An Improved Pass Transistor Synthesis Method for Low Power, High Speed CMOS Circuits

Tudor Vinereanu
National Microelectronics Research Centre
University College
Cork, Ireland
tudorv@nmrc.ucc.ie

Sverre Lidholm
National Microelectronics Research Centre
University College
Cork, Ireland
lidholm@nmrc.ucc.ie

ABSTRACT

A synthesis method for generating hybrid pass gate circuits is presented. These circuits combine features from both complementary CMOS and pass gates architectures. The simulation results using a 0.7 μm technology show that circuits synthesized according to the proposed method may achieve significant improvements in terms of area, power and delay over traditional full swing pass transistor logic and complementary CMOS.

1. INTRODUCTION

New logic families using pass transistor circuits have been proposed in the recent years, using solely NMOS logic networks [7, 3, 6] or using both NMOS and PMOS logic networks [5, 2]. Avoiding PMOS transistors is attractive, but NMOS-only circuits suffer from the threshold voltage drop at the output, hence the need for a level restorer. This has a negative impact on the performance, especially when lowering the supply voltage [8]. On the other hand, the circuits using both NMOS and PMOS logic networks provide full swing outputs, and are more robust with respect to voltage and transistor scaling. A synthesis method for Pass Gates Logic (PGL) circuits was developed in [4], using a modified Karnaugh map minimization procedure. The aim of this work has been to improve this synthesis method, such as to give a further reduction in area, power consumption and delay.

2. PASS TRANSISTOR LOGIC STYLES

The general structure of a pass transistor network is shown in fig. 1. The source side of transistor networks is connected to variable input signals instead of constants (VDD or GND). The variables associated with the input signals connected to pass transistors' sources are called pass variables. The variables associated with the signals driving the gates of the transistors are called control variables. For a full swing output, the pass transistor network consists of

ISLPED, '00 Rapallo, Italy

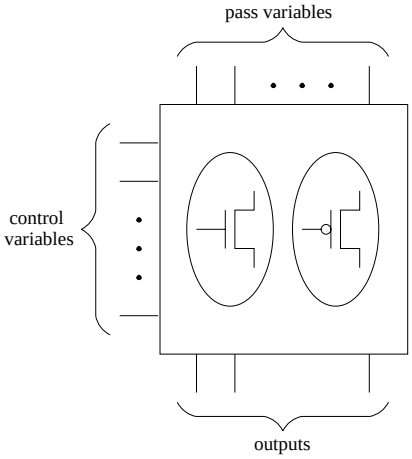

Figure 1: General structure of a pass transistor network

both NMOS and PMOS logic networks, thus avoiding the threshold drop due to the exclusive use of NMOS transistors.

Double Pass Transistor Logic (DPL) was proposed in [5] as a dual-rail pass transistor logic style. An example of basic logic gates implemented in DPL is given in fig. 2(a). However, the high transistor count, especially large PMOS transistors, and the requirement for complementary signals are drawbacks of DPL, making it not so attractive in many cases.

The Dual Value Logic (DVL) was obtained from DPL in [2], by elimination of redundant branches and signal rearrangement. It demonstrates the reduction in the number of transistors and interconnections, while preserving or improving the speed. An example of DVL logic gates is given in fig 2(b). To the best of our knowledge, a thorough comparison between DVL and DPL with respect to power consumption has not been made. However, one may assume that DVL is more power-efficient than DPL for the same reasons it is faster: smaller transistor count and fewer interconnections.

Even though DVL and PGL were developed following different paths, there are no major differences between them. Practically, they differ in that any PMOS transistor branch is doubled by a complementary NMOS transistors branch

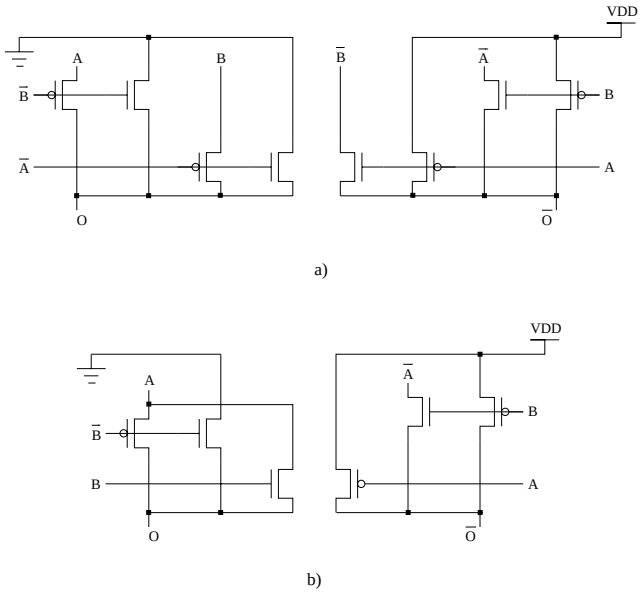

Figure 2: Example of AND/NAND gate implementation in a) DPL, b) DVL

in DVL, while PGL uses a single PMOS transistors branch when passing a logic "1". The complementary NMOS transistor branch in DVL increases the speed of the corresponding PMOS transistors branch, but in the same time has a negative effect on the power consumption of the circuit.

The speed degradation due to a PMOS-only branch driving a logic "1", as in PGL, can be reduced if one would merge that branch with a pass gate branch, such as the first will take advantage of the speed of the latter. This is made possible by choice of pass implicants, so it is achieved as part of pass transistor synthesis. The advantage of using only PMOS transistors to drive a logic "1" is reduction in area and power consumption of the circuit. An example of branch merging is given in fig. 3. Here, branch merging not only reduces the transistor count, but it also improves the speed of the circuit when the PMOS-only branch is driving the output.

In this paper, we will compare our implementations with PGL circuits, which we consider as being performant compared to the other full swing pass transistor logic styles.

3. IMPROVED SYNTHESIS METHOD FOR PASS TRANSISTOR LOGIC

The Karnaugh map minimization procedure in [4] relies on the use of pass implicants. If V_i is the pass variable and P_i is the product term, $P_i(V_i)$ is called the pass implicant, where V_i is in the set [0, 1, X_i, $\bar{X}_i$], with X_i being any input variable. The pass function is determined by covering every cell in the Karnaugh map at least once, and it consists of pass implicants.One may notice that the conventional Karnaugh map minimization method for CMOS circuits is a particular case of the method described in [4]. Indeed, if only pass implicants whose pass variables are either 0 or 1 are considered, the resulting circuit is but a conventional CMOS circuit.

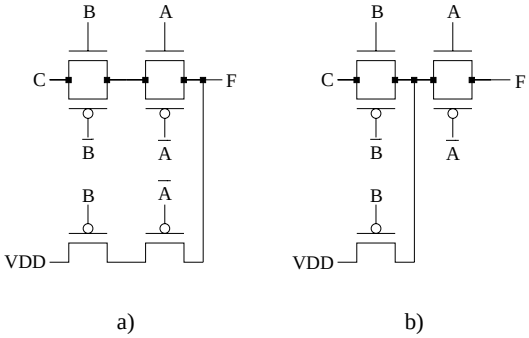

Figure 3: Branch merging for pass transistor networks: a) before merging, b) after merging

Our proposed method relies on the algorithm in [4], but differs with respect to choice of prime implicants and implementation with pass transistor branches. It uses overlapped implicants and eliminate redundancies in order to reduce transistor count and the number of transistors in series, aiming better performance in terms of speed and power over the aforementioned method of synthesis.

For conciseness and clarity reasons, we will use throughout the rest of this paper a set of naming conventions as described below.

A homogenous pass implicant is an implicant who covers only 0's or only 1's in the Karnaugh map (its pass variable is 0 or 1 respectively, hence it connects the output to either GND or VDD respectively). To implement a homogenous pass implicant only a single pass transistors branch is required (made of NMOS or PMOS transistors).

A non-homogenous pass implicant is an implicant who covers both 0's and 1's (its pass variable is a literal). For its implementation both NMOS and PMOS transistors branches are required.

An N-implicant is a non-homogenous pass implicant who covers both 0's and 1's in the Karnaugh map, but all the 1's were previously covered by other implicants. This doesn't necessarily mean that none of the 0's are previously covered, the only requirement is that at least one minterm was left uncovered by previously chosen implicants. Its pass variable is a literal, but a single branch made of NMOS transistors is required to implement it. The reason behind this is that only "good" 0's have to be passed by this implicant, since passing the "good" 1's is already taken care of by previously chosen implicants. So, a redundant PMOS transistors branch is eliminated. Same definition for P-implicants, for whom all 0's were already covered, and is implemented by a single PMOS transistors branch. Note that N- and P-implicants can only be defined during the minimization process, after the Karnaugh map has been partly covered.

It is worth noticing that N- and P-implicants require only a single transistor branch to implement, same as for the homogenous implicants. However, homogenous implicants pass a constant, either 0 or 1, so they don't add extra capacitive load to the input signal associated to a pass variable.

The pass variable of an N- or P-implicant is a literal, hence it increases the capacitive load on the input signal connected to the source side of the pass transistor branch. Hence, an homogenous pass implicant would be preferred over an N- or P-implicant to be selected in a pass function.

The synthesis rules are based on the rules proposed in [4]. The modified set of rules are described as follows:

1. Each and every cell has to be covered at least once.
2. To cover a cell which has not been covered before, always use the highest order pass implicant (a pass implicant is of order i if it covers 2^i cells, where $i = 2, 3, ...$)
3. Use a prioritized search for pass implicants: first find the homogenous implicants, then N-implicants, P-implicants and non-homogenous implicants, in this order.
   (a) For homogenous implicants passing a logic "0" and N-implicants only an NMOS transistor branch is needed. The control signals for the branch are the signals associated with the literals in the product term.
   (b) For homogenous implicants passing a logic "1" and P-implicants only a PMOS transistor branch is needed. The control signals for the transistors in the branch are the signals associated with the complements of the literals in the product term.
   (c) For non-homogenous pass implicants, both NMOS and PMOS branches are required.

The focus of this approach is on the priority order when selecting pass implicants. Pass implicants who require a single transistor branch to implement (homogenous, N- and P-implicants) are preferred, so the resulting circuit will have a reduced transistor count. It is rather the type of implicant than the number of previously uncovered cells it contains or the number of implicants in the minimized function which is important. The possibility of using these implicant types depends on the logic function to be minimized, hence the improvement in performance over conventional method depends on the function as well. It is fair to say that in cases

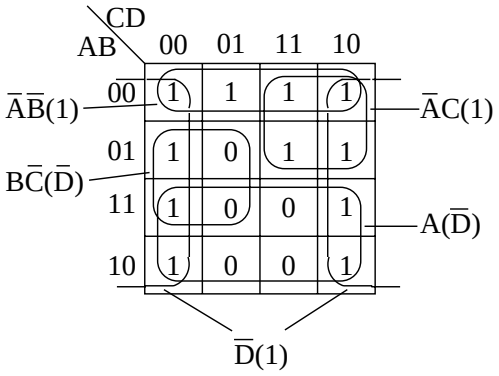

Figure 4: Example of Karnaugh map minimization using the proposed synthesis method

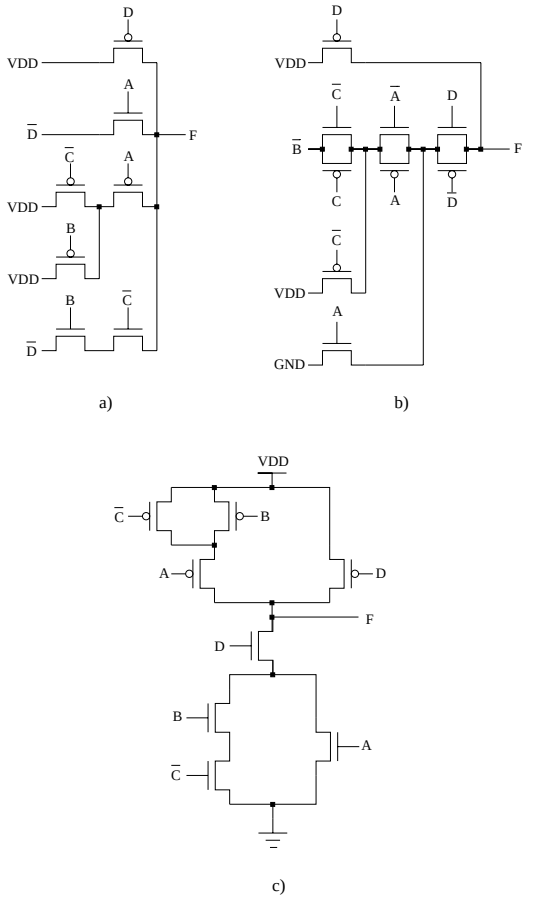

Figure 5: Function $F = \bar{D} + \bar{A}\bar{B} + \bar{A}C$ synthesized in a) HPGL, b) PGL, c) CMOS

when no N- and P-implicants can be identified, the resulting circuit is identical to the PGL version (e.g. sum generation for a 1-bit full adder) or CMOS version (e.g. carry generation for a 1-bit full adder).

An example of how the synthesis method works is shown for the function $F = \bar{D} + \bar{A}\bar{B} + \bar{A}C$ in fig. 4. The highest order implicants in the example are $\bar{D}(1)$ and $A(\bar{D})$, of order $i = 3$. First choose the homogenous implicant $\bar{D}(1)$, and then choose $A(\bar{D})$ as an N-implicant (all the 1's it covers are already passed correctly to the output by $\bar{D}(1)$). Notice that $A(\bar{D})$ can be considered an N-implicant only after $\bar{D}(1)$ was selected as part of the minimized function. To cover the remaining 4 cells in the Karnaugh map we will use order $i = 2$ implicants: homogenous implicants $\bar{A}\bar{B}(1)$ and $\bar{A}C(1)$, and N-implicant $B\bar{C}(\bar{D})$. The minimized pass function thus obtained is $F = \bar{D}(1) + A(\bar{D})_N + \bar{A}\bar{B}(1) + \bar{A}C(1) + B\bar{C}(\bar{D})_N$, where N designates an N-implicant.

An implementation of the minimized pass function requires 4 PMOS and 3 NMOS transistors, see fig. 5(a). We will refer to the logic circuits synthesized using the synthesis guidelines given in this paper as HPGL (Hybrid Pass Gate Logic). Using the conventional synthesis rules gives the pass function for the PGL implementation as $F = \bar{D}(1) + AD(0) + \bar{A}\bar{C}D(\bar{B}) + \bar{A}CD(1)$, which requires 5 PMOS and 4 NMOS

transistors, see fig. 5(b). A DVL implementation would require two more NMOS transistors (in parallel with the PMOS-only branches). A CMOS implementation is shown in fig. 5(c), using 4 PMOS + 4 NMOS transistors.

A consequence of reducing the transistor count in HPGL is the reduction in power consumption, but in the same time the circuit may suffer from speed degradation compared to PGL. This is one reason to cover the cells in the Karnaugh map with implicants of the highest possible order. By doing this, the number of transistors in series required to implement a pass transistor branch can be reduced, thus compensating for the speed degradation. This is especially true because of the quadratic delay characteristic of pass transistor branches.

Apart from the transistor count, we may also consider the input signals required for a particular implementation. For instance, in fig. 5, the CMOS implementation needs A, B, $\bar{C}$, D as input signals, while the PGL implementation requires A, $\bar{A}$, $\bar{B}$, C, $\bar{C}$, D, $\bar{D}$. The requirement for complementary input signals is one of the often mentioned drawbacks of pass transistor logic, which translates at physical level into a larger number of interconnects and inverters. The large number of input signals required for PGL implementations could be explained by looking at the way non-homogenous pass implicants are implemented. Both NMOS and PMOS transistors are required, and the control signals for the PMOS branch are the complements of those for the NMOS branch. In the case of CMOS implementations, there are the same signals driving the NMOS transistors and their PMOS counterparts, usually leading to a smaller number of inputs.

However, the HPGL implementation in fig. 5(a) requires only A, B, $\bar{C}$, D, $\bar{D}$, which is a smaller number of input signals than PGL requires, but larger than in the case of CMOS. The reason for a reduced number of input signals of HPGL compared to PGL is the priority order set for implicants. We consider first the homogenous implicants, then N- and P-implicants, whose implementation requires only a single transistor branch, thus no need for complementary signals. Non-homogenous implicants are taken into account only if there are still uncovered cells after homogenous, N- and P-implicants were considered. Avoiding non-homogenous implicants in the pass function, especially the lower order implicants, usually leads to a smaller number of inputs.

The HPGL and CMOS implementations have some similarities. See figures 5(a) and 5(c). Their PMOS transistors networks are identical, while the NMOS transistors networks of HPGL could be obtained from CMOS by eliminating the transistor driven by D and connecting $\bar{D}$ to the source of the resulting NMOS branch. However, this cannot be regarded as a rule and it strictly depends on the logic function to be minimized. To illustrate this, the implementation of function $F = \bar{A}C + A\bar{B}D + A\bar{C}D$ in HPGL, PGL and CMOS is shown in fig. 6. Here, the HPGL implementation is made of pass gates and single branches, thus combining features derived from PGL with features derived from CMOS. The resulting circuit has a reduced transistor count (4 PMOS + 5 NMOS compared to 5 PMOS + 5 NMOS for PGL and 6 PMOS + 6 NMOS for CMOS), and the same number of inputs as CMOS (6 compared with 8 for PGL). The synthesis method presented in this paper generates a circuit which is a hybrid between PGL and CMOS.

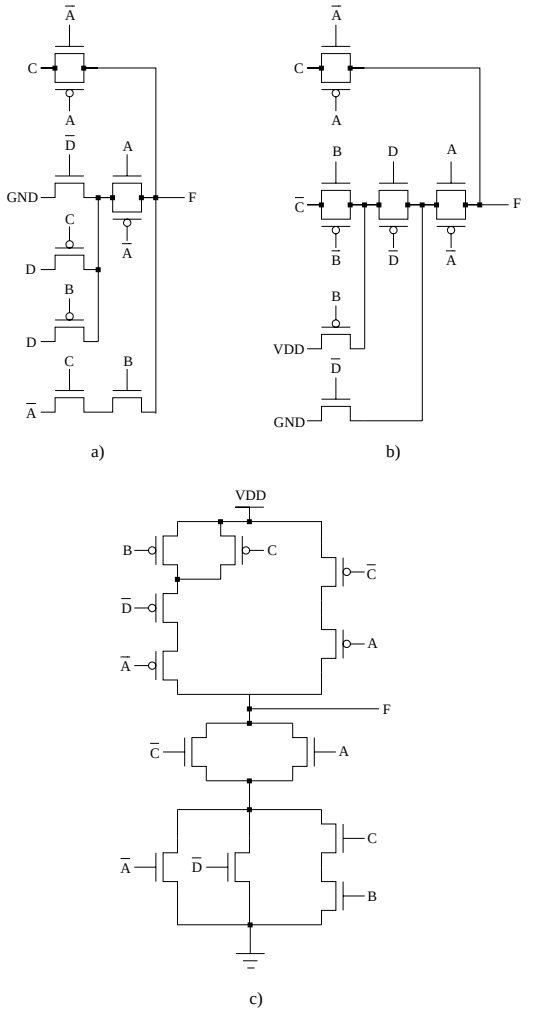

Figure 6: Function $F = \bar{A}C + A\bar{B}D + A\bar{C}D$ synthesized in a) HPGL, b) PGL, c) CMOS

4. SIMULATION RESULTS

The simulations were performed using HSPICE and a standard 0.7 μm process technology. A layout-based netlist similar to the one described in [1] was used. For all transistors the diffusion capacitances were estimated with respect to the layout, considering the situations when there are shared diffusions and contacts. An exception was made for the input inverters when measuring the power consumption, no diffusion capacitances being taken into account in this case, for a higher accurancy of measurements. Transistors were sized manually for lowest power-delay product (PDP), and an equal optimization effort was considered for all logic styles. Note that by transistor sizing, power and delay can be traded off considerably, so the circuit performance can be adjusted further for high speed or low power. The size reported represents the total widths of transistors. The number of inputs required are shown separately.

Table 1: Simulation results for $F = \bar{D} + \bar{A}\bar{B} + \bar{A}C$ (fig. 5)

Logic Style	Delay [ns] 5V	Delay [ns] 2V	Power [μW] 5V	Power [μW] 2V	PDP [fJ] 5V	PDP [fJ] 2V	No. of transistors	Size [W_{min}]	No. of inputs
PGL	0.34 (1.13)	1.93 (1.51)	325 (1.44)	55 (1.49)	110 (1.64)	106 (2.26)	9	20.5	7
CMOS	0.36 (1.20)	1.52 (1.19)	234 (1.04)	36 (0.97)	84 (1.25)	54 (1.15)	8	18.5	4
HPGL	0.30 (1.00)	1.28 (1.00)	226 (1.00)	37 (1.00)	67 (1.00)	47 (1.00)	7	15.5	5

Table 2: Simulation results for $F = \bar{A}C + A\bar{B}D + A\bar{C}D$ (fig. 6)

Logic Style	Delay [ns] 5V	Delay [ns] 2V	Power [μW] 5V	Power [μW] 2V	PDP [fJ] 5V	PDP [fJ] 2V	No. of transistors	Size [W_{min}]	No. of inputs
PGL	0.36 (1.03)	2.01 (1.29)	367 (1.34)	62 (1.44)	132 (1.43)	124 (1.85)	10	21.5	8
CMOS	0.56 (1.60)	2.31 (1.48)	273 (1.03)	43 (0.96)	152 (1.65)	103 (1.54)	12	23	6
HPGL	0.35 (1.00)	1.56 (1.00)	265 (1.00)	45 (1.00)	92 (1.00)	67 (1.00)	9	12.5	6

All possible input transitions combinations were simulated, and the worst case delay and the average power dissipation were obtained from simulation. The input vectors frequency was 100MHz, and the simulations were performed at 5V and 2V supply voltage.

The simulation results of the circuits in fig. 5 and fig. 6 are presented in table 1 and table 2, respectively. The numbers in paranthesis are the normalized values for delay, power consumption and power-delay product, with respect to HPGL. For the considered circuits, HPGL implementations have the smallest delay and the smallest power-delay product, at both supply voltages. HPGL circuits have also the smallest power consumption at 5V, but when the supply voltage is lowered down to 2V, CMOS performs slightly better. However, because HPGL circuits are the fastest at both supply voltages, delay can be traded off for power by transistor sizing if power consumption is the main issue for a specific application.

When compared to CMOS, both PGL and HPGL performance gets worse with supply voltage reduction. However, HPGL implementations considered are less sensitive to supply voltage downscaling than PGL circuits.

5. CONCLUSIONS

In this paper, an improved synthesis method based on Karnaugh map minimization for pass transistor logic was presented. It uses N- and P-implicants and a priority order for selecting pass implicants to eliminate redundancies and reduce transistor count. It was shown that important savings in area, power and delay could be achieved over PGL and CMOS. Reductions in delay, power consumption and power-delay product of up to 35%, 33% and 56% respectively were demonstrated when compared to PGL at 2V supply voltage. Improvements in delay and power-delay product over CMOS of up to 32% and 35% respectively were obtained at the same supply voltage. Similar results were obtained at 5V supply voltage. The efficiency of the synthesis method is directly related to the logic function to be minimized, and the resulting circuit becomes a hybrid between PGL and CMOS.

The synthesis method presented in this paper was demonstrated for logic functions with small number of inputs, being based on Karnaugh map minimization. Work is currently ongoing towards integrating the synthesis method presented here in a Binary Decision Diagram (BDD)-based [6] synthesis environment.

6. REFERENCES

[1] F. Mu and C. Svensson. A layout-based schematic method for very high-speed cmos cell design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):144–148, March 1999.

[2] V. G. Oklobdzija and B. Duchene. Pass-transistor dual logic value for low-power cmos. In *Proceedings of the 1995 International Symposium on VLSI Technology, Taipei, Taiwan*, pages 341–344, May 31-June 2 1995.

[3] A. Parameswar, H. Hara, and T. Sakurai. A swing restored pass-transistor logic-based multiply and accumulate circuit for multimedia applications. *IEEE Journal of Solid-State Circuits*, 31(6):804–809, June 1996.

[4] D. Radhakrishnan, S. R. Whitaker, and G. K. Maki. Formal design procedures for pass transistor switching circuits. *IEEE Journal of Solid-State Circuits*, SC-20(2):531–536, April 1985.

[5] M. Suzuki, N. Ohkubo, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome. A 1.5-ns 32-b cmos alu in double pass-transistor logic. *IEEE Journal of Solid-State Circuits*, 28(11):1145–1151, November 1993.

[6] K. Yano, Y. Sasaki, K. Rikino, and K. Seki. Top-down pass-transistor logic design. *IEEE Journal of Solid-State Circuits*, 31(6):792–803, June 1996.

[7] K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu. A 3.8-ns cmos 16x16-b multiplier using complementary pass-transistor logic. *IEEE Journal of Solid-State Circuits*, 25(2):388–395, April 1990.

[8] R. Zimmermann and W. Fichtner. Low-power logic styles: Cmos versus pass-transistor logic. *IEEE Journal of Solid-State Circuits*, 32(7):1079–1090, July 1997.