1 An introduction to domino logic

1.1 CMOS and NMOS

By the late 1970s complementary metal oxide semiconductor (CMOS) started to become the process of choice for digital semiconductor designs. CMOS had originally been proposed by Frank Wanlass in 1963 as a low standby power technology, since CMOS logic gates dissipate almost no power when the inputs to the gate do not change [1]. This follows as CMOS contains both PMOS field effect transistors (FETs), which can efficiently drive a high voltage, or logic one value, and NMOS transistors, which are good at driving a zero voltage. The presence of complementary transistors allows CMOS logic gates to be implemented so that the output voltage level is connected to the power or ground line, but not both. This ability to avoid contention ensures that if the inputs are not changing, then no power is dissipated. This was a major advantage of CMOS over the other manufacturing processes then available, which dissipated constant leakage or bias currents.

In Figure 1.1 the schematic representation of a CMOS static NAND logic gate is shown. The logic gate has two inputs A and B. A high logic value at inputs A and B turns on transistors MN1 and MN2, while turning off transistors MP1 and MP2. This causes the output Z to be low. When either input A or B is off, however, the path to the ground line is ruptured, with a path to the power supply (by convention called Vdd) being established. This causes Z to rise. While a NAND gate represents a simple function, it does show how contention between the power and ground supplies can be avoided in CMOS circuits. This lack of contention means that when the inputs to a CMOS circuit do not change, often called a standby or idle state, almost no power dissipation occurs, except for a small leakage current which flows through the transistors due to the imperfect manner in which a MOSFET acts as a switch (due to the relentless scaling in the physical dimensions of CMOS processes, driven by the cost advantages of having a smaller silicon area for digital functions, MOS transistors have become less perfect switches, leading to greater leakage current).

The fact that CMOS logic would lead to substantial power savings was apparent to its inventor Frank Wanlass, who in 1963 was working at Fairchild Semiconductor. Wanlass attempted to prove the viability and technical advantages of CMOS with a monolithic implementation of the technology [2]. When this proved infeasible, he proved the concept with discrete transistors. His CMOS implementations reduced standby power by six orders of magnitude over equivalent bipolar and PMOS implementations [2]. While

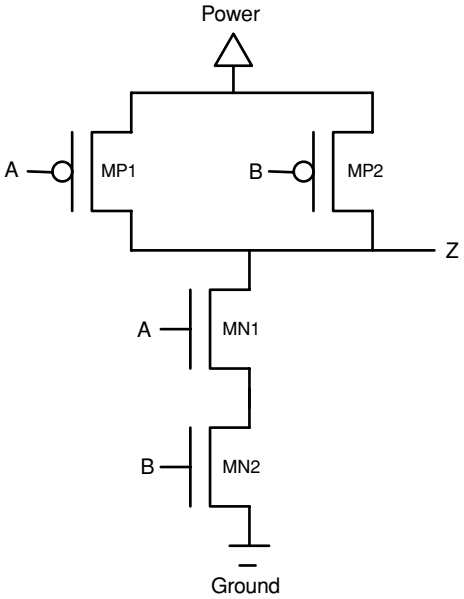

Figure 1.1. A static CMOS two-input NAND cell.

impressive, this advantage of CMOS would not prove decisive for many years. Early monolithic designs were very small, with the standby power consequently being very small as an absolute quantity. The inferior maturity of MOS transistors meant that in the 1960s, bipolar logic raced ahead of MOS transistors in applications. Transistor–transistor logic (TTL) and emitter-coupl ed logic (ECL), developed in 1962 and 1966, respectively, provided effective digital design techniques for bipolar transistors in the rapidly increasing semiconductor industry, which by 1962 had surpassed a billion dollars in annual sales [2]. The major user of CMOS in its early years was the watch industry, where battery life was a more important attribute than speed [3]. Starting in the 1970s, MOS technology began to mature rapidly, with much of the early industrial development being driven by Intel, then a small Silicon Valley company. In 1971 Intel released the 4004, the world's first microprocessor. The 4004 was built using a 10 μm line width PMOS transistor and used 2300 transistors running at 108 kHz [4]. In 1974 Intel released the 8-bit 8080, manufactured in a 6 μm NMOS process. The chip ran at 2 MHz and had 6000 transistors. Yield and cost concerns at the time ensured manufacturers preferred to use a single type of MOS transistor. Since NMOS transistors were faster than PMOS ones, due to the higher mobility of electrons over holes, the move to an NMOS process was natural.

Figure 1.2 shows the schematic implementation of a NAND gate using NMOS transistors only. The PMOS transistors MP1 and MP2 shown for the CMOS implementation in Figure 1.1 are removed here and replaced by a resistor, R1. This conceptual resistor is actually implemented by a depletion mode NMOS transistor [5]. The NMOS NAND gate output is at Vdd, or a logic one value, when either of the inputs, A or B, is low. When input A and input B are both high, the output is driven low. The current-driving

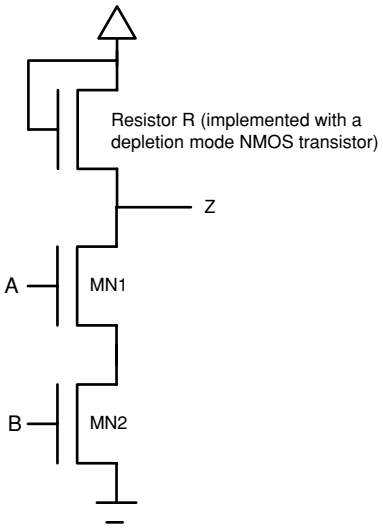

Figure 1.2. An NMOS two-input NAND cell.

ability of pull-down NMOS transistors must be much greater than that of the pull-up resistor. This ensures that the output can be driven to a low voltage at the cost of higher power dissipation. In addition to the standby power dissipation, NMOS circuits tend to be slower than equivalent CMOS circuits. This is due to the need for a weak pull-up resistor, which results in very slow low-to-high transitions. While these disadvantages may make NMOS appear to be unappealing, NMOS designs are more compact than CMOS circuits. Figure 1.2 uses only two transistors and a resistor, compared with the four transistors needed by a CMOS design. Since the pull-up resistor is implemented by another NMOS MOSFET, the NMOS design uses fewer transistors and a simpler process than the CMOS design. The need to move to CMOS therefore arose only when the integration level on integrated circuits (ICs) made the large standby power on the NMOS design unacceptable. For Intel this transition occurred in 1978, when the 8088/8086 family of microprocessors was introduced (the designs were almost identical to the 8088, having an 8-bit bus while the 8086 has a 16-bit bus). With 29,000 transistors and a clock rate of 5 to 10 MHz, the 8086 dissipated 1.5 W. This exceeded the 1 W per chip power limit for plastic packaging. Increases in integration levels meant that a 32-bit processor would dissipate 5 to 6 W, leading to severe reliability problems [6]. The CMOS version of the 8086, the 80C86, consumed only 250 mW [6]. The ability of CMOS to reduce power dissipation with increasing integration meant that it rapidly emerged as the technology that could best utilize fabrication advances. It is an advantage that CMOS maintains till today (2007), with the overwhelming majority of digital IC designs in the world being manufactured in CMOS, and the increased convergence of systems onto chips leading CMOS to make strong inroads into analog and radio frequency (RF) designs. In 1980, Intel's 8088 was chosen by IBM as the microprocessor for its personal computer (PC) [4], a step that would lead to Intel becoming, within a few years, the largest semiconductor

company in the world, with its semiconductor revenues far exceeding that of IBM itself. The rest, as they say, is history.

As semiconductor manufacturing progressed, the largest challenge to the nascent industry was the ability to design and verify designs using the increasing number of transistors available. This need was met by the development of a new field of software, often closely tied to dedicated hardware in its early years, called electronic design automation (EDA). EDA developments started in the 1960s, with software developed in-house by different semiconductor companies. In the early years of EDA the most common tools developed were circuit and logic simulators, which allowed designers to verify the expected functionality of a design before manufacturing. Alberto Sangiovanni-Vincentelli states in his excellent history of EDA (*The Tides of EDA*) that the early tools had limited loyalty due to the perceived limited value-added of the tools [7]. By the late 1980s continued developments in EDA had resulted in the development of logic synthesis, which could map a register transfer level (RTL) description to a set of standard cell gates and memory instances, and automated physical design tools, which could physically instantiate and route the wires needed to complete the physical design. These tools led to a marked improvement in productivity, allowing digital designs to be quickly implemented based on a higher abstraction level, behavioral RTL description [7]. The increasing complexity of EDA tools, along with the realization of their tremendous usefulness, led to the rise of independent EDA companies and a rapid reduction in EDA tool development within semiconductor companies. By the end of 2006, the EDA industry had a total available market (TAM) of 5.3 billion dollars [8], which is about 2% of the worldwide semiconductor TAM of 260 billion dollars [9]. The success of CMOS manufacturing technology, along with the availability of powerful EDA tools, allowed for the widespread penetration of electronics into a multitude of applications.

People who are drawn into, and ultimately stay in, engineering are generally somewhat private people. Our work must, by definition, be cooperative, but the bread and butter of our daily tasks tends to be very solitary exercises. I am aware that such an audience feels extremely uncomfortable with broad, historical utterances, reminding them of overly optimistic forecasts they have had to sit through in darkened conference rooms with a roll of the eye and a quick, knowing smile to a colleague. Still, I feel compelled to state the following: I have no doubt that looking back from the future, the most important historical event of our age will be the development and promulgation of digital technology. It will also be seen as a profoundly positive development. I believe that all of us working in this field should be proud of our achievements. I have said my piece and now return to the theme of digital ASIC design.

It may have been assumed that the emergence of ASIC design methodologies would displace all other techniques for implementing digital CMOS logic. This has not happened, as many digital designs have specific needs that cannot be achieved by using standard ASIC techniques. In recent years the capabilities of ASIC tools have increased greatly, largely due to the tremendous competition among the companies in the field. Many logical and behavioral optimizations that previously had to be hand-coded for efficient implementation are now automatically incorporated in the synthesis tools. The two most common benefits of custom design are its ability to optimize across the different

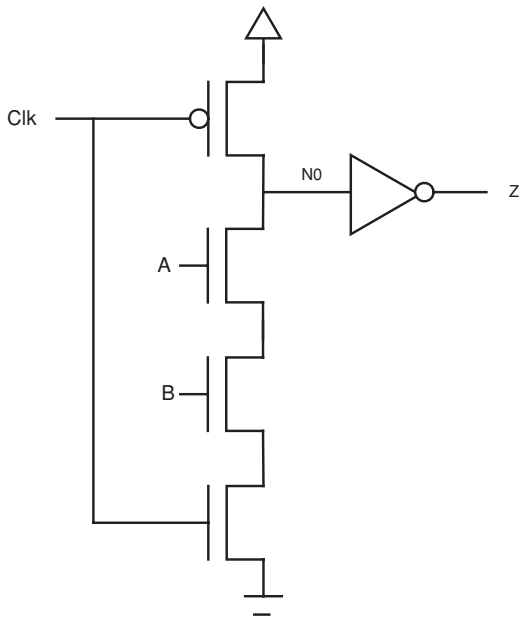

Figure 1.3. A CMOS domino logic two-input AND gate.

levels of abstractions in the ASIC design framework and the opportunity it provides for using logic families other than standard static logic. The first of these advantages relates to the sequential approach that an ASIC design methodology uses, by which standard cell library development, logic synthesis, and physical design are broadly separate processes. It is true that synthesis tools are increasingly aware of physical design constraints and physical design tools can perform logic optimizations. This is still very far away from them providing a common design framework. For example, if a very high-speed design can be made fast and small using a very specific cell that is instantiated and placed in a particular manner, it is improbable that an automated design flow will be able to reach that exact solution. If the library does not have the particular standard cell in its library, then the automated solution obviously cannot use it.

The second advantage of custom design is that it can utilize certain logic families, specifically dynamic logic, that automated design frameworks have not traditionally been able to support. In this book we describe our experiences in incorporating domino logic into an ASIC design flow. This journey starts with a short description of what domino logic is.

1.2 Domino logic circuits

A picture, it has been said, is worth a thousand words. We therefore begin our description of domino logic with Figure 1.3, which shows the schematic representation of a domino logic two-input AND gate.

The AND gate shown in Figure 1.3 can be used to illustrate the functionality, the speed advantage, and also some of the challenges involved in using this logic family. In Figure 1.3 it can be seen that the two functional inputs, A and B, are also attended by the clock signal, Clk. At first glance this may seem strange, since an AND gate should be a purely combinational circuit, which unlike latches and flip-flops does not require the presence of the clock signal. Domino logic is, however, a clocked logic family, which means that every single logic gate has a clock signal present. When the clock signal turns low, node N0 (which is called the evaluation or internal node – some authors refer to it as the dynamic node) goes high, causing the output of the gate to go low. This represents the only mechanism for the gate output to go low once it has been driven high. The operating period of the cell when its input clock and output are low is called the precharge phase or cycle. The next phase, when the clock is high, is called the evaluate phase or cycle. During the evaluate phase the output of the domino AND cell can go high provided that both inputs A and B are high, which causes the evaluation node, N0, to be driven to a low value. The evaluate phase is the functional operating phase in domino cells, with the precharge phase enabling the next evaluate phase to occur. The appropriate application of the clock signal ensures that the critical path in domino cells only traverses through cells in the evaluate phase. One of the advantages of domino logic over static logic can also be garnered from the schematic in Figure 1.3. Since the domino cell only switches from a low to a high direction, there is no need for the inputs A and B to drive any pull-up PMOS transistors. The lack of a PMOS transistor means that the effective transistor width that loads down a previous stage of logic, for a particular current drive, favors domino over static logic. This is critical since the key to high speed is ensuring that a speed advantage can be gained without loading down the cell greatly [10]. For example, if a design is constructed with a set of cells with transistors of a certain size, replacing the transistors in every cell with ones ten times larger will almost certainly lead to a design that is faster. Provided that the initial design is properly sized, i.e., without weak cells which have very long rise or fall times, the new design will not, however, be ten times faster. The reason for this is that, while the drive strengths of each cell have increased by a factor of 10, the output loading due to the input transistor capacitance seen by each cell has also increased by approximately a factor of 10. Since larger cells are now used in the design, its area will be larger, leading to greater wiring capacitance. Thus, while speed gains can be achieved by optimizing cell drives, the indiscriminate increase in drive strengths tends to limit the improvement in speed due to the increased self-loading.

In order to see how domino logic alters the relationship between input capacitance and output drive strength, compared with an equivalent static cell, the reader is directed to Figure 1.4. A static buffer is shown in the figure, with input PMOS and NMOS transistor widths of 2 μm and 1 μm, respectively. Assuming that the gate capacitance of a PMOS and NMOS transistor is the same per unit micrometer of transistor width, the total load seen by the cell driving the buffer is 3 μm of transistor gate width. For a domino cell it is possible to construct a buffer with the same drive strength but which has only 1 μm of transistor width as input capacitance. Alternately, with the same input capacitance it is possible to build a stronger and faster domino buffer. This is shown in Figure 1.4,

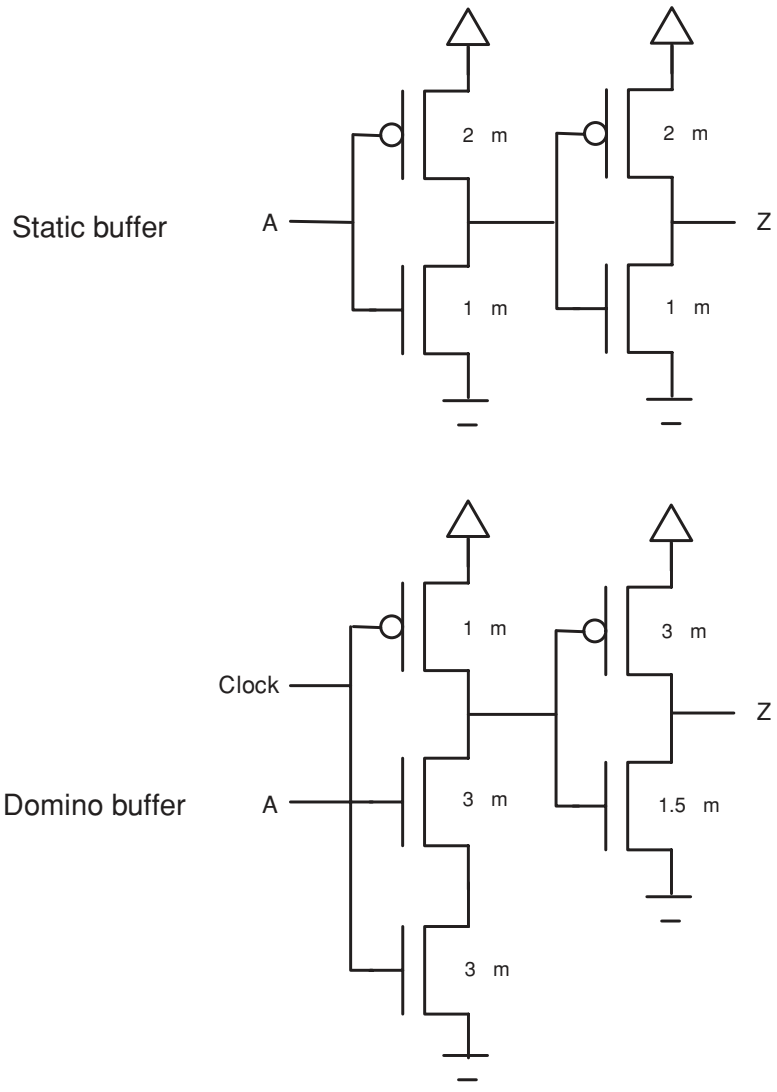

Figure 1.4. A static and domino logic buffer.

where a domino buffer with input transistor width of 3 μm is shown. This particular domino buffer is called a footer transistor, i.e., a series NMOS transistor connected to the clock. It is possible to use domino cells with and without footer transistors, although the absence of the footer transistor makes the design more complicated. Since the footed clock transistor adds to the resistance of the pull-down path, the two 3 μm transistor widths are considered roughly equivalent in terms of drive strength to a single 1.5 μm transistor. This follows, as the resistance of a MOS transistor is inversely proportional to its gate width, and as the total resistance of a set of resistors in series is equal to the sum of the resistors. Since, for the static buffer, 1 μm of NMOS transistor length is driving a 2 μm PMOS and a 1 μm NMOS transistor, the effective 1.5 μm of NMOS for the domino

cell has 50% greater drive strength. Thus, for the same input capacitance a domino cell can result in greater output drive strength than a static equivalent.

There are a few points to note. Firstly, the degradation in drive strength due to the addition of the footer transistor in a domino buffer is worse than in other domino cells with more than two NMOS transistors in series. For example, in a three-input domino AND cell, the number of NMOS transistors in series goes from three to four when the footed transistor is considered. This is much better than a domino buffer where a doubling in the height of the NMOS series stack occurs. Secondly, the PMOS pull-up transistor for the domino cell in Figure 1.4 is shown as 1 μm. The actual size of the PMOS transistor will depend on the time available for the output of the domino cell to turn low when the clock falls. This delay is called the precharge delay. In general, the PMOS pull-up transistor is smaller in a domino cell than the static equivalent. Thirdly, the ratio of PMOS to NMOS transistor width for the static cell is given as 2. In Chapter 3 we will see that the actual ratios tend to be lower in static logic. Finally, for stability purposes domino cells tend to use weak feedback keepers placed between the output and the evaluation node driving the output. For simplicity, that circuit is not shown in Figure 1.4.

In addition to being able to achieve better output drive strength for input loading, domino cells also have a speed advantage as they avoid contention when the cells switch. In order to understand this, one must note that the input to a static cell drives both PMOS and NMOS transistors. Any input transition that causes the cell to switch logical states results in a PMOS transistor being turned off and an NMOS transistor being turned on, or vice versa. Since the inputs to the cell have finite rise and fall times, this means that during the transition period both the PMOS and the NMOS transistors are weakly on. This contention between the two transistors increases the input voltage level at which the cell switches. It is possible to speed up the rise or fall transition of a static cell by increasing or decreasing the ratio of the PMOS to NMOS transistor size. This, however, leads to the alternate transition becoming slower. Since both transitions are equally important in static cells, it is difficult to gain very much by skewing a particular transition. For this reason, the switching point of most static cells tends to be close to half the supply voltage level (Vdd). For domino cells only the rising transition is critical. If an input rise causes the evaluation node of the domino cell to discharge, no contention exists between PMOS and NMOS transistors. This allows domino cells to start switching when the input voltage level reaches an NMOS transistor threshold voltage level. Figure 1.5 illustrates the switching behavior of a static and a domino buffer as the data input to the cell rises. The lower switching voltage of a domino cell leads to a speedup since the input driving cells will reach the lower NMOS threshold voltage quicker than a higher voltage level. These factors lead to domino cells being significantly faster than equivalent static cells. The speed advantage of a domino cell over an equivalent static design is in the range of 1.5× to 2.5×.

Domino logic is an uninverting style of logic [11]. This follows since every domino cell is a single-stage dynamic cell followed by an inverter. Consequently, the only valid transitions at the output of the gate during the evaluate phase are from a low to a high value. The uninverting nature of the logic means that while AND gates, OR gates, and

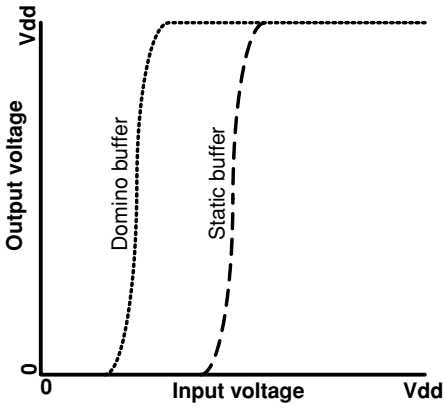

Figure 1.5. The output voltage of a static and domino buffer as the input switches from low to high.

buffers can be implemented with domino logic, NAND gates, NOR gates, and inverters cannot be. Since inverting functions are unavoidable in most designs, this would appear at first thought to preclude the general applicability of domino logic. Furthermore, as the evaluation node is precharged in domino logic cells, the only valid input transitions are from a low to a high value. Again, this is generally not acceptable. In the next paragraph we will describe how it is possible to construct general logic functions using domino logic. Before that, however, one other major difference between static and domino logic is discussed. Signal nodes, which toggle several times before reaching a final steady-state condition (referred to as glitching), are relatively common in static logic, but absent in domino designs. This is because once the output of a domino cell rises, no change in the inputs to the cell will cause the output to fall. Only when the clock falls at the end of the evaluation phase will the output of the cell fall. Thus, within a clock period there is the possibility of only one rise and fall at the output of the domino cell.

There are basically three approaches to constructing functionally correct domino logic designs in which inverting functions are present. The first approach is to stop the domino cone of logic when inverting functions are encountered. This approach was suggested in the original paper in which domino logic was introduced [11], where the XOR at the end of an adder is implemented as a static cell. The advantage of this scheme is that it allows for the easy incorporation of inverting functions with domino logic. The disadvantage with this scheme is that if an inverting function is encountered early in a path of logic, most of the gates in the path will be implemented with static logic. In addition, since some form of a latch must be placed at the boundary between domino and static logic, to ensure that the precharge value of the domino cells does not propagate through to the static logic, this will involve a timing penalty. These disadvantages could easily diminish the speed advantages to the point where it is not worthwhile. Nevertheless, if a single inverting function is present near the end of a critical path, the use of static logic at the end of the path can be a useful solution.

The other two approaches provide mechanisms to ensure that the entire path, generally from a register to a register, can be fully implemented with domino logic. The second

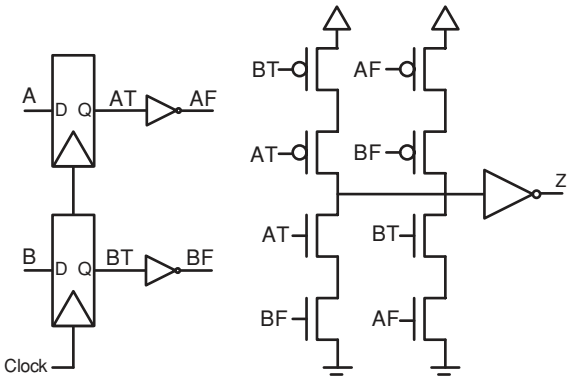

Figure 1.6. A static logic two-input XOR cell.

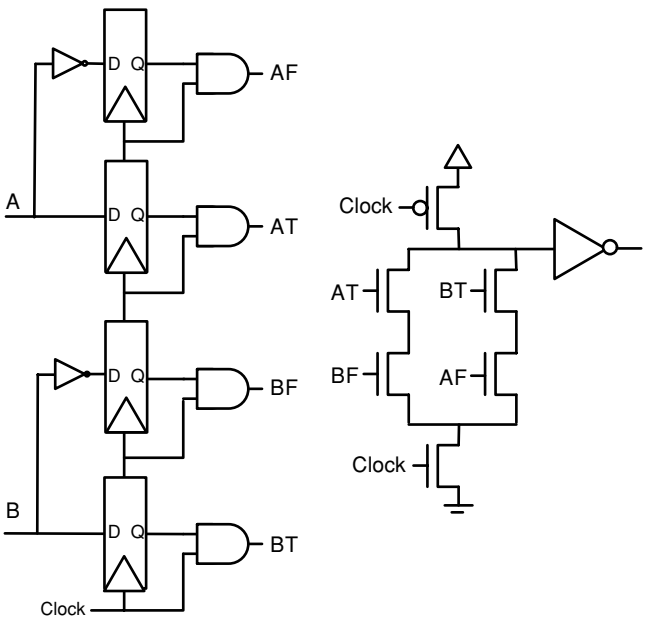

Figure 1.7. A domino logic two-input XOR cell.

approach for implementing inverting functions is to ensure that all inverting signals needed in a design are provided from the primary inputs which are low when the clock is low. In most digital designs the primary inputs to a block are the outputs of the flip-flops from a previous block. In Figure 1.6 a static logic implementation of an XOR function whose inputs are coming from a flip-flop is shown. In Figure 1.7 an implementation of the same function using domino logic is shown. To ensure that the primary inputs are initially 0, the output of the rising edge-triggered flip-flop is ANDed with the clock. This AND gate can be incorporated directly into the flip-flop. It is shown here as a separate gate for conceptual clarity. Since the clock is low before it rises, the inputs to the domino