevier, 2006.

[5] R. Drechsler, J. Shi, and G. Fey, "Synthesis of fully testable circuits from BDDs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 3, pp. 440–443, March 2004.

[6] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, Aug 1986.

[7] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design - A Systems Perspective*. Addison-Wesley Publishing Company, 1993.

[8] K. Yano, Y. Sasaki, K. Rikino, and K. Seki, "Top-down pass-transistor logic design," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 6, pp. 792–803, Jun 1996.

[9] G. P. R. Reddy, J. Ghosh, A. P. C. R. Mandal, and B. B. Bhattacharya, "Power-delay efficient technology mapping of bdd-based circuits using dcvspg cells," in *2008 3rd International Design and Test Workshop*, Dec 2008, pp. 123–128.

[10] C. Scholl and B. Becker, "On the generation of multiplexer circuits for pass transistor logic," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537)*, 2000, pp. 372–378.

[11] V. Bertacco, S. Minato, P. Verplaetse, L. Benini, and G. D. Micheli, "Decision diagrams and pass transistor logic synthesis," *Technical Report - Departments of Electrical Engineering and Computer Science, Stanford University*, no. CSL-TR-97-748, Dec 1997.

[12] R. Chaudhry, T. H. Liu, A. Aziz, and J. L. Burns, "Area-oriented synthesis for pass-transistor logic," in *Computer Design: VLSI in Computers and Processors, 1998. ICCD '98. Proceedings. International Conference on*, Oct 1998, pp. 160–167.

[13] G. Gristede and W. Hwang, "A comparison of dual-rail pass transistor logic families in 1.5v, 0.18um cmos technology for low power applications," in *GLSVLSI-2000*, 2000, pp. 101–106.

[14] F.-S. Lai and W. Hwang, "Design and implementation of differential cascode voltage switch with pass-gate (dcvspg) logic for high-performance digital systems," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 4, pp. 563–573, Apr 1997.

[15] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8-ns cmos 16 times;16-b multiplier using complementary pass-transistor logic," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 388–395, Apr 1990.

[16] R. Zimmermann and R. Gupta, "Low-power logic styles : Cmos vs cpl," in *Solid-State Circuits Conference, 1996. ESSCIRC '96. Proceedings of the 22nd European*, Sept 1996, pp. 112–115.

[17] A. Rauchenecker and T. Ostermann, "Examination of different adder structures concerning di/dt in a 180nm technology," in *Electromagnetic Compatibility of Integrated Circuits (EMC Compo), 2015 10th International Workshop on the*, Nov 2015, pp. 103–108.

[18] M. Song, G. Rang, S. Kim, and B. Kang, "Design methodology for high speed and low power digital circuits with energy economized pass-transistor logic (eepl)," in *Solid-State Circuits Conference, 1996. ESSCIRC '96. Proceedings of the 22nd European*, Sept 1996, pp. 120–123.

[19] F. Somenzi, "Cudd: Cu decision diagram package," *http://bessie.colorado.edu/ fabio/CUDD*.

[20] A. Mishchenko, M. Case, R. Brayton, and S. Jang, "Scalable and scalably-verifiable sequential synthesis," in *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, Nov 2008, pp. 234–241.