[VLSI Digital Circuit Design], by [Oklobdzija, Yano].
ISBN 0-471-XXXXX-X   Copyright © 2000 Wiley[Imprint], Inc.

Chapter 4a: Draft – NOT TO BE REPRODUCED

Dynamic CMOS Circuits

Vojin G. Oklobdzija, Kazuo Yano

Introduction

Historically, dynamic CMOS was used sparsely by using the property of dynamic nodes. If the transistor leakage current is relatively low so that a circuit node can retain its charge for a relatively long time, the presence and absence of charge can be used to interpret particular information. This is similar to the principle used in dynamic memories and it brings the same benefits of simpler and therefore denser and more economical circuit. A typical use of dynamic CMOS circuit was to store the information. A dynamic CMOS latch is shown in Fig. 1.

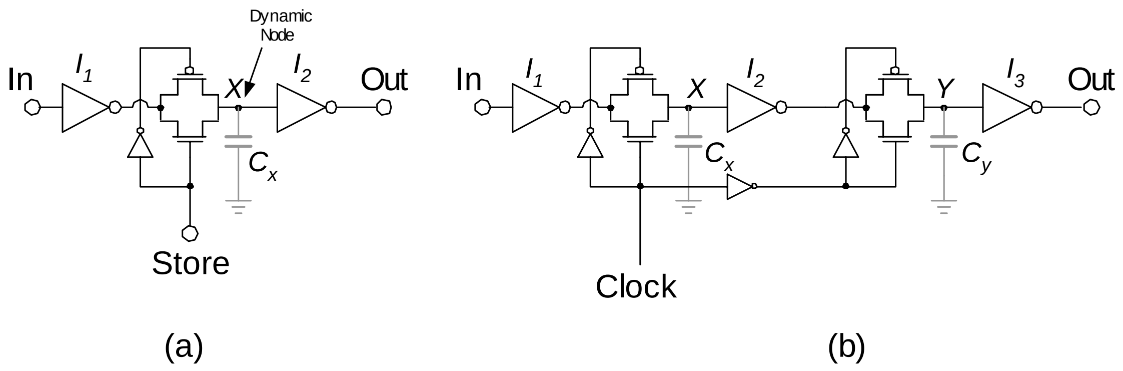

Fig. 1. (a) Dynamic CMOS Latch (b) Dynamic CMOS Master-Slave Latch

In the example shown in Fig.1.a, dynamic node X consisting of the input capacitance C_x of the inverter I_2 is charged / (or discharged) while the signal *Store=1*. When *Store=0* the input capacitor C_x is disconnected and the input node is retaining its charge. The voltage level on the input of the inverter I_2 is corresponding to the voltage level of its input. Assuming that it takes a relatively long time for the capacitance C_x to discharge the node X will retain the necessary information for the needed period of time. The storage element, in this case, is very simple; consisting of the capacitor C_x only which comes in basically for free as a natural property of the inverter. We should notice that the node *X* will eventually loose its charge. Therefore the logic that uses dynamic node properties must be refreshed from time to time as a re-assurance that the information does not get lost. In Fig.1.b. we see a dynamic *Master-Slave* Latch. The *Master* latch is a dynamic latch consisting of the inverters I_2 and dynamic node *X*. The *Slave* latch consists of dynamic node *Y* and inverter I_3. While Clk=1 the Master latch is sampling, while the Slave latch is outputting the value stored on the node *Y*. When the clock changes to Clk=0 the logic value at the output of the inverter I_2 changes the node *Y* accordingly and it is propagated to the Output. It should be noticed that inverters I_2 and I_3 could be replaced by a gate, implementing useful logic function. The latching function does not take any logic stages, thus virtually coming for free.

An example of a good use of the dynamic logic is the so called *"Manchester Carry Chain"* used to propagate carry signal in a typical VLSI adder. A CMOS implementation of the *Manchester Carry Chain* is shown in Fig. 2.

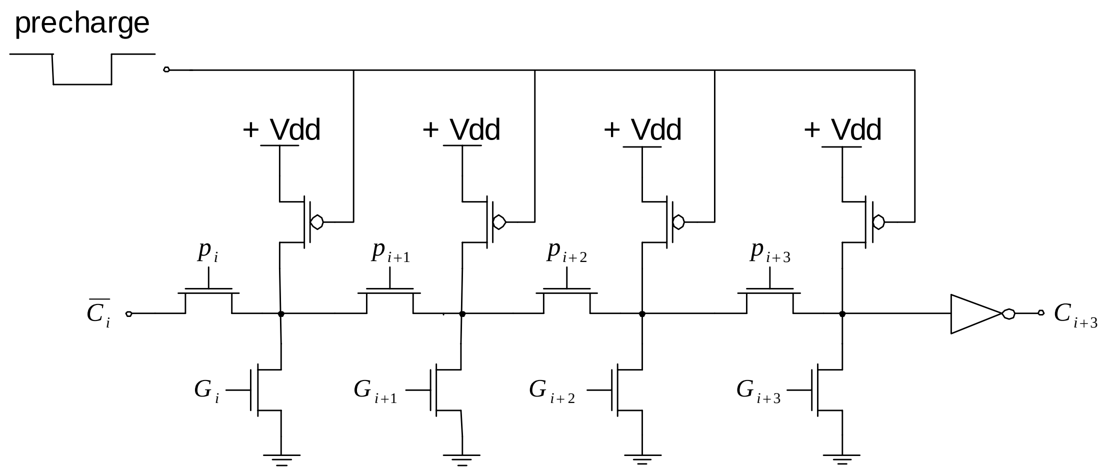

Fig. 2. Dynamic Manchester Carry Chain

It is assumed that all the *Propagate* P_i and *Generate* G_i signals are kept at logic zero level during the precharge phase and that they change *monotonically* (only a transistion from zero to one is allowed). This feature helps to avoid any possible signal conflicts during the precharge phase. Using dynamic CMOS combined with pass-transistor logic yields one of the simplest and fastest implementation of the carry function and it has been widely used for implementing VLSI adders.

Though efficient and simple, dynamic CMOS also brings certain problems and risks associated with its use. One of them is the stability of the node. If left for a sufficiently long time the transistor leakage currents can discharge the node, thus the information will be lost. However, designers are aware of those limitations and they do provide frequent refreshing of the information which prevents this from happening. Radiation and alpha particles are more of a problem. If the particle hits the node X, it generates many pairs of electron and acceptor atoms which can discharge the node X, fully or partially. Radiation induced charge problem is illustrated in Fig. 3.

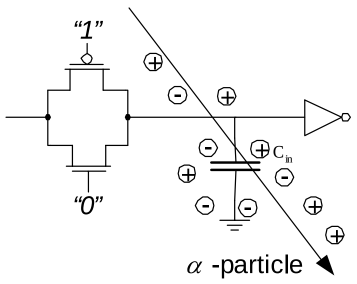

Fig. 3. Radiation induced charge

Another problem can be caused by the capacitive and inductive coupling between the signal lines. One such a problem is illustrated in Fig. 4.a. If the line coupling causes a positive going spike on the signal linked to the node Y, an accidental charge of the node X can occur, in spite of the node X being disconnected from the node Y. When the node Y becomes positive beyond V_{dd}, the transistor MP1 turnes ON, because its source is now positive with respect to the gate. If the *overshot* on the node Y is of sufficient duration and has sufficient energy, it can charge the node capacitance of the node X, thus turning the signal from 0 to 1 and causing a false operation. In order to prevent that, an inverter is always inserted to precede the transistor switch as shown in the Fig. 4.b. when the switch is connected to long lines or to the nodes that are exposed to excessive capacitive or inductive coupling. An *overshot* at the input of the inverter can not cause its output to change.

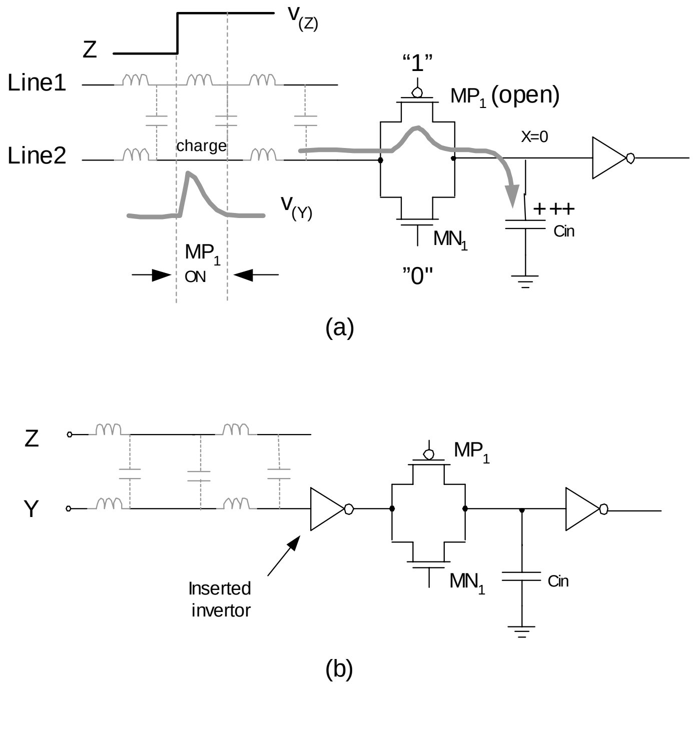

Fig. 4.(a) Accidental charge caused by capacitive or inductive coupling between the signal lines Y and Z. (b) prevention by inserting and inverter between the affected line and the pass-transistor switch.

CMOS Domino Logic

CMOS-Domino logic was developed while designing the first 32-bit microprocessor, called “Belmac”, at the AT&T Bell Laboratories by Krambeck, Lee and Law in the early 1980s. This microprocessor was also the first 32-bit CMOS processor which really started the transition into the CMOS era. This was the first serious departure from the static CMOS.

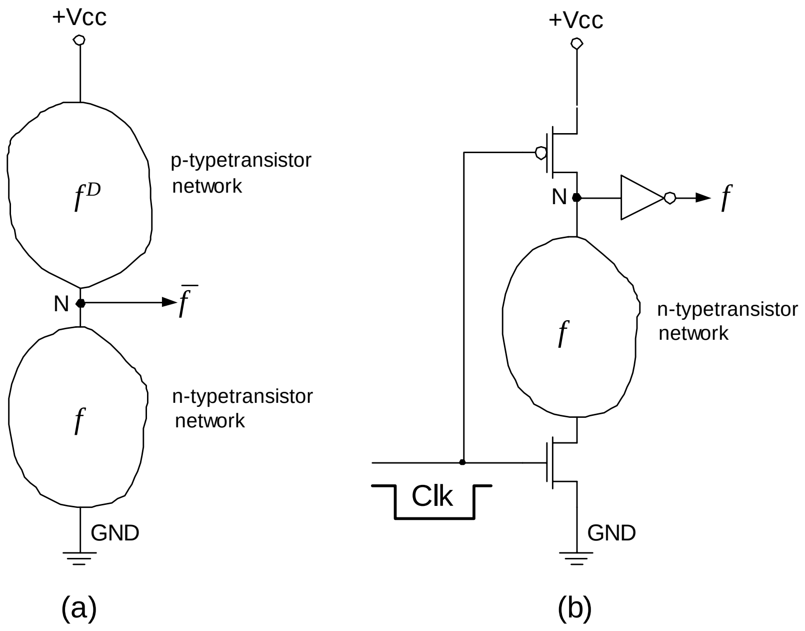

Fig. 5. (a) CMOS logic block (a), Domino Logic (b)

CMOS circuit requires twice as many transistors as n-MOS in order to implement the same function, because both functions *f* and f^D should be realized. The switching time of p-transistor of a comparable size to n-transistor is twice as long as compared to the n-transistor. This is due to the different carrier mobility of *n* and *p*-type carriers in silicon. The p-type network has to be twice as large as compared to n-type transistor network, if we want to make the *low-to-high* output voltage transition roughly equal to the *high-to-low* transition. This makes CMOS logic block almost three times as large as n-MOS logic implementing the same function.

To overcome this inherent CMOS problem it was suggested to build CMOS logic containing only *n*-type transistors implementing the switching function *f*.

This logic is a dynamic type because there are two clock-phases necessary for its proper operation. First, there is a *precharge phase* during which the clock is low, followed by the *evaluation phase* during which the clock is high. During the *precharge phase*, all of the nodes *N* are precharged via the *p*-type transistor *Q1*. The nodes *N* will stay *charged* if there is no discharge path in the switching function *f* during the evaluation phase when the transistor *Q2* is conducting. However, if the nodes *N* were taken directly as outputs, thus driving the inputs of the next logic blocks, all of the subsequent blocks would start to discharging immediately following the *precharge phase,* simply because all of the outputs are at the *logic one*. Therefore, the final state of the functional blocks would be undetermined. Domino Logic resolves this problem by passing all of the nodes *N* through the regular CMOS inverters. Only the output of an inverter can drive the next logic block. In Domino logic, all of the outputs are at *logic zero* immediately following the *precharge stage*. Therefore, no discharge can exist in the logic blocks that are themselves driven by other Domino logic blocks. The *evaluation phase* starts by the logic blocks being driven by the *primary inputs.* Some of the blocks will be selectively discharged, if a path to ground is established by the logic function represented by that particular block. This would change their outputs from *logic zero* to *one* driving the inputs of subsequent logic blocks. Now if this change in turn, creates a discharge path in the subsequent logic blocks, they would discharge, changing their outputs from *zero* to *one*. This change would further propagate through the logic from the primary inputs to the primary outputs like falling dominos. This is where the logic obtained its name: *Domino Logic*.

The benefits of CMOS Domino Logic were not obtained without a cost. First, a part of the cycle used for *precharge* is essentially lost because no logic operation is possible during this time. Ability to rapidly discharge the output node N via the *n*-type MOS transistor switching network was supposed to offset this loss. This feature is implementation dependent and not always true. Further, CMOS Domino is inherently *non-inverting* logic which represents difficulties (e.g. XOR gate implementation with Domino as described by Krambeck, Lee and Law is not possible). This inability is to be to some extent compensated by using dual polarity latches at the inputs.

However, another feature plagued CMOS Domino logic. This is a *charge re-distribution problem* described in the paper by Oklobdzija et al. Charge re-distribution is manifested by the loss of charge from the node *N*, due to the creation of paths leading into the previously discharged parts of the transistor switching network *f*, but not to the ground node. The output of the Domino logic block was to stay at logic zero level, because there is no path in the switching network *f* leading to ground. However, due to the distribution of charge from the node *N* into the various nodes in the switching network the voltage of the node *N* will drop (Fig.1.2.). This voltage drop might exceed the threshold voltage of the inverter INV and its output will assume the *logic one*. The *logic one* value of the output might in turn discharge the next block (though it was not supposed to do so) leading to the erroneous value. One of several techniques to alleviate the *charge re-distribution* employs a small feedback p-type transistor Q_f connected to the node *N* whose function is to replenish the charge lost in the re-distribution and return the node *N* back to the *logic one* value. If the amount of re-distributed charge was not large this might be sufficient. However, depending on the amount of charge that was lost from the node N, a negative voltage *glitch* of a different magnitude will result on the node *N*. This *glitch* may be amplified in the inverter (*INV*) and it might be sufficient to discharge the next logic block. The *charge re-distribution* mechanism is illustrated in Fig.1.3.

The problem posed by the non-inverting property of the CMOS Domino logic was treated in the paper by Goncalves and De Man in which they have proposed a new type of Domino logic consisting of p-type as well as n-type switching networks which are employed alternatively. Both n-type logic and p-type logic are in the *precharge* phase where the output node of the n-type switching function is precharged to logic *one* while the output of the p-type switching function is *discharged* to logic *zero*. This is achieved by clocking the n-type and p-type switching functions with the two opposite clock phases. During the *evaluation phase,* n-type logic blocks will discharge their output to logic *zero,* if there exists a path in the n-transistor switching function connecting the output to ground. In turn, this may create a path to Vcc in a p-type transistor switching network of the next block bringing the output to logic *one*.

The change would propagate from the primary inputs to the primary outputs alternating between n-type and p-type of switching blocks. If a particular output is destined to be an input in the next logic block of the same type, the inverters make its operation very similar to Domino circuit which consist of n-type and p-type switching networks. The operation of NORA logic is illustrated in Fig.1.4. A tri-state buffer at the output of a logic section holds its logic value during *precharge* phase, thus enabling pipelining of logic section. NORA logic is also sensitive to charge re-distribution and the large p-type switching transistor blocks do not add to its performance. Though a very interesting concept, even today, this type of logic has not found many applications.

CMOS Domino family was further enhanced by Heller, Griffin and Thoma of IBM, which resulted in CVSL logic. The IBM authors developed two types of CMOS logic which they called Cascode Voltage Switch: *static* and *dynamic,* shown in Fig.1.5. The first of these types of CMOS logic: static CVSL, consists of two n-type transistor switching functions f and $\overline{f}$ which are connected to Vcc via a cross-coupled p-type transistor combination. The advantages of this logic over regular CMOS are obvious. While CVSL implements two functions f and $\overline{f}$, so does CMOS except that in CMOS f is implemented with p-type transistor switching network. Given the inferior speed of p-type transistors, the p-type transistor switching function will usually end up being twice as large as the one implemented with the n-type transistors. In terms of the transistor numbers, CVSL contains two p-type transistors in the cross-coupled combination that are in excess of the number of transistors used by regular CMOS. It can be argued that in terms of size CVSL is not larger than regular CMOS, though this is dependent on the complexity of the logic function implemented. Certainly operation involving only n-type transistor switching function is faster. Further, Heller, Griffin, Davis and Thoma showed that implementation of f and $\overline{f}$ does not necessary mean duplication. A number of transistors in the switching functions f and $\overline{f}$ can be shared, therefore, setting duplication only as an upper limit. Creation of CVSL circuits was supported by an automated synthesis tool (one of the first of that kind) that would create optimal and shared n-type transistor switching functions f and $\overline{f}$ as shown in the example in Fig.1.5.(a).

The second type of the new CMOS logic, dynamic CVSL, is a clocked version of static CVSL and in essence it represents a dual output CMOS Domino logic. Transistor *Qf* is added to ensure more robust operation of the dynamic CVSL circuit. If charge re-distribution occurs resulting in the loss of charge at the inverter's input, the output voltage will drop toward zero. This will in turn activate the transistor *Qf* and the charge will be restored pulling the output back to logic *one*. In order to discharge this node the pass-transistor network implementing the function $\overline{f}$ has to overcome the transistor *Qf*. *Qf* is made to be a *weak* transistor, thus its dimensions are smaller compared to the rest of the transistors implementing the switching function.

An interesting new concept is presented in the paper by Leo Pfennings on Differential Split-Level CMOS Logic. This is essentially a very clever enhancement of CVSL, which allowed the use of one micron devices at that time and under a reduced voltage of operation. The logic transistors were made of $L_{eff} = 1\mu$ devices, operating at 2.5 V, while the entire circuit operates on the commonly used $V_{DD} = 5$ V. The logic swing of the interconnection lines was reduced to 2.5 V allowing for a sub-nanosecond speed (in 1985). The essence of DSL logic can be best understood from the Fig.1.6. The difference between DSL and CVSL is in introduction of the two n-MOS transistors Q_3 and Q_4 between the load transistors Q_1 and Q_2 acting as source followers. Therefore the impedance seen from the side of the logic is low, allowing for a fast change of state, while their output resistance is high, allowing for the use of small devices Q_1 and Q_2 representing the *load* transistors. This enables for a fast change of the logic state, further facilitated by the reduced voltage swing of 1/2 V_{DD}. The load transistors Q_1 and Q_2 are not completely turned off due to their gates being driven from the lower voltage potential. This makes an active loop capable of switching the state in a rapid manner. However, this also represents a power problem because DSL contains a static power component.

A comparative analysis of conventional CMOS versus Differential Cascode Voltage Switch Logic (which includes: CVSL, DSL and NORA), both static and dynamic have been presented in the paper by Chu and Pulfrey. The analysis is somewhat limited because it involves a section of a full adder only. Their conclusion is that DCVS logic is faster than conventional CMOS, however, this advantage is counterbalanced by a somewhat larger circuit and more active power consumption. The fastest logic technique seems to be DSL, however the problems with DSL involve static power dissipation and increased sensitivity on circuit parameters in order to make the operation reliable. In dynamic operation, differential logic seems to have a speed advantage over single-ended logic.

Similar comparative studies to that of Chu and Pulfrey were conducted extensively at IBM in the early 80s in order to reach an important decision to which CMOS family should be adopted for the future products. This studies involved several large pieces of the control logic besides the parts from the data path as test examples. The findings, which were not published, were in some agreement with those of Chu and Pulfrey. However, they showed that CVSL has an advantage over regular CMOS. Though, this advantage was not sufficient to justify extensive changes in the computer aided design tools and methodology, and test generation in particular. Therefore regular CMOS was chosen, but the real deciding factor was the status of the existing CAD tools, not the performance of the logic families compared. In spite of its advantages CVSL failed to become a technology of the future. Very often the choice is influenced by other factors, such as CAD and testability as it was the case. Those issues will be covered in later chapters of this book.