

Chapter 4

Differential and Pass-Transistor CMOS Digital Circuits

Oklobdzija, Yano

Introduction

Computational and market demands have driven VLSI processors to double their performance every two to three years. The same trend has been observed in the computer market represented by the X'86 architecture, as shown in Fig.1. The clock frequencies have reached 3.8GHz [28]. This demand for higher speed has been exercising a significant pressure on the circuit design and the circuit design style. Keeping this rate of performance increase is not possible only through the advances in fabrication technology. Therefore the improvements in all the other aspects of the design are necessary to support the rate of this progress.

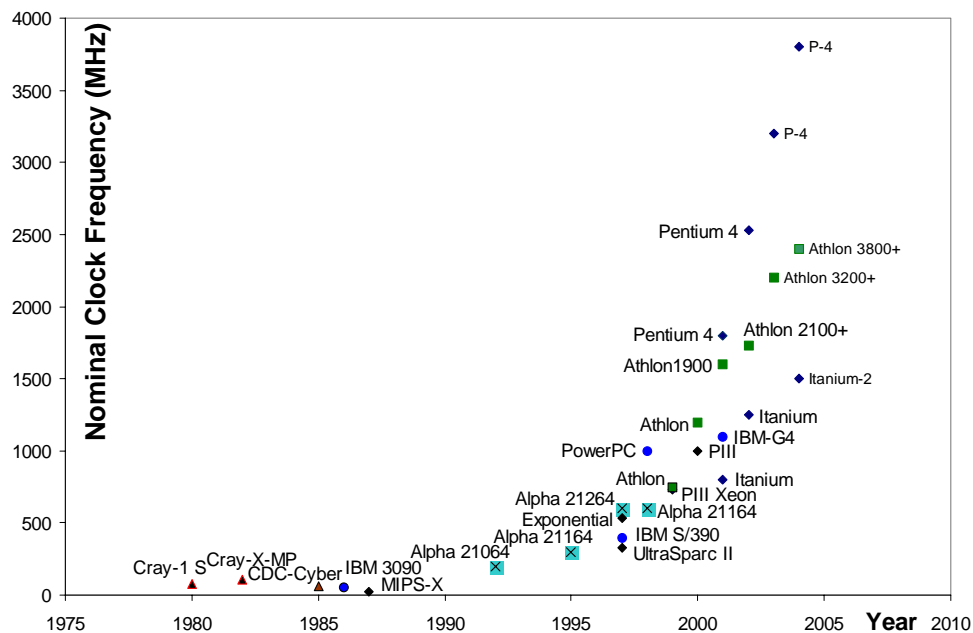


Fig. 1. Microprocessor clock frequency increase over years (source: MDR)

As the technology reaches into the deep sub-micron region, the use of regular CMOS is coming to its limits. The problems associated with the power and speed required that the other types of logic be invented and the old ones re-examined. In order to reach the required performance goals, the use of dynamic logic in the critical paths of a processor is quite common, if not standard. Every high-performance processor today uses non-conventional CMOS circuits such as: Domino logic and Skew-Tolerant Domino, (also known as Opportunistic Time Borrowing) as well as pass-transistor circuits []. The circuits in the critical part of a high-performance processor are so important that it is essential for the leading processor design centers today to have a very skilled circuit design team. The interaction between the circuit and the architecture team in those organizations is so close that the logic design stage is eliminated for those parts of the processor or confined to verification [].

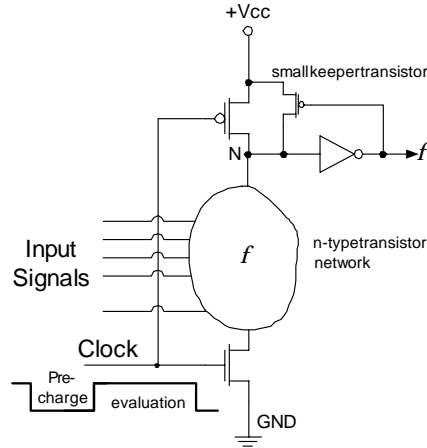
The use of pass-transistor logic is regaining interest because it may behave better than the ordinary CMOS as the power-supply voltage scales down. It was not only the performance gain over the conventional CMOS that is of interest, but the energy-delay product yielding the optimal design point. The power has increasingly becoming more important than speed. This is an issue as the processor has been migrating into the consumer market, portable and hand-held devices in particular.

Differential Logic

The introduction of differential CMOS logic evolved from the development of dynamic CMOS, “Domino Logic” in particular. A significant effort was undertaken in the early 1980s while exploring the circuit families that are to be replacement for nMOS logic []. This development took place within IBM and AT&T Bell Laboratories and resulted in several new circuit and logic configurations.

CVS Logic

Cascode Voltage Switch Logic (CVSL) was developed in IBM [1] as an improvement over the use of pseudo n-MOS. It comes in two forms: single-output and differential output (or double-rail). The later form of the logic is also known as DCVSL (Differential Cascode Voltage Switch Logic).



CVS IBM

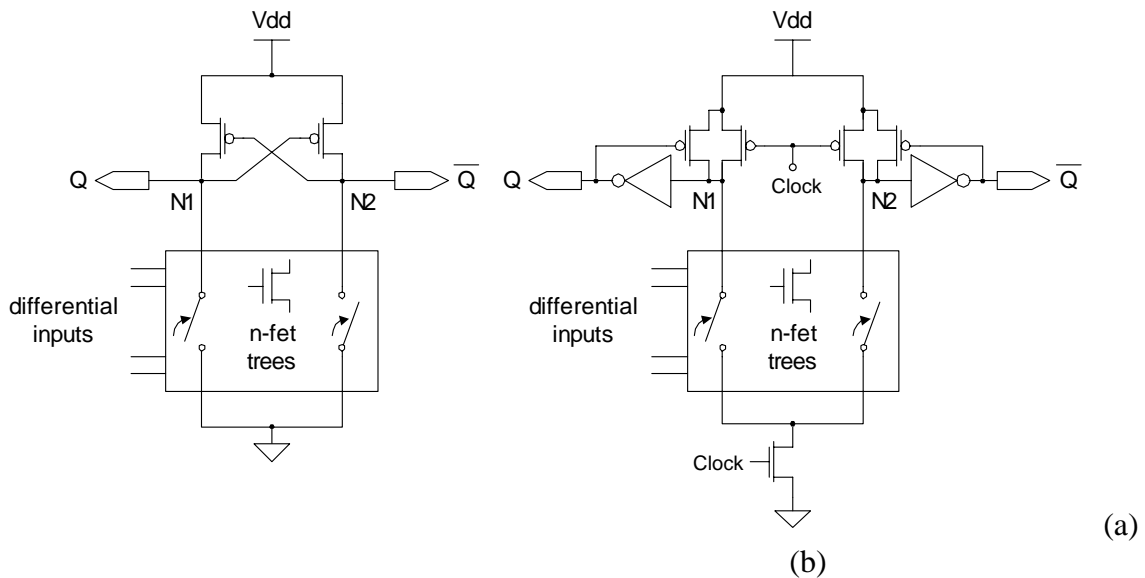


Fig. 2. Differential Cascode Voltage Switch Logic (DCVSL): (a) Static DCVLS (b) Dynamic DCVSL

DCVSL is made of two n-type switching networks, one implementing \bar{f} and the other f , and of two p-type transistors, connected in a cross-coupled combination to V_{dd} , used as pull-up devices (Fig.2.a,b). Depending on the state of the differential inputs, either node N1 or N2 is pulled down by one of the nMOS logic tree (but never both). The regenerative action of the pMOS latches keeps the outputs Q and \bar{Q} static and assures the full voltage swing, V_{dd} or ground, of its outputs.

The two logic trees are capable of processing complex functions within a single circuit delay. The advantage of DCVSL logic is that both polarities of the output are represented, thus inversion operation is not necessary. This eliminates the need for the inverter and makes this type of logic inherently faster. The presence of both polarities of

the output has other advantages as well. If the circuit is operating correctly, the values of the output signals can only assume 0-1 or 1-0, i.e. the 0-0 or 1-1 combination can never occur. This gives this logic “self-checking” properties. If one of the forbidden combinations is detected by a circuit with self-checking properties (known as self-checking checker), a failure is immediately signaled. Depending on the system logic, the system could switch to “fail” state and take the appropriate action.

Another variation of CVSL is Static and Dynamic CVSL circuit. Dynamic logic is available in two forms: single-ended (single-output) and double-ended (where true and complements of the function are present). Dynamic DCVSL is shown in Fig. 2.b.

One the problem of static DCVSL is the signal asymmetry which can appear during the transition. Given that the pMOS transistors are the only pull-up devices there may be a time window during which both the pMOS and the nMOS are ON. This situation will create a current from V_{dd} to ground node causing current spikes and additional delay. The choice of the size of the pMOS is thus very important. If the pMOS is made too small the transition of the signal from GND to V_{dd} is too slow. If on the other hand, the pMOS devices are made too big the transition of the output node from V_{dd} to ground is too slow. This makes static CVS to be a “ratioed” logic. In general to assure a good “pull-up” of the output signal the pMOS devices should be twice the size of the nMOS devices. There is no direct current from V_{dd} to ground after the transition, however because of the asymmetry of the circuit the power consumption of CVSL can be high.

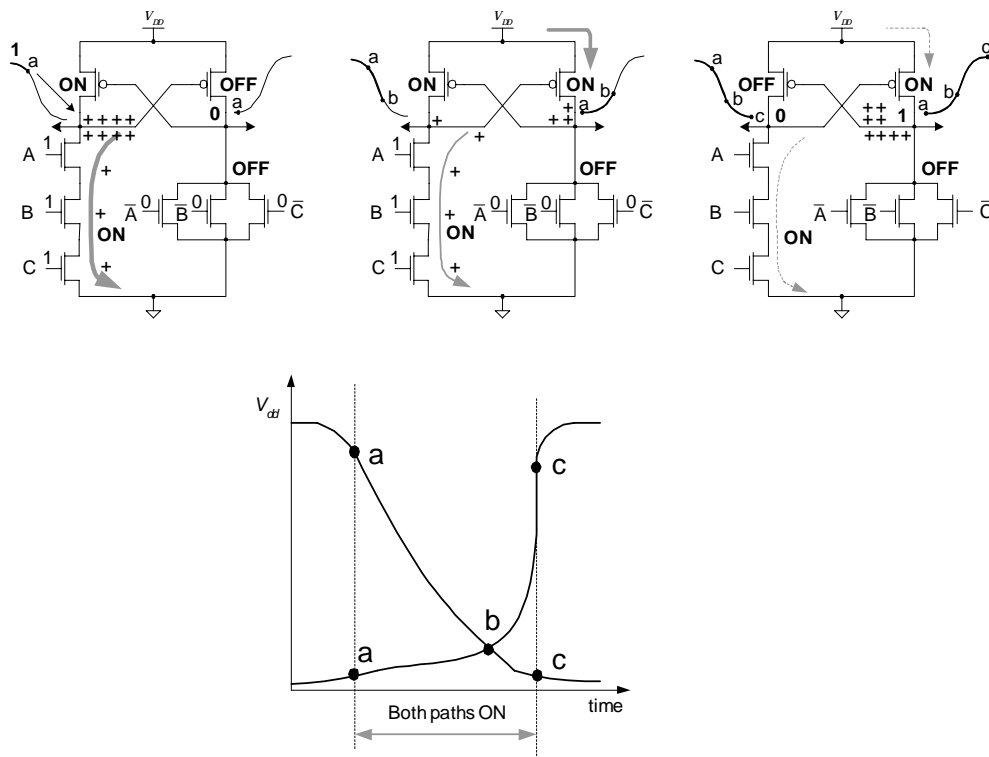


Fig. 3. Switching asymmetry in CVSL. This asymmetry causes current spikes and increased power consumption.

CVSL versus CMOS

The main difference between CMOS and DCVSL is in the way the switching function is implemented. While both CMOS and DCVSL implement the true function and its complement, DCVSL uses only n-type devices for both switching trees whereas CMOS use p-type for the f tree and n-type for \bar{f} tree, as shown in Fig. 4.

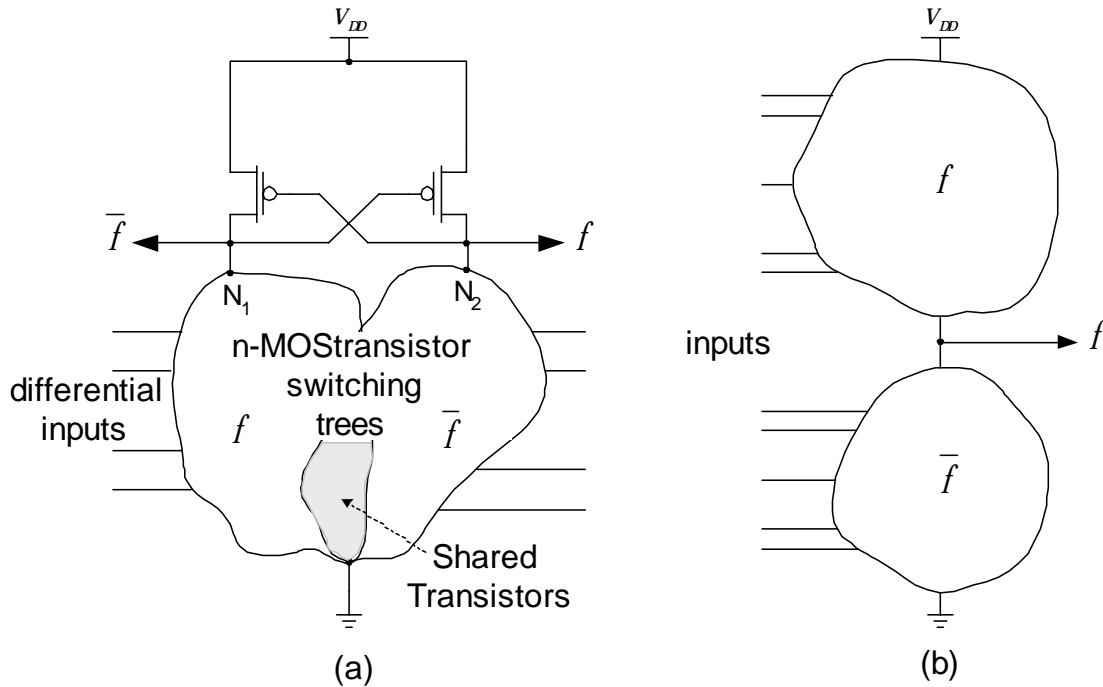


Fig. 4. (a) DCVSL Logic consisting of two shared nMOS transistor switching networks (b) CMOS consisting of two separate: nMOS and pMOS transistor switching networks.

In terms of the area this allows CVSL to be smaller than CMOS. Since the carrier mobility in the pMOS transistor is half of that in the nMOS, the pMOS transistors need to be made twice as large. Therefore the area taken to implement the switching network representing the function f is usually twice as large as compared to the switching network representing \bar{f} . In DCVSL those switching networks are approximately the same. It is also possible to share transistors between the f and \bar{f} switching networks for further reduction of the total number of transistors, Fig.4. In addition to the area reduction, the use of nMOS transistors results in a reduced input capacitance thus contributing to the speed of the circuit. DCVSL uses both polarities of all input signals, thus doubling the number of wires. However, due to its differential nature, both output polarities are available, eliminating the need for an inverter. Inversion is achieved without delay penalty, by simply choosing the appropriate polarity of the output signal. This feature also helps to improve speed of DCVSL logic as compared to CMOS.

In general, due to the lower input capacitance and a better intrinsic transistor speed CVSL is faster as compared to CMOS using the same transistor sizes. In studies done by IBM, CVSL has shown an overall performance improvement [1]. Other studies [2], show an improvement of performance but at the cost of increased power consumption.

In terms of the number of transistors, CVSL uses two extra pMOS transistors in the cross-coupled combination, as compared to CMOS. However the implementation of both functions f and \bar{f} doesn't necessarily mean that duplication of the transistors is necessary given that a number of transistors are usually shared between f and \bar{f} switching trees. The amount of such overlap is dependent on the function. Thus the number of transistors in CVSL is generally the same or lower as compared to CMOS.

The sharing of the transistors is illustrated in the example of a 3 input XOR gate shown in Fig. 5.

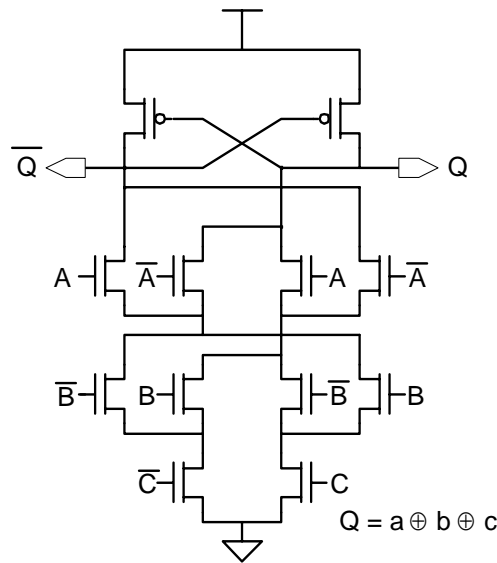


Fig. 5. Example of transistor sharing in DCVS Logic: Implementation of 3-input XOR function

Pass-Transistor Logic

With the technology moving into the sub-micron region, the new circuit structures were rapidly introduced trying to overcome the limitations of the regular CMOS. The driving force was an ever increasing requirement for performance and higher level of integration. This commanded that other types of logic be examined. An extensive consideration was given to the use of pass-transistors which saw its resurrection from the early days of VLSI. Several experimental prototypes were built to demonstrate the feasibility and advantages of the new circuit topologies.

One of the first papers examining pass-transistor logic and formalizing pass-transistor design style was published by Whitaker, in 1983 [Whitaker, Electronics]. The paper describes how the logic functions can be efficiently and optimally built using pass-transistor logic. It shows some fundamental pass transistor building blocks which became very popular in VLSI design practices. A methodology for synthesis of pass-transistor functions was established presenting a modified Karnaugh map which employs the pass variables, not only logic zeroes and ones. A formal design procedure for pass transistor switching circuits was developed later in [Whitaker, Jo SSC]. This work was further extended and formalized in [Markovic, Oklobdzija, Microelectronics].

For example let us analyze a simple pass-transistor circuit shown in Fig. 6.a. We define the variable connected to the gate of the pass-transistor, thus controlling its output as the “*control variable*”, while the variable connected to its source (or drain) as “*pass variable*”. The two branches are OR-ed together by wiring them together, which is known as “*wired-OR*”. The *pass variable* can propagate to the output only if this is allowed by the *control variable*. Thus, we have at our disposal a basic circuit topology capable of implementing a Boolean function, and AND-OR structure. The structure that is shown in Fig. 6.a. implements an XOR function. We could have derived it from the Karnaugh map shown in Fig. 6.b, by simply choosing A to be the *control variable* and then realizing that the desired output is equal to B when A is not true and not B when A is true. Connecting B and not B to the appropriate inputs will produce desired XOR function. In the same manner, we could have generated any one of the 16 possible functions.

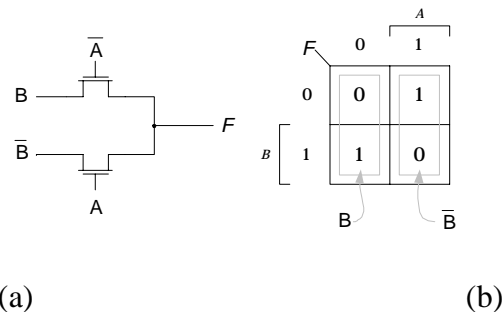


Fig. 6. (a) XOR function implemented with pass-transistor circuit, (b) Karnaugh map showing derivation of the XOR function.

We can generalize this derivation by connecting variables X and Y at the input of this simple network. The number of the functions that can be realized in this manner is equal to the 2^k where $k=2^N$, N representing the number of variables that can be formed at the input. For $N=2$, this represents 16 functions. This is illustrated in Fig. 7.a. and b. If we use variable B as a control variable instead of the variable A, we will obtain the same set of 16 functions. Given that we are using only two transistors in this structure to implement 16 possible functions, this illustrates the power of pass transistor realization. However, while such a structure was very convenient for nMOS circuits, in CMOS the threshold drop across the pass transistor represents a problem. At this point we want to explain the synthesis of pass-transistor logic, but we will return to this problem later and will show some possible solutions that work for CMOS.

If we extended this concept further, using two control variables A and B and fixed controls P_0, P_1, P_2, P_3 at the input, we can create a pretty powerful function generator as shown in Fig. 8. It is quite common to see this type of general function generator implementation in VLSI processors. The variables P_0, P_1, P_2, P_3 are come from the controller portion of the processor and they determine the function implemented by the variables A and B which are connected as control variables. This functional generator implements the same 16 possible functions of two variables A and B, but this is done in a more structured manner.

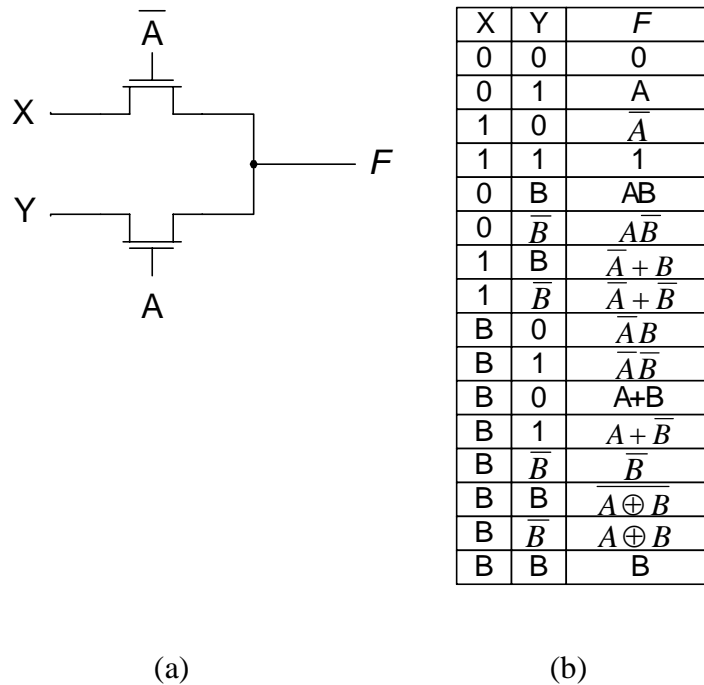


Fig 7. (a) General topology of pass-transistor function generator (b) Karnaugh map of 16 possible functions that can be realized.

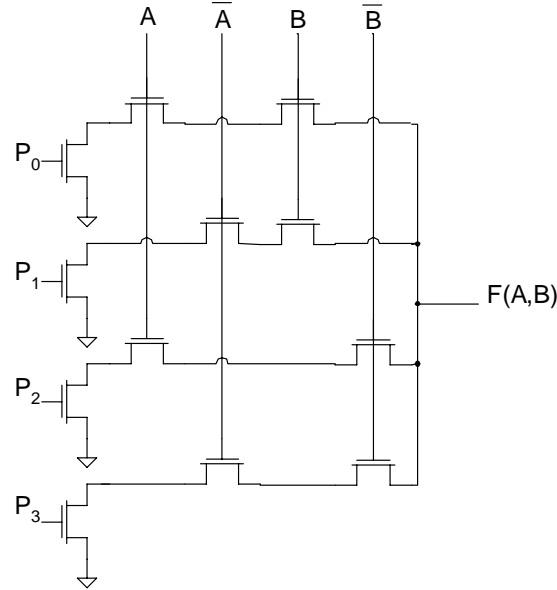


Fig 8. Function generator implemented with pass-transistor logic.

The function generator, shown in Fig. 8 is missing a pull-up portion. In nMOS, that is realized by simply connecting the output F to V_{dd} through a depletion mode nMOS transistor. In CMOS this requires duplication of the entire generator portion using pMOS transistors. It should not be difficult for the reader to figure out how. The simple grid layout style makes this implementation compact and efficient.

At this point we may address the problem of the threshold voltage drop between the input and output when the pass transistor logic is used. The logic shown in Fig. 6 represents no problem when the input is at the zero voltage level because the one applied to the gate will simply make the output voltage zero. The problem arises when the input is at the logic one level because the output can not reach one. For that purpose, let us examine the circuit in Fig. 9.

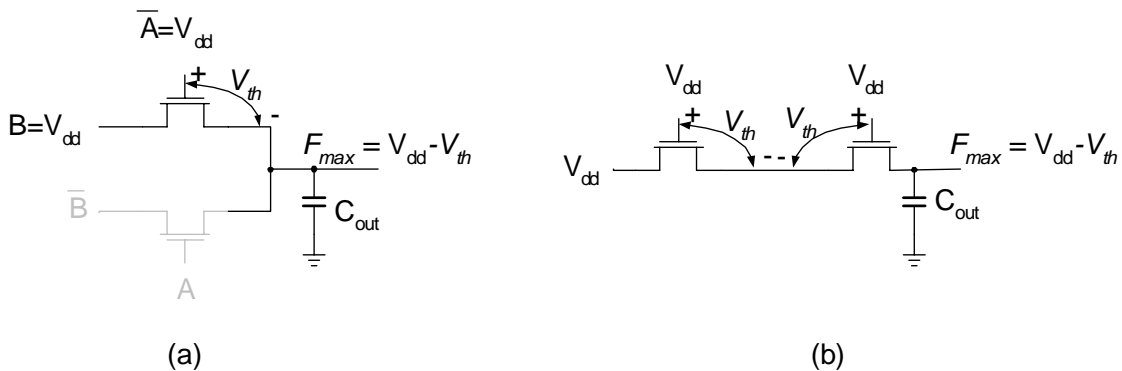


Fig 9. (a) Threshold voltage drop at the output of the pass-transistor gate (b) Voltage drop does not exceed V_{th} when there are multiple transistors in the path

When the input voltage is equal to V_{dd} the output can not reach V_{dd} simply because if both transistor terminals, source and drain, were to be at V_{dd} the voltage at the transistor gate would not be above either of the terminals and transistor would not conduct. Thus, the output capacitor will be charged through the pass-transistor until its voltage is one transistor threshold voltage V_{th} below the gate voltage, as shown in Fig. 9.a. One may naively think that when two or more transistors are connected in series the output voltage be lower for two or several transistor threshold voltages V_{th} . This thinking is not correct because source and drain in the MOS transistor are indistinguishable. The existing voltage drop caused by the first transistor in the path makes needed voltage difference for the second transistor (or third and any subsequent transistor in the chain) so that the output voltage will be lower for only one V_{th} voltage drop and equal to: $V_{dd} - V_{th}$.

While this voltage drop did not represent serious problem in nMOS logic it does so in CMOS because this can bring the next stage into the region when both transistors (p and n) are conducting while seriously degrading the noise margin of the CMOS circuit. An obvious, but more complex solution is to always pair the nMOS transistor with a pMOS which would bring the output voltage to the required full swing as shown in Fig. 10.a. Another solution is to always terminate the problematic node with a full-swing restoring inverter, as shown in Fig. 10.b.

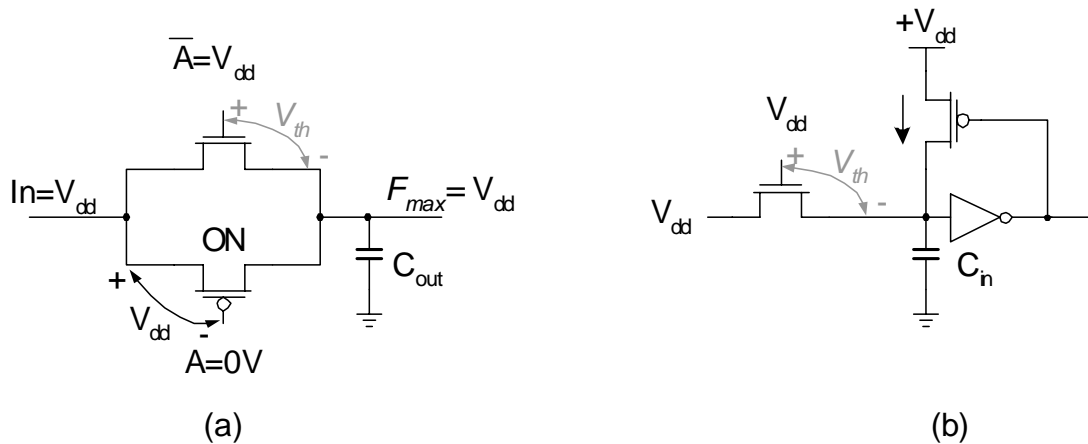


Fig 10. Elimination of the threshold voltage drop by: (a) pairing nMOS transistor with a pMOS (b) using a swing-restoring inverter

CPL and DPL Logic

New CMOS logic families using pass-transistor circuits were introduced with the objective of improving speed and power [4,6]. Two of them, simultaneously developed by Hitachi: CPL [4] and DPL [6], are the most notable. The Double Pass-Transistor Logic, developed by Hitachi in 1993 demonstrated an 1.5nS 32-bit ALU in 0.25 μm CMOS technology [4] and a 4.4nS 54X54 bit multiplier [9]. New developments followed from IBM and from Toshiba introducing DCVSL-PG [3] and SRPL [5]. Recent studies have shown that the use of pass-transistor logic not only brings speed and area improvement, but also results in lower power.

Complementary Pass-Transistor Logic (CPL)

In 1990, researchers from Hitachi Research Center in Japan published the structure known as Complementary Pass-Transistor Logic, CPL [9]. The CPL structure was significant in the fact that it brought back the pass-transistor efficiency in implementation of logic circuits. The logic function built from the pass-transistors not only efficiently utilized the space on silicon, but resulted in a very fast logic. Importantly, this type of logic also consumed less power.

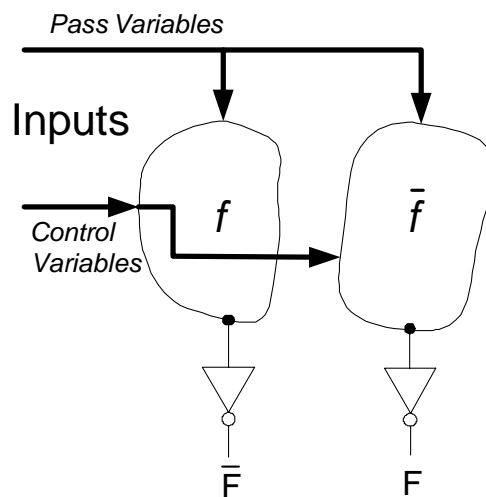


Fig.11. Complementary Pass-Transistor Logic structure

The general structure of CPL is shown in Fig.11. The given function f is implemented from two pass transistor logic blocks implementing the function f and its complement \bar{f} . The transistor logic is built from the pass-transistor networks taking advantage of pass-transistor design efficiency. Resulting logic is differential, which means that both the function f and its complement \bar{f} are present at the outputs, while requiring both polarities of the inputs, i.e. every variable is represented in its *true* and *complement* form. This does not represent a significant problem since the logic itself

produces both polarities. It is also assumed that both polarities are readily available from the registers. In an AND gate implementation, for example, a NAND output is readily available. Thus, complementation does not require additional inverters. Instead it rather consists of choosing the proper polarity of the signal.

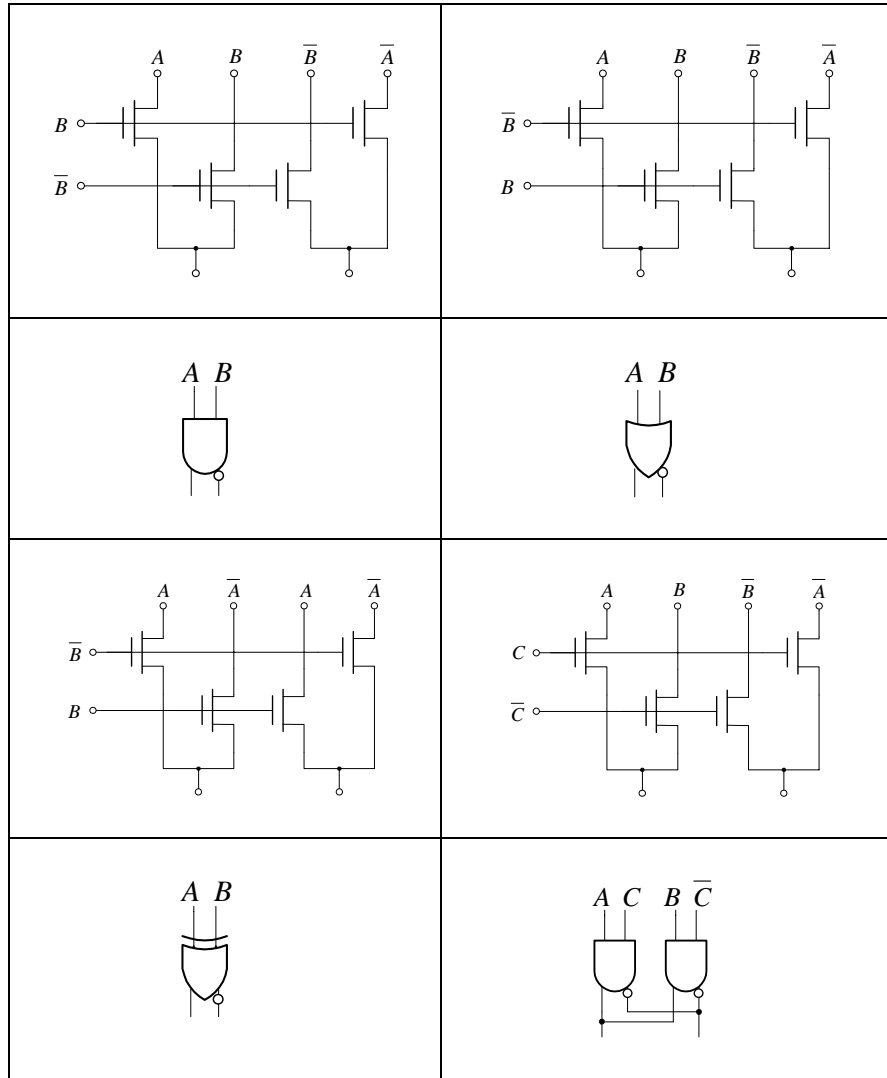


Fig.12. Basic logic functions in CPL: circuit diagram is followed by a logic symbol

CPL went one step further from CVSL because it de-coupled the p-transistor latching combination used in CVSL by replacing it with two separate inverters instead. A family of gates is implemented in this fashion including the XOR / XNOR combination as well as multiplexer as shown in Fig. 12.

A distinguished feature of CPL circuits is that the implementation of the multiplexer circuit is especially effective and fast. The same circuit topology is used to implement an XOR gate resulting in equally fast and efficient realization. This feature has much importance in digital system design given that multiplexer and XOR gates are

essential building blocks which are found in the critical paths of various components. A CPL implementation of an AND gate and XOR gate and sum bit of full-adder are shown in Fig.13.

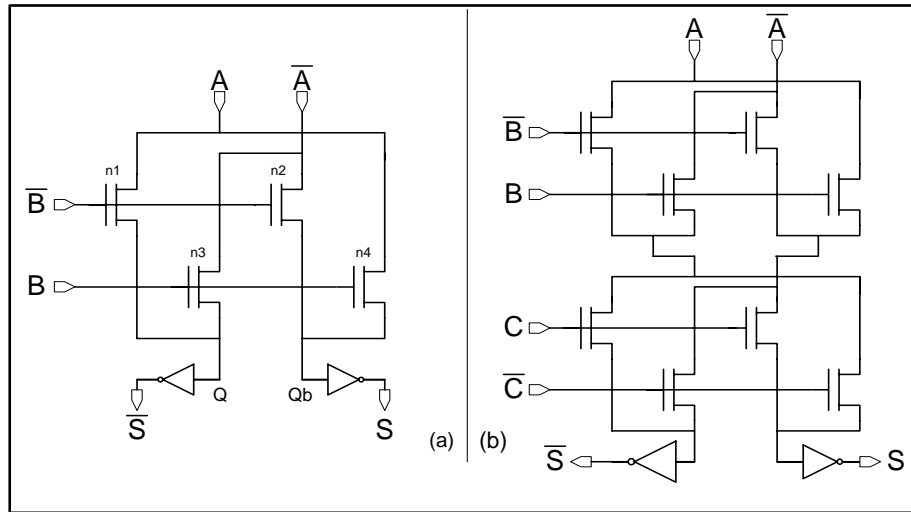


Fig.13. CPL provides an efficient implementation of XOR function: (a) XOR gate (b) Sum circuit

Using CPL was proven not only to yield efficient implementation, but also a fast logic. In double metal 0.5μ CMOS technology of its introduction it yielded an $3.8nS$ $16X16b$ multiplier, which was very fast for that time.

However, at its introduction CPL suffered from the *threshold voltage drop* problem. When passed through a series of pass-transistors, the signal voltage is degraded by one V_T (*threshold voltage drop*). This brings both transistors in the output inverter to the conducting region, causing static current to flow from V_{cc} to GND thus, resulting in static power dissipation. To alleviate this problem, two types of transistors were used: *logic transistors* (with $V_T = 0$ V) and *inverter transistors* (with $V_T = 0.4V$ and $-0.4V$). Though this reduced static power dissipation and delay time, it increased the process complexity and the sensitivity to noise. In the new version of CPL, the problem of the "threshold drop" was alleviated by using a special type of inverter which has the ability to restore the voltage level to its full potential. This inverter is shown in Fig.14.

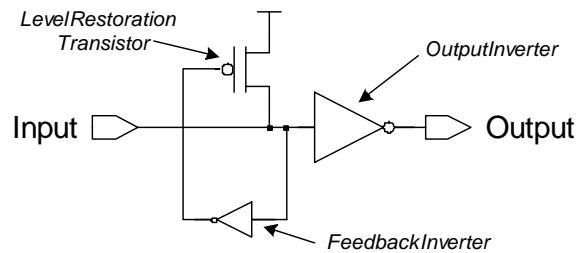


Fig.14. CPL inverter

The clever property of CPL inverter is that the level restoration transistor is connected to a separate small *Feedback Inverter*. This inverter is connected to the input of the *Output Inverter*, thus it is not sensitive to the load on the output. If this was not the case, the level restoration in circuit with the loaded output would occur at a slower rate and would be output load dependent, which is not the case here. In CPL the restoration of the signal level is independent on the load at the output, thus resulting in faster signal level restoration and decrease in power during the signal transition.

The concept of CPL was further extended into a design style associated with the tool for automatic generation of the logic block named "*Lean Integration*" [8]. The use of this design style has provided improvement in performance, power and area of the ASIC and micro-processor units. Another advancement of the CPL concept termed LEAP is described in [15].

Double Pass-transistor Logic (DPL)

Another pass-transistor logic family, developed at Hitachi Research Laboratories, is Double Pass-transistor Logic (DPL) (see Okhubo [6]). DPL was originally developed to overcome the "*threshold drop*" problems of CPL and to provide an alternative "pass-transistor logic". In creating the switching network f , DPL uses both: n-MOS and p-MOS transistors in parallel. This eliminates the problem of the "*threshold drop*" and the use of inverters after each logic block is not necessary, thus enhancing the speed of DPL. Elimination of inverters results in enhanced speed, however, buffering of the signal after every 2-3 stages is necessary.

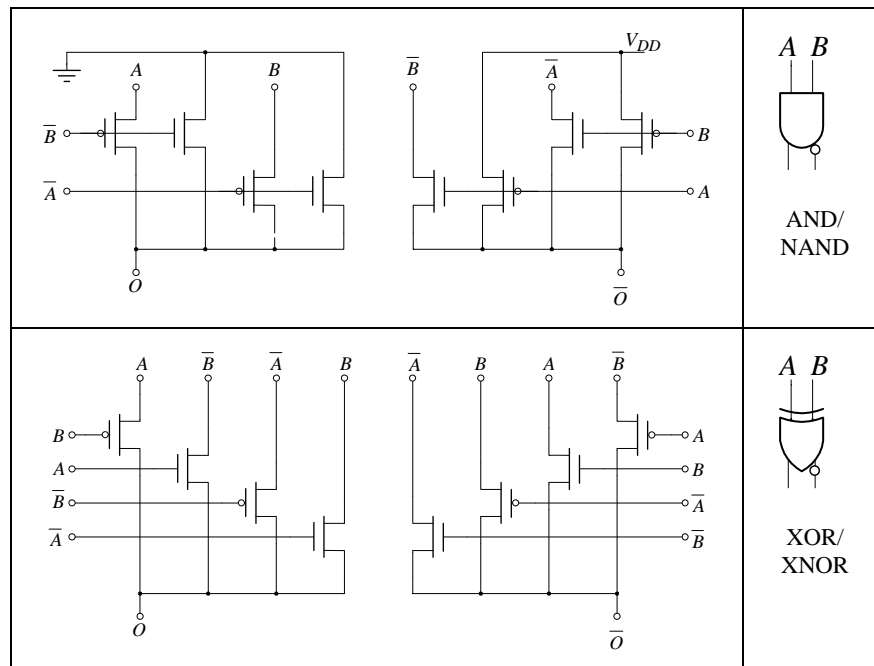


Fig.15. Double Pass-Transistor Logic (DPL): circuit implementation of basic logic functions

The two basic gates used in DPL are shown in Fig.15. The simplicity of DPL family is apparent. For this logic family to be complete it is necessary to implement only one logic function (AND/NAND) and inversion which is obtained by simply choosing an appropriate output. An efficient implementation of XOR gate was also achieved. As in CPL the basic circuit structure in DPL is a multiplexer which topology is equivalent to that of an XOR gate. However, unlike in CPL those two basic building blocks (XOR and MUX) do not necessarily have to be followed by an inverter, thus making an implementation of a pass-transistor chain possible. When the signal is propagated through several stages of pass-transistors, restoration of the signal is necessary which is achieved by inserting inverters. Unlike in CPL it is not necessary for this inverter to be of a special kind.

Hitachi has shown two very fast implementations using DPL: one a 1.5nS 32-b ALU [6] and another 4.4nS 54X54-b parallel multiplier [9], both built in 0.25u technology. DPL multiplier was the fastest 54-bit multiplier for quite some time.

An XOR and Sum bit of a DPL full-adder are shown in Fig. 16.

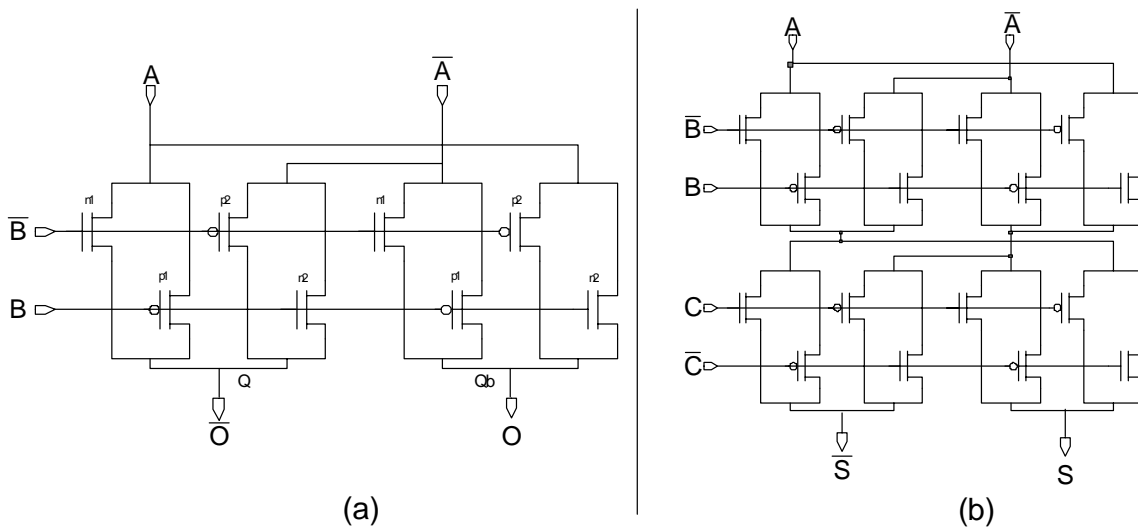


Fig.16. DPL Logic: (a) XOR (b) One bit full-adder: Sum circuit

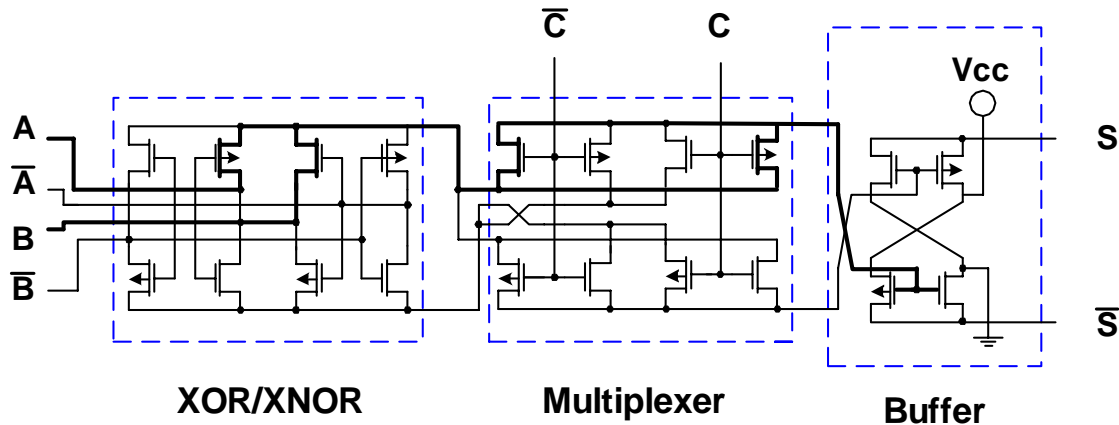


Fig. 17. DPL Full Adder. The critical path traverses two transistors only (not counting the buffer)

Extensions to CPL and DPL

An extensive consideration was given to the use of pass-transistor logic networks in the design of high-speed digital systems, [1]-[12]. However, no systematic approach in the synthesis of pass-transistor circuits exists, that takes into account the impact of circuit technology and signal arrangements on circuit performance. Traditional approach to pass-transistor circuit synthesis was based on binary decision diagrams (BDD), which uses both NMOS and PMOS devices in circuit implementation. BDD approach does not guarantee full swing of output voltage – it can be degraded by the threshold voltage of NMOS or PMOS devices for some input test vectors [6]. Another commonly used design practice was library-based design, with library cells realized as combination of simple logic gates. Therefore, more attention should be directed to optimization of pass-transistor logic gates, which is often overlooked.

In this section, we present the rules for algorithmic and systematic synthesis of complementary and dual logic functions, with emphasis on different signal arrangements. The method is based on Karnaugh-map coverage and circuit transformations as an approach to logic gate synthesis. An algorithmic method for synthesis of basic logic gates in three conventional pass-transistor techniques will be treated here.

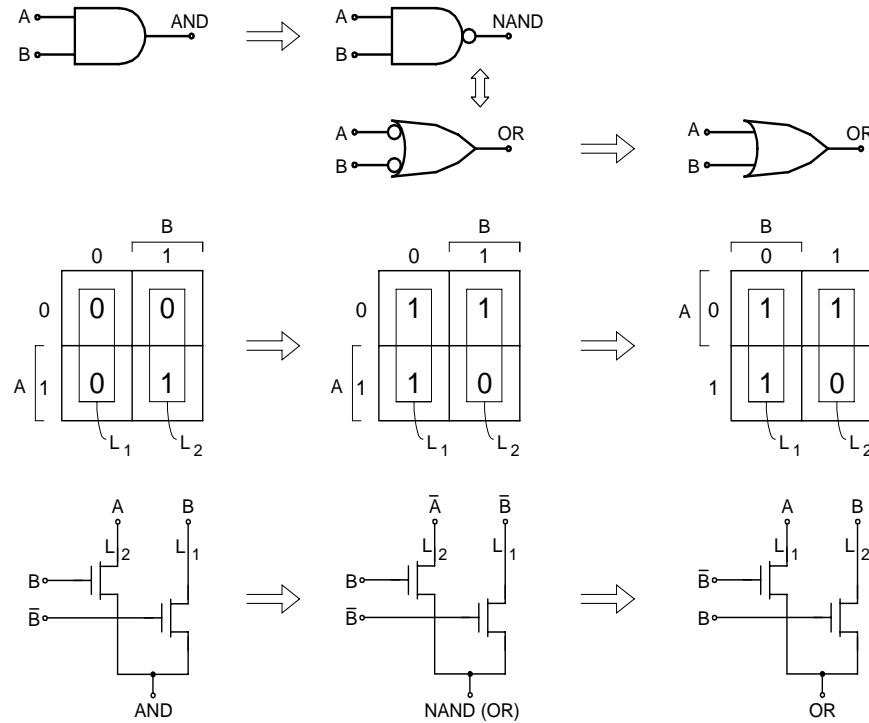


Fig. 17. Construction of dual logic function

A. CPL

Logic function can be implemented in CPL following the algorithm given below:

1. Cover the Karnaugh-map with largest possible cubes (overlapping allowed)
2. Express the value of the function in each cube in terms of input signals
3. Assign one branch of transistor(s) to each of the cubes and connect all the branches to one common node, which is the output of NMOS pass-transistor network

Complementary function can be implemented from the same circuit structure by applying complementarity principle, as given below. By applying duality principle, a dual function is synthesized.

Complementarity Principle: Using the same circuit topology, with pass signals inverted, complementary logic function is constructed in CPL.

Duality Principle: Using the same circuit topology, with gate signals inverted, dual logic function is constructed. Following pairs of basic functions are dual:

- AND-OR (and vice-versa)
- NAND-NOR (and vice-versa)
- XOR and XNOR are self-dual (dual to itself)

Proof of complementarity principle is trivial since the pass variables are directly passed from inputs to the outputs, so for obtaining complementary function an inversion of pass variables is needed.

Proof of duality principle follows from De Morgan rules and it is illustrated on the example how OR gate can be obtained from AND gate, Fig. 17.

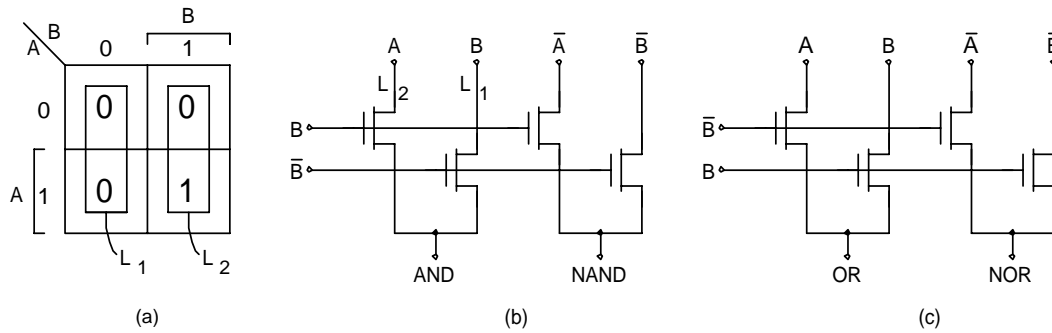


Fig. 18. Two-input CPL logic: (a) AND function Karnaugh map, (b) AND/NAND circuit, (c) OR/NOR circuit

The logic gate synthesis procedure is shown on two-input AND example, Fig. 18.a, b. All the input vectors of Fig. 18.a are covered with cubes L_1 and L_2 . The value of the function covered with cube L_1 is equal to B , which becomes pass signal at the source of transistor branch, representing L_1 . Branch L_1 is driven with \bar{B} on its gate. Therefore B is passed when \bar{B} is high. Similarly for the cube L_2 pass signal is input variable A , driven by gate signal B (A is passed when B is high). These two cubes are represented with two NMOS transistor branches, implementing the desired function. By applying complementarity principle on AND circuit we obtain NAND circuit, Fig. 18.b. Furthermore, by applying duality principle on AND we synthesize two-input OR function. NOR circuit is generated from OR (complementarity) or from NAND (duality), Fig. 18.c.

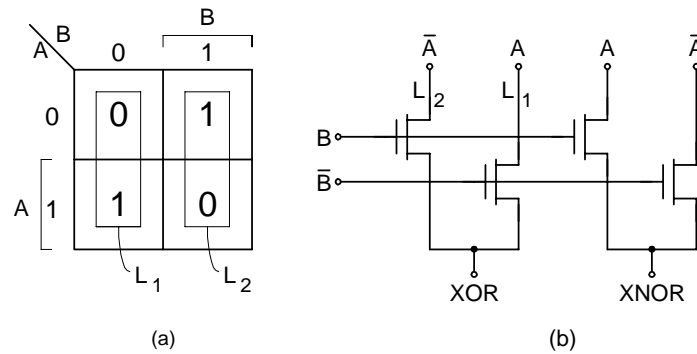


Fig. 19. Synthesis of two-input CPL logic: (a) XOR function Karnaugh map, (b) XOR/XNOR circuit

Implementation of 2-input XOR/XNOR function is shown in Fig. 19.

The synthesis of three input AND function is shown in Fig. 20. The cubes L_1 and L_3 are overlapped. It has a consequence that the corresponding branches are active for the input vectors at which the cubes are overlapped. That saves the circuit area.

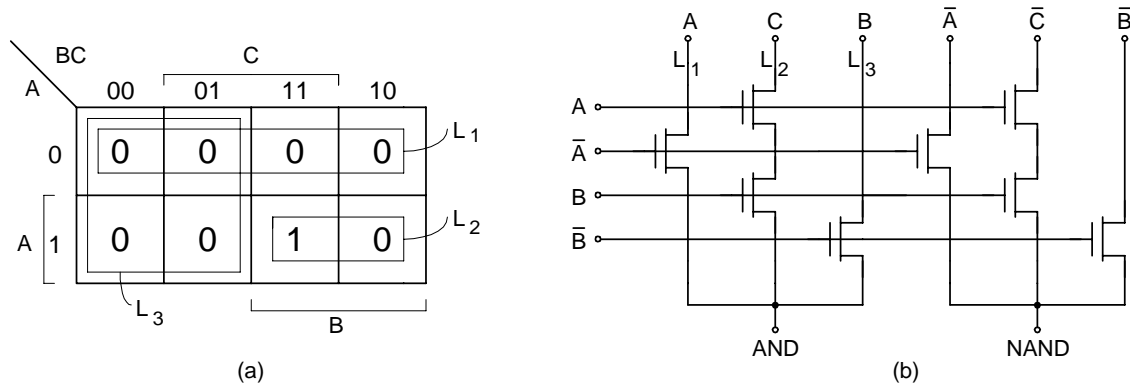


Fig. 20. Synthesis of three-input CPL logic: (a) AND function Karnaugh map, (b) AND/NAND circuit

Three-input OR/NOR circuit is shown in Fig. 21. Three-input XOR/XNOR circuit in CPL is implemented as cascade of two-input XOR/XNOR modules [7].

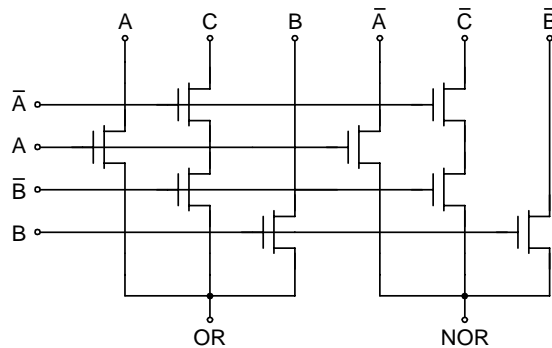


Fig. 21. Three-input CPL logic: OR/NOR circuit

B. DPL

Random logic function in DPL can be synthesized from Karnaugh-map as follows:

1. Two NMOS branches can not be overlapped covering logic 1s. Similarly, two PMOS branches can not be overlapped covering logic 0s.
2. Pass signals are expressed in terms of input signals or supply. Every input vector has to be covered with exactly two branches.

At any time, excluding transitions, exactly two transistor branches are active (any of the pairs NMOS/PMOS, NMOS/NMOS and PMOS/PMOS are possible), i.e. they both provide output current.

Complementarity Principle: Complementary logic function in DPL is generated after the following modifications:

- Exchange PMOS and NMOS devices
- Invert all pass and gate signals

Duality Principle: Dual logic function in DPL is generated when PMOS and NMOS devices are exchanged, and signals V_{DD} and GND are exchanged.

In obtaining complementary function it is necessary to invert pass signals and gate signals also, since we exchanged PMOS and NMOS devices. Proof of duality principle is similar to that in CPL.

The synthesis method is illustrated on the example of two-input AND function, Fig. 22. Cube L_1 of Fig. 22.a is represented with NMOS transistor, with the source connected to GND and the gate connected to \bar{A} . PMOS branch of L_2 passes B , when gate signal A is low etc. Complementary circuit (NAND), Fig. 22.b, is generated from AND, according to complementarity principle.

According to duality principle OR circuit is formed from AND circuit, its dual counterpart, Fig. 23.

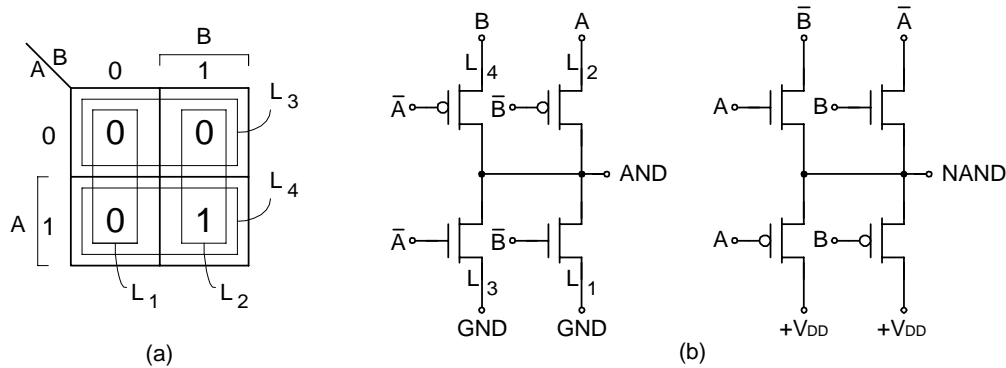


Fig. 22. Two-input DPL logic: (a) AND function Karnaugh map, (b) AND/NAND circuit

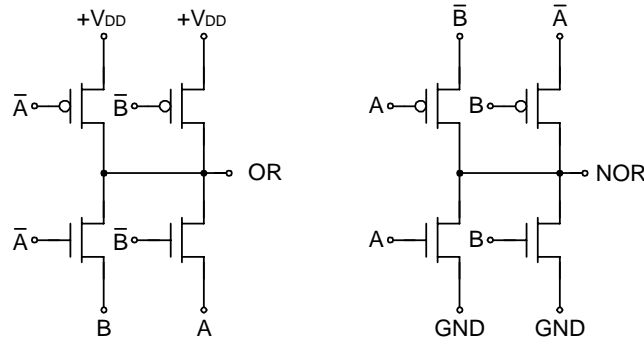


Fig. 23. Two-input DPL logic: OR/NOR circuit

Different two-input XOR/XNOR circuit arrangements are shown in Fig. 24 and Fig. 25. Mapping from Fig. 24 results in balanced load on both true and complementary input signals. But, it is also possible to cover map as shown in Fig. 25.

This mapping strategy results in different load on input signals A and B , which could be advantageous in case when switching probability of signals A and B is different.

Three-input functions in DPL are implemented as cascaded combinations of two-input DPL modules.

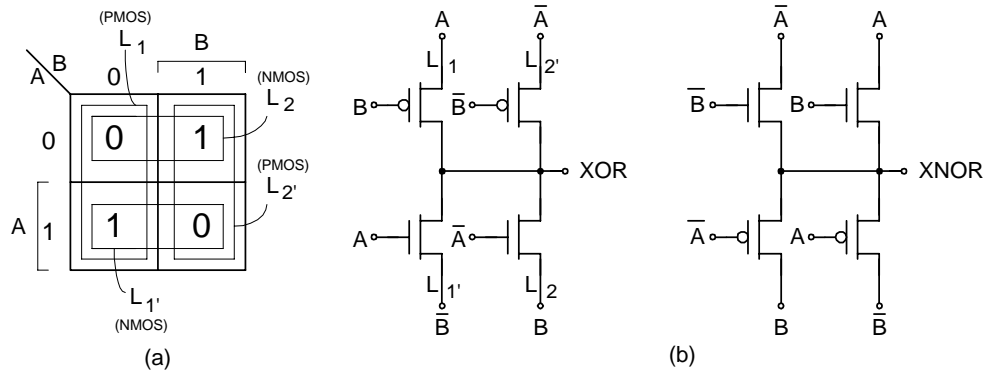


Fig. 24. Two-input DPL logic: (a) Mapping strategy I (b) XOR/XNOR circuit I

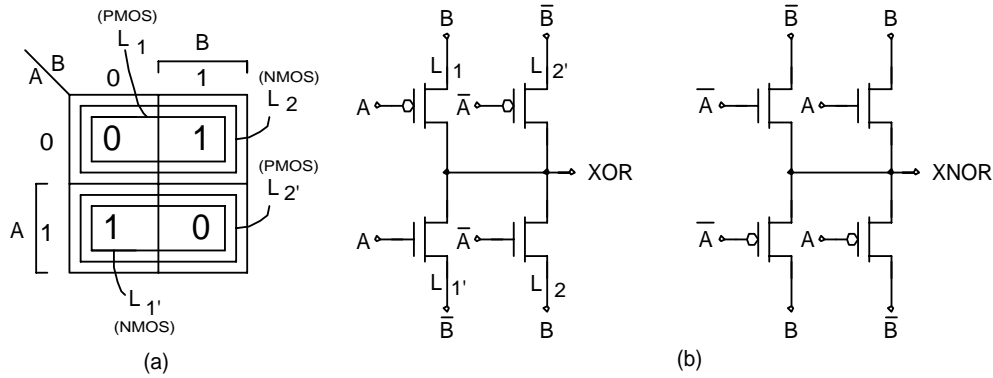


Fig. 25. Two-input DPL logic: (a) Mapping strategy II (b) XOR/XNOR circuit II

C. DVL

Random logic function in DVL can be synthesized from Karnaugh-map according to the algorithm given below:

1. Cover all input vectors, which produce 0 at the output, with largest possible cubes (overlapping allowed) and represent those cubes with NMOS devices, which sources are connected to *GND*
2. Repeat step 1 for input vectors, which produce 1 at the output, and represent those cubes with PMOS devices, which sources are connected to V_{DD}
3. Finish with mapping input vectors, not mapped in steps 1 and 2 (overlapping with cubes from steps 1 and 2 allowed), which produce 0 or 1 at the output. Represent this cubes with parallel NMOS (good pull-down) and PMOS (good pull-up) branches, which sources are connected to one of the input signals

Mapping strategy and circuit implementation in DVL is shown on three-input AND example, Fig. 26.

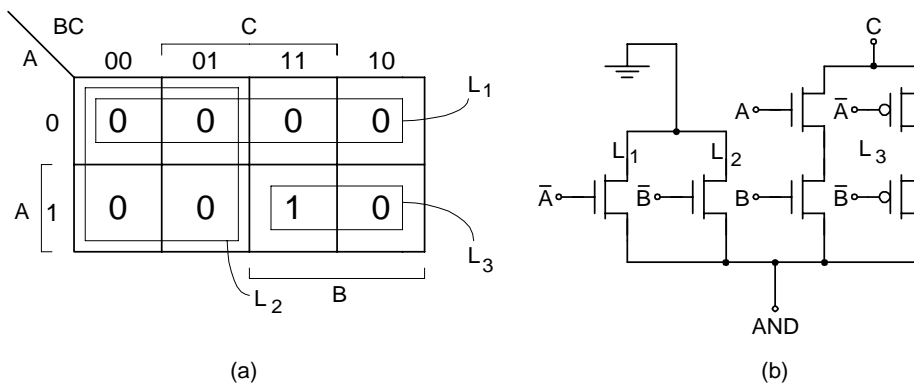


Fig. 26. Three-input DVL logic: (a) AND function Karnaugh map, (b) AND circuit

From Fig. 26 we see that by overlapping cubes L_1 and L_2 we save the area, which would be otherwise wasted on additional transistor in L_1 or L_2 . That allows wider transistors of cube L_3 . OR/NOR circuit, which is directly generated from AND circuit, is shown in Fig. 27.

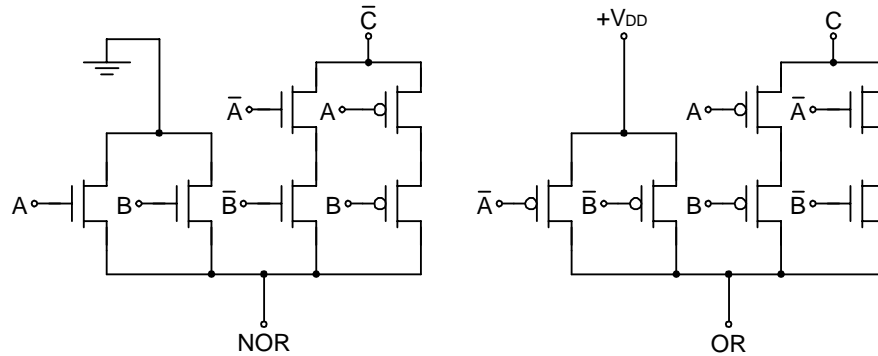


Fig. 27. Three-input DVL logic: OR/NOR circuit

Complementarity and duality principles in DVL are the same as in DPL (both are CMOS structures).

In this section general rules for synthesis of pass-transistor gates in three representative conventional pass-transistor techniques were established. From those rules an algorithmic way for generation of various circuit topologies (complementary and dual circuits), is possible. This lays the foundation for development of computer aided design (CAD) tools capable of generating fast and power-efficient pass-transistor logic.

CVSL Pass-Gate

Further enhancement of differential CMOS family was done by Lai and Hwang [ref]. They modified CVSL by using pass-variables instead of the ground connection at the opposite end of the pass-transistor block implementing the function f , while leaving the cross-coupled p -transistor latch at the opposite end as shown in Fig.28.

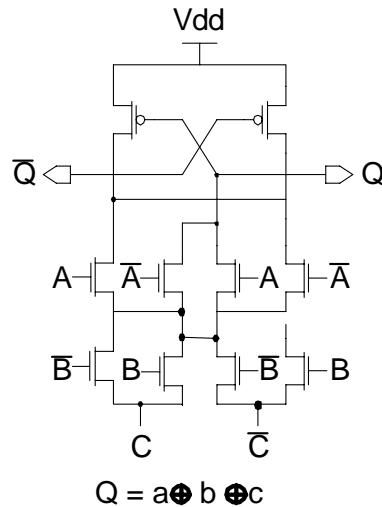


Fig.28. 3-input XOR Gate implementation in CVSL-PG

They have solved the problem of current spikes in CVSL by having the CVSL-PG switching tree act as the pull-down (accelerating the shut down of the p-transistors). The cross-coupled pMOS is acting as a load to regenerate the output signal levels. Therefore the size of the pMOS transistors is not critical anymore. Those transistors can be of the minimal size, thus unlike DCVSL, DCVSL-PG is not a ratioed logic. Further, there are fewer transistors in DCVSL-PG leading to smaller and faster circuits as compared to DCVSL. The main difference between DCVSL and DCVSL-PG is in the logic trees implemented with nMOS pass-transistors. In DCVSL-PG topology they are not always connected to ground but are, most of the time, connected to other pass variables, even, to the supply voltage. The switching network thus does not act as a path to ground but as a passing network. The cross-coupled pMOS pair is only used as a regenerative load to bring the outputs to full-swing level.

DCVSL-PG logic showed better performance than DCVSL. This was demonstrated by an implementation of 2nS 64-bit adder in 0.5 μ CMOS technology.

Swing Restored Pass-Transistor Logic (SRPL)

Toshiba Corp. developed their version of differential CMOS pass-transistor logic that does not suffer from degraded pull down performance [Iya ref]. They named it: Swing Restored Pass-Transistor Logic (SRPL). In SRPL the generic gate consists of a pass-transistor logic constructed of nMOS transistors (similar to CPL) and a latch type swing restoring circuit consisting of two cross-coupled CMOS inverters as shown in Fig.29. The nMOS transistor logic network implements any Boolean logic function while

the outputs of the pass-transistor logic are restored to full swing by the cross-coupled combination at the circuit output. In this way SRPL solves a major problem of the CPL logic. However, it is argued that the input variable can potentially be connected to a long chain through several gates, thus making the total output capacitance of the circuit quite large. Toshiba has built an experimental MAC (Multiply Accumulator) in a 0.4μ CMOS technology achieving a 150MHz speed at 3.3V supply voltage.

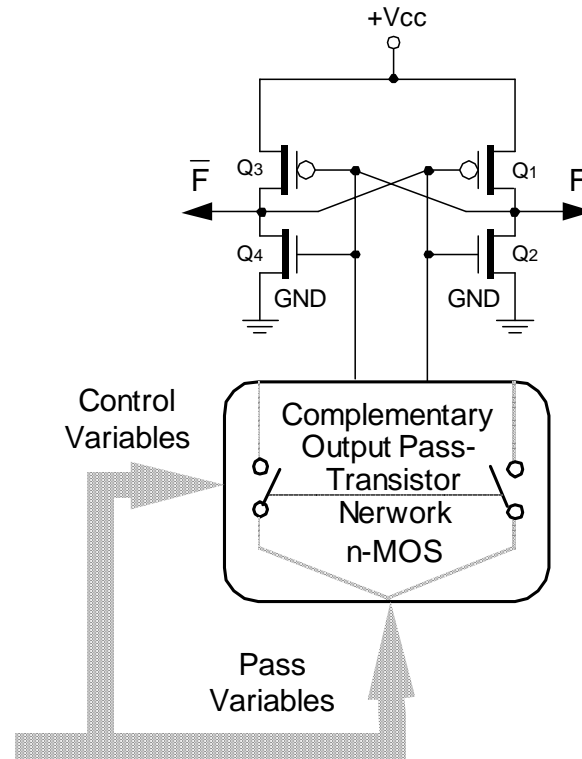


Fig.29. Generic Swing Restoring Pass-Transistor Logic (SRPL) Gate

Logic Comparison

Comparisons of full adder circuits implemented with CMOS, CPL, DPL, DCVSPG and SRPL showed CPL to be the fastest followed by SRPL and DCVSPG logic. However, SRPL had the best power-delay product, which amounted to 21% of that of CMOS.

References:

- [1] Masaki, “*Deep-Submicron CMOS Warms Up to High-Speed Logic*”, IEEE Circuits and Devices Magazine, November 1992.
- [2] Krambeck, C.M. Lee, H.S. Law, “*High-Speed Compact Circuits with CMOS*”, IEEE Journal of Solid-State Circuits, Vol. SC-13, No 3, June 1982.
- [3] V.G. Oklobdzija, R.K. Montoye, “*Design-Performance Trade-Offs in CMOS-Domino Logic*”, IEEE Journal of Solid-State Circuits, Vol. SC-21, No 2, April 1986.
- [4] N.F. Goncalves, H.J. DeMan, “*NORA: A Racefree Dynamic CMOS Technique for Pipelined Logic Structures*”, IEEE Journal of Solid-State Circuits, Vol. SC-18, No 3, June 1983.
- [5] L.G. Heller, W.R. Griffin, et al, “*Cascode Voltage Switch Logic: A Differential CMOS Logic Family*”, in 1984 Digest of Technical Papers, IEEE International Solid-State Circuits Conference, Vol.27, p.16-17, February 1984.
- [6] L.C.M.G. Pfenning, et al, “*Differential Split-Level CMOS Logic for Subnanosecond Speeds*”, IEEE Journal of Solid-State Circuits, Vol. SC-20, No 5, October 1985.
- [7] K.M. Chu, D.L. Pulfrey, “*A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic Versus Conventional Logic*”, *IEEE Journal of Solid-State Circuits*, Vol. SC-22, No.4, p.528-532, August 1987.
- [8] KM Chu and DL Pulfrey, “*Design Procedures for Differential Cascode Voltage Switch Circuits*”, *IEEE J.Solid State Circuits*, vol sc 21, no 6, December 1986.

References (CPL, DPL, DVL Synthesis)

- [9] S. Whitaker, “*Pass-transistor networks optimize n-MOS logic*”, *Electronics*, September 1983.
- [10] K. Yano, et al, “*A 3.8-ns CMOS 16x16-b Multiplier Using Complementary Pass-Transistor Logic*”, IEEE Journal of Solid-State Circuits, Vol. 25, No 2, p.388-395, April 1990.
- [11] K. Yano, et al, “*Lean Integration: Achieving a Quantum Leap in Performance and Cost of Logic LSIs*”, Proceedings of the Custom Integrated Circuits Conference, San Diego, California, May 1-4, 1994.
- [12] 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, June 1996.
- [13] M. M.Suzuki, N.Ohkubo, T.Shinbo, T.Yamanaka, A.Shimizu, K.Sazaki, and Y.Nakagome, “*A 1.5ns CMOS 16x16 Multiplier Using Complementary Pass-Transistor Logic*,” *IEEE Journal of Solid-State Circuits*, vol.28, no.11, pp.599-602, Nov. 1993.

- [14] N. Ohkubo, et al, "A 4.4-ns CMOS 54x54-b Multiplier Using Pass-transistor Multiplexer", Proceedings of the Custom Integrated Circuits Conference, San Diego, California, May 1-4, 1994.
- [15] P.Y.K.Cheung et al., "High Speed Arithmetic Design Using CPL and DPL Logic," *Proceedings of the 23rd European Solid-State Circuits Conference*, pp.360-3, Sept. 1997.
- [16] V. G. Oklobdzija and B. Duchêne, "Pass-Transistor Dual Value Logic For Low-Power CMOS," Proceedings of the 1995 International Symposium on VLSI Technology, Taipei, Taiwan, May 31-June 2nd, 1995.
- [17] V.G.Oklobdzija and B.Duchene, "Synthesis of high-speed pass-transistor logic," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol.44, no.11, pp. 974-6, Nov.1997.
- [18] D. Markovic, B. Nikolic, V.G. Oklobdzija, "[General Method in Synthesis of Pass-Transistor Circuits](#)" 22st International IEEE Conference on Microelectronics, May, 2000, Nis, Yugoslavia.
- [19] F.S. Lai, W. Hwang, "Differential Cascode Voltage Switch with the Pass-Gate (DCVSPG) Logic Tree for High Performance CMOS Digital Systems", Proceedings of the 1993 International Symposium on VLSI Technology, Taipei, Taiwan, June 2-4, 1995.
- [20] Parameswar, A., H. Hara, T. Sakurai, "A Swing Restored Pass-Transistor Logic Based Multiply and Accumulate Circuit for Multimedia Applications", Proceedings of the Custom Integrated Circuits Conference, San Diego, California, May 1-4, 1994.
- [21] T. Fuse, et al, "0.5V SOI CMOS Pass-Gate Logic", Digest of Technical Papers, 1996 IEEE International Solid-State Circuits Conference, San Francisco February 8, 1996.
- [22] T.Kuroda and T.Sakurai, "Overview of Low-Power ULSI Circuit Techniques," *IEICE Transactions on Electronics*, E78-C, no.4, pp.334-44, April 1995. INVITED PAPER, Special Issue on Low-Voltage Low-Power Integrated Circuits.
- [23] J.H.Pasternak, A.S.Shubat, and C.A.T.Salama, "CMOS differential pass-transistor logic design," *IEEE Journal of Solid-State Circuits*, vol.SC 22, no.2, pp.216-22, April 1987.
- [24] J.H.Pasternak and C.A.Salama, "Design of Submicrometer CMOS Differential Pass-Transistor Logic Circuits", *IEEE Journal of Solid-State Circuits*, vol.26, no.9, pp.1249-58, Sept. 1991.
- [25] I.S.Abu-Khater, A.Bellaouar, and M.I.Elmasry, "Circuit techniques for CMOS low-power high-performance multipliers," *IEEE Journal of Solid-State Circuits*, vol.31, no.10, pp.1535-46, Oct. 1996.
- [26] P.Buch, A.Narayan, A.R.Newton, and A.Sangiovanni-Vincentelli, "Logic Synthesis for Large Pass Transistor Circuits," *Proceeding of the IEEE International Conference on Computer-Aided Design (ICCAD)*, pp.633-670, Nov. 1997.

[27] R.Zimmermann, W.Fichtner, "Low-Power Logic Styles: CMOS Versus Pass-Transistor Logic," *IEEE Journal of Solid-State Circuits*, vol.32, no.7, pp.1079-90, July 1997.

[28] *Microprocessor Report*, various issues, 1995-2004.

References:

[1] L. Gwennap, "Processor Performance Climbs Steadily", *Microprocessor Report*, p.18, January 23, 1995.

[2] Dobberpuhl, D., et al, "A 200 MHz 64 b Dual-Issue CMOS Microprocessor," *1992 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, San Francisco, CA, USA, 19-21 Feb. 1992, pp. 106-7, 256.

[3] B. Gieske, et al, "A 600MHz Superscalar RISC Microprocessor with Out-of-Order Execution", *1997 ISSCC Dig. Tech. Papers*, p.176-177, February 7, 1997.

[4] B. Benschneider, et al, "A 300MHz 64-b Quad-Issue CMOS RISC Microprocessor", *IEEE Journal of Solid-State Circuits*, Vol.30, No.11, November.