Automated synthesis of netlists for ternary-valued n-ary logic functions in CNTFET circuits

Halvor Nybø Risto Steven Bos Henning Gundersen

Ternary Research Group, Department of Science and Industry Systems, University of South-Eastern Norway, Norway, {henning.gundersen}@usn.no

Abstract

This paper is an investigation of automated netlist synthesis for ternary-valued n-ary logic functions, based on a static ternary gate design methodology. We present an open-source C++ implementation, which outputs a ready-to-simulate SPICE subcircuit netlist file for ternary-valued n-ary function circuits. A circuit schematic of the 3-operand carry is demonstrated as synthesized by the netlist generator. We investigate a holistic (non-compound) approach to designing balanced full-adders by using 3-operand func-tions as compared to a traditional 2-operand compound design methodology. Three gate-level design approaches (compound, non-compound and hybrid) for the balanced full-adder have been simulated in HSPICE and are com-pared to each other and the state-of-the-art with simulation results. Furthermore, we propose to standardize the ternary functions by indexing them. This indexing system allows for the convenience of referencing any possible logic func-tion with no ambiguity. This indexing is necessary as most ternary functions do not have semantic names (e.g. AND, OR) and the amount of unique 3-valued functions grows exponentially with higher arity.

Keywords: ternary, netlist, synthesis, simulation

1 Introduction

In recent years, along with developments of the carbon nanotube field-effect transistor (CNTFET), there have been a handful of papers on the designs and synthesis of ternary or 3-valued logic gates implemented in simulations of CNTFET circuits. One paper in particular (Kim et al., 2018) proposes a design method for ternary logic gates, with the use of pull-up and pull-down networks constructed from a truth table for the circuit. In ternary logic, with only two operands, there are 19683 possible logic gates. With three operands, 7.6e12 logic gates are possible. The process of designing the circuit and writing the netlists of these circuits can be a tedious process, especially for circuits with more than two operands. Therefore, an open-source netlist synthesizer is of much use. The study (Lee et al., 2019) reports to have automated this process, however their code is not open-source.

2 Function Indexing

To unambiguously refer to any of the many logic functions, we propose a simple indexing system.

2.1 Range of index in arities

The number of possible functions for a specific arity and radix can be calculated as in Equation 1, where R is the radix and A is the arity.

$$F_{range} = R^{R^A} \tag{1}$$

Table 1. Range of functions in arities and radices

Arity	Radix 2	Radix 3
1	$2^{2^1} = 4$	$3^{3^1} = 27$
2	$2^{2^2} = 16$	$3^{3^2} = 19683$
3	$2^{2^3} = 256$	$3^{3^3} = 7{,}625{,}597{,}484{,}987$

While in binary, there are few enough functions that naming the useful functions (AND, OR, XOR, etc.) has been a feasible practice, for ternary logic the quantity of possible functions is more unwieldy, as can be seen in Table 1. Therefore, an indexing system is proposed to refer to specific ternary-radix functions.

The indexing system maps every truth table to an index by counting up from 0 to the function range, along with the values of the truth table. As an example, a function always outputting the low value would be the 0th index. Then, the first row of the truth table acts as the three lowest-significance trits of the index, and so on. With a truth table listed vertically, the output values for a specific function can be read in ternary as the function index.

2.2 Heptavintimal index encoding

We adopt the usage of the base-27 heptavintimal notation for ternary values (Jones, 2012), as it conveniently covers three ternary digits (trits) per symbol, as shown in Table 2. As one operand can have one of three values, the truth table of a function is three trits long in each dimension. Therefore, a logic function index can conveniently be encoded with the heptavintimal notation.

Table 2. The heptavintimal notation

Weight(Decimal)	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Ternary	000	001	002	010	011	012	020	021	022	100	101	102	110	111
Heptavintimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D

Weight(Decimal)	14	15	16	17	18	19	20	21	22	23	24	25	26	
Ternary	112	120	121	122	200	201	202	210	211	212	220	221	222	
Heptavintimal	E	F	G	H	K	M	N	P	R	T	V	X	Z	

3 Methodology

Based on the static ternary gate design methodology of (Kim et al., 2018), we have implemented an algorithm in C++, which produces a ready-to-simulate SPICE subcircuit netlist file from a circuit truth table.

The program takes an n-dimensional truth table, and constructs the four truth tables for the pull-up and pull-down networks. Then, for each network, a set of n-dimensional rectangular groupings are found. Each of these groupings will provide a transistor path to the output within each pull-up and pull-down network.

3.1 Usage

To use the program, compile the open-source code in a C++ compiler. The netlists will be generated in the same file directory as the compiled program. When the program starts, it asks for the function arity, and the values of each element in the three-by-three n-dimensional truth table, with values low(0), middle(1), high(2), don't care(x). The filename will be generated as the function index of the specific function. The transistor parameters, as well as which CNTFET model is being used, can be specified with the string variables p0, p1, p2, n0, n1, n2.

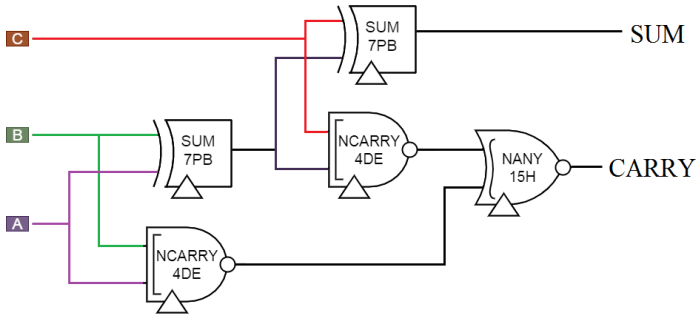
The program will produce a subcircuit which must be connected externally to a 0.9V voltage supply, and the operand inputs. The circuits rely on external 2-transistor Positive Ternary Inverters (PTI) and Negative Ternary Inverters (NTI) to achieve the four different transistor operations detailed in (Kim et al., 2018).

3.2 Logic minimization algorithm

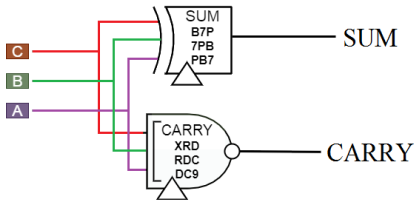
The logic minimization done to produce an optimized circuit is similar to karnaugh-mapping. The grouping algorithm takes the truth tables for each transistor network and draws n-dimensional rectangular groupings which covers every '1' on the truth table for each network, with as few groupings as possible. These groupings represent the transistor-paths in the circuit towards the output within each of the four transistor networks. Each transistor in series narrows down the throughput, until the logical rectangle of a grouping is achieved.

4 Circuit schematics

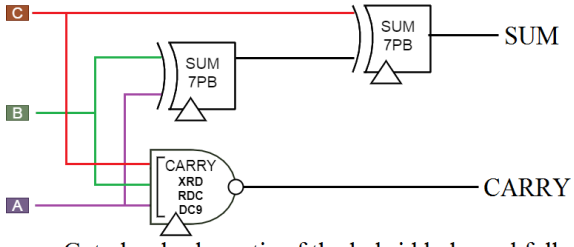
The common procedure for constructing functions with more than two operands is to combine smaller functions to create bigger compound functions. However, circuits with more operands can also be generated with our program. Therefore we investigate the usage of 3-operand functions in a 1-trit balanced full-adder circuit, in the form of a non-compound and a hybrid gate architecture. These architectures are a more holistic view of the function as a relation between input and output. Figure 1 shows three different balanced full-adder circuit design approaches.



Gate-level schematic of the compound balanced full-adder



Gate-level schematic of the non-compound balanced full-adder



Gate-level schematic of the hybrid balanced full-adder

Figure 1. Three approaches for a balanced full-adder

Figure 2 shows the circuit schematic for the 3-carry circuit used in the hybrid 1-trit full-adder, with diameters 1.487 *nm* and 1.018 *nm* being depicted as blue and green respectively.

Table 3. Simulation results with 2fF load capacitor

Circuit	Transistors	Avg. power 500MHz	Avg. power 50MHz	Worst Delay	PDP 500MHz	PDP 50MHz
2-sum (7PB)	40 (32)	0.61μW	0.35μW	530ps	0.327e-15 J	0.185e-15 J
2-ncarry (4DE)	10	0.80μW	0.80μW	20ps	0.016e-15 J	0.016e-15 J
2-nany (15H)	18	0.33μW	0.32μW	40ps	0.013e-15 J	0.013e-15 J
3-sum (B7P7PBPB7)	150 (138)	0.56μW	0.34μW	1530ps	0.856e-15 J	0.526e-15 J
3-carry (XRDRDCDC9)	50 (38)	0.87μW	0.82μW	30ps	0.026e-15 J	0.024e-15
(Lee et al., 2019) Unbalanced FA	106	(no data reported)	0.47μW	0.89ns	(no data reported)	**0.421e-15 J**
(Vudadha et al., 2018) Unbalanced FA	98	(no data reported)	**0.43**μW*	1.57ns*	(no data reported)	0.667e-15 J*
Proposed Compound Balanced FA	118 (102)	2.73μW	2.29μW	**0.55ns**	1.44e-15 J	1.262e-15 J
Proposed Non-compound Balanced FA	188 (176)	**1.67μW**	1.18μW	1.53ns	2.55e-15 J	1.805e-15 J
Proposed Hybrid Balanced FA	118 (102)	1.96μW	1.50μW	0.56ns	**1.10e-15 J**	0.840e-15 J

* see (Lee et al., 2019)

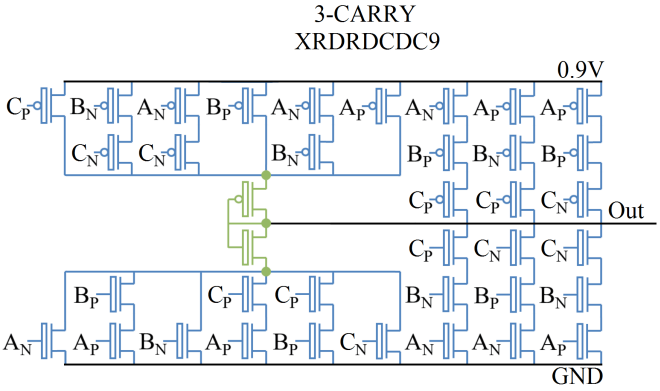


Figure 2. The balanced 3-operand carry function

5 Simulation results

The simulations were done in HSPICE, with the standard 32nm CNFET model technology from Stanford University. (Deng and Wong, 2007)

For the sake of these simulations, voltages below 200mV is considered "low", 250mV to 650mV is "middle", and above 700mV is "high".

Table 3 shows simulation results for some balanced functions, and compares three different circuit concepts of a balanced full-adder. The average current is measured at 500MHz and 50MHz. All measurements include the external PTI and NTI inverters of 2 transistors each where they are required. The transistor count is shown with and without the external inverters. A capacitive load of 2fF was put on the output to ground.

6 Discussion

The runtime performance of the synthesizer can be further optimized for > 7 arity. Under that condition circuit solutions can be found in reasonable time on standard hardware. Due to the sheer number of possible functions, only a minority of the circuit solutions were tested. However, all the tests produced the correct output values.

It should be possible to optimize the circuit solutions even further as we found by manual inspection. This is especially true for circuits with high arity functions. It is interesting to see that the performance of a 1-trit balanced ternary full-adder compared is comparable to an unbalanced version, commonly found in literature.

7 Conclusion

For up to 7 operands, a circuit of any 1-output ternary-valued function can be produced. We show that 3-operand functions can be implemented in circuits such as a 1-trit balanced full-adder, and may in some cases outperform a traditional 2-operand design strategy, as the hybrid full-adder was shown to outperform the compound full-adder in terms of power-delay-product (PDP) performance.

This paper has provided three contributions:

1. An open-source implementation for synthesis of n-ary ternary-valued CNTFET circuits (Risto, 2020).

2. An indexing system has been proposed which allows for any possible ternary-valued logic function to be referenced unambiguously.

3. A novel 3 operand, classical 2 operand, and a hybrid 1-trit balanced full-adder circuits have been simulated and compared with simulation results.

References

J. Deng and H. . P. Wong. A compact spice model for carbon-nanotube field-effect transistors including nonidealities and its application—part i: Model of the intrinsic channel region. *IEEE Transactions on Electron Devices*, 54(12):3186–3194, 2007.

Douglas W. Jones. The Ternary Manifesto · Heptavintimal encoding of ternary values, 2012. URL `http://homepage.divms.uiowa.edu/~jones/ternary/hept.shtml`.

S. Kim, T. Lim, and S. Kang. An optimal gate design for the synthesis of ternary logic circuits. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 476–481, 2018.

S. Lee, S. Kim, and S. Kang. Ternary logic synthesis with modified quine-mccluskey algorithm. In *2019 IEEE 49th International Symposium on Multiple-Valued Logic (ISMVL)*, pages 158–163, 2019.

Halvor Nybø Risto. Automated synthesis of netlists for ternary-valued n-ary logic functions in cntfet circuits, September 2020. URL `https://doi.org/10.5281/zenodo.4015574`.

C. Vudadha, A. Surya, S. Agrawal, and M. B. Srinivas. Synthesis of ternary logic circuits using 2:1 multiplexers. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65 (12):4313–4325, 2018.