CHAPTER1

INTRODUCTION

1.1 OVERVIEW

As the advancement of technologies is at its peak, rising market strain has driven cutting-edge microprocessors to enhance their performance with each passing year. This progress rate is maintained and kept in flow by using micro-architectural methodologies such as dynamic implementation and execution of designs and pipeline. But the usage of these technologies provides decrementing returns, however, in the present fast developing world, it is now more than ever necessary to use faster circuit techniques to further increase performance [1].

The circuit implementation of a logic function is done using 2N devices when Static Complementary MOS (CMOS) logic has a fan-in of N. A range of methods have been provided to decrease the amount of transistors that are needed to perform a particular logic function, including pseudo-NMOS, dynamic logic, pass transistor logic etc. There occurs static power dissipation when the functions are implemented using pseudo – NMOS logic since it requires N+1 number of transistors in order to design a logic gate that has N inputs [2]. An alternative logic type called dynamic logic is described in this report that achieves a comparable outcome while avoiding the usage of static power.

With the enhancement in the electronics industry, domino logic topology is being actively utilized for the designing of enhanced speed and better performance micro-controllers and microprocessors. The usage of domino logic enables to attain the adequate timing objectives [3] [4]. Their improved performance and enhanced efficiency can be attributed to factors such as reduced input capacitance, lower value of switching thresholds

and decreased use of logic gates to implement the design. However, a trade-off occurs between power dissipation and speed. As the speed of a design is improved it leads to enhanced power dissipation [5].

1.2 DYNAMIC LOGIC : PRINCIPLE

The primary implementation of a N-type dynamic logic gate is depicted in Fig. 1.1. From Fig. 1.1 it can be perceived that the implementation of PDN (pull-down network) is exactly similar to that of the implementation done using complementary MOS (CMOS). The operation of n-type dynamic logic gate is divided into two major phases:

1) Precharge
2) Evaluation

The mode of operation of the designed circuit is determined by the clock signal (CLK) provided.

1) Precharge Mode

When the clock input applied to the circuit is 0, i.e., *CLK = 0*, the PMOS transistor (represented as M_p) precharges the output node (*OUT*) to a value V_{DD}. Thus, when clock input is 0, the pull-down path is disabled as the evaluate NMOS transistor (represented as *Me*) is switched off. The evaluate transistor eliminates the consumption of static power that occurs during the precharge period. This static power consumption occurs when both the pull-down and pull-up devices are turned on simultaneously and as a result static current flows between the supplies [6].

2) Evaluation Mode

When the clock input applied to the circuit is 1, i.e., *CLK = 1*, the evaluation transistor (M_e) is in 'ON' state which implies that it is in operating condition whereas the precharge transistor (M_p) is in switched 'OFF' state. The output value OUT is discharged conditionally which depends upon the input values applied to the logic circuit as well as the

pull-down topology [7]. A low resistance path is created between OUT node and GND when the pull-down network (PDN) is conducting which causes the output value to be discharged to 0. However, if the inputs applied are such that the PDN is in conducting state, the value that is contained by the load capacitance (C_L) is the value that has been precharged at the earlier state [8].

During the evaluation mode of operation, GND is the only possible path that exists as a connection between the output node and the supply voltage. As a result, it is not possible to charge the output load capacitance once OUT have been discharged. This can be done only when the subsequent precharge phase takes place [9]. Therefore, the input values applied to the gates can make at most of one transition while in evaluation phase. While in evaluation phase it might happen that when the pull-down network of the logic function is in 'off' state, the output node enters into high impedance state [10]. Consequently, this functionality of the dynamic circuit design is essentially different from the static equivalent of the same logic function in which a low resistance path always exists between the output node and one of the supply voltages.

There are two output levels: Low output level V_{OL} represented by GND and High output level V_{OH} represented by V_{DD}. The Voltage Transfer Characteristics parameters of dynamic logic circuits are extensively different from that of static CMOS logic gates. It is not possible to apply pure static analysis to the dynamic logic gates because in order to be functional dynamic circuits needs a periodic sequence of precharges as well as evaluations [11] [12].

During the evaluation phase, the pull-down network (PDN) of the dynamic inverter becomes active in nature, which occurs in case the input signal applied to the gate exceeds the threshold voltage of the NMOS pull-down transistor. As a result, the values of switching threshold (V_M), V_{IH} and V_{IL} of the logic gate are set equal to threshold voltage (V_{tn}). This translates to a low value for the NM_L.

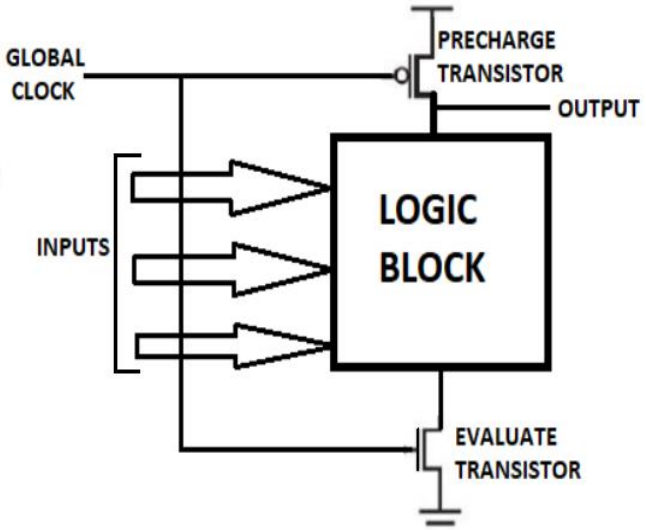

Figure 1.1 Dynamic Logic Circuit

1.3 SPEED AND POWER DISSIPATION IN DYNAMIC LOGIC

The capability that makes the dynamic logic circuits so efficient and highly used is the enhanced speed and reduction in implementation area. As the number of logic elements being utilized for the implementation of the logic function is decreasing, it signifies that the overall load capacitance of the circuit becomes much smaller. Some unique characteristics of logic circuits come to light when the switching behavior of the gate is analyzed [14]. After the end of precharge phase, the output is in high state. When the applied input signal is low, no additional switching takes place. Thus, as a result, $tp_{LH} = 0$. The discharging of output load capacitance takes place when the pull-down circuit makes a high-to-low transition. Thus, it can be concluded that tp_{HL} is proportional to the load capacitance (C_L) and the current-sinking capabilities of pull-down network. However, the use of the evaluation transistor presents an extra series resistance that leads to the slowing of gate operation. However, if we eliminate the evaluation transistor, static power dissipation will occur and performance would degrade [15] [16] [17].

The precharge charge is basically determined by the time that the circuit takes for charging the load capacitance (C_L) through the PMOS precharge transistor. During the charging of the load capacitance, it is not possible to utilize the logic being implemented by gate. Most of the time, the digital system maybe implemented such that the precharge time gets overlapped with the other logic design functions. Thus, while designing the dynamic logic circuit, the designer must be careful about this "overlap zone" [18].

At this point, dynamic logic design comes into light as it provides faster speed and less power dissipation. Although dynamic logic has several advantages, it suffers from "cascading issue" in extensive circuits [19]. There are two phases of functionality of dynamic logic:

1) Precharge phase (Set-up phase): In this phase, irrespective of the input values applied, output rises to an extremely high value. The load capacitance gets charged in this phase.

2) Evaluation Phase: In this phase, there exist a path between the GND and the output terminal. This causes the load capacitance to be discharged.

It might happen that the second logic gate of the dynamic logic design may discharge prematurely. This condition comes into play when the first logic gate is in precharge state.

This condition causes the precharge state of the second logic gate to be used up and it can't be restored until the next clock cycle. Thus, error occurs. Therefore, to be able to implement cascaded logic circuits, the one solution is Domino Logic. In domino logic, a static inverter is inserted between the logic states [20].

1.4 DOMINO LOGIC CIRCUITS

Domino logic topology is widely utilized in high-speed applications that are used for the implementation of logic circuits having large fan-in. But the area where domino logic circuits suffer is their vulnerability to unwanted noise. Domino logic circuits have low switching threshold voltage that makes them more sensitive to noise [21]. This low switching threshold voltage is equal to the threshold voltage of NMOS devices used for the implementation of evaluation network of the domino logic. The substantial increment in the noise with technology scaling severel affects the usefulness of domino logic topology [22]. With the help of technology scaling, it is possible to scale down the supply voltage which leads to the reduction in power dissipation.

The main concern is to design a circuit that has improved performance. This could be achieved by maintaining a high-drive current that can be attained by ensuring that the threshold voltage is proportionately scaled. However, scaling of the threshold voltage leads to a substantial increase in the sub threshold leakage current [23].

As the technology is being scaled down, it leads to the exponential increase in the leakage of evaluation NMOS transistors because of their lower threshold voltage. Domino logic circuits offers a significant edge over the existing technology due to their faster transitions and glitch-free operation. One of the most effective technology to implement high speed logic functions is the execution of circuits using Domino Logic. Domino logic is evolved from the dynamic logic technology and is based on the CMOS-based implementation of logic function using either PMOS or NMOS transistors. It provides a speed at least twice as faster than the corresponding static complementary MOS (CMOS) logic. In domino logic, a single clock is used for the process of precharge and evaluation of a cascaded implementation of dynamic logic blocks [24] [25].

1.4.1 Domino Logic: Principle

Domino logic module is basically comprising of an. n-type dynamic logic with a static inverter following it (Fig. 1.2). While operating in precharge phase, the output of the n-type dynamic logic gate is charged up to V_{DD}, and the output of the inverter is set to a value of 0. During evaluation phase, the dynamic logic gate conditionally discharges, and the output of the inverter makes a conditional transition from 0 to 1 [26]. If an assumption is taken such that the inputs being applied to a Domino logic gate are the outputs of the previous Domino logic gate , then it can be made sure that all the applied inputs are set to 0 at the end of the evaluation phase and the only transactions that takes place during evaluation phase are 0 to 1 transitions.

The inclusion of the static inverter in between the logic states provides with an additional advantage of increased noise immunity. The enhanced noise immunity can be attributed to the fact that the static inverter that is driving the fan-out of the logic gate has a low impedance output [27]. The use of the buffer further decreases the load capacitance of the output node of the dynamic circuit by providing separation between the internal and load capacitances. Let us take into consideration the implementation of the cascaded structure of Domino logic gates. In the precharge phase, all the inputs have been initially set to 0. In the evaluation phase, the output of the. first domino logic block either stay in the state 0 or makes a transition from 0 to 1, which in turns affects the output of the second logic gate [28] [29]. This transition effect might ripple from one stage to another throughout the chain, similar to a line of falling dominoes – hence given the name Domino Logic Circuit. Domino CMOS has the following characteristics:

- Domino logic can be used for the implementation of only non-inverting logic because each dynamic logic gate consists of a static inverter that has been inserted between the two logic states. This situation is a major limiting factor for the rare existence of pure domino logic design [30] [31].

- The logic functions implemented using Domino CMOS provides with very high speed. There only exists a rising edge delay and t_{pHL} equals to 0. The sizing of the inverter can be done in order to meet the fan-out characteristics of the logic gate. Also, the fan-out of the logic gate is already much smaller as compared to the corresponding static CMOS implementation of the same logic design, as there exists only one output load capacitance for a gate per fan-out [32].

Due to the low input being applied while in precharge phase, it is a practical approach to eliminate the evaluation transistor as this would lead to reduction in the clock load and would increase pull-down drive. But eliminating the evaluation device would result in extension of the precharge cycle which would further lead to the precharge to ripple through the logic network as well [33]. Thus, the critical path of a logic circuit can be signified as the time taken to precharge the logic circuit. Another issue that has been faced in domino logic implementation of a function is the excess power dissipation that occurs when both the pull-up and pull-down transistors are turned on. Therefore, it is vital to always utilize evaluation devices.

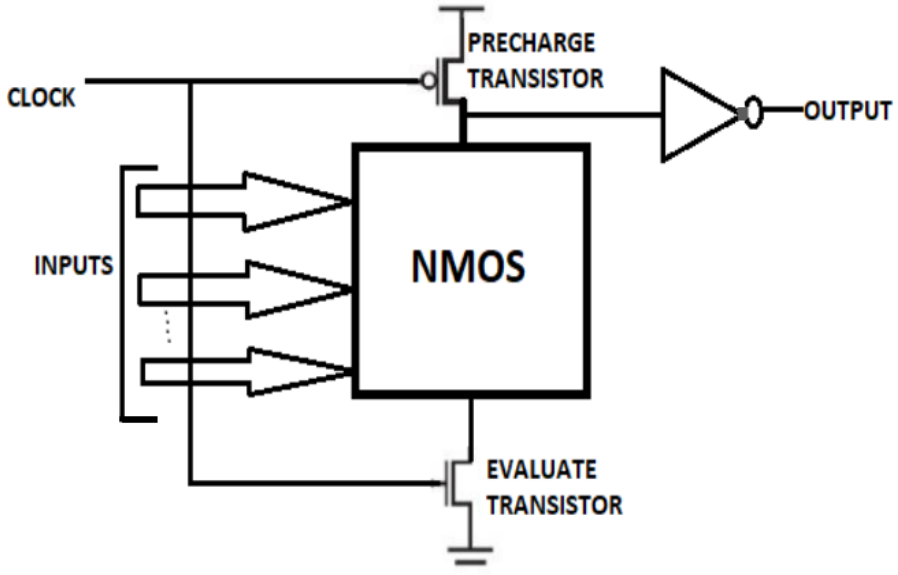

Figure 1.2 Domino Logic Circuit

From the Fig. 1.3, it can be seen that two functional inputs are controlled by a clock signal, CLK. Domino logic topology is a clocked logic family, which shows that each logic gate has a clock signal present.

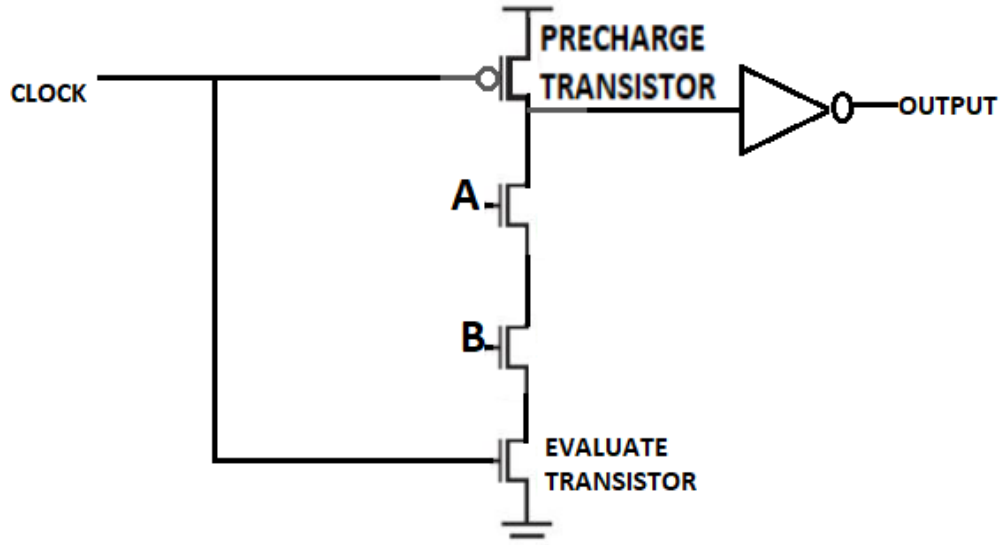

Figure 1.3 A CMOS domino logic 2-input AND gate.

When the clock signal becomes low, the evaluation node reaches a high state, which causes the output of the gate to attain a low value. The time period for which the cell is under operating condition when the input clock and the output are low is known as precharge phase or cycle.

The evaluation cycle is the phase when the applied input clock signal is high. In the evaluation mode of operation, the output of the AND logic using domino circuit, is capable of attaining a high value when both the applied inputs, A and B, are '1', i.e., high. The high inputs drive the evaluation node to a low value. The evaluation phase is the functional operating phase in the domino logic cells. The precharge phase basically enables the next evaluate phase to occur. The correct application of the clock signal ensures that the critical path in domino logic cells only traverses through cells in the evaluate phase [34].

1.5 DOMINO LOGIC BUFFER

Fig. 1.4 shows the design of a conventional clock-controlled domino logic circuit. Domino topology has a dynamic N-type gate (pull-down network PDN) followed by a static inverter.

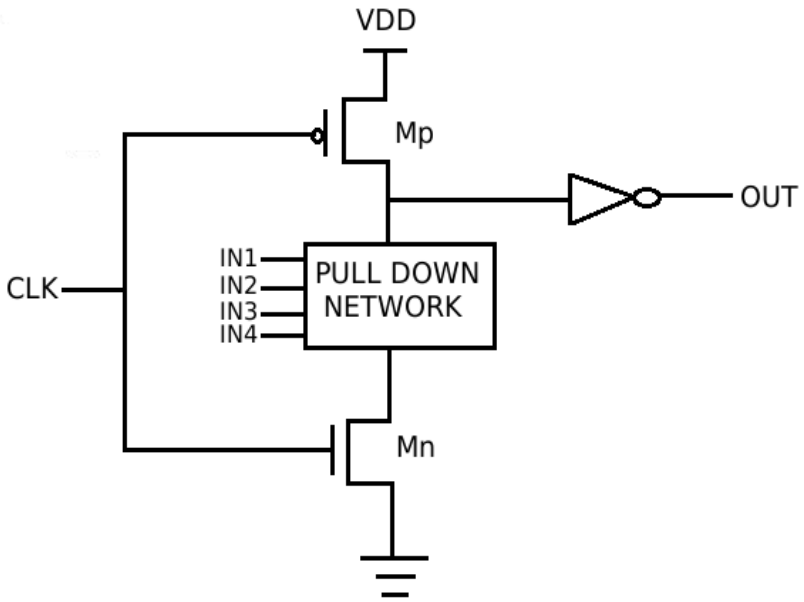

Figure 1.4 Domino logic implementation

Fig. 1.5 represents the implementation of a domino logic buffer. There are two phases of operation, precharge phase and evaluation phase.

- During the precharge phase, the clock signal is in '0' state. The 0 input signal causes the PMOS (M1) transistor to be turned on thus precharging the dynamic node, Z.

- During the evaluation phase, the clock signal is pulsed high. The high input signal causes the NMOS (M2) transistor to be turned on.

Thus, when the applied input signal (A) is low, the logic at the node Z is kept high regardless of the present operating phase. However, when the applied input signal (A) is high, the precharge and evaluation phase takes place.

As depicted in the Fig. 1.5, when the circuit is in precharge phase, node Z and node B will be charged up to V_{DD}. The voltage at node OUT will dropdown to '0'. This would lead to the propagation of the precharge phase value to the output of the buffer. This precharge pulse propagation from node Z via the static buffer leads to higher power dissipation. Moreover, the output state is not stable when the circuit is in precharge phase as thus the cascading characteristics and performance of the domino logic is limited [35] [36].

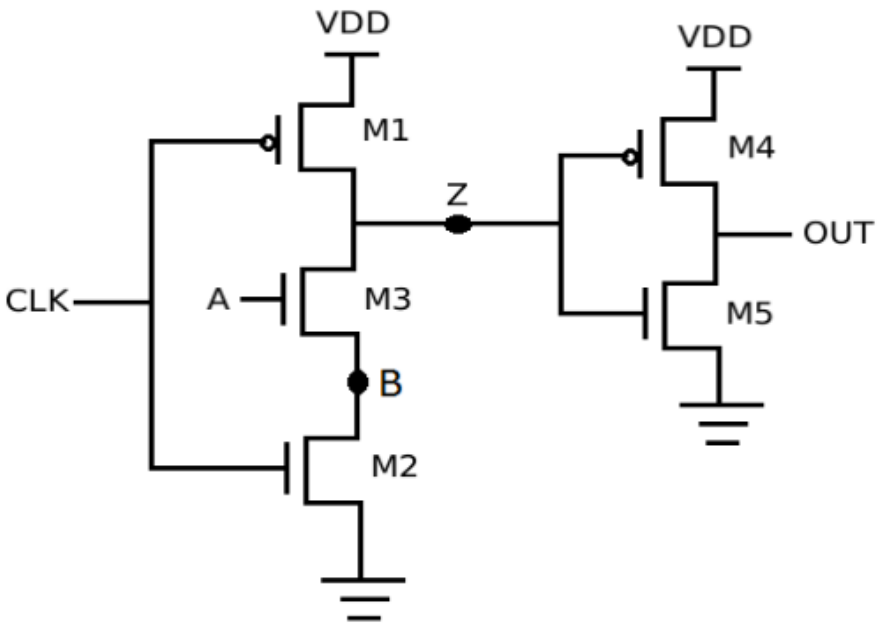

Figure 1.5 Domino buffer implementation

1.6 PSEUDO DYNAMIC BUFFER (PDB)

The main drawback that degrades the performance of conventional domino logic buffer is the precharge pulse propagation. The topology that overcomes this issue is PDB-based implementation that has been illustrated in Fig. 1.6. In this buffer implementation, the source of the buffer's NMOS transistor M5 is connected to node B instead of being connected to GND. By implementating a logic function using such

topology, the value present at node Z cannot propagate to the output node OUT during the precharge phase of the gate. This happens because during precharge phase, the evaluation transistor M2 is turned off. When the input logic A is low, the floating node Z is always in high state and then, the output node OUT is kept low regardless of the operating phase [37] [38]. On the other hand, if the input signal A is high, the precharge and evaluation phases will lead to the following operational conditions:

- During the evaluation phase, node Z and node B gets discharged to GND, resulting in enabling the PMOS transistor M4, while pulling up the output node voltage OUT to V_{DD}.

- During the precharge phase, node Z is charged upto V_{DD}, followed by the voltage at node B. Since the NMOS evaluation transistor M2 is disabled, the output node Z is held high (same value as the previous evaluation phase).

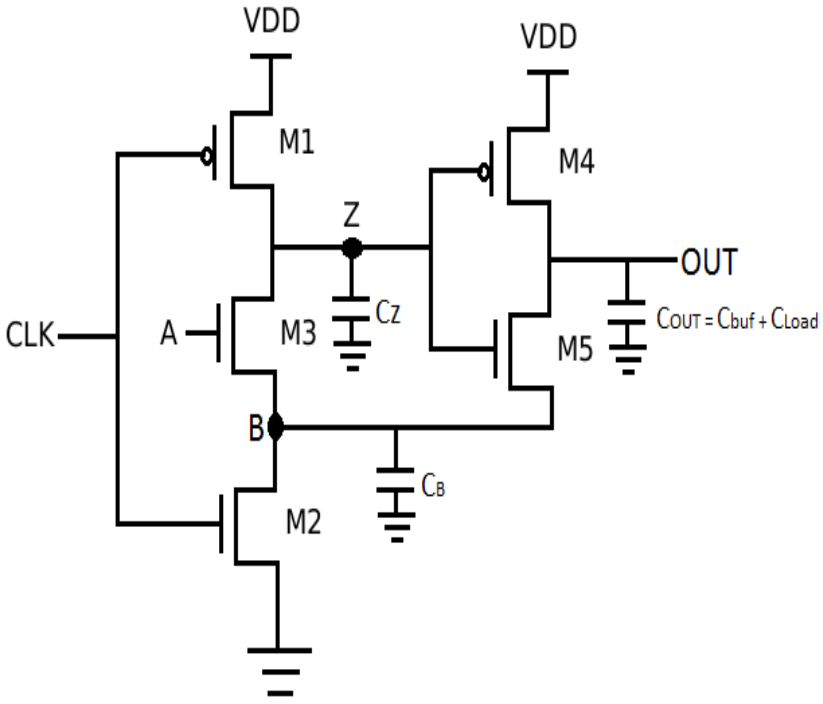

Figure 1.6 Domino logic circuit using pseudo dynamic buffer.

1.7 FOOTED QUASI RESISTANCE (FQR) MODEL

The major issue that occurs in PDB implementation is the cascading problem at the output node, when the input signal applied makes a transition from logic l to logic 0. Implementing logic functions using Footed Quasi Resistance (FQR) technique can overcome this issue. The structure for FQR implementation is as shown in Fig. 1.7 [39].

In this circuit, the logic function is being designed by using depletion PMOS and NMOS, both of which are being driven by same input node Y. When the node Y is at logic l logic 1, depletion PMOS is *OFF* thus making quasi resistance acting as open circuit, which is same as in case of logic 0 at node Y but at this time depletion NMOS is *OFF* which results same. This process can be explained as follows:

When the input signal (A) being applied is logic 1, then in precharge mode of operation, the node Y is connected to V_{DD} and in evaluation phase it is connected to ground which makes FQR work as PDB only.

When the input signal (A) being applied is logic 0, the footed quasi resistance functionality in evaluation and precharge phase can be explained as below:

- During precharge phase, both the transistors NM2 and NM5 are in cut-off condition which will lead node Y to act as an open circuit. This happens due to the fact that we are using depletion NMOS and PMOS in FQR which provides a path to discharge the output parasitic capacitance.

- During evaluation phase, both the NMOS transistors, NM2 and NM5 are turned on thus causing the output node to be at voltage 0.

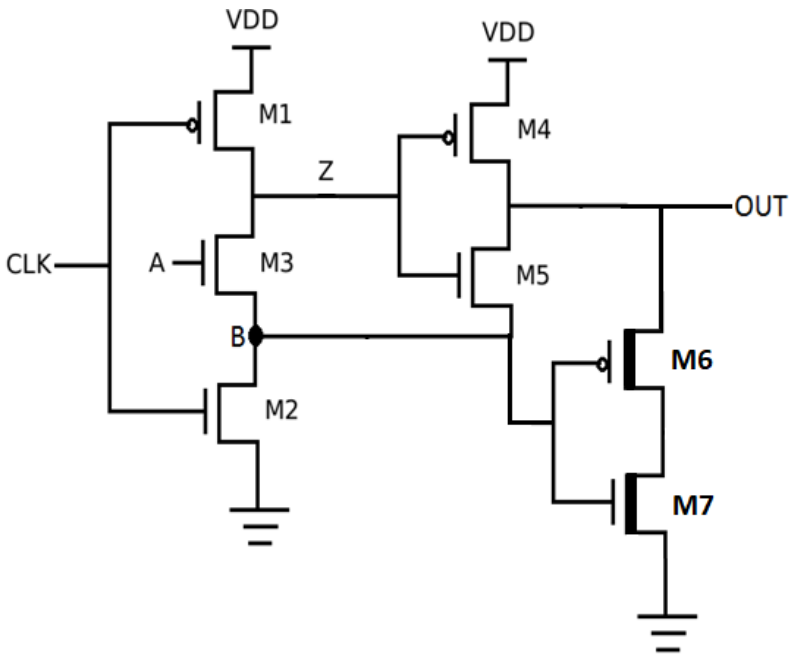

Figure 1.7 Domino logic circuit using footed quasi resistance

1.8 AND GATE

An AND gate is a digital logic gate which is provided with two or more input signals and as a result produces a single output that performs AND operation. The output would be high if all the applied inputs are high, i.e., logic '1'. The output would be low if any of the applied input is low, i.e., logic '1'.

$$\mathbf{Q = A.B} \qquad (1.1)$$

The AND gate basically performs multiplication operation which results in the output of the gate to be '0', when either or all the applied inputs are '0'.

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.1 Truth Table of 2-Input AND Gate

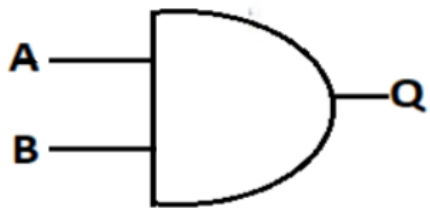

Figure 1.8.1 2-input AND Gate

The following characteristics can be derived from the AND gate:

- A.A = A
- 1.A = A
- 0.A = 0
- A.A' = 0
- A.B = B.A
- A.(B.C) = (A.B).C = A.B.C

1.9 OR GATE

The OR gate is a digital logic gate which is provided with two or more input signals and as a result produces a single output that performs OR operation. The output would be low if all the applied inputs are low, i.e., logic '0'. The output would be high if any of the applied input is high, i.e., logic '1'.

$$\mathbf{Q = A+B} \qquad (1.2)$$

The OR gate basically performs addition operation which results in the output of the gate to be '1', when either or all the applied inputs are '1'.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Table 1.2 Truth Table of 2-Input OR Gate

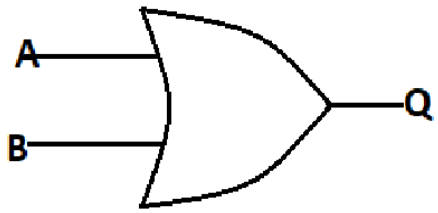

Figure 1.9.1 2-input OR Gate

The following characteristics can be derived from the AND gate:

- A+A = A
- 1+A = 1
- 0+A = A
- A+A' = 1
- A+B = B.A
- A+(B+C) = (A+B)+C = A+B+C

1.10 NAND GATE

The NAND gate is basically a digital logic gate which is provided with two or more input signals and as a result produces a single output that performs NAND operation. The output would be low if all the applied inputs are high, i.e., logic '1'. The output would be high if any of the applied input is low, i.e., logic '0'. It performs the complementary operation of that is implemented by AND Gate. The output of the NAND gate would be ‘0’ only when exactly both the inputs applied is ‘l’. If any of the input is ‘0’, then the output is also ‘1’.

The logic or boolean expression represents that the operation that is being executed by logic NAND is of logical addition and this operation is performed when the inputs applied are being complimented.

$$\mathbf{Q = (A.B)'} \qquad (1.3)$$

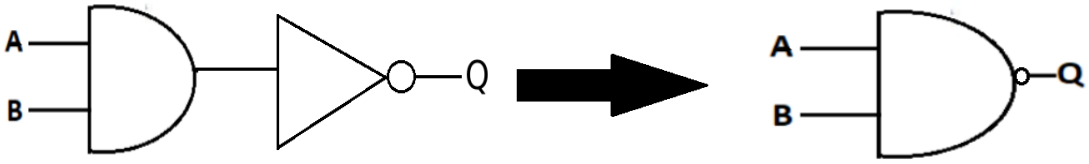

Figure 1.10 2-input NAND Gate

A	B	Q
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

Table 1.3 Truth Table of 2-Input NAND Gate

1.11 NOR GATE

The NOR gate is a digital logic gate which is provided with two or more input signals and as a result produces a single output that performs NOR operation. It performs the reverse or complementary operation of OR Gate. The output would be high in case all the inputs applied is low, i.e., logic '0'. The output would be low if any of the applied input is high, i.e., logic '1'.

The logic or boolean expression represents that the operation that is being executed by logic NOR is of logical multiplication and this operation is performed when the inputs applied are being complimented.

$$Q = (A+B)' \qquad (1.4)$$

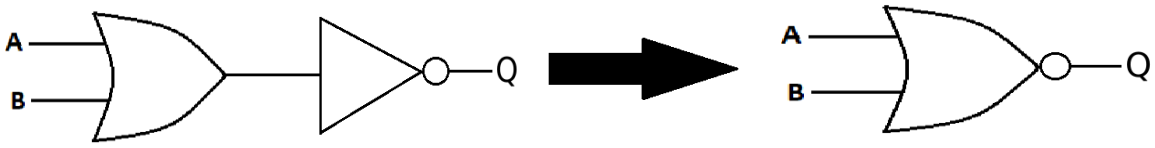

Figure 1.11 2-input NOR Gate

A	B	Q
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**0**

Table 1.4 Truth Table of 2-Input NOR Gate

1.12 XOR Gate

XOR gate (also called as Exclusive OR) represents a digital logic gate. This logic gate is provided with two or more inputs and thus produces one output that implements exclusive disjunction. The output that results would be '1' only when exactly one of its input signals is '1'. If both the inputs applied to an XOR gate are '0', or if both of its inputs are '1', then the resulting output would be '0'. However, in case that the inputs being applied to the XOR gate is greater than two, then the functioning of the gate would be determined by its implementation.

As observed from a wide range of scenarios, XOR gate will result in an output '1' in case the number of inputs '1' applied is odd. Thus, the output of the XOR gate only goes high when the inputs being applied to the logic gate are at different logic levels with respect to each other.

$$\mathbf{Q = (A \oplus B) = A.B' + A'.B} \qquad (1.5)$$

Ex-OR logic gate can be designed by the strategic combination of standard logic gates. These logic gates can be used for the implementation of complex design systems. Ex-OR logic gate with two input signals basically represents a modulo two adder, because it implements the addition of two binary numbers and hence are more complicated to design and implement as compared to standard logic gates [40].

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

Table 1.5 Truth Table of XOR Gate

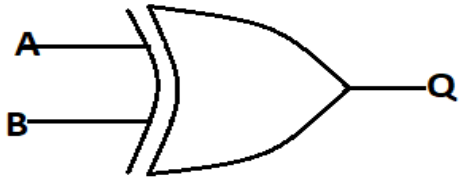

Figure 1.12 2-input Ex-OR Gate

XOR gate is one of the most widely used arithmetic unit and it is used in many VLSI applications such as microprocessors, adders, subtractors, shift registers etc. Design of XOR gate using static CMOS requires a pull-up and pull-down network, which leads to increased area requirement and higher power. Implementation of XOR gate using domino CMOS requires reduced layout area due to the elimination of pull-up network [41]. As the pull-up network in the design is eliminated, the parasitic capacitance at dynamic and output node get reduced which provides the designer with enhanced speed for domino logic XOR gate.

Due to the increased speed and low area requirement of domino circuit topology, it is used in various VLSI applications. Standard domino XOR gate inputs require two phase signals, one is original and the other is inverted signal. Application of inverted input consumes extra circuitry. This extra hardware not only increases the power consumption but also deviates the performance of the domino XOR circuit [42].

In conventional domino logic implementation, more power is dissipated. In order to overcome this problem, Pseudo dynamic buffer based domino logic and footed quasi resistance model has been introduced to implement the XOR circuit.

1.13 XNOR Gate

XNOR gate (also called as Exclusive NOR) represents a digital logic gate. This logic gate is provided with two or more inputs and thus produces one output that implements exclusive disjunction. The output that results would be '0' only when exactly one of its input signals is '1'. If both the inputs applied to an XOR gate are '0', or if both of its inputs are '1', then the resulting output would be '1'. However, in case that the inputs being applied to the XOR gate is greater than two, then the functioning of the gate would be determined by its implementation. While taking into consideration a vast range of scenarios, it has been observed that XNOR gate will result in an output '1' in case all inputs applied are at same logic level.

$$\mathbf{Q = (A \oplus B)' = A.B + A'.B'} \qquad (1.6)$$

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

Table 1.6 Truth Table of XNOR Gate

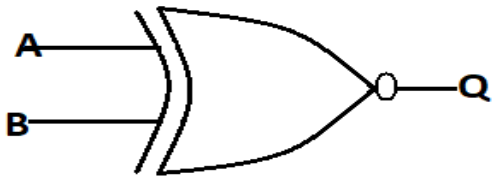

Figure 1.13 2-input XNOR Gate

1.14 HALF ADDER

Adder is a combinational digital circuit which has been used for the implementation of function of adding two numbers. The addition of two numbers eventually results in a sum bit (SUM) and a carry bit (CARRY) as the output. Typically, adders are realized to implement the addition of binary numbers, however they can also be realized for executing addition operation for other formats such as BCD (binary coded decimal), Excess-3 etc. Adder finds use in a wide range of other applications such as in digital electronics like address decoding, table index calculation etc. [43].

Half adder is a combinational arithmetic circuit that performs addition of two input numbers and hence produces two outputs: sum bit (SUM) and a carry bit (CARRY). Half adder is the simplest of all adder circuit, but it suffers from a major drawback. The half adder can implement addition of only two input bits (A and B) and can take no action if any carry is present as the input. Therefore, if a carry input is provided to the half adder, then the input carry bit will be neglected and this poses a serious problem. This implies that the process of binary addition is incomplete and hence it is known as half adder.

$$\mathbf{SUM = (A \oplus B) = A.B' + A'.B} \quad (1.7)$$

$$\mathbf{CARRY = A.B} \quad (1.8)$$

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 1.7 Truth Table of HALF ADDER

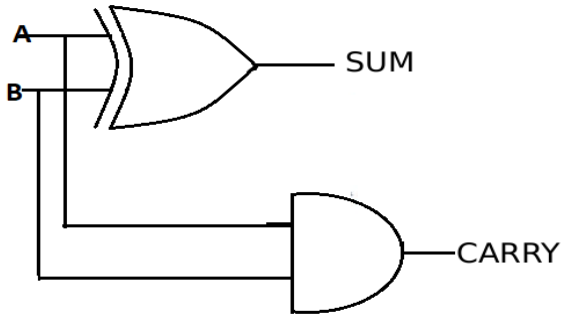

Figure 1.14 HALF ADDER

1.15 HALF SUBTRACTOR

In the process of binary subtraction, the process of subtracting two numbers is similar to that of arithmetic subtraction. While subtracting two numbers in arithmetic subtraction the base 2 number system is used whereas while performing binary subtraction, binary numbers are being used for subtraction [44]. The resultant terms obtained are given as the difference and borrow.

Half subtractor is a combinational arithmetic circuit that implements the operation of subtraction of two numbers and produces two outputs: a difference bit (DIFFERENCE) and a borrow bit (BORROW).

$$\mathbf{DIFFERENCE = (A \oplus B) = A.B + A'.B} \quad (1.9)$$

$$\mathbf{BORROW = A'.B} \quad (1.10)$$

A	B	DIFF	Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Table 1.8 Truth Table of half subtractor

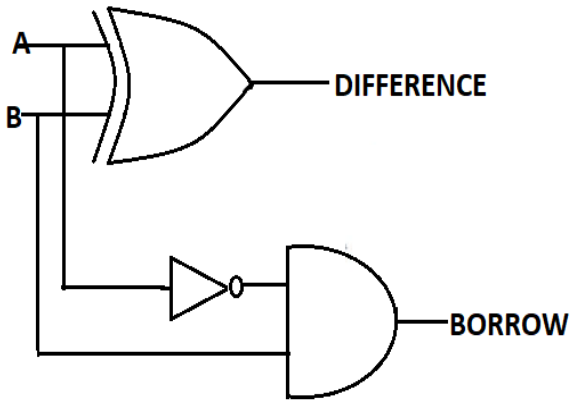

Figure 1.15 Half Subtractor

1.16 1-BIT FULL ADDER

Adder is a combinational digital circuit which has been used for the implementation of function of adding numbers. In several computer systems and different types of operating processors, adders are used for the calculation of addresses and to perform other arithmetic functions. Typically, adders are realized to implement the addition of binary numbers, however they can also be realized for executing addition operation for other formats such as BCD (binary coded decimal), Excess-3 etc. [44].

1-bit Full adder is a combinational arithmetic circuit which performs addition of three input numbers and hence results in two outputs: a sum bit (SUM) and carry bit (C_{out}). The full adder circuit has three inputs: A and B, and an input carry given as C_{in}. The addition of these three inputs results in a sum bit (SUM) and an output carry (C_{out}).

$$\mathbf{SUM = (A \oplus B \oplus C_{in})} \quad (1.11)$$

$$\mathbf{C_{out} = (A.B)+(B.C_{in})+(A.C_{in})} \quad (1.12)$$

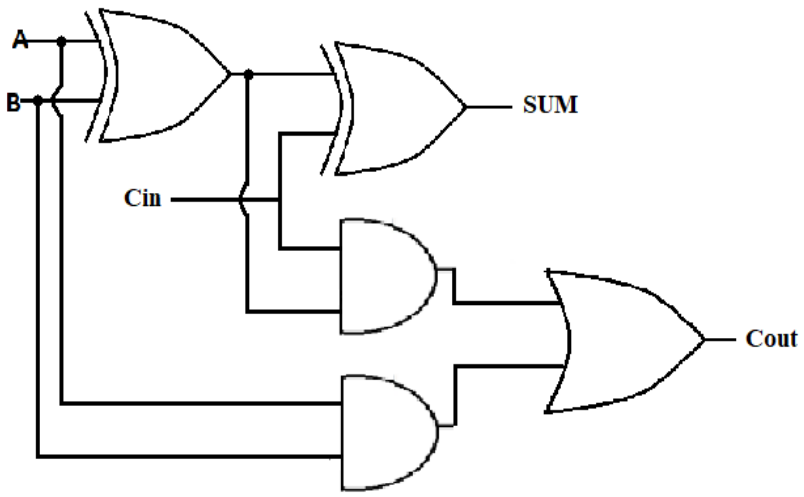

Figure 1.16 1-bit Full Adder

With the help of truth-table, the full adder logic can be implemented. From the above equation, it can be derived that the SUM output is obtained by performing XOR between the three input signals. The output carry, C_{out} will be true only under the condition when any two of the three applied input signals are HIGH.

A	B	C_{in}	SUM	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 1.9 Truth Table of Full Adder

A 2-bit full adder is basically a ripple carry adder. In this logic circuit two full adders are cascaded to execute the addition of 2-bit numbers. The output carry generated from the first full adder acts as in input carry for the subsequent full adder.

Therefore, final carry is generated by the second full adder. As a result, we obtain two sum bits (S1 and S2) and one carry bit (C_o2) as the output.

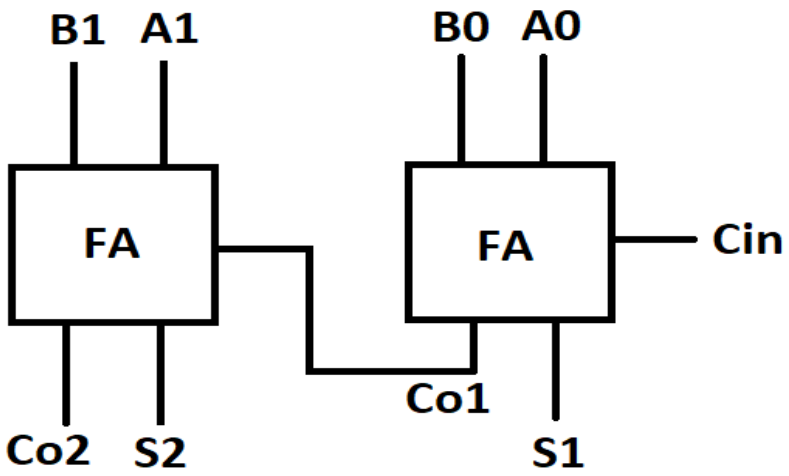

Figure 1.17 2-bit Full Adder

1.17 1-BIT FULL SUBTRACTOR

A full subtractor is a combinational logic function that executes the process of subtracting one bit from the another. Subtraction is implemented between two bits: minuend and subtrahend. While subtraction, the borrow that has generated from the previous adjacent lower minuend bit is also taken into consideration. 1-bit full subtractor is a combinational arithmetic circuit which performs addition of three input numbers and hence results in two outputs: a difference bit (DIFF) and borrow bit (B_{out}). The full subtractor circuit has three inputs: A and B, and an input borrow given as Bin.

$$\mathbf{DIFF = (A \oplus B \oplus B_{in})} \quad (1.13)$$

$$\mathbf{B_{out} = A'B_{in} + A'B + BB_{in}} \quad (1.14)$$

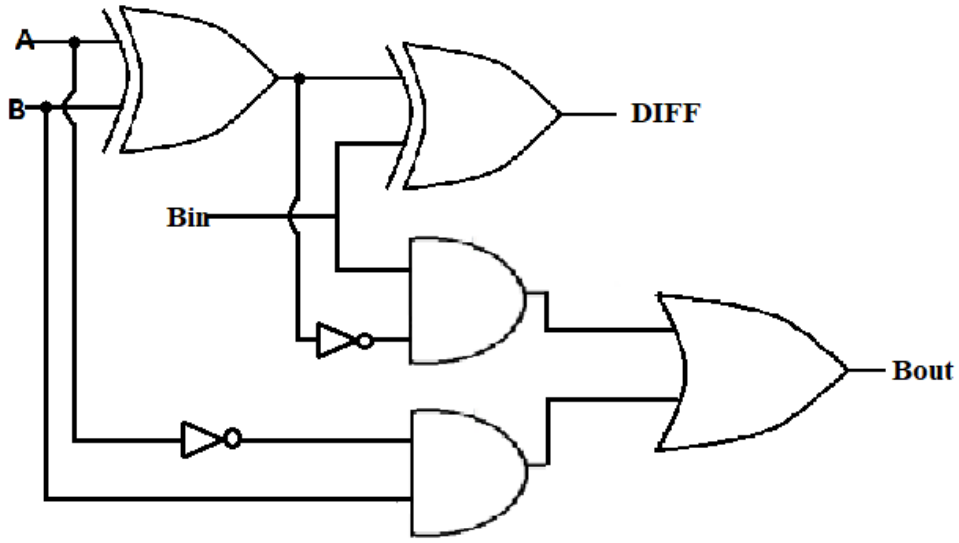

Figure 1.18 1-bit Full Subtractor

A	B	B_{in}	DIFF	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table 1.10 Truth Table of Full Subtractor

1.18 INCREMENTER

Incrementing is a micro-operation which executes the operation of adding one binary value to the applied input value. For example, if a 3-bit register has a value of 011, then when incremented by one will result in a value of 100 as the output.

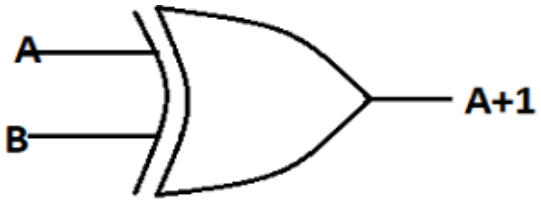

Figure 1.19 Implementation of A+1 State

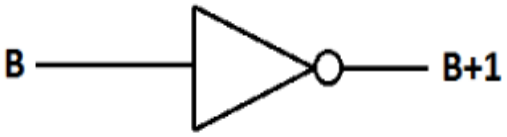

Figure 1.20 Implementation of B+1 State

PRESENT STATE		NEXT STATE	
A	**B**	**(A+1)**	**(B+1)**
0	**0**	**0**	**1**
0	**1**	**1**	**0**
1	**0**	**1**	**1**
1	**1**	**0**	**0**

Table 1.11 Truth table for 2-bit Incrementer

1.19 DECREMENTER

Decrementing is a micro-operation which executes the operation of subtracting one binary value from the applied input value. For example, if a 3-bit register has a value of 100, when this value is decremented by one, it will result in a value of 011 as the output.

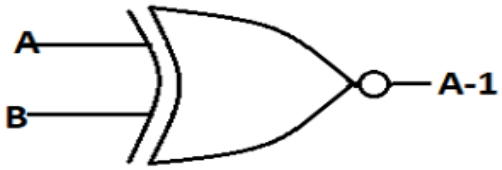

Figure 1.21 Implementation of A-1 State

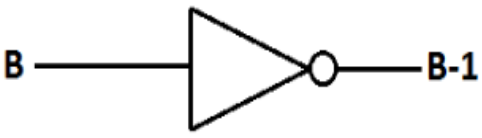

Figure 1.22 Implementation of B-1 State

PRESENT STATE		NEXT STATE	
A	**B**	**(A-1)**	**(B-1)**
0	**0**	**1**	**1**
0	**1**	**0**	**0**
1	**0**	**0**	**1**
1	**1**	**1**	**0**

Table 1.12 Truth table for 2-bit decrementer

1.20 MOTIVATION

Addition of static inverter is used in the domino logic circuit to eliminate the logic 1 (in precharge phase) to logic 0 (in evaluation phase) transition. This elimination of transition from logic 1 to logic 0 signifies that additional charging and discharging is performed at the output capacitance of static inverter when it is in precharge phase and meanwhile the output of its subsequent evaluation phase is logic 0, which causes significant power dissipation.

This problem is solved through various techniques, whereby the pseudo-dynamic buffer model significantly reduces power dissipation and hence saves power. However, in the PDB model we have logic 1 (in precharge) to logic 0 (in evaluation) transition which results in faulty output when used for designing of cascaded circuits. As compared to the PDB model, FQR topology solves the problems caused due to cascading. Moreover, FQR model has a lower power dissipation as compared to domino logic model.

In this work, various arithmetic logic functions are implemented by using the above three techniques and their corresponding delay and power consumption is evaluated using 180nm technology using VIRTUOSO CADENCE.

CHAPTER2

LITERATURE SURVEY

2.1 DYNAMIC LOGIC CIRCUITS

As explained by J. M. Rabaey [45] dynamic logic circuits have several advantageous features such as low device count, high speed, elimination of short circuit power and glitch free operation. All these features enable dynamic logic circuits to be used for wide range of applications in high speed, low power areas such as microprocessors, digital signal processing, dynamic memories etc. Moreover, a dynamic logic unit that is smaller than its static equivalent can also be designed.

Dynamic logic circuits provide enhanced performance for higher fan-in and complex logic circuit design. As the level of integration of design increases, lower power dissipation and higher speed becomes imperative requirements for the design of logic functions along with improved performance [46] [47] [48].

During the digital implementation of high density and high-performance circuits, it is imperative to reduce the circuit delay as well as the silicon area of the design. In such implementations, dynamic logic circuits are significantly advantageous over static logic circuits.

There are two phases of functionality of dynamic logic:

1) Precharge phase: In this phase, irrespective of the input values applied, output rises to an extremely high value. The load capacitance gets charged in this phase.

2) Evaluation Phase: In this phase, there exist a path between the GND and the output terminal. This causes the load capacitance to be discharged [49].

The major issue faced while using dynamic logic is that the second logic gate of the dynamic logic design may discharge prematurely, before arriving at the first gate when the first logic gate is in precharge state. The occurrence of this operating condition causes the precharge state of the second logic gate to be used up and it can't be restored until the next clock cycle. Therefore, to be able to implement cascaded logic circuits, the one solution is Domino Logic. In domino logic, a static inverter has been inserted between the logic states [50] [51].

2.2 Domino Logic Buffer

As depicted in the Fig. 2.1, when the circuit is in precharge phase, node Z and node B will be charged up to V_{DD}. The voltage at node OUT will dropdown to '0'. This would lead to the propagation of the precharge phase value to the output of the buffer. This propagation of the precharge phase pulse from node Z through the static buffer leads to high power consumption. Moreover, the output state is not stable when the circuit is in precharge phase as thus the cascading characteristics and performance of the domino logic is limited [52].

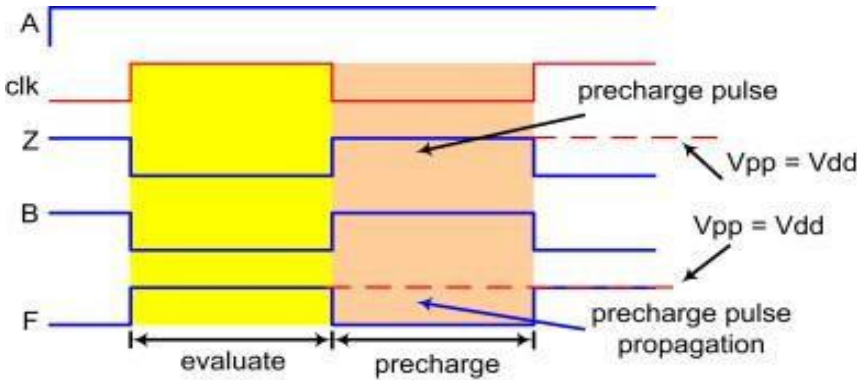

Figure 2.1 Timing diagram of the domino logic circuit [52]

The conventional domino logic buffer illustrates the issue of performance degradation which occurs due to the propagation of the precharge pulse inherent in domino logic gates. The PDB-based implementation overcomes this problem using the circuit structure shown in Fig. 2.2. In this buffer implementation, the source of the buffer's NMOS transistor M5 is connected to node B instead of being connected to GND. By implementing a logic function using such topology, the value present at node Z cannot propagate to the output node OUT during the precharge phase of the gate. This happens because during precharge phase, the evaluation transistor M2 is turned off. When the input logic A is low, the floating node Z is always in high state and then, the output node OUT is kept low regardless of the operating phase. On the other hand, if the input signal A is high, the precharge and evaluation phases will be executed [53] [54] [55].

2.3 FAULTY OUTPUT IN CASCADED CIRCUITS: PDB TOPLOGY

Implementation of logic functions using PDB topology leads to a problem that occurs when logic 0 is applied as input after logic l in evaluation phase. This condition arises because the charge accumulated on the output parasitic capacitance in evaluation phase ,when input A is logic l, does not have a discharge path in precharge phase when input A is logic 0. The precharge pulse propagation is overcome by using PDB design technique. However, it catches on a major problem: candid scrutiny of pseudo dynamic buffer logic to design more complex boolean functions results in faulty output logic. The issue can be best illustrated with the two cascaded pull down n-type pseudo dynamic buffer circuits, as shown in Fig. 2.2. During the precharge phase (i.e., the precharge phase after when logic is high in evaluation phase), the output of both buffers would be charged to V_{DD}. Now let us consider that the primary input signal (A) applied makes a transition from logic l to logic 0. On the rising edge of the clock signal, the output O_1 begins to discharge. The final output O_2 should be logic low as the expected input to the circuit now is logic low. However, in practice, there is a small propagation delay for the input to discharge O_1 to logic low. Therefore, the precharged voltage at node Z begins to discharge. By the time O_1 crosses the threshold voltage of the transistor Q7, the node voltage present at node Z

gets discharged through the transistors Q7 and Q8. This discharging makes the final output as logic high. Thus, the correct logic level can't be retrieved due to the fact that PDB relies on parasitic capacitance voltage. This situation leads to erroneous output logic in cascaded PDB logic circuits. This problem arises because of logic high to logic low transition at output which can be resolved by implementing logic functions using Footed Quasi Resistance (FQR) technique [56] [57].

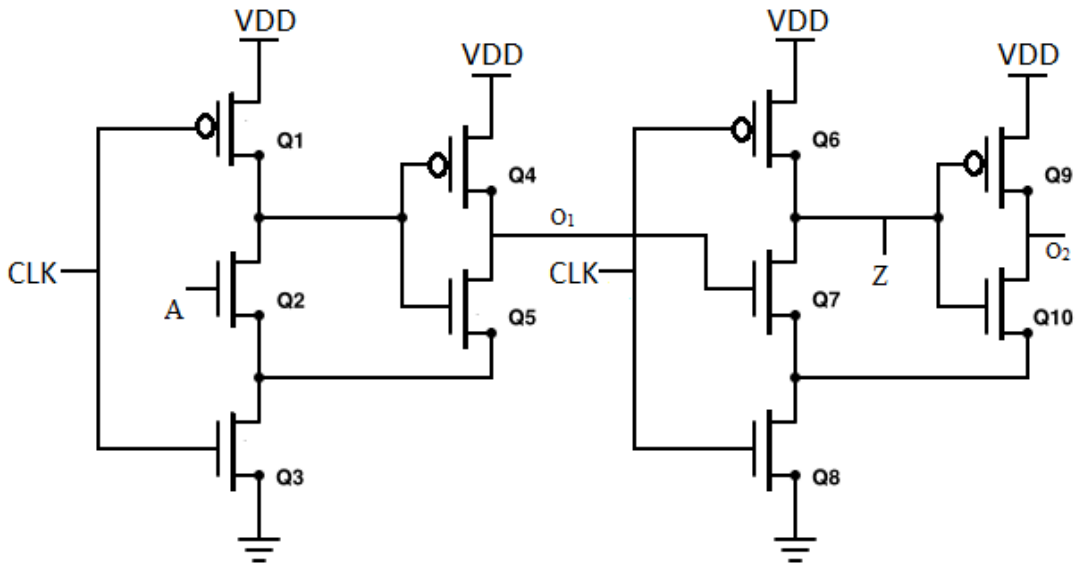

Figure 2.2 Cascaded Buffer using PDB Model

2.4 FOOTED QUASI RESISTANCE(FQR) MODEL

As illustrated in the paper [39], the major issue that occurs in PDB implementation is the cascading problem at the output node, when the input signal applied makes a transition from logic l to logic 0. Implementing logic functions using Footed Quasi Resistance (FQR) technique can overcome this issue. The structure for FQR implementation is as shown in Fig. 1.7. In this circuit, the logic function is being designed by using depletion PMOS and NMOS, both of which are being driven by same input node Y. When the node Y is at logic l logic 1, depletion PMOS is OFF thus making quasi resistance acting as open circuit, which is same as in case of logic 0 at node Y but at this time depletion NMOS is OFF which results same [39].

CHAPTER3

STUDY AND SIMULATION

3.1 CHARACTERIZATION OF DOMINO LOGIC BUFFER

Domino logic operates in two phases:
(i) Precharge phase
(ii) Evaluation phase

The above two operational phases can be esentially differentiated by the CLK signal. In case the applied CLK signal is at *logic 0*, the PMOS transistor PM1 is turned *ON* which allows the charging of the parasitic capacitance present at node X to V_{DD}. This mode of operation is precharge mode.

When the applied CLK signal is at *logic 1*, the NMOS transistor NM5 is turned *ON*. The voltage at node X depends on the value of the input signal applied. This mode of operation is precharge mode. When the applied input signal, A, is *logic 1* the node voltages are as shown in the Fig. 3.2 for both, precharge and evaluation, phases. In evaluation phase, the value which is present at the output node will get charged to V_{dd} which will then discharge in precharge phase leading to precharge propagation at output buffer. Therefore, this propagation of the precharge phase pulse from node Z through the static buffer leads to high power consumption. This precharge pulse propagation can be solved by using different techniques such as true single phase CLK model, pseudo dynamic buffer.

The VIRTUOSO schematic of domino logic Buffer is illustrated in Fig. 3.1. The timing waveform of Domino Logic, when input logic A is logic 1 is shown in Fig. 3.2.

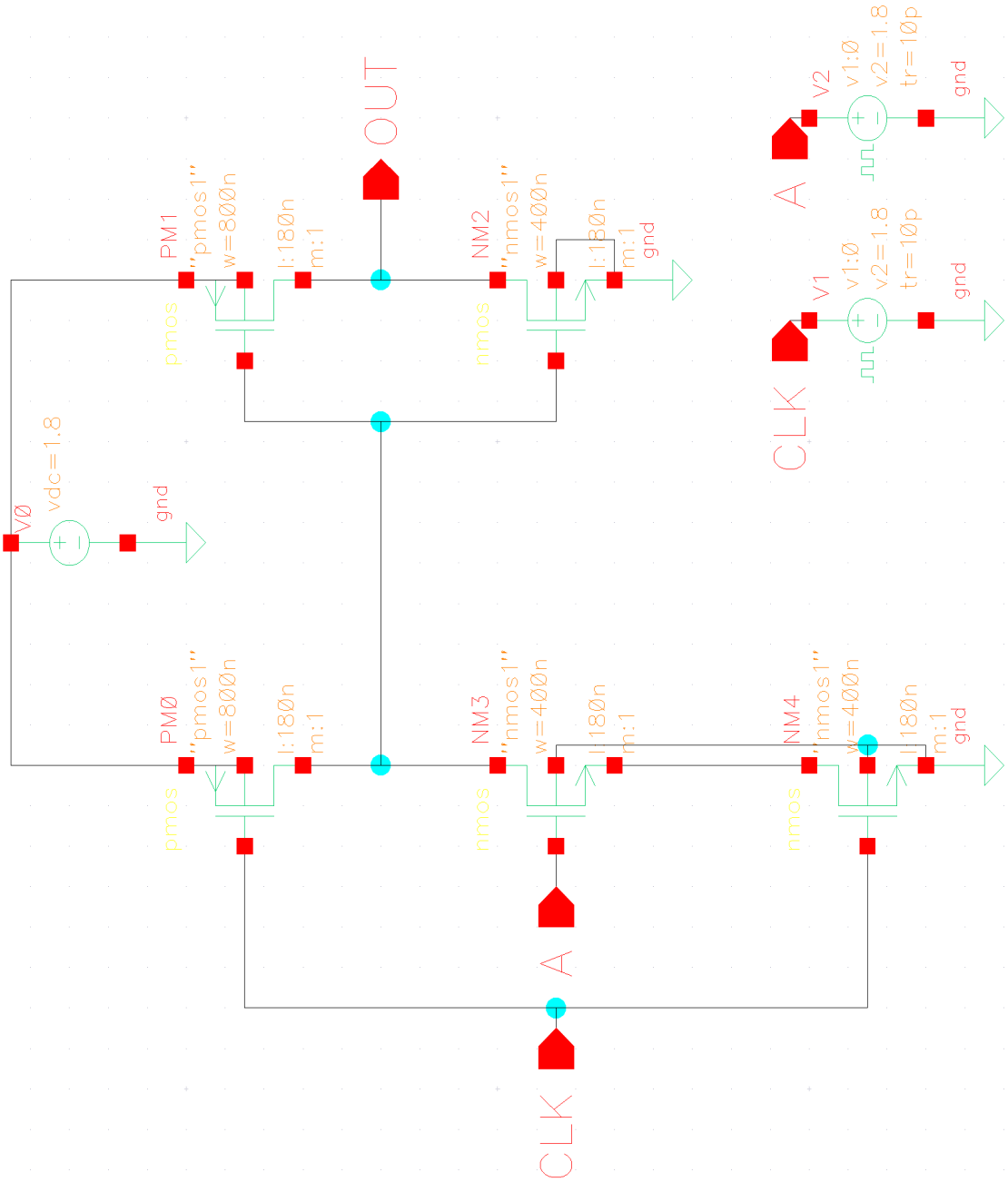

Figure 3.1 Virtuoso Schematic of Domino Logic Buffer

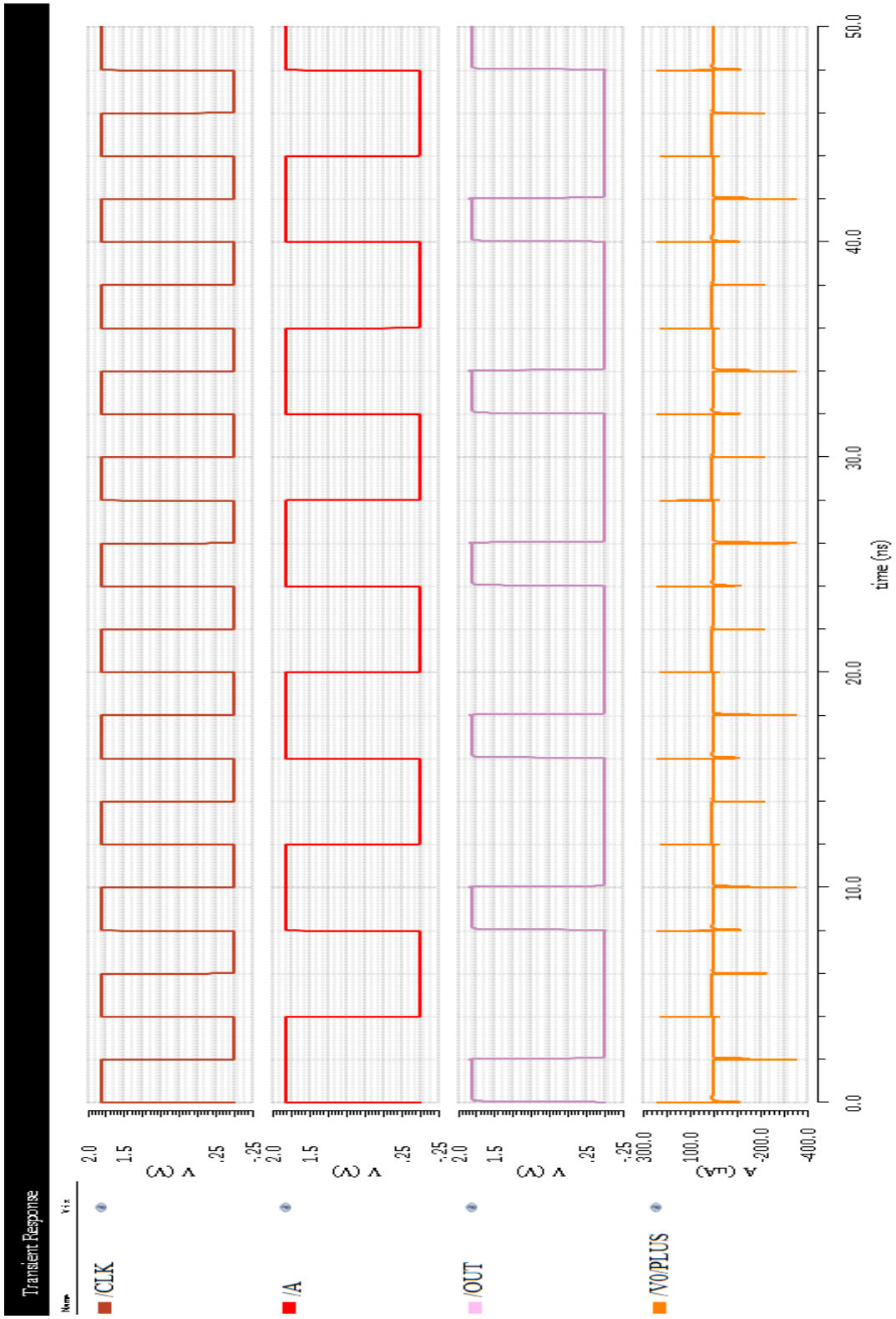

Figure 3.2 Timing Waveform of Conventional Domino Logic Buffer

3.2 CHARACTERIZATION OF PSEUDO DYNAMIC BUFFER DOMINO LOGIC

In PDB model, the source of the NMOS transistor NM3 in output inverter is connected to node Y instead of being connected to ground as shown in Fig. 3.3 For this particular model, when input A is *logic 1*, evaluation and precharge will lead to following case:

- During evaluation phase, node X will discharge to *logic 0* which makes PMOS transistor to turn *ON* and output parasitic capacitance charge to V_{DD}.
- In precharge operational mode, output node X charges to a value of VDD as of node Y which results in NMOS transistor NM3 to be turned *OFF*.

The NMOS transistor NM3 operates as following in both the phases. In precharge mode, previously source of NM3 is connected to GND. The presence of logic high at node X leads to the NMOS transistor NM3 to switch ON because V_g-V_s (V_{gs}) is greater than threshold voltage (V_{th}). In the PDB topology, the source of the NMOS transistor NM3 is connected to node Y which causes the operation of the NMOs transistor NM3 to depend on the value present at the node X. During the precharge phase, when the CLK signal is low, transistor NM5 turns on thus causing the source of the NM3 transistor to be connected to GND, similar to what happens in domino logic buffer topology. However, when the CLK signal applied is high, the operation of the transistor NM3 depends upon both, voltage at node X and input logic applied for domino logic circuit [39].

The VIRTUOSO schematic of pseudo domino logic Buffer is illustrated in Fig. 3.3. The timing waveform of Pseudo Domino Logic, for logic 1 to logic 0 transition is displayed in Fig. 3.4.

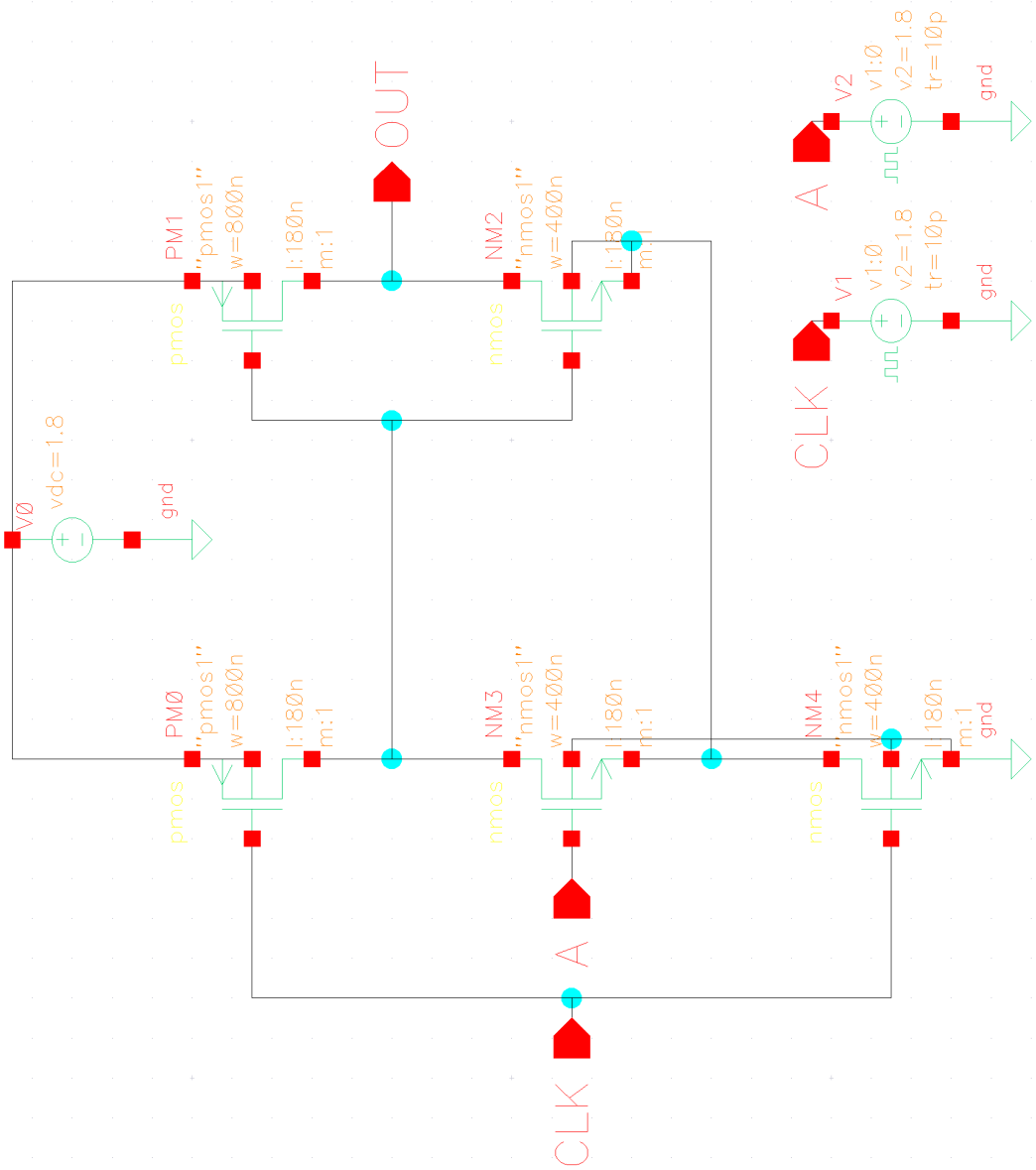

Figure 3.3 PDB Domino Logic Buffer Implementation

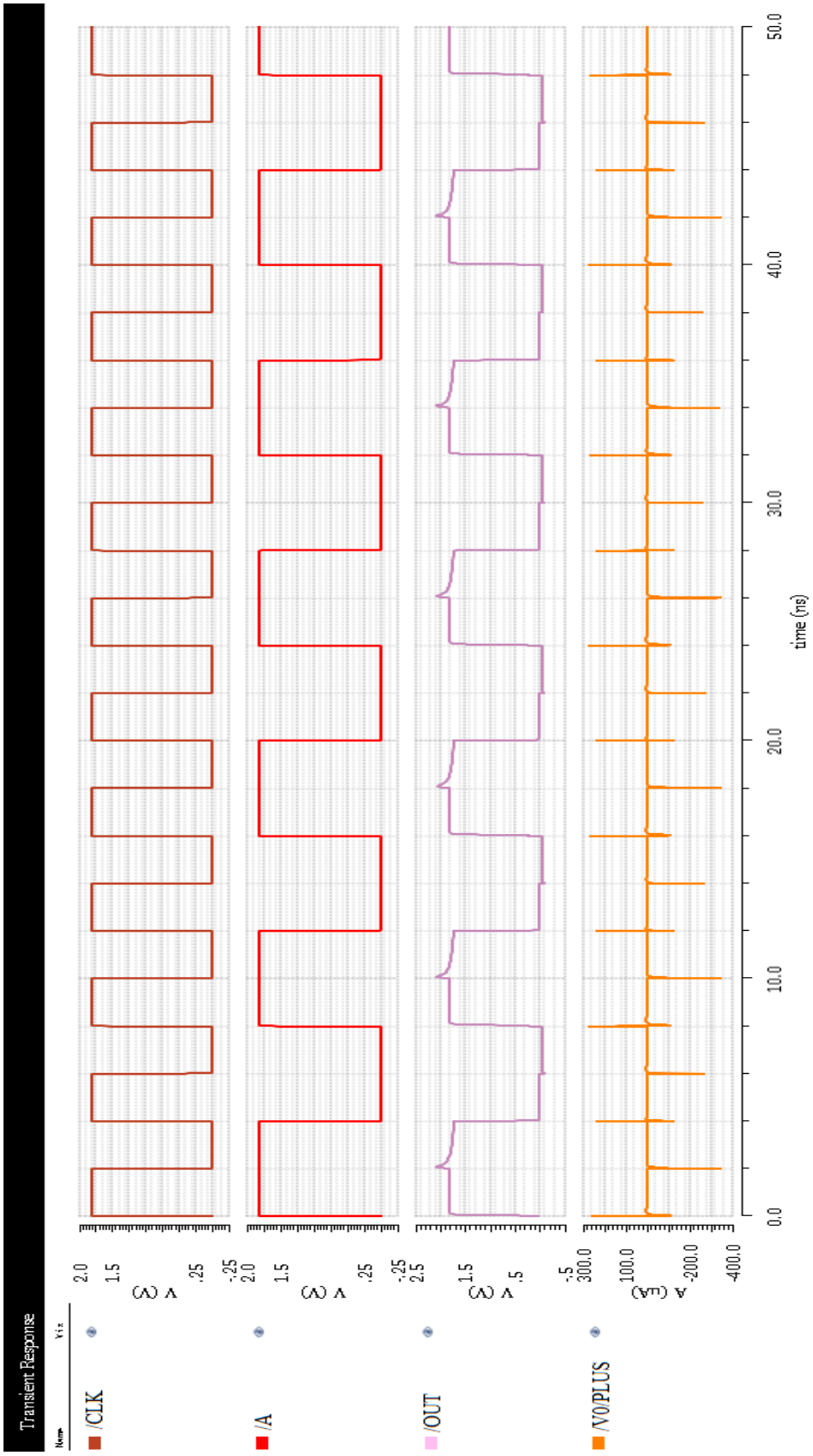

Figure 3.4 Timing Waveform of Pseudo Domino Logic

Implementation of logic functions using PDB topology leads to a problem that occurs when *logic 0* is applied as input after logic l in evaluation phase which is shown in Fig 3.3. This condition arises because output parasitic capacitance charge accumulated in evaluation phase when input A is logic l does not have a discharge path in precharge phase when input A is *logic 0*. Due to this, logic l to logic 0 transition is not a suitable input for dynamic logic circuits.

The precharge pulse propagation is overcame by using PDB design technique. However, it catches on a major problem: pseudo dynamic buffer logic to design more complex boolean functions results in faulty output logic. This situation leads to erroneous output logic in cascaded PDB logic circuits. This problem arises because of logic high to logic low transition at output which can be resolved by implementing logic functions using Footed Quasi Resistance (FQR) technique.

3.3 CHARACTERIZATION OF FOOTED QUASI RESISTANCE MODEL

When the input signal (A) being applied is *logic 1*, then in precharge phase, the node Y is connected to V_{DD} and in evaluation phase it is connected to ground which makes footed quasi resistance work as PDB only. When the input signal(A) being applied is *logic 0*, the footed quasi resistance functionality in evaluation and precharge phase can be explained as below [39]:

- During precharge phase, both the transistors NM2 and NM5 are in cut-off condition which will lead node Y to act as an open circuit. This happens due to the fact that we are using depletion NMOS and PMOS in FQR which provides a path to discharge the output parasitic capacitance.

- During evaluation phase, both the NMOS transistors, NM2 and NM5 are turned on thus causing the output node to be at voltage 0.

The VIRTUOSO schematic of FQR Buffer is illustrated in Fig. 3.5. In Fig. 3.6 output voltage waveform is given where we can observe logic l to 1ogic 0 transition is removed.

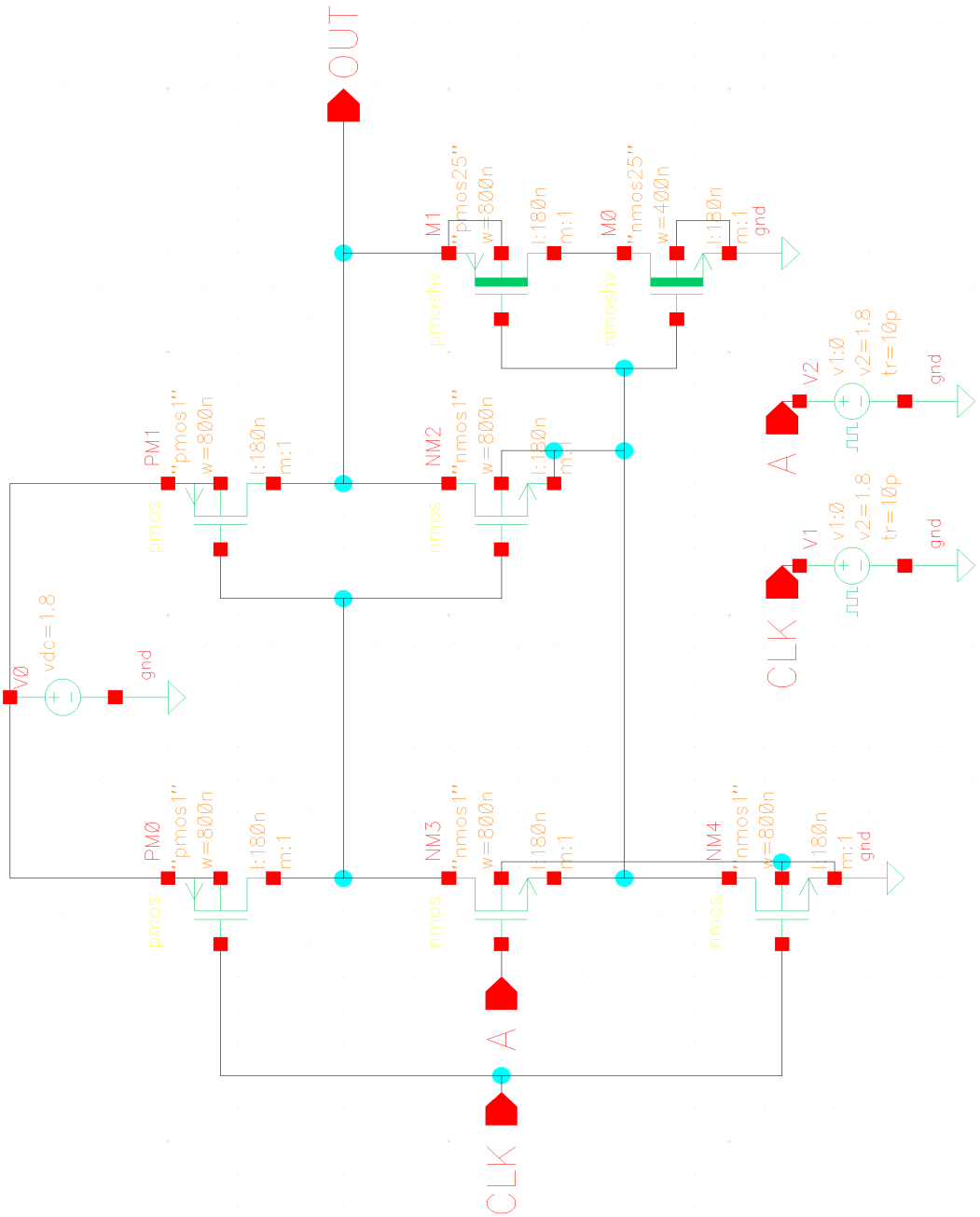

Figure 3.5 VIRTUOSO schematic of FQR Buffer

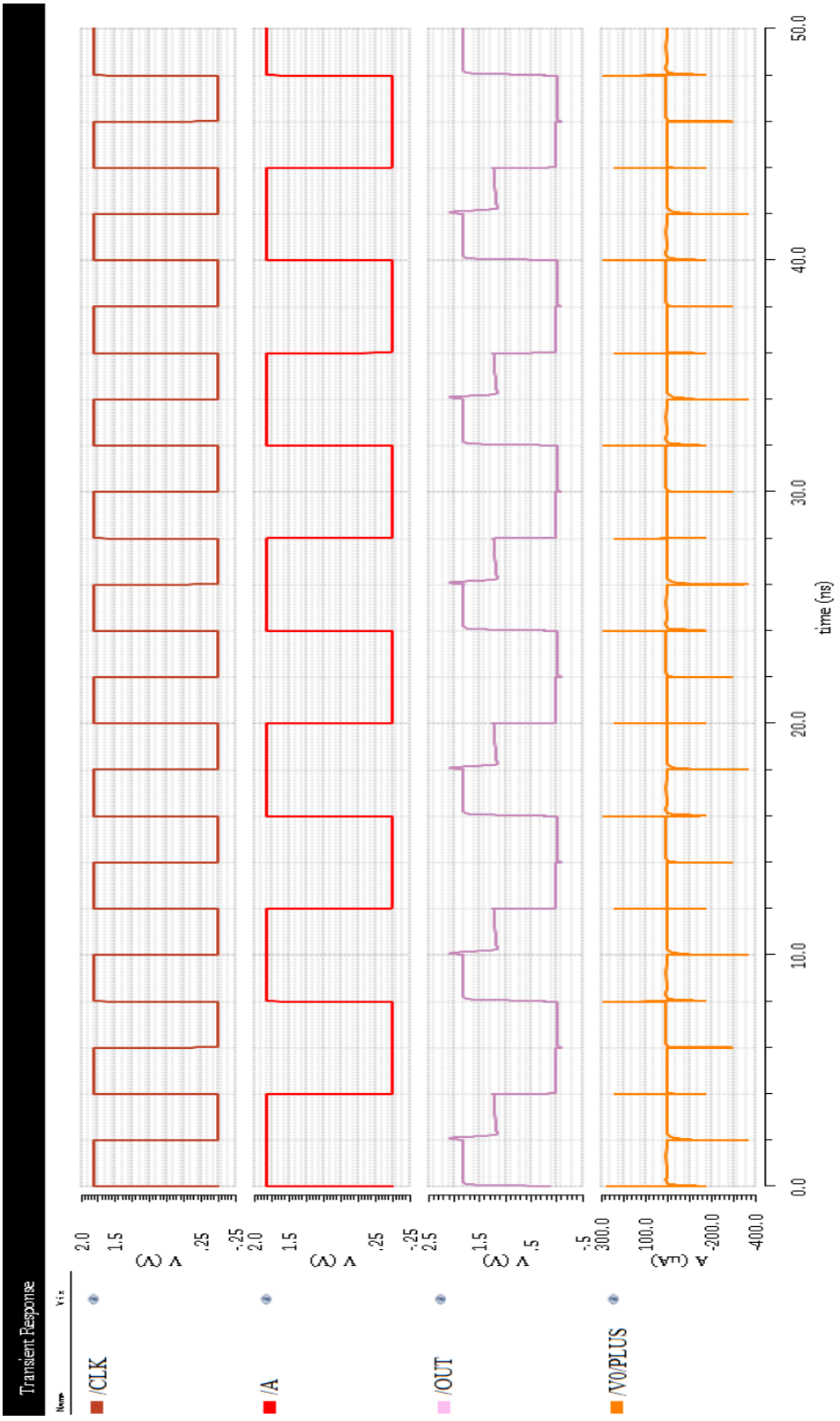

Figure 3.6 Timing Waveform of FQR Buffer

3.4 CHARACTERIZATION OF AND GATE

3.4.1 AND Gate using domino logic

The VIRTUOSO schematic of AND Gate using domino logic is illustrated in Fig. 3.7. The Timing Waveform of AND Gate, is illustrated in Fig. 3.8.

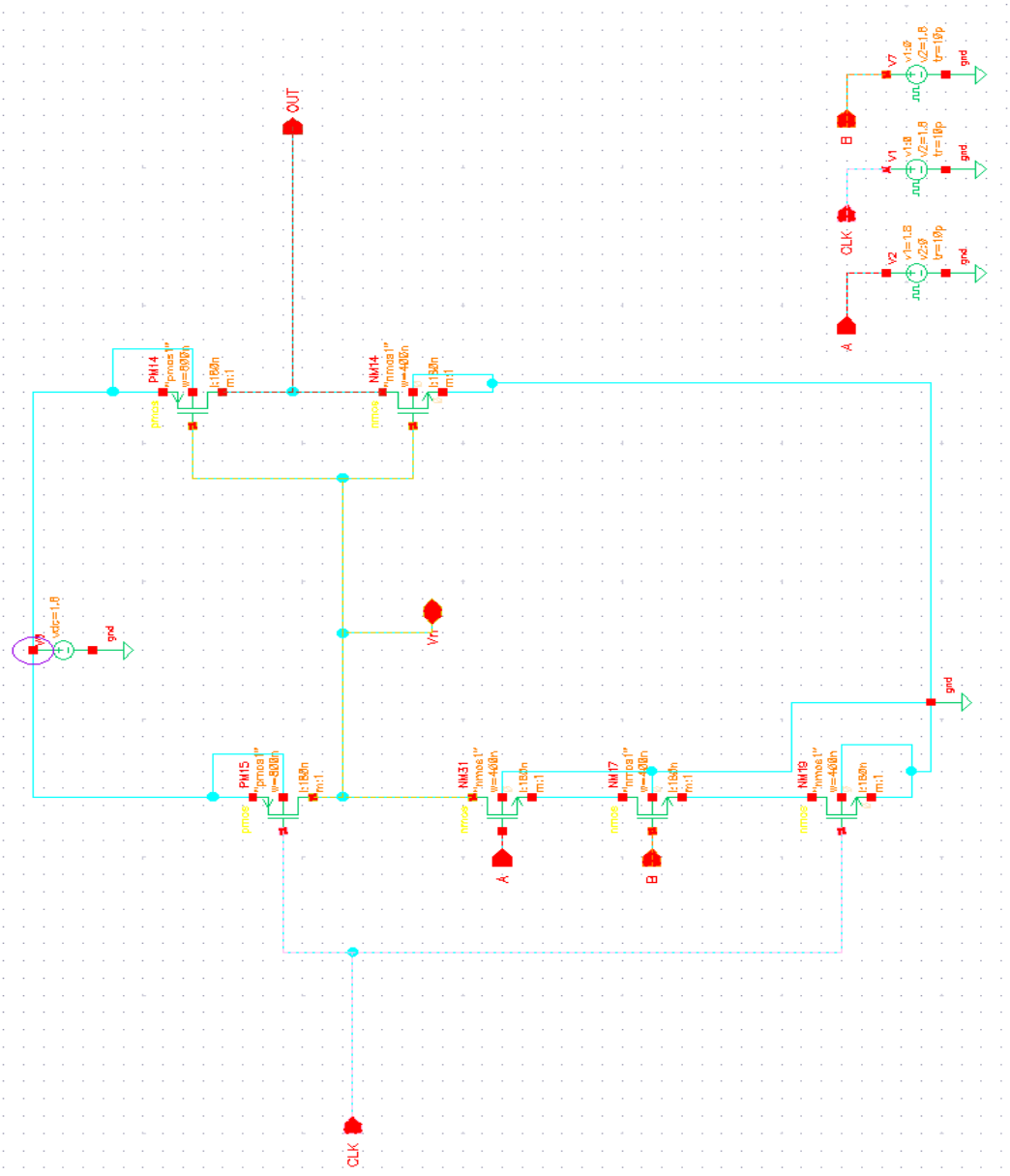

Figure 3.7 VIRTUOSO Schematic of AND Gate using domino logic

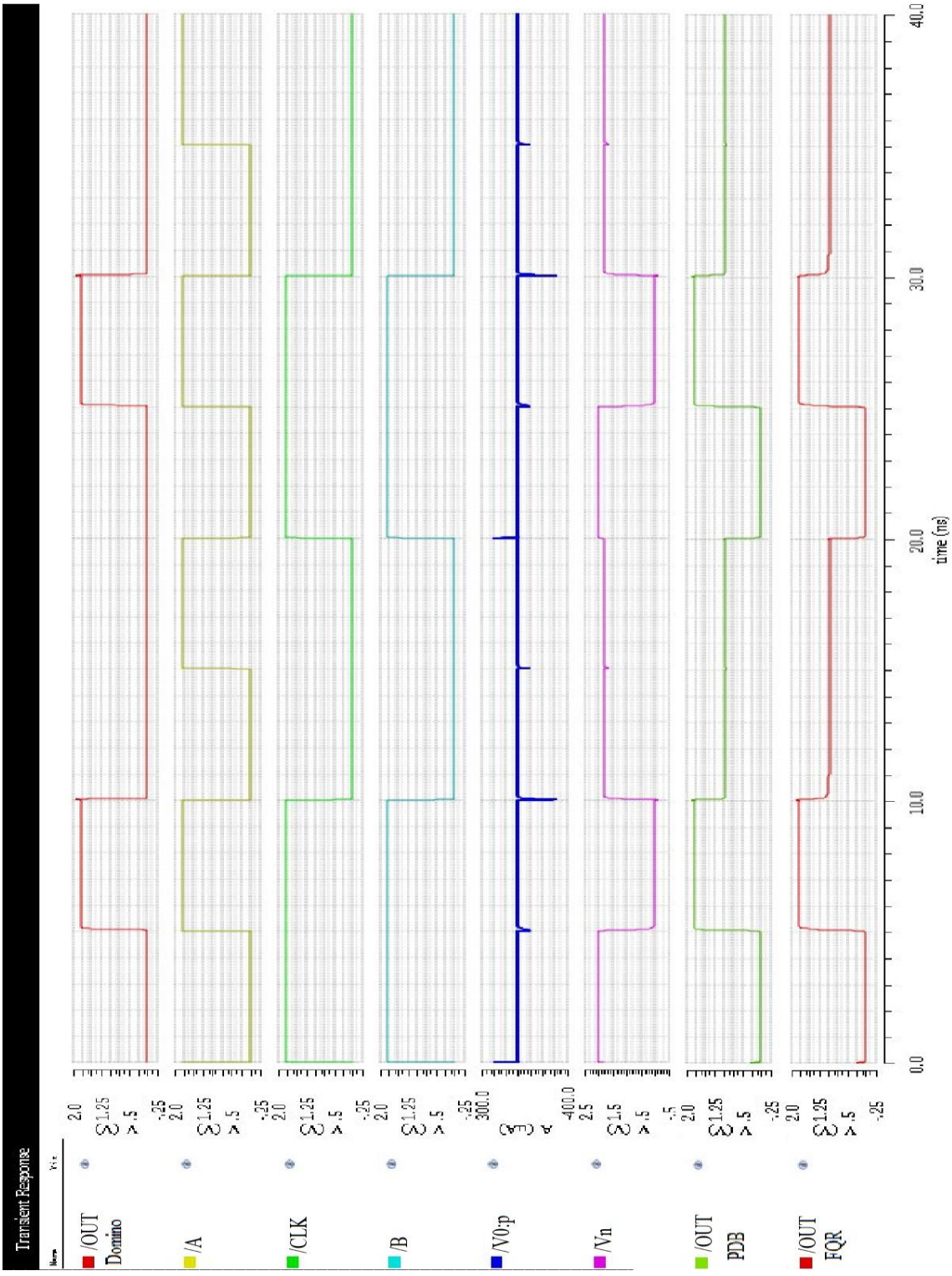

Figure 3.8 Timing Waveform of AND Gate

3.4.2 AND Gate using PDB logic

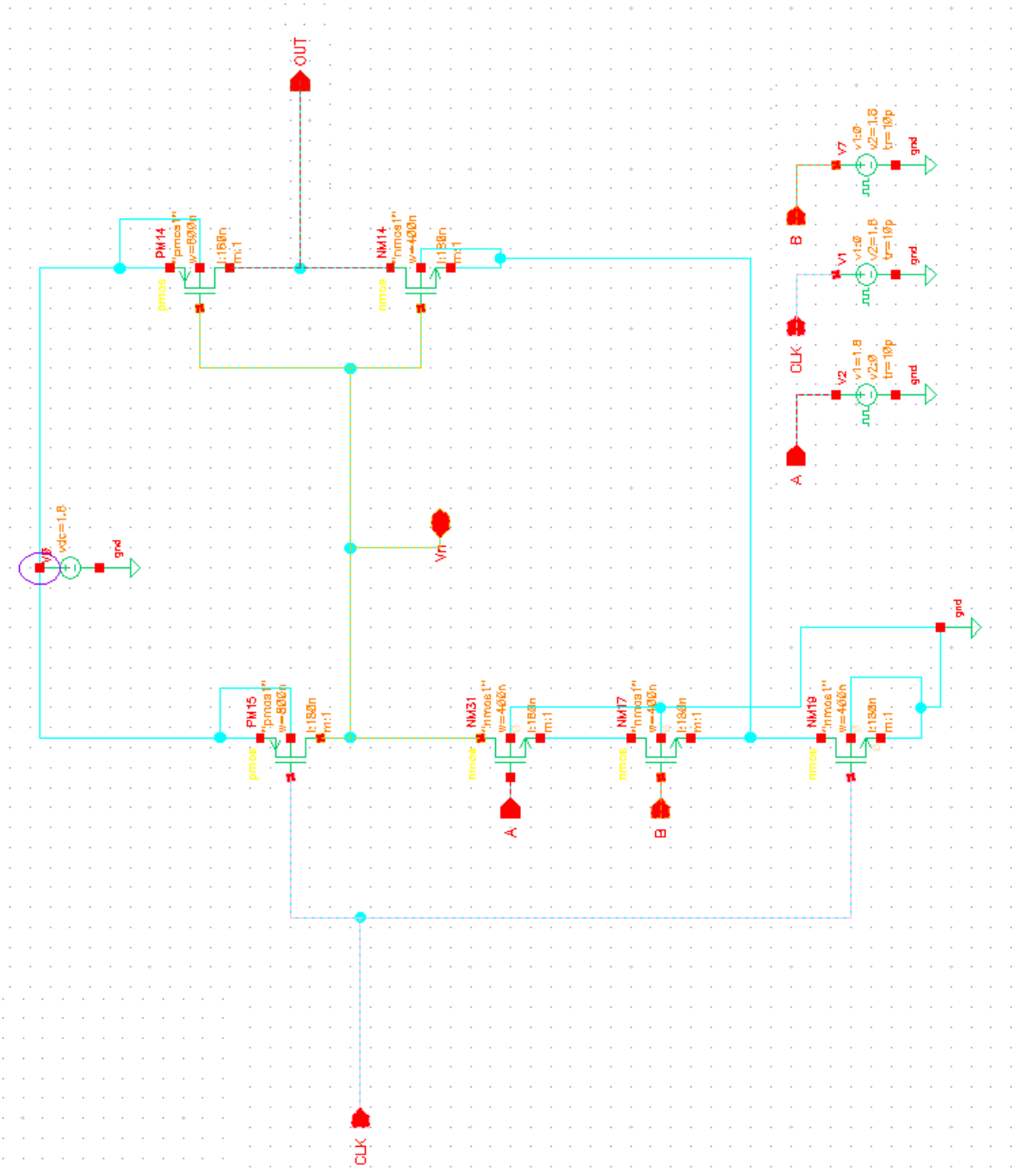

Figure 3.9 VIRTUOSO Schematic of AND Gate using PDB logic

The VIRTUOSO schematic of AND Gate using PDB logic is illustrated in Fig. 3.8.

3.4.3 AND Gate using FQR logic

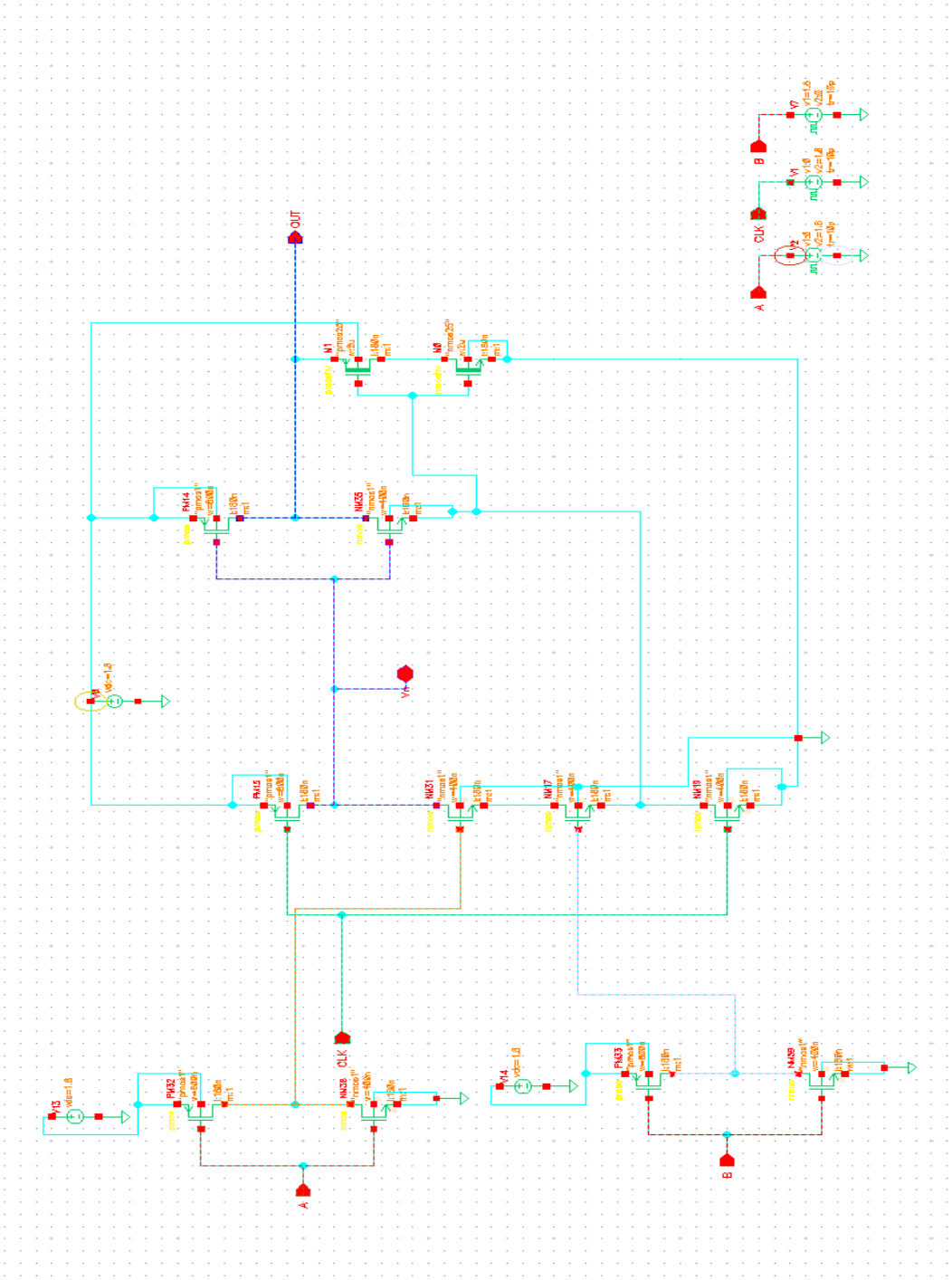

Figure 3.10 VIRTUOSO Schematic of AND Gate using FQR Model

The VIRTUOSO schematic of AND Gate using FQR Model is illustrated in Fig. 3.10.

3.5 CHARACTERIZATION OF OR GATE

3.5.1 OR Gate using domino logic

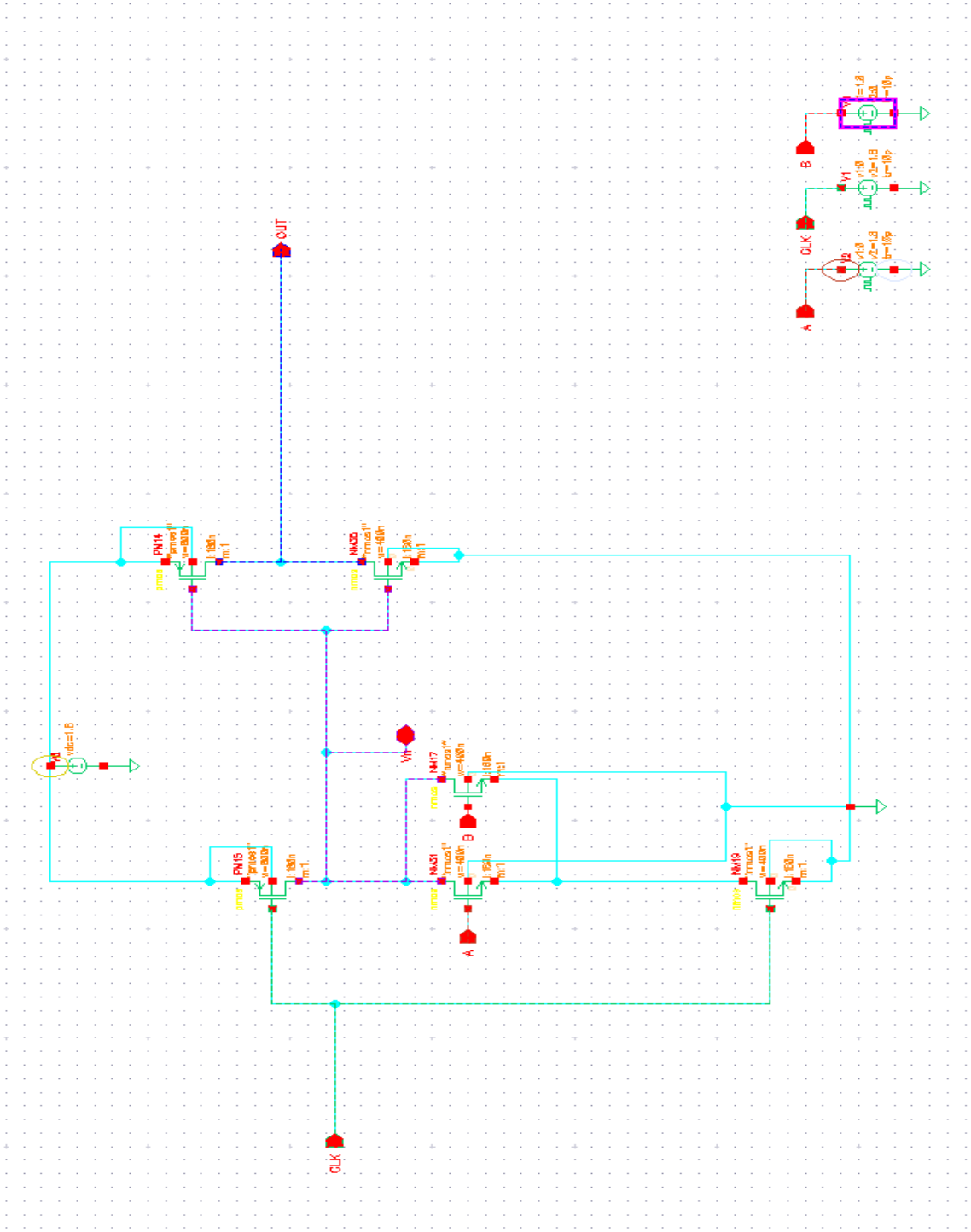

Figure 3.11 VIRTUOSO Schematic of OR Gate using domino logic

The VIRTUOSO schematic of OR Gate using domino logic is illustrated in Fig. 3.11.

The Timing Waveform of OR Gate, is illustrated in Fig. 3.12.

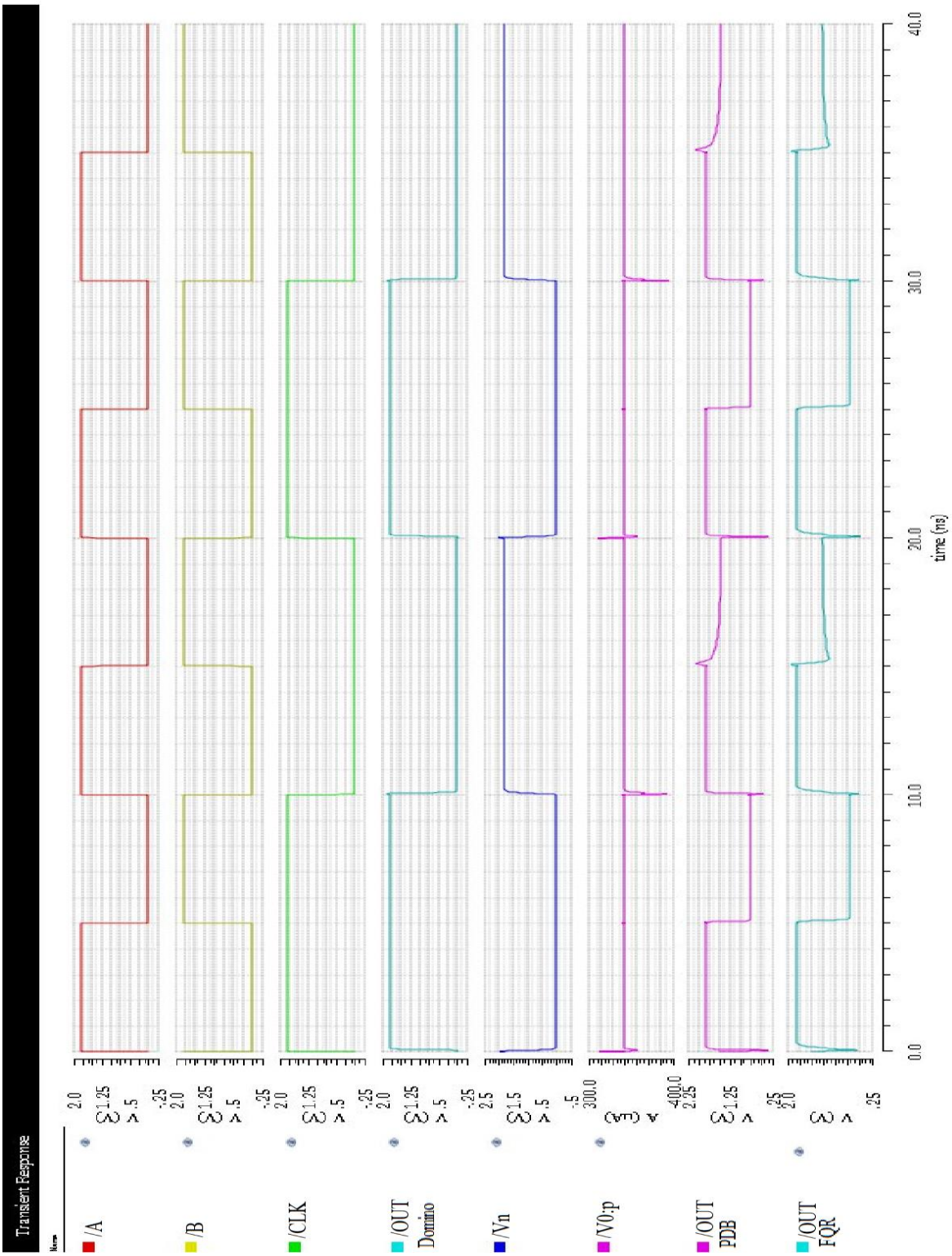

Figure 3.12 Timing Waveform of OR Gate

3.5.2 OR Gate using PDB logic

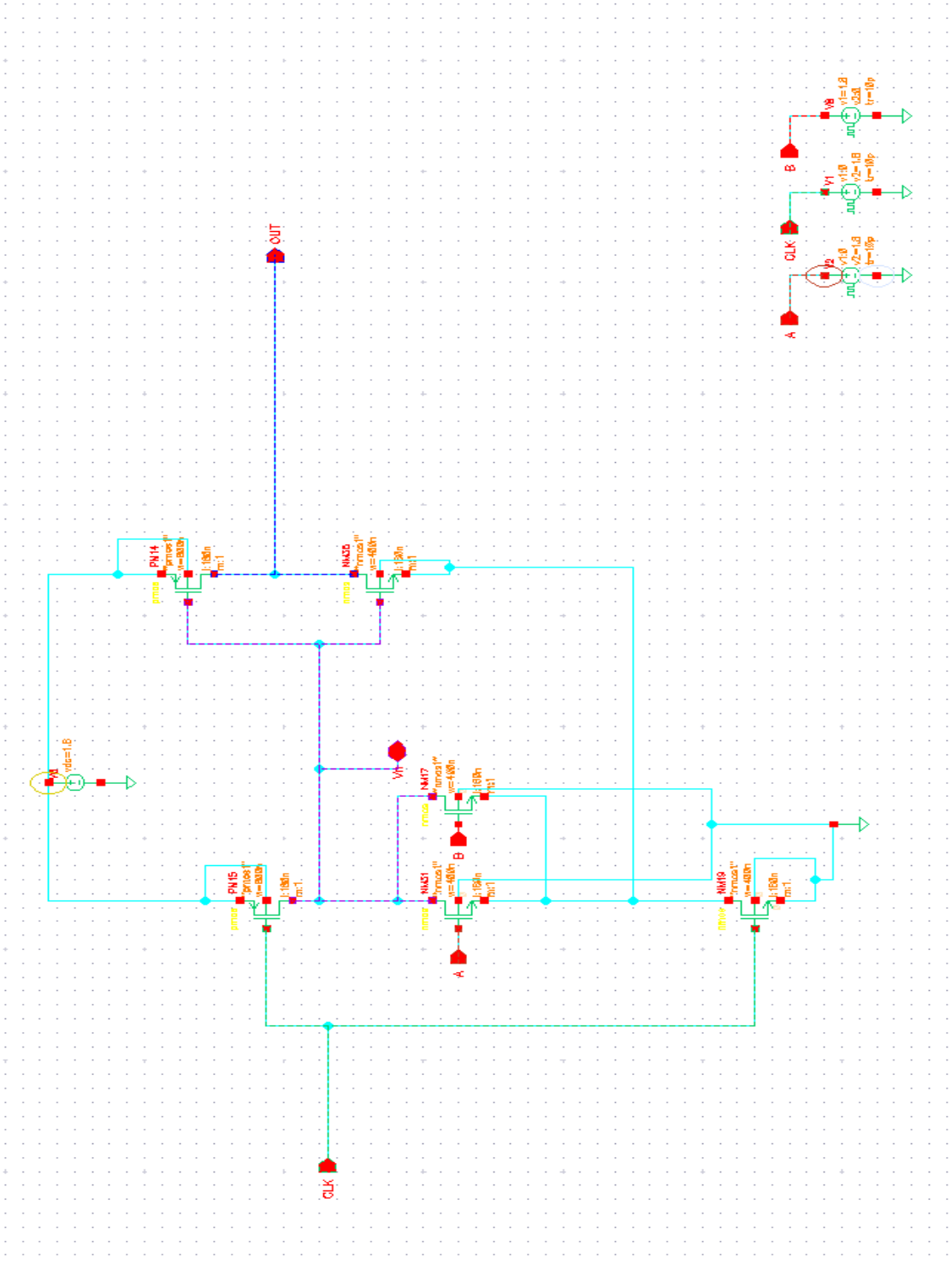

Figure 3.13 VIRTUOSO Schematic of OR Gate using PDB logic

The VIRTUOSO schematic of OR Gate using PDB logic is illustrated in Fig. 3.13.

3.5.3 OR Gate using FQR logic

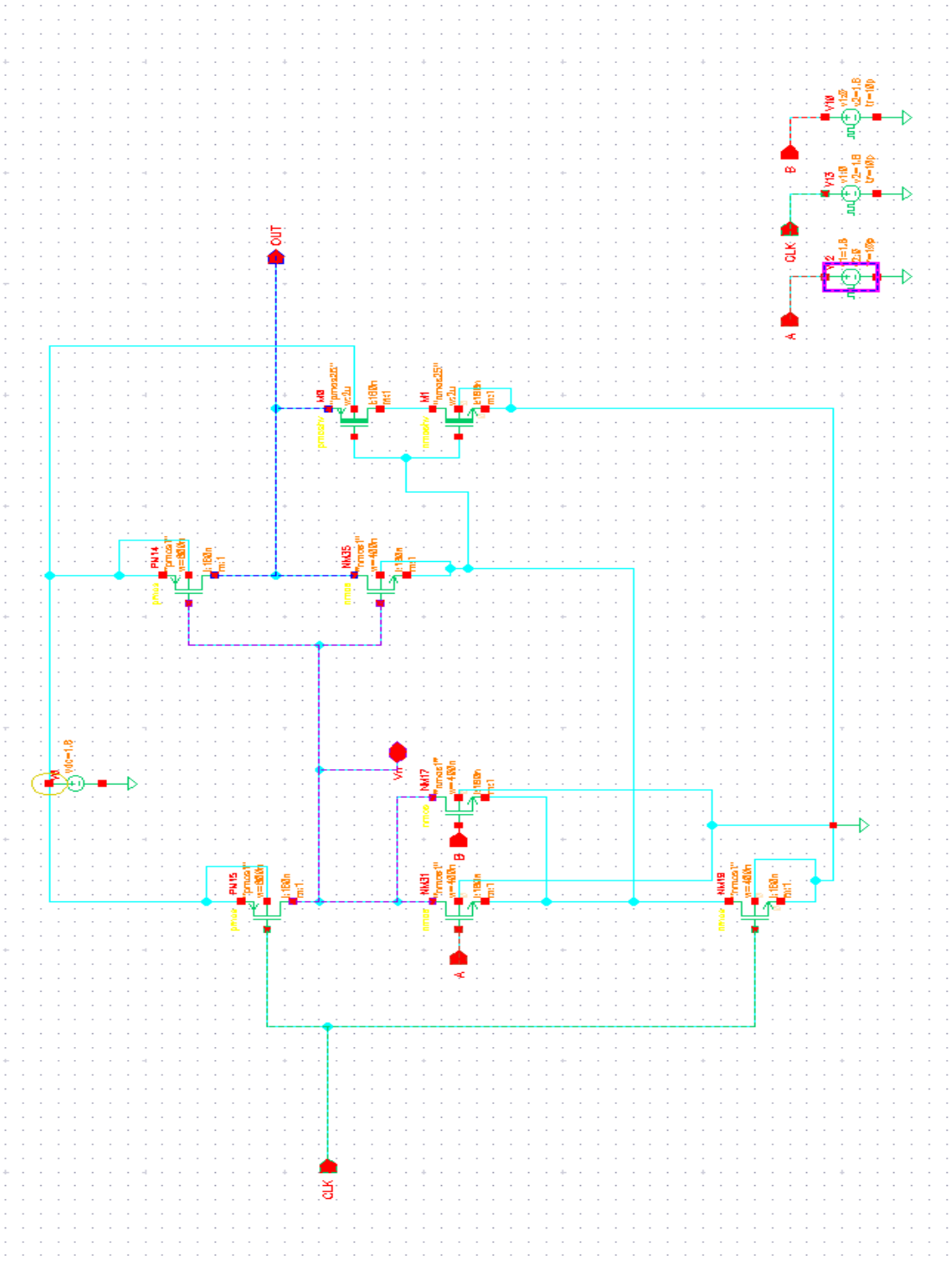

Figure 3.14 VIRTUOSO Schematic of OR Gate using FQR Model

The VIRTUOSO schematic of OR Gate using FQR Model is illustrated in Fig. 3.14.

3.6 CHARACTERIZATION OF NAND GATE

3.6.1 NAND Gate using domino logic

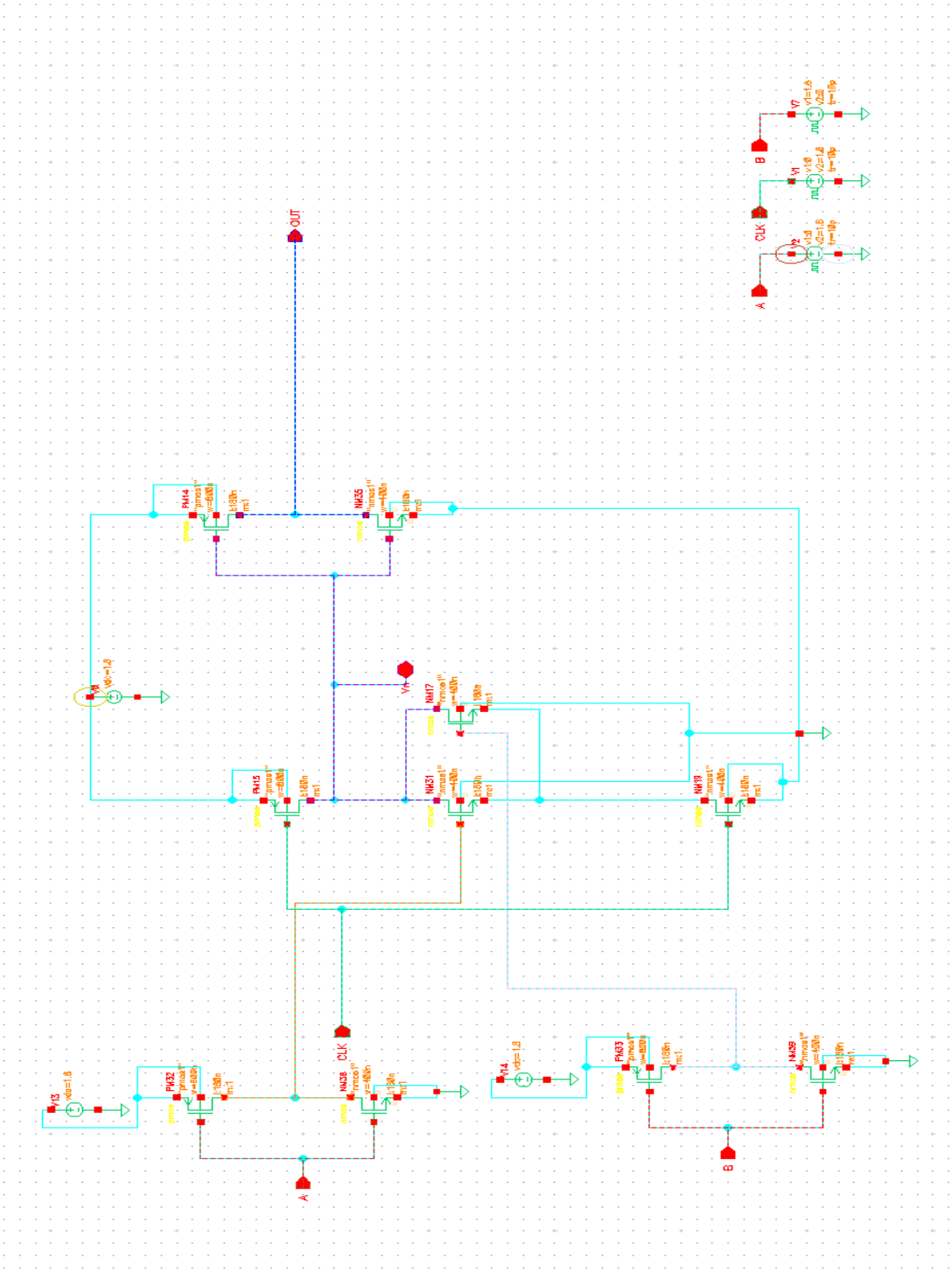

Figure 3.19 VIRTUOSO Schematic of NAND Gate using domino logic

The VIRTUOSO schematic of NAND Gate using domino logic is illustrated in Fig. 3.19. Timing Waveform of NAND Gate, is illustrated in Fig. 3.20.

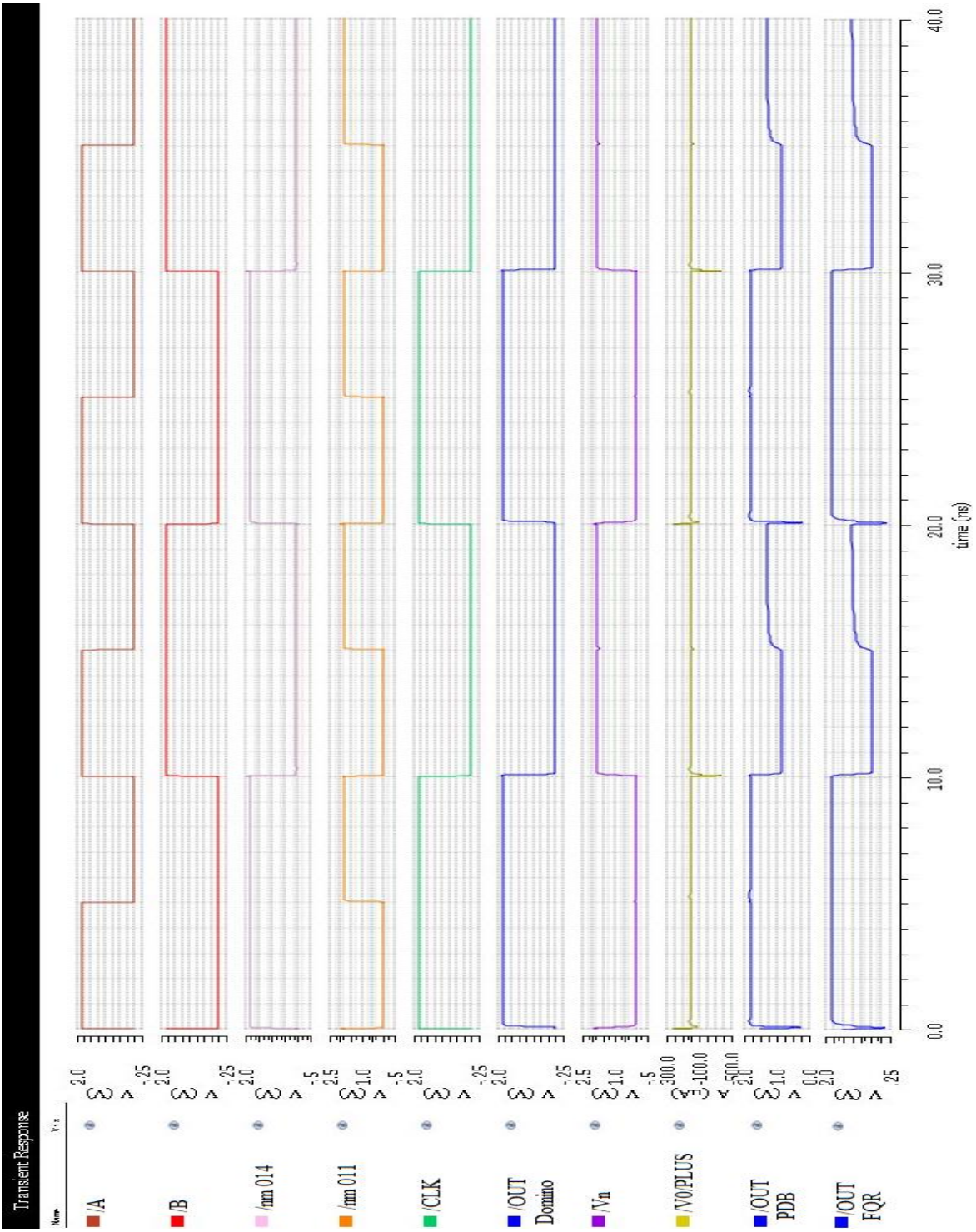

Figure 3.20 Timing Waveform of NAND Gate

3.6.2 NAND Gate using PDB logic

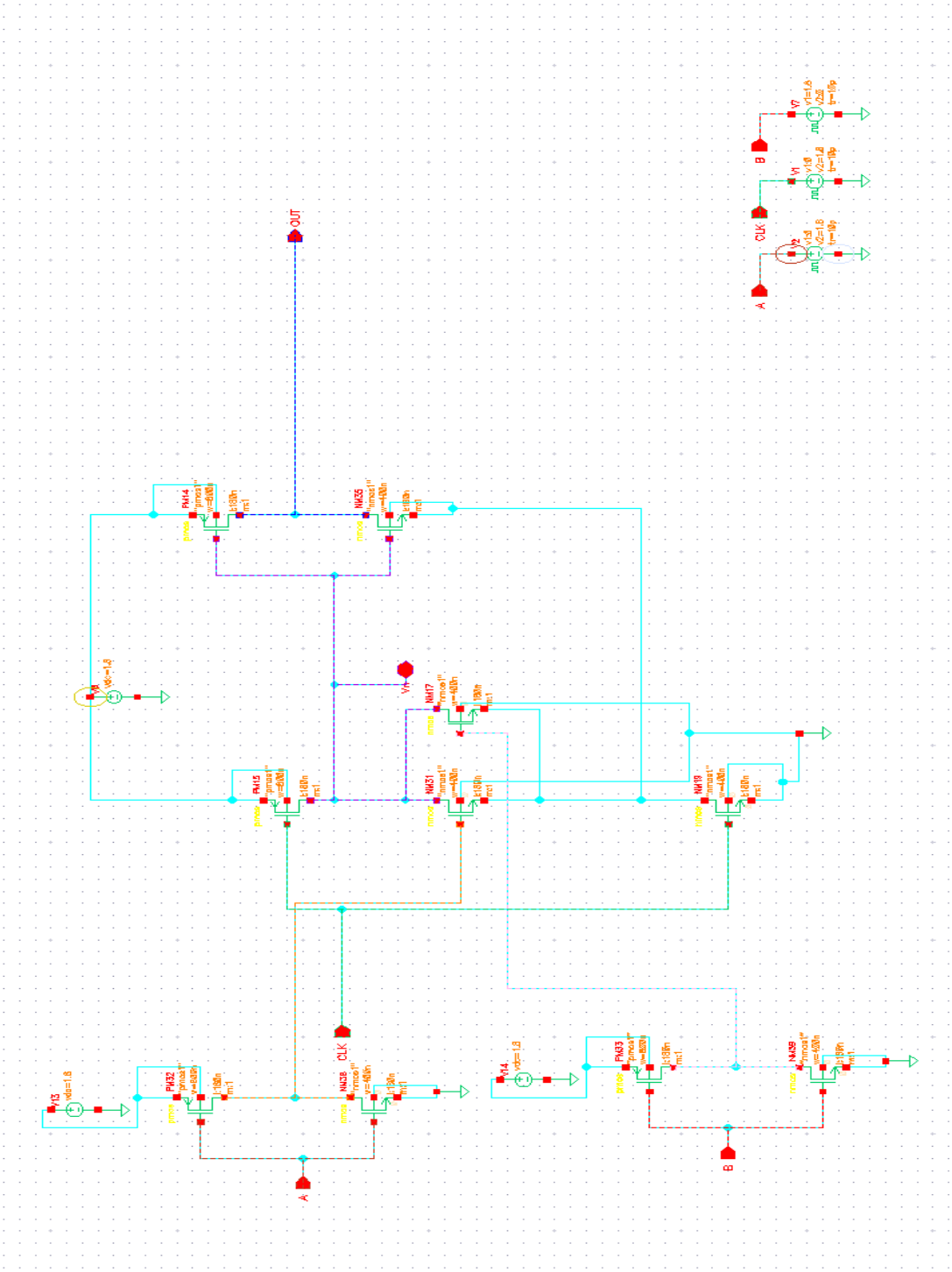

Figure 3.21 VIRTUOSO Schematic of NAND Gate using PDB logic

The VIRTUOSO schematic of NAND Gate using PDB logic is illustrated in Fig. 3.21.

3.6.3 NAND Gate using FQR logic

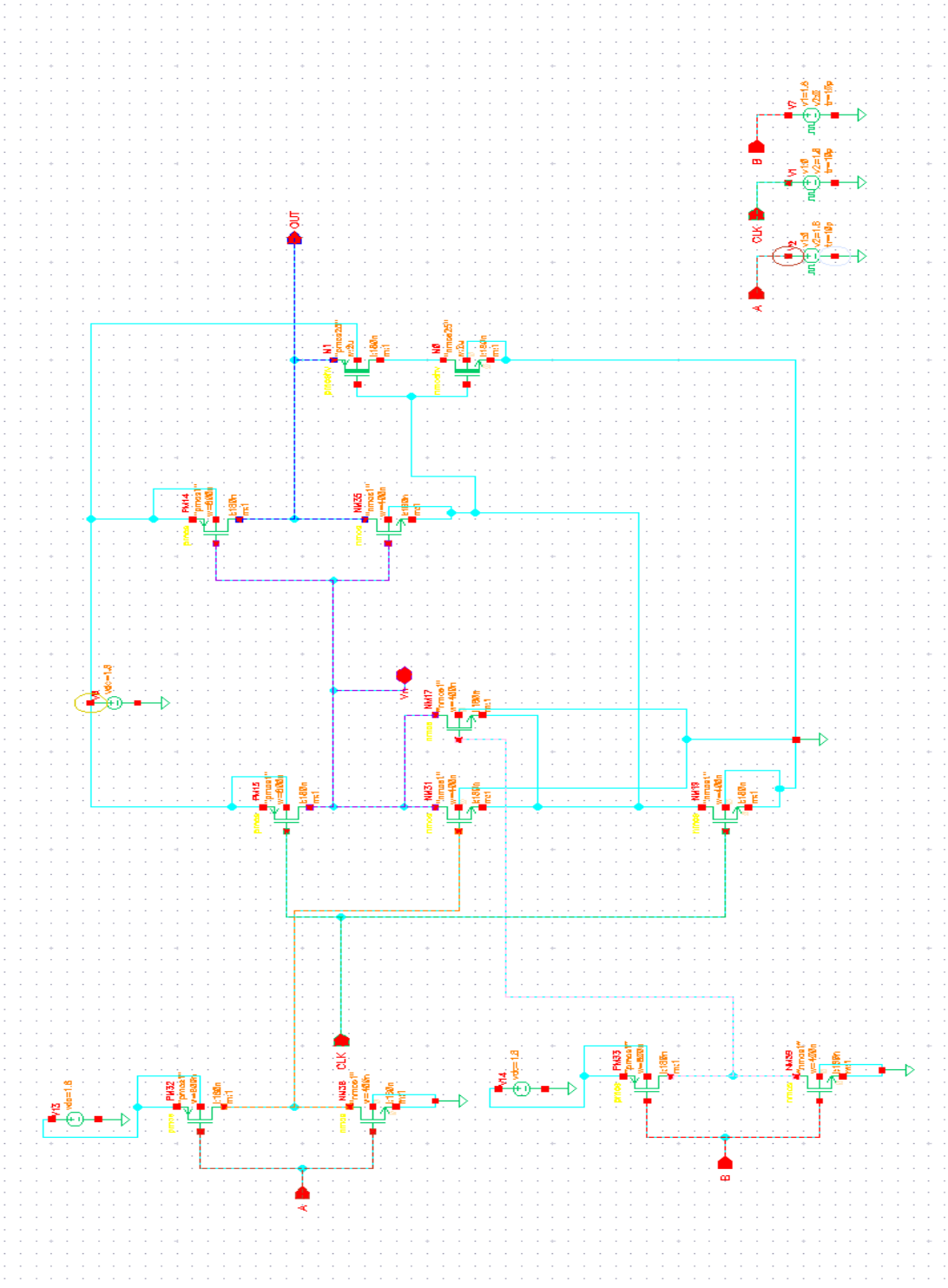

Figure 3.22 VIRTUOSO Schematic of NAND Gate using FQR logic

The VIRTUOSO schematic of NAND Gate using FQR logic is illustrated in Fig. 3.22.

3.7 CHARACTERIZATION OF NOR GATE

3.7.1 NOR Gate using domino logic

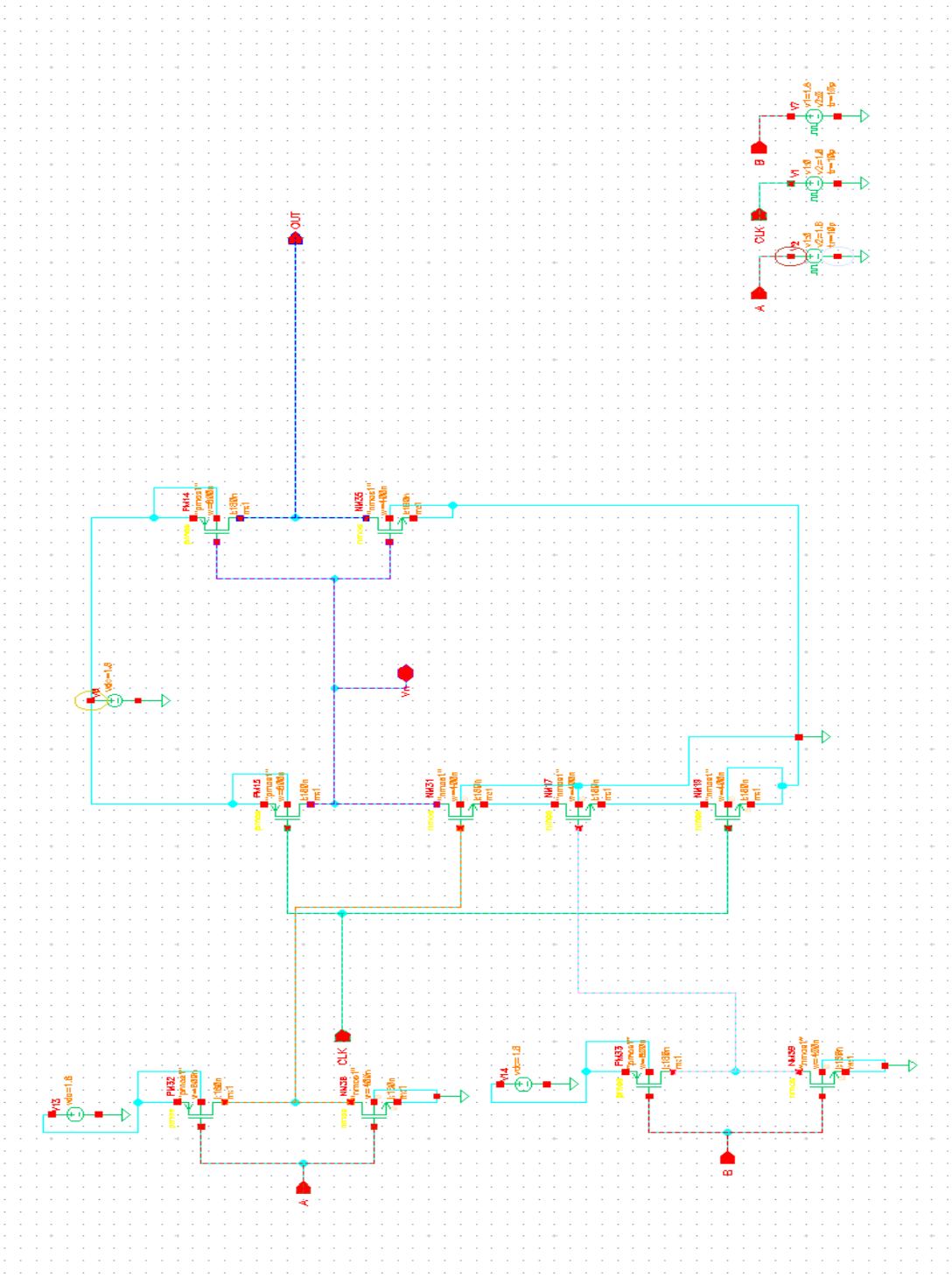

Figure 3.23 VIRTUOSO Schematic of NOR Gate using domino logic

The VIRTUOSO schematic of NOR Gate using domino logic is illustrated in Fig. 3.23. Timing Waveform of NOR Gate, is illustrated in Fig. 3.24.

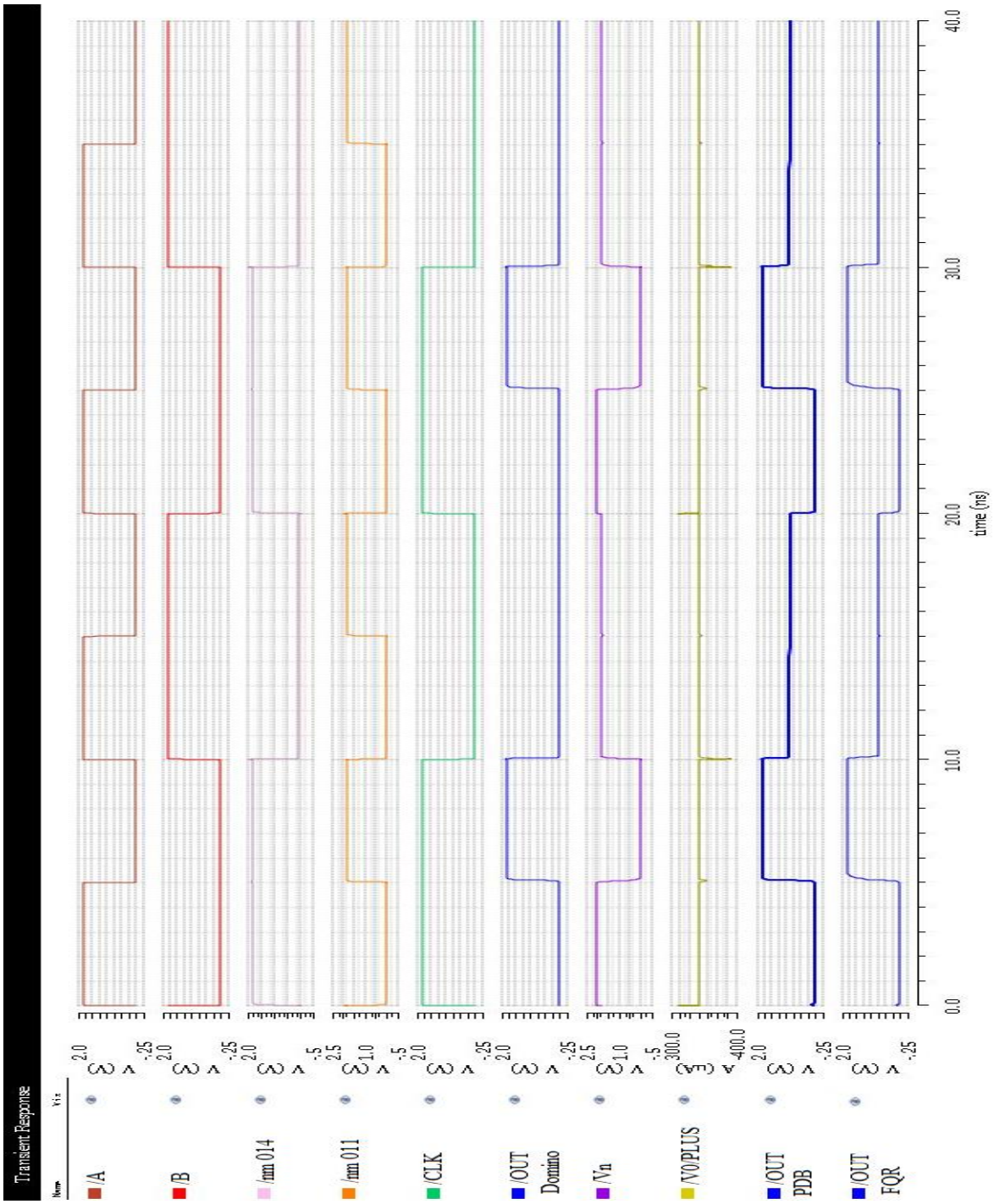

Figure 3.24 Timing Waveform of NOR Gate

3.7.2 NOR Gate using PDB logic

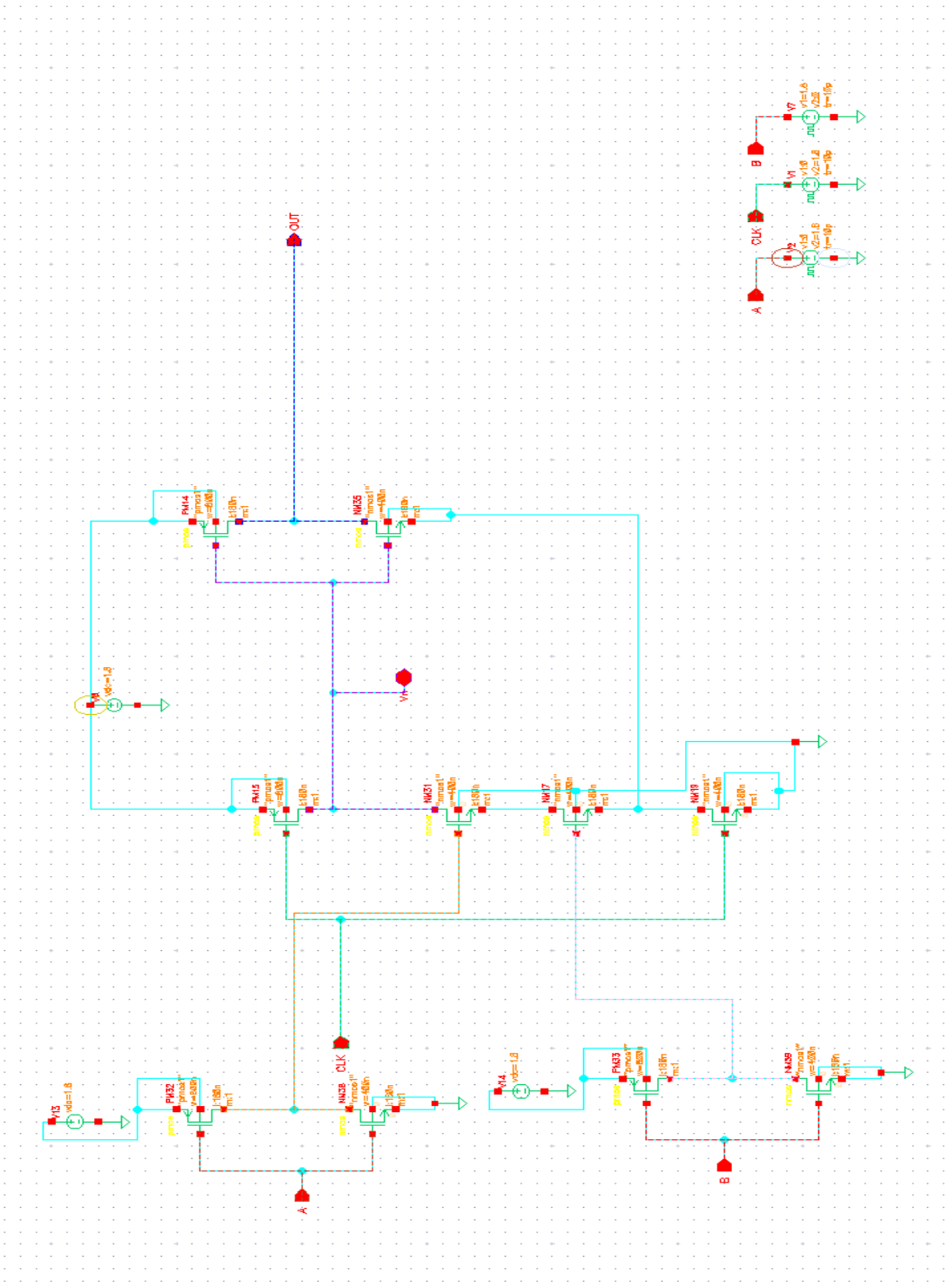

Figure 3.25 VIRTUOSO Schematic of NOR Gate using PDB logic

The VIRTUOSO schematic of NOR Gate using PDB logic is illustrated in Fig. 3.25.

3.7.3 NOR Gate using FQR logic

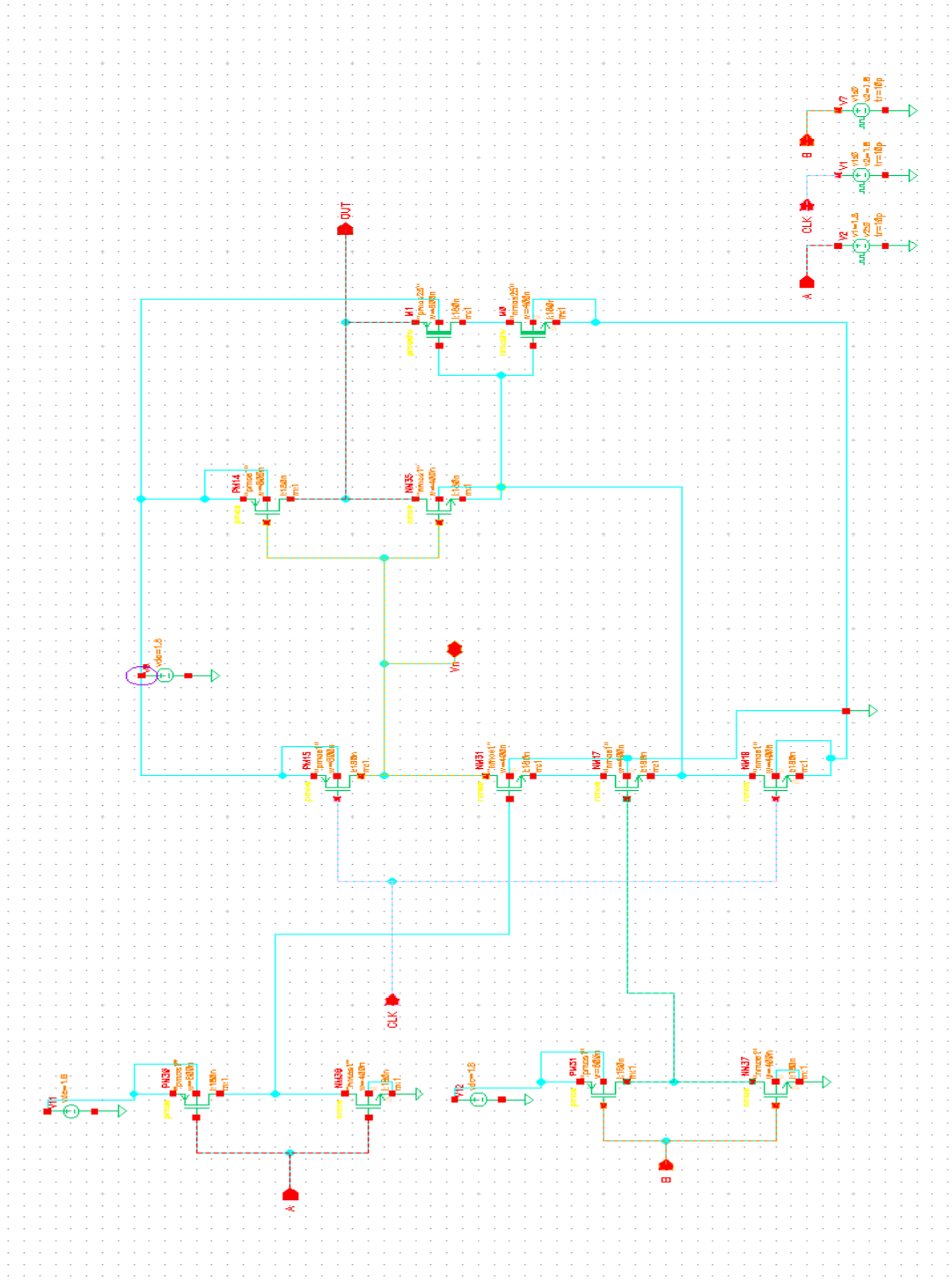

Figure 3.26 VIRTUOSO Schematic of NOR Gate using FQR logic

The VIRTUOSO schematic of NOR Gate using FQR logic is illustrated in Fig. 3.26.

3.8 CHARACTERIZATION OF XOR GATE

3.8.1 XOR Gate using domino logic

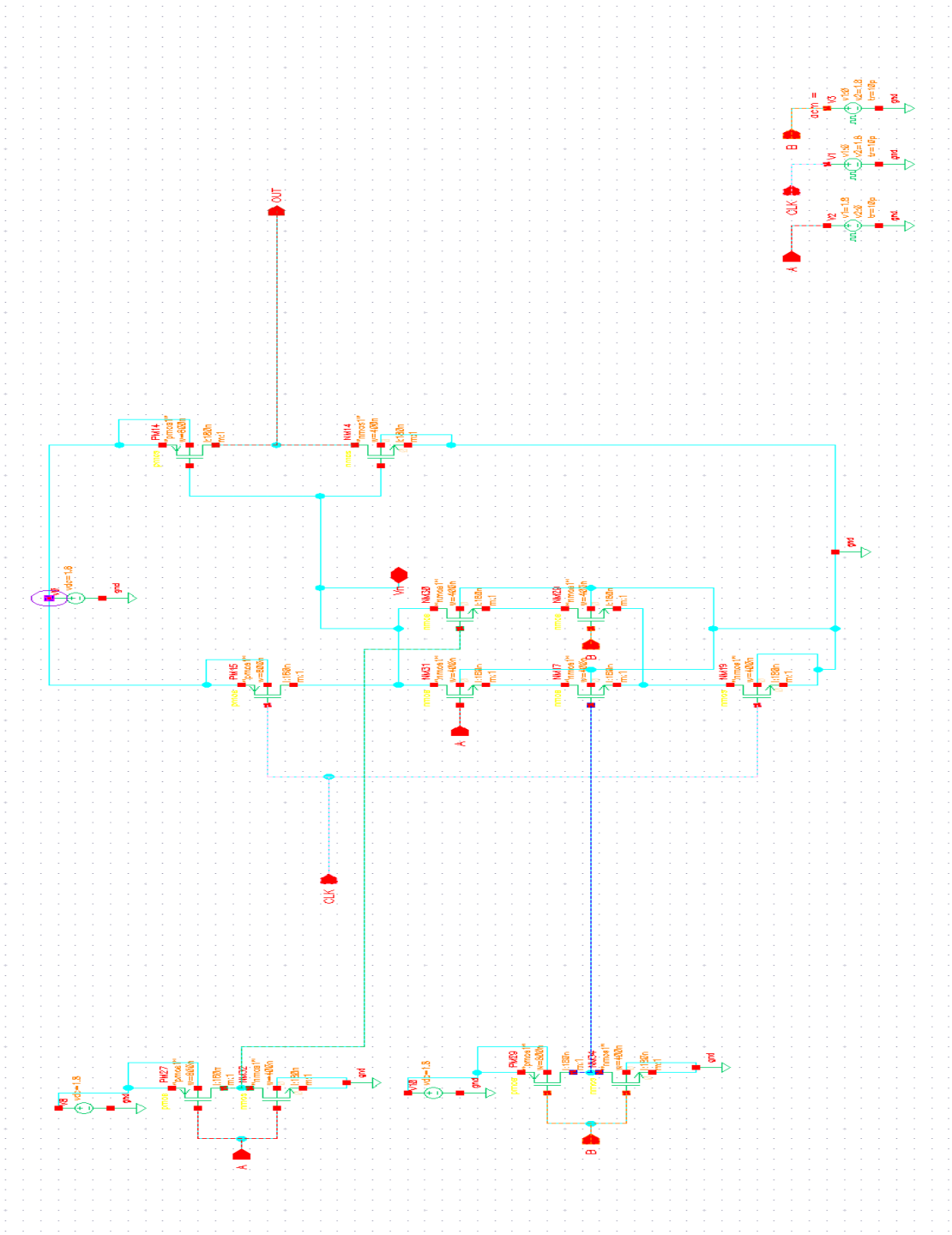

Figure 3.27 VIRTUOSO Schematic of XOR Gate using domino logic

The VIRTUOSO schematic of XOR Gate using domino logic is illustrated in Fig. 3.27. The Timing Waveform of XOR Gate, is illustrated in Fig. 3.28.

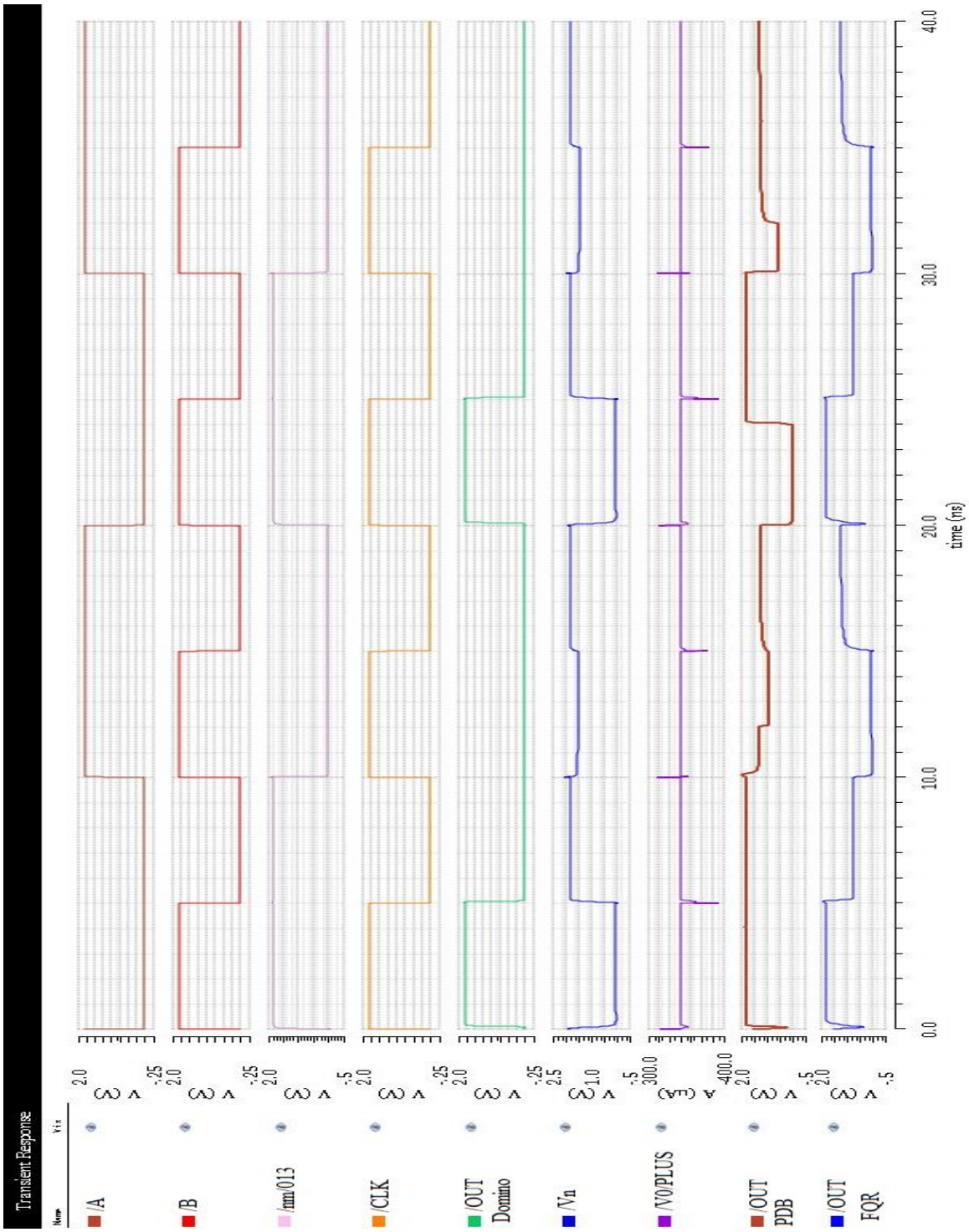

Figure 3.28 Timing Waveform of XOR Gate

3.8.2 XOR Gate using PDB logic

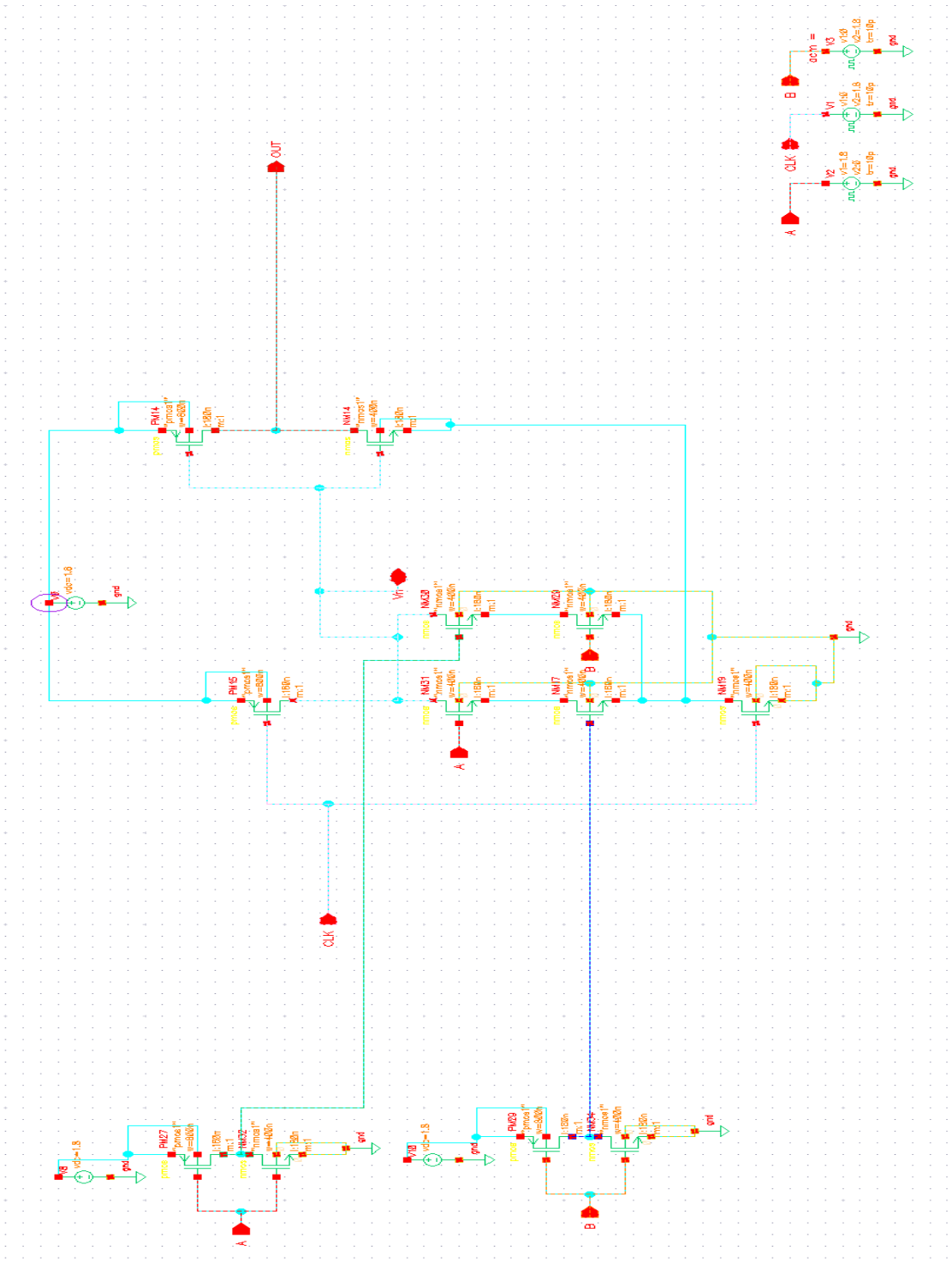

Figure 3.29 VIRTUOSO Schematic of XOR Gate using PDB logic

The VIRTUOSO schematic of XOR Gate using PDB logic is illustrated in Fig. 3.29.

3.8.3 XOR Gate using FQR logic

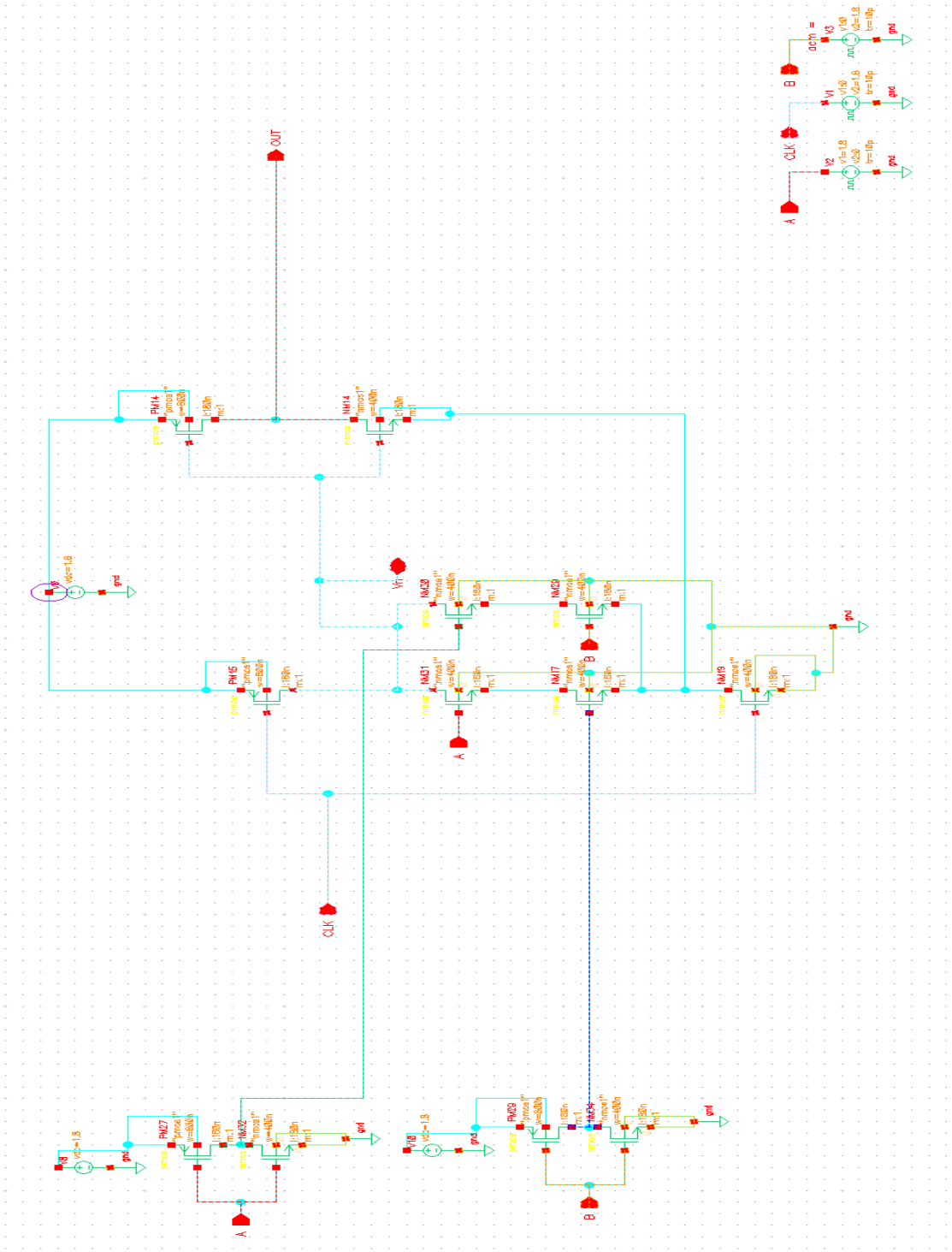

Figure 3.30 VIRTUOSO Schematic of XOR Gate using FQR logic

The VIRTUOSO schematic of XOR Gate using FQR logic is illustrated in Fig. 3.30.

3.9 CHARACTERIZATION OF XNOR GATE

3.9.1 XNOR Gate using domino logic

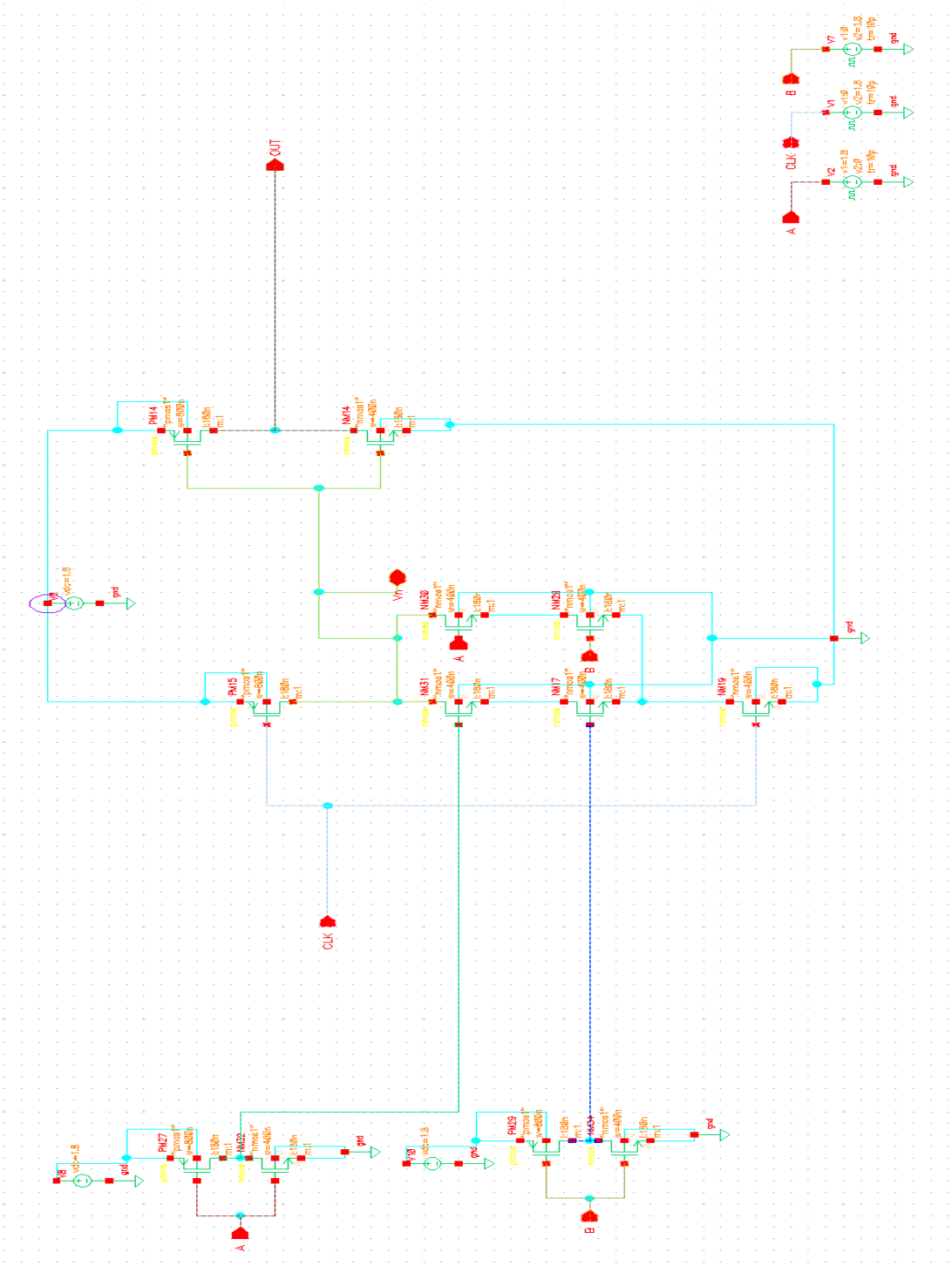

Figure 3.31 VIRTUOSO Schematic of XNOR Gate using domino logic

The VIRTUOSO schematic of XNOR Gate using domino logic is illustrated in Fig. 3.31. The Timing Waveform of XNOR Gate, is illustrated in Fig. 3.32.

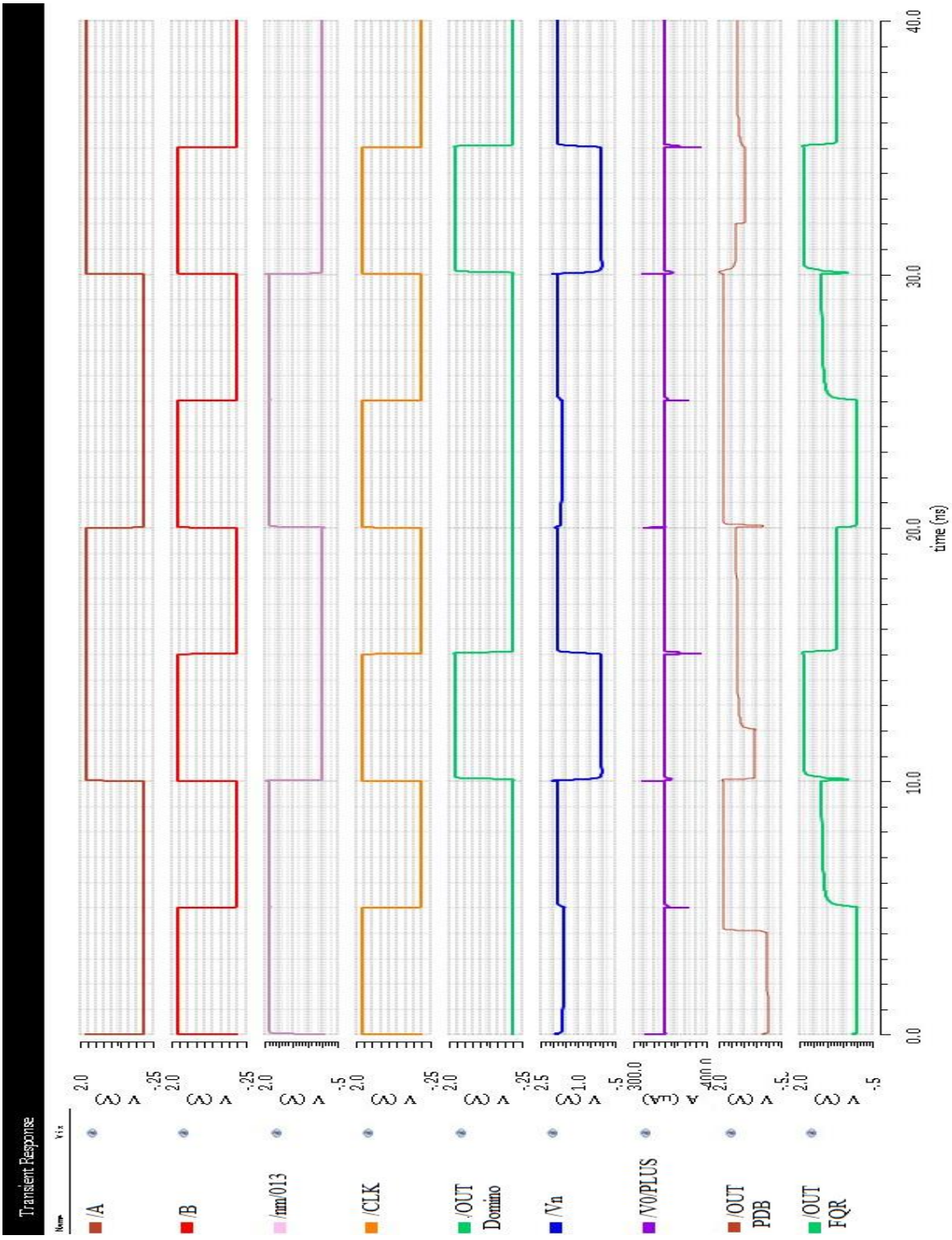

Figure 3.32 Timing Waveform of XNOR Gate

3.9.2 XNOR Gate using PDB logic

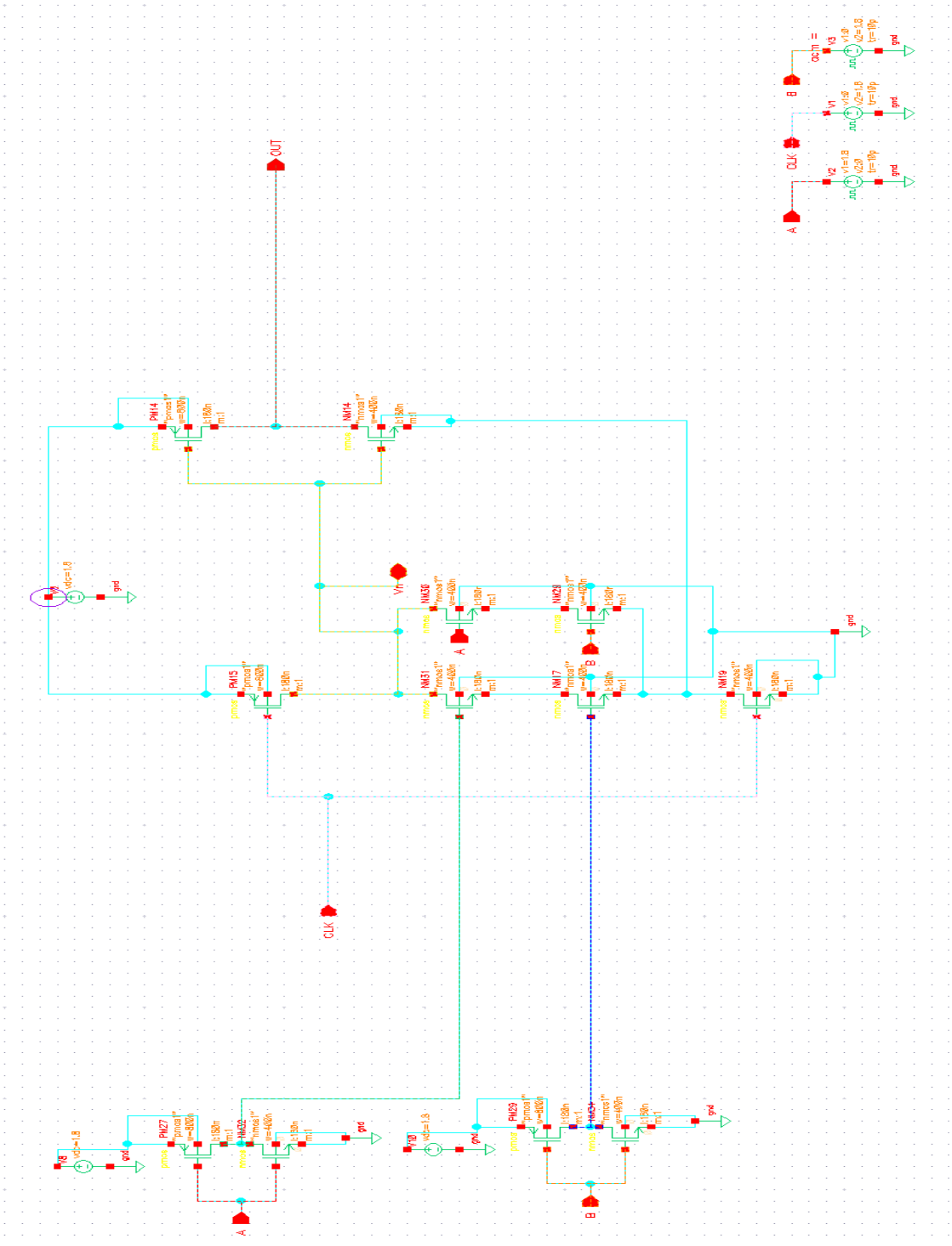

Figure 3.33 VIRTUOSO Schematic of XNOR Gate using PDB logic

The VIRTUOSO schematic of XNOR Gate using PDB logic is illustrated in Fig. 3.33.

3.9.3 XNOR Gate using FQR logic

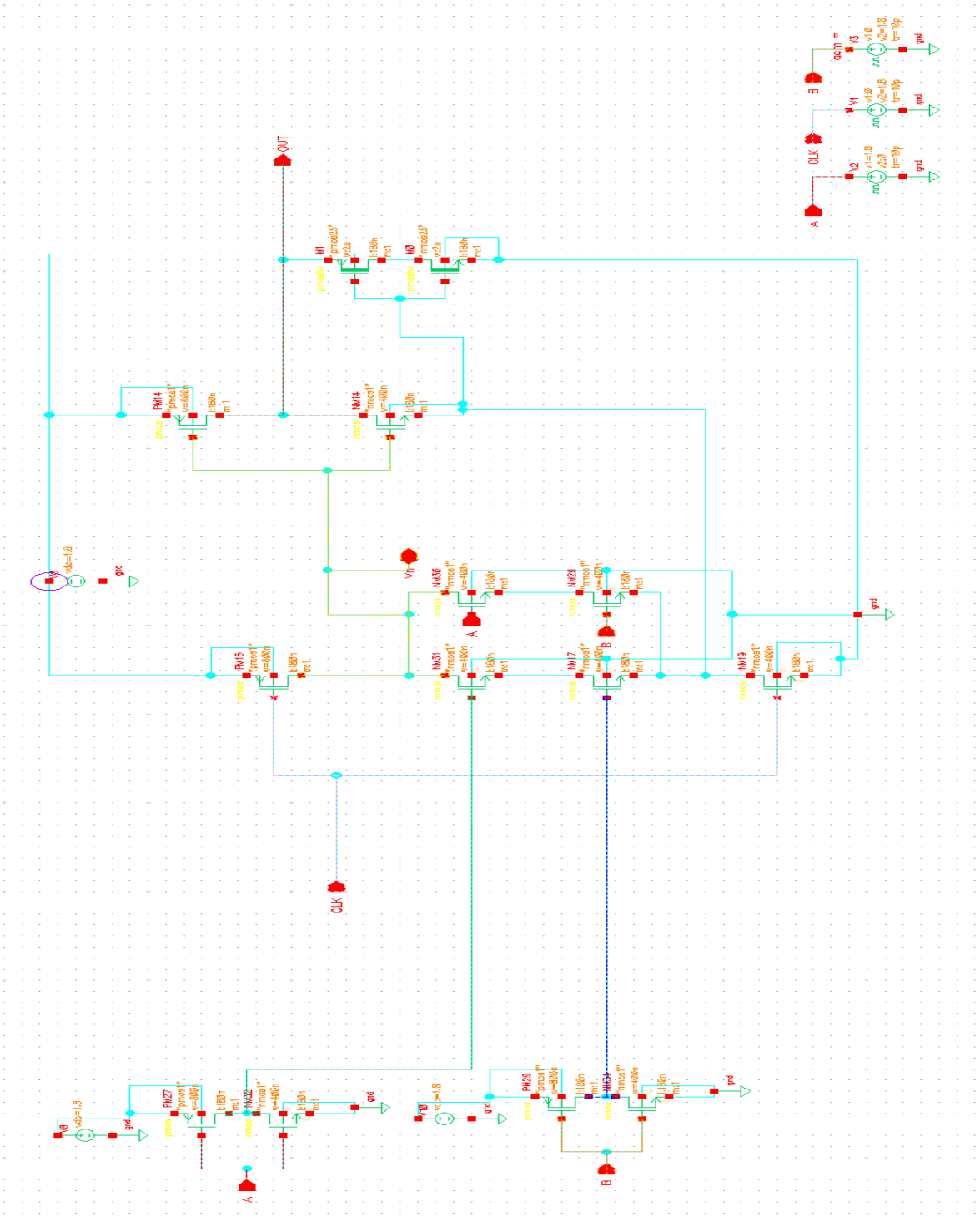

Figure 3.34 VIRTUOSO Schematic of XNOR Gate using FQR logic

The VIRTUOSO schematic of XNOR Gate using FQR logic is illustrated in Fig. 3.34.

3.10 CHARACTERIZATION OF XOR GATE

3.10.1 Half Adder using domino logic

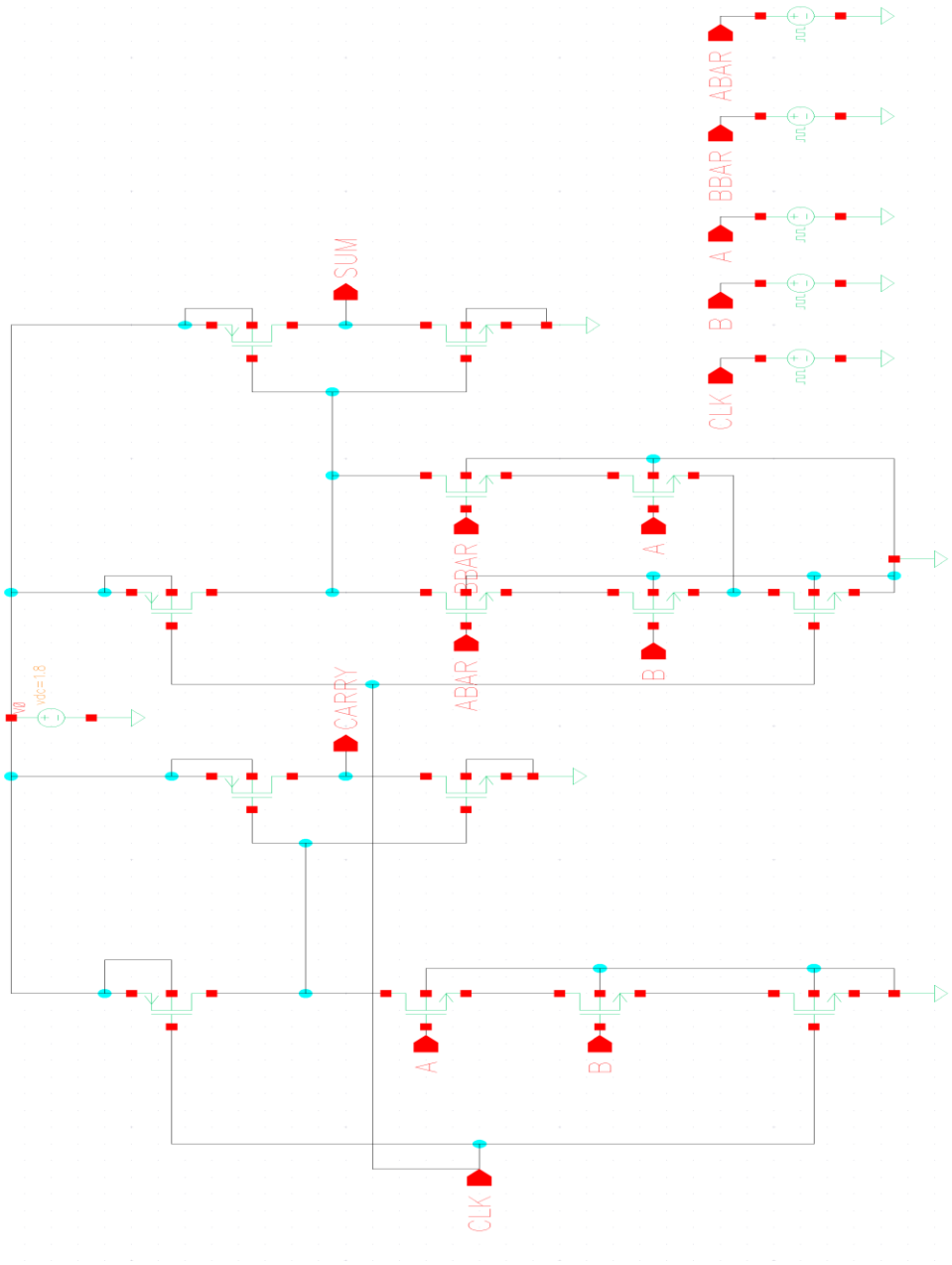

Figure 3.35 VIRTUOSO Schematic of half adder using domino logic

The VIRTUOSO schematic of half adder using domino logic is illustrated in Fig. 3.35. The Timing Waveform of half adder, is illustrated in Fig. 3.36.

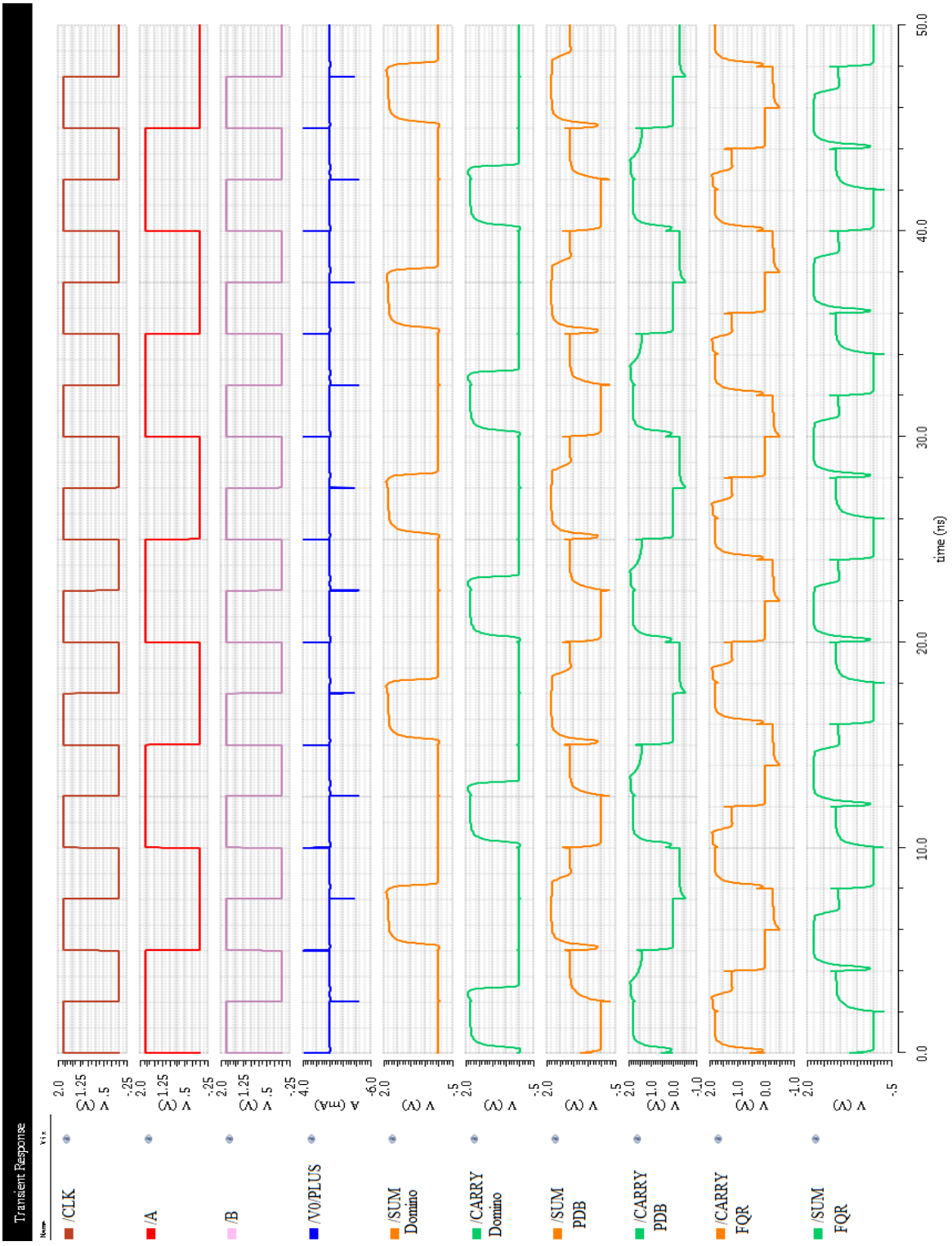

Figure 3.36 Timing Waveform of half adder

3.10.2 Half Adder using PDB logic

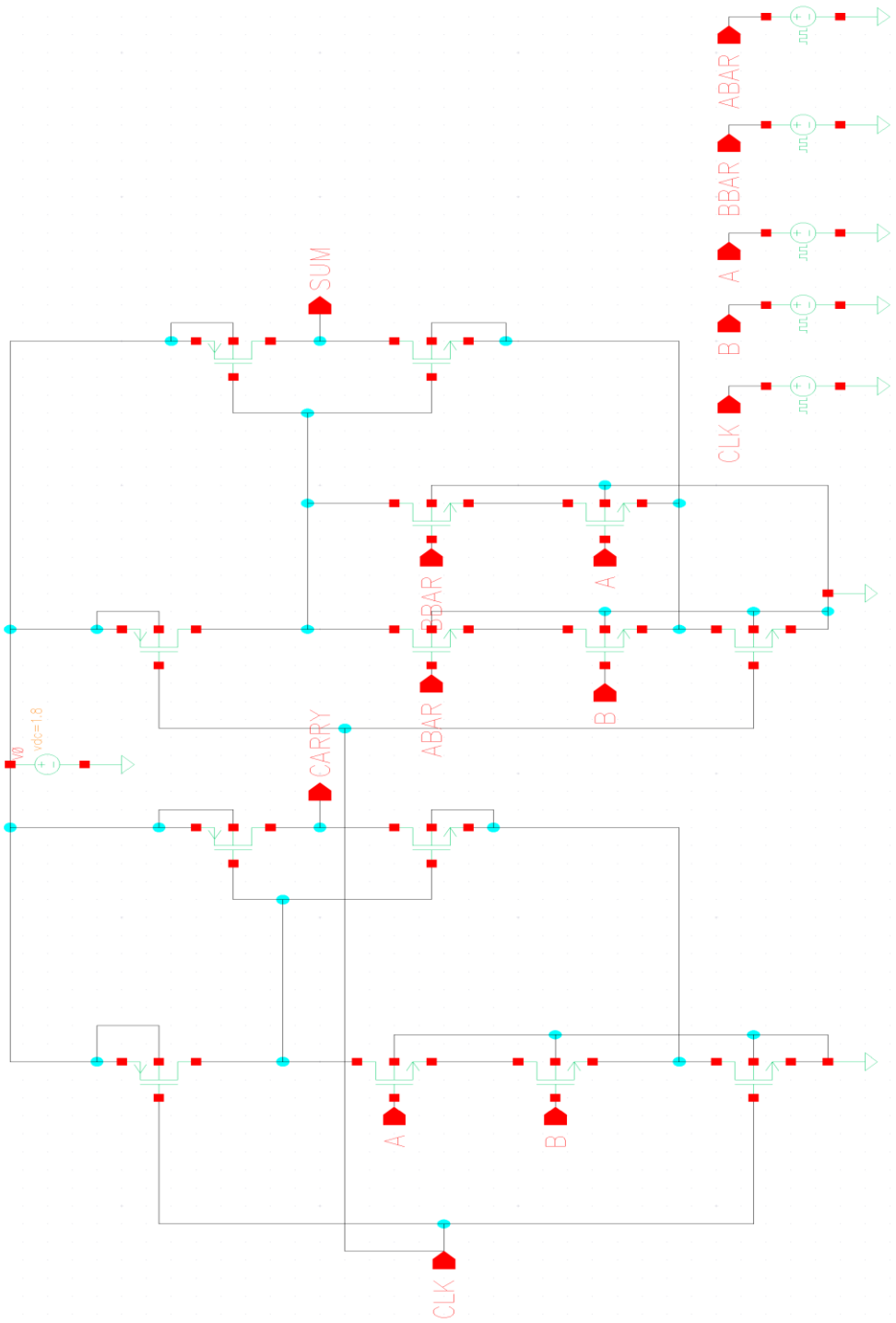

Figure 3.37 VIRTUOSO Schematic of half adder using PDB logic

The VIRTUOSO schematic of half adder using PDB logic is illustrated in Fig. 3.37.

3.10.3 Half Adder using FQR logic

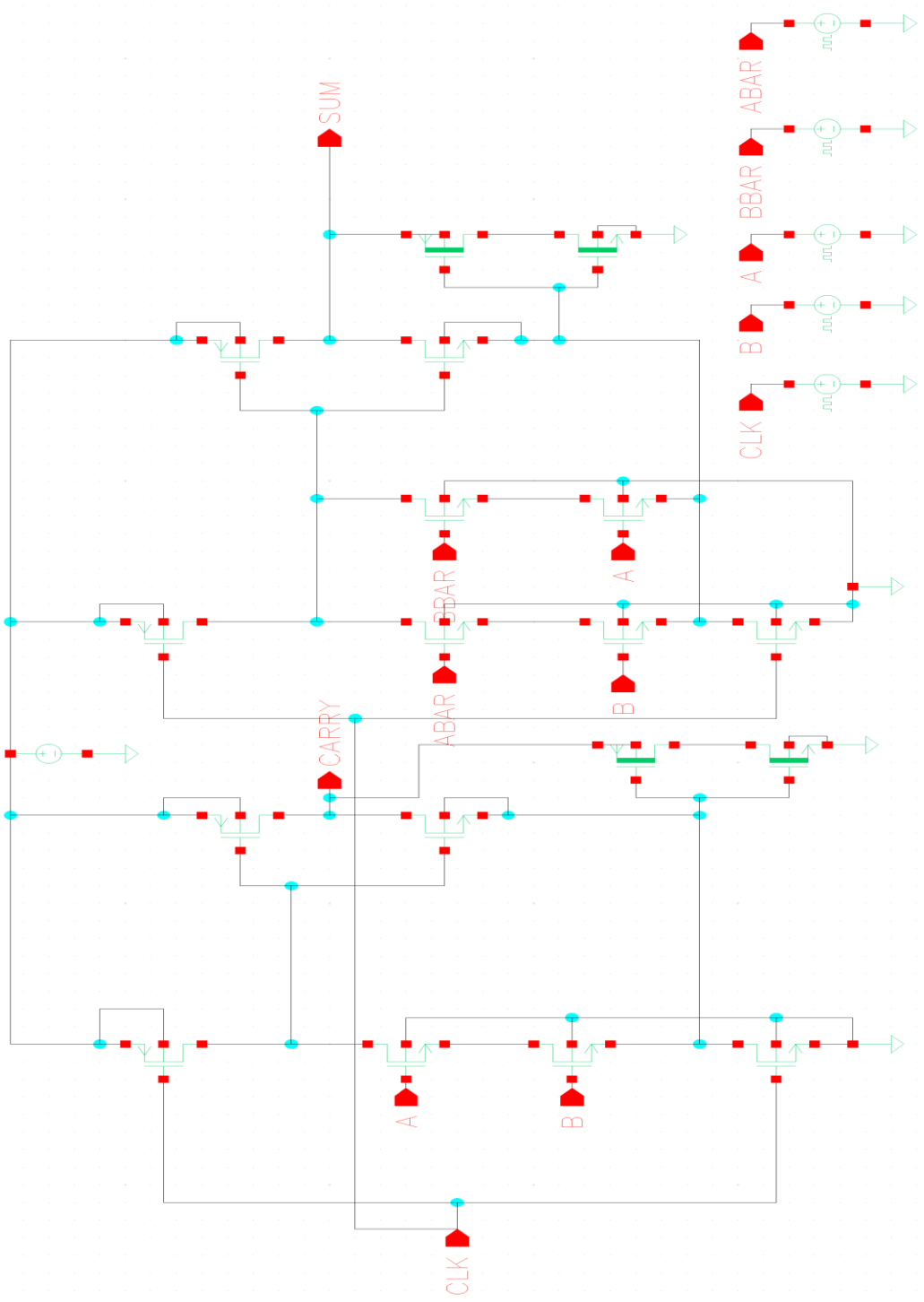

Figure 3.38 VIRTUOSO Schematic of half adder using FQR logic

The VIRTUOSO schematic of half adder using FQR logic is illustrated in Fig. 3.38.

3.11 CHARACTERIZATION OF HALF SUBTRACTOR

3.11.1 Half Subtractor using domino logic

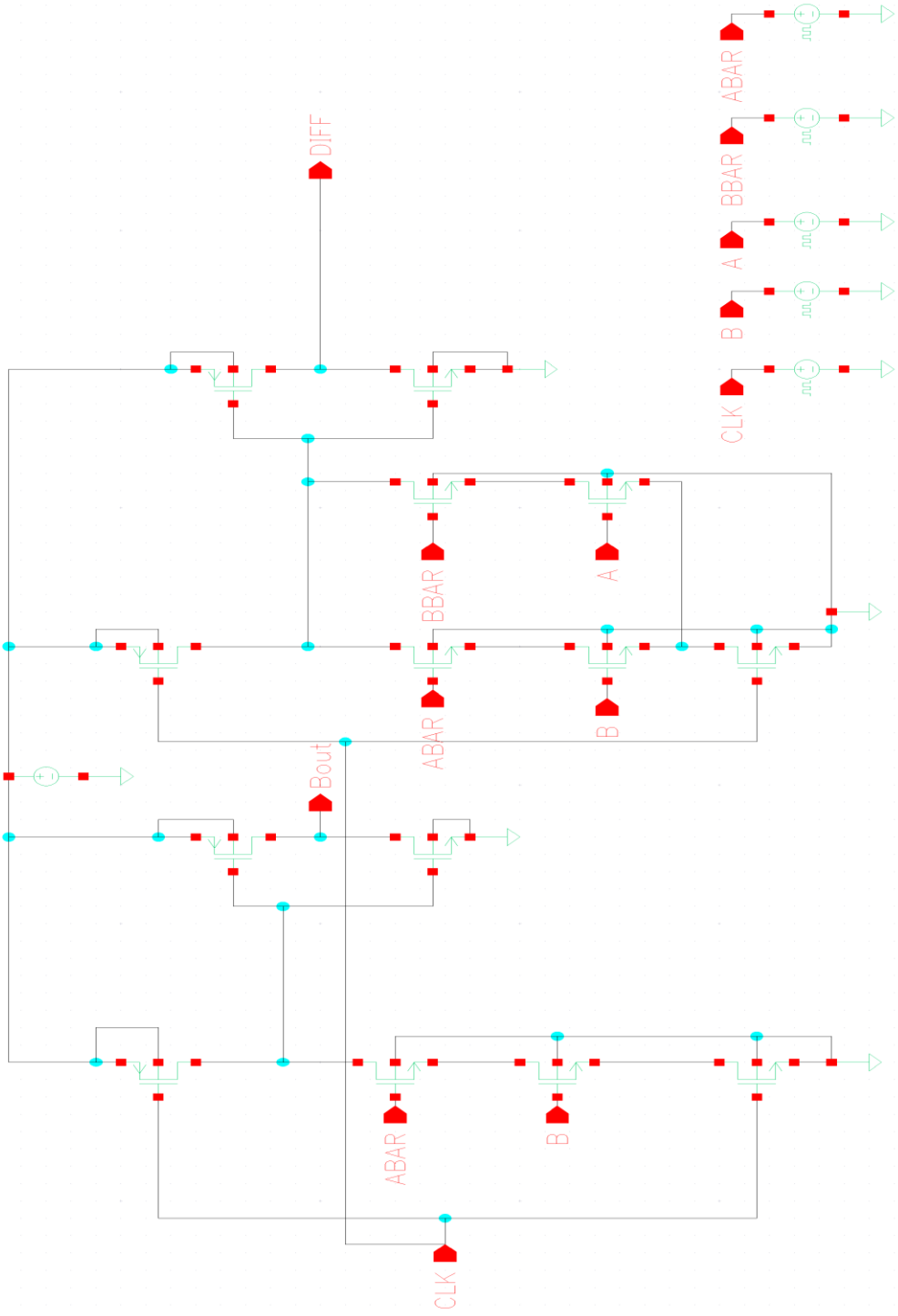

Figure 3.39 VIRTUOSO Schematic of half subtractor using domino logic

The VIRTUOSO schematic of half subtractor using domino logic is illustrated in Fig. 3.39. Timing Waveform of half subtractor, is illustrated in Fig. 3.40.

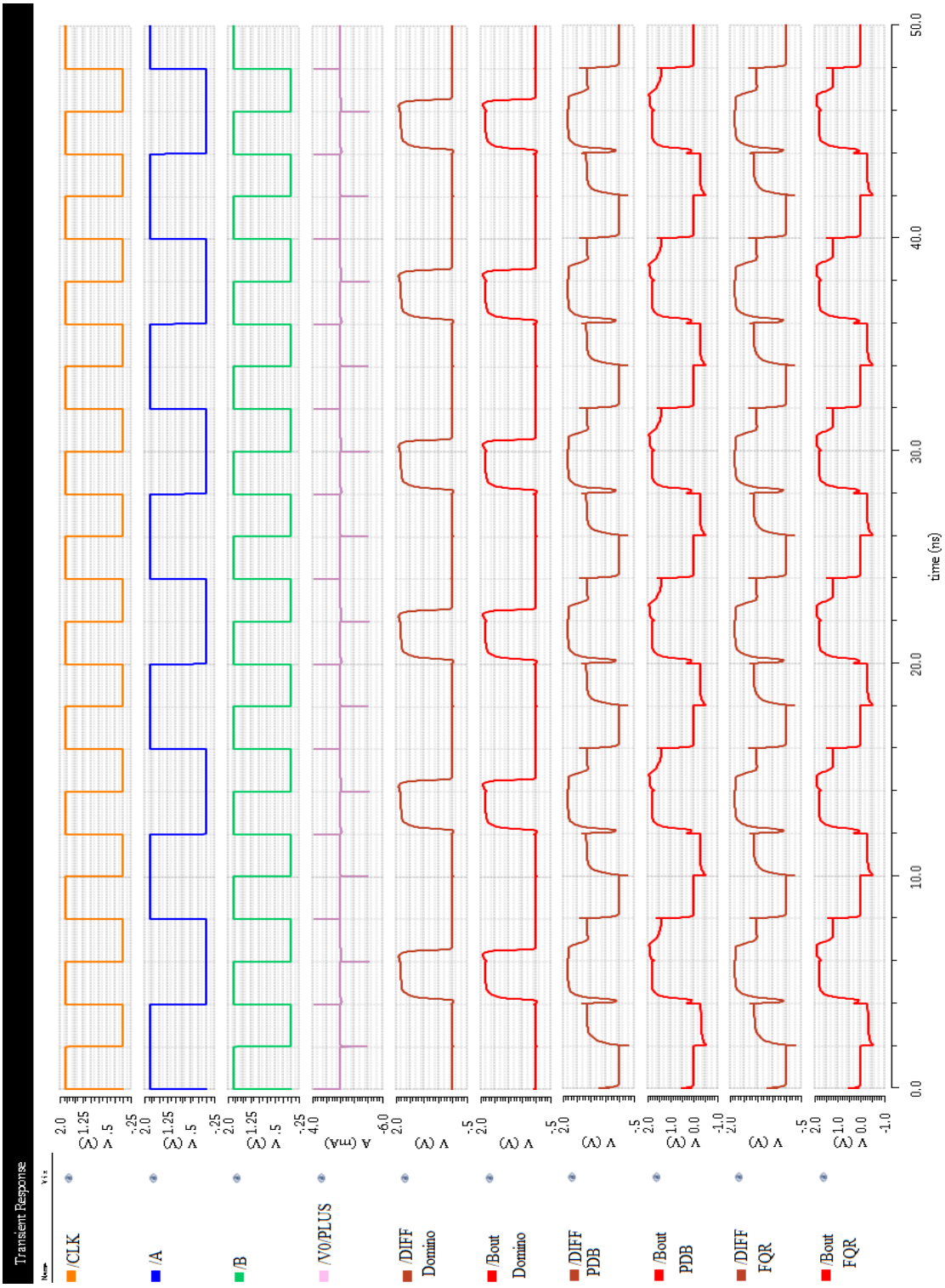

Figure 3.40 Timing Waveform of half subtractor

3.11.2 Half Subtractor using PDB logic

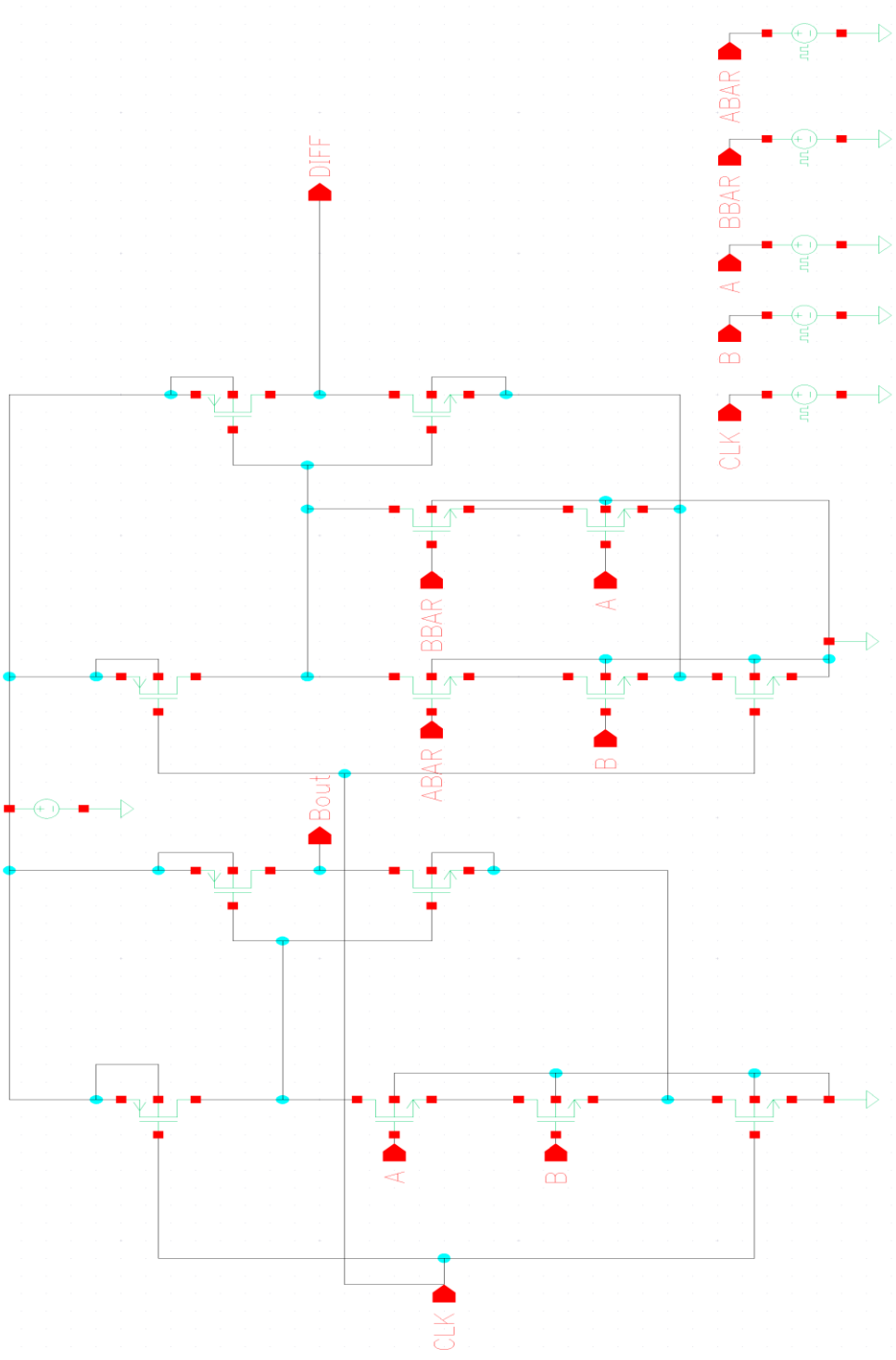

Figure 3.41 VIRTUOSO Schematic of half subtractor using PDB logic

The VIRTUOSO schematic of half subtractor using PDB logic is illustrated in Fig. 3.41.

3.11.3 Half Subtractor using FQR logic

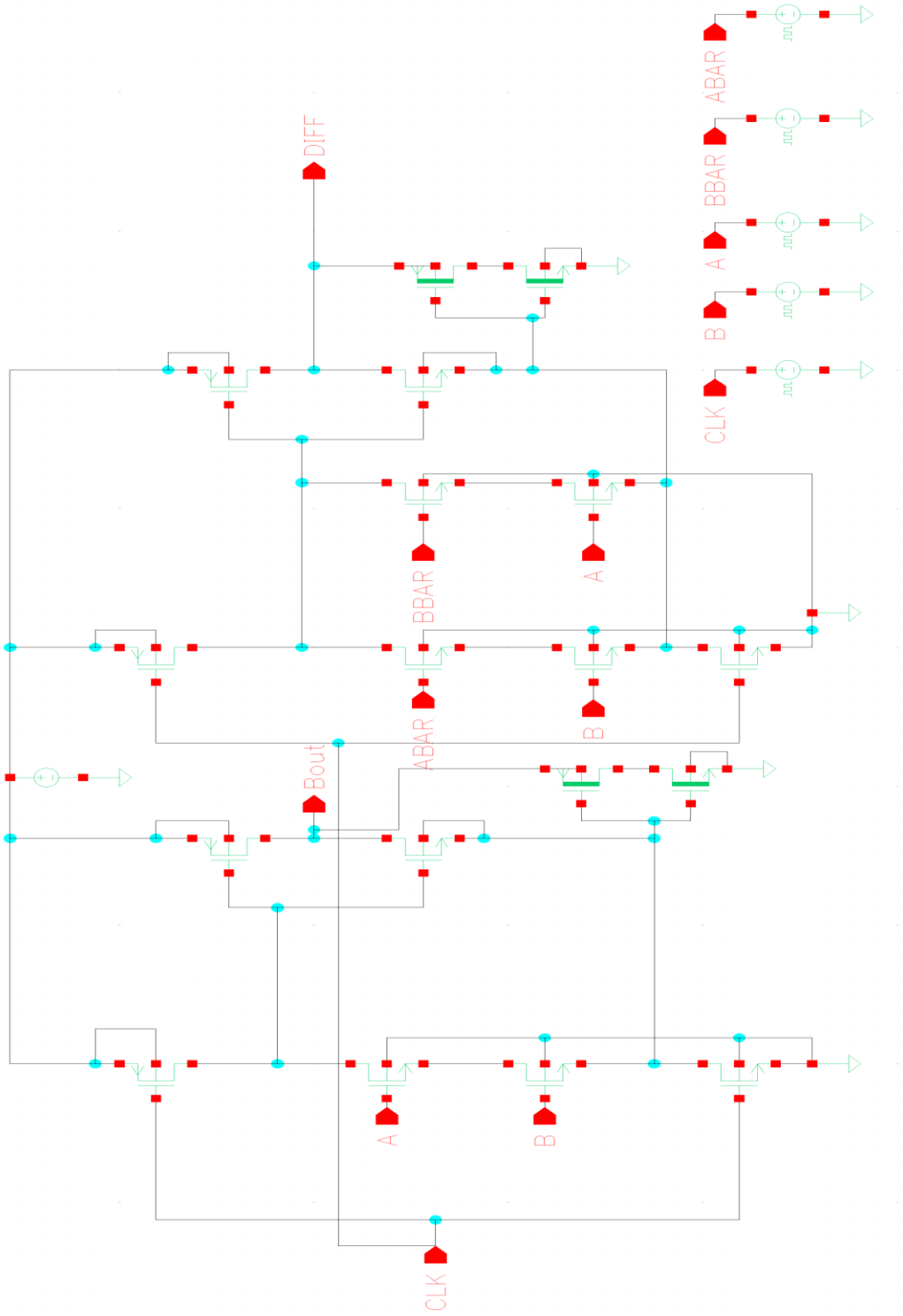

Figure 3.42 VIRTUOSO Schematic of half subtractor using FQR logic

The VIRTUOSO schematic of half subtractor using FQR logic is illustrated in Fig. 3.42.

3.12 CHARACTERIZATION OF 1-BIT FULL ADDER

3.12.1 1-bit Full Adder using domino logic

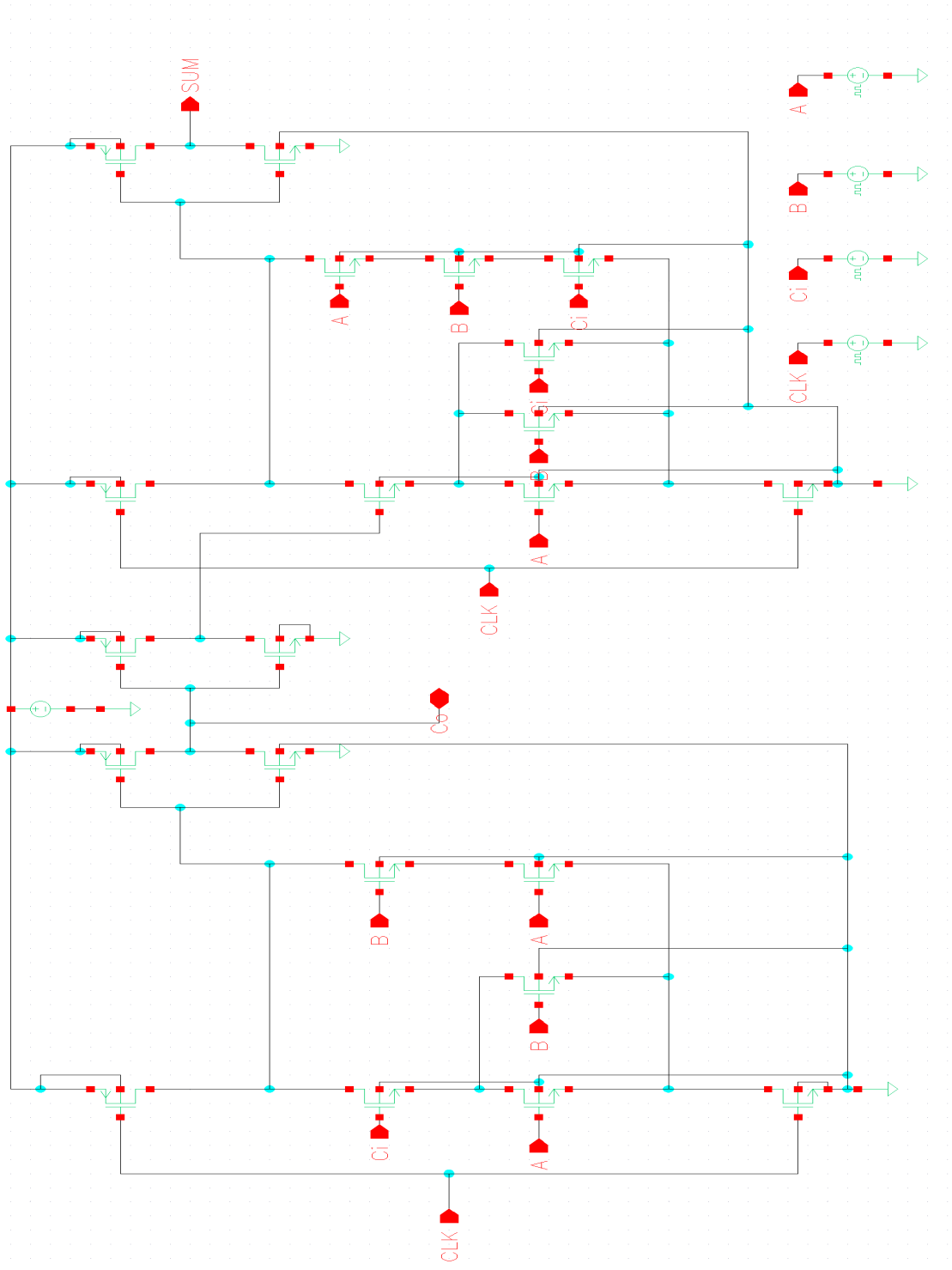

Figure 3.43 VIRTUOSO Schematic of 1-bit full adder using domino logic

The VIRTUOSO schematic of 1-bit full adder using domino logic is illustrated in Fig. 3.43. Timing Waveform of 1-bit full adder, is illustrated in Fig. 3.44.

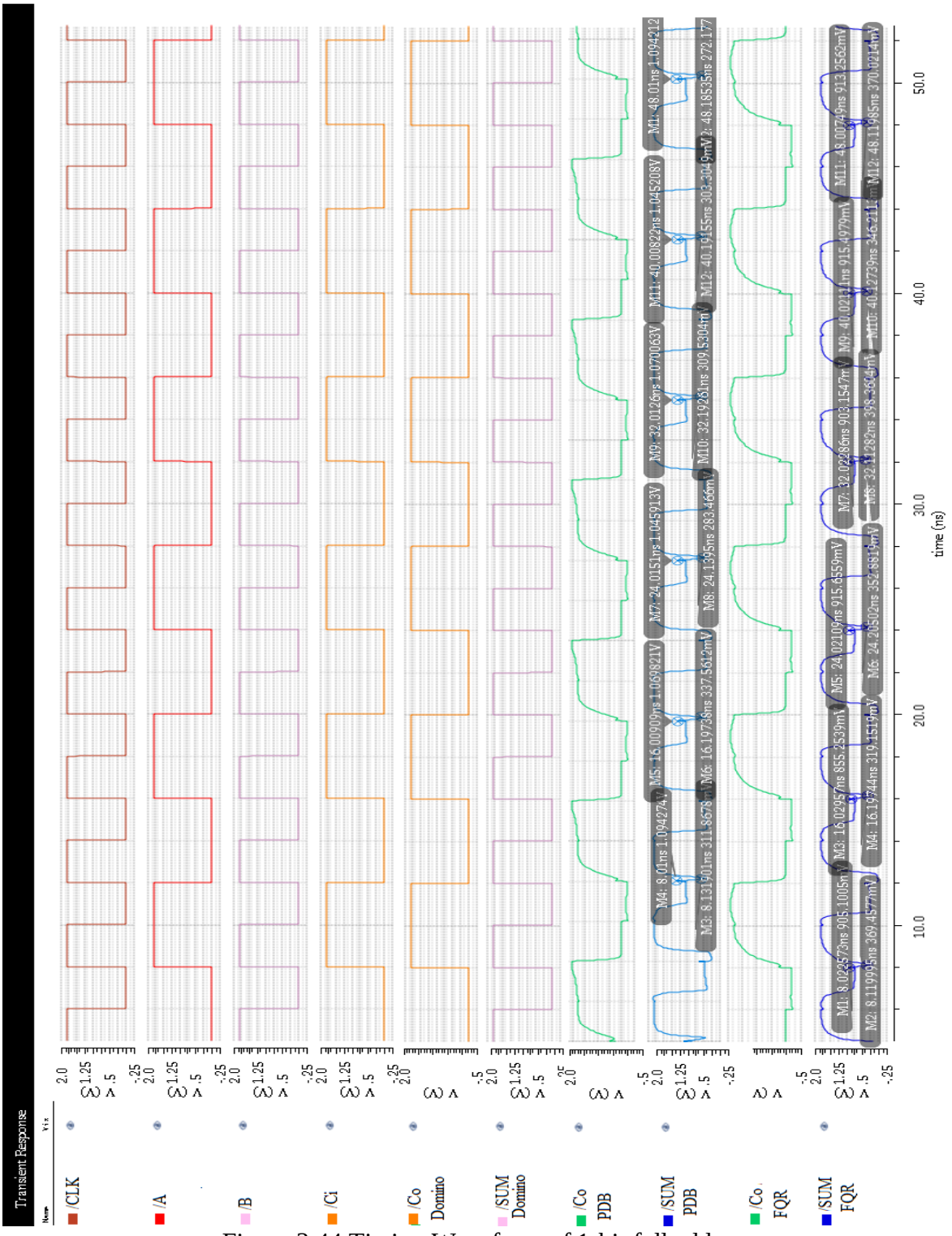

Figure 3.44 Timing Waveform of 1-bit full adder

3.12.2 1-bit Full Adder using PDB logic

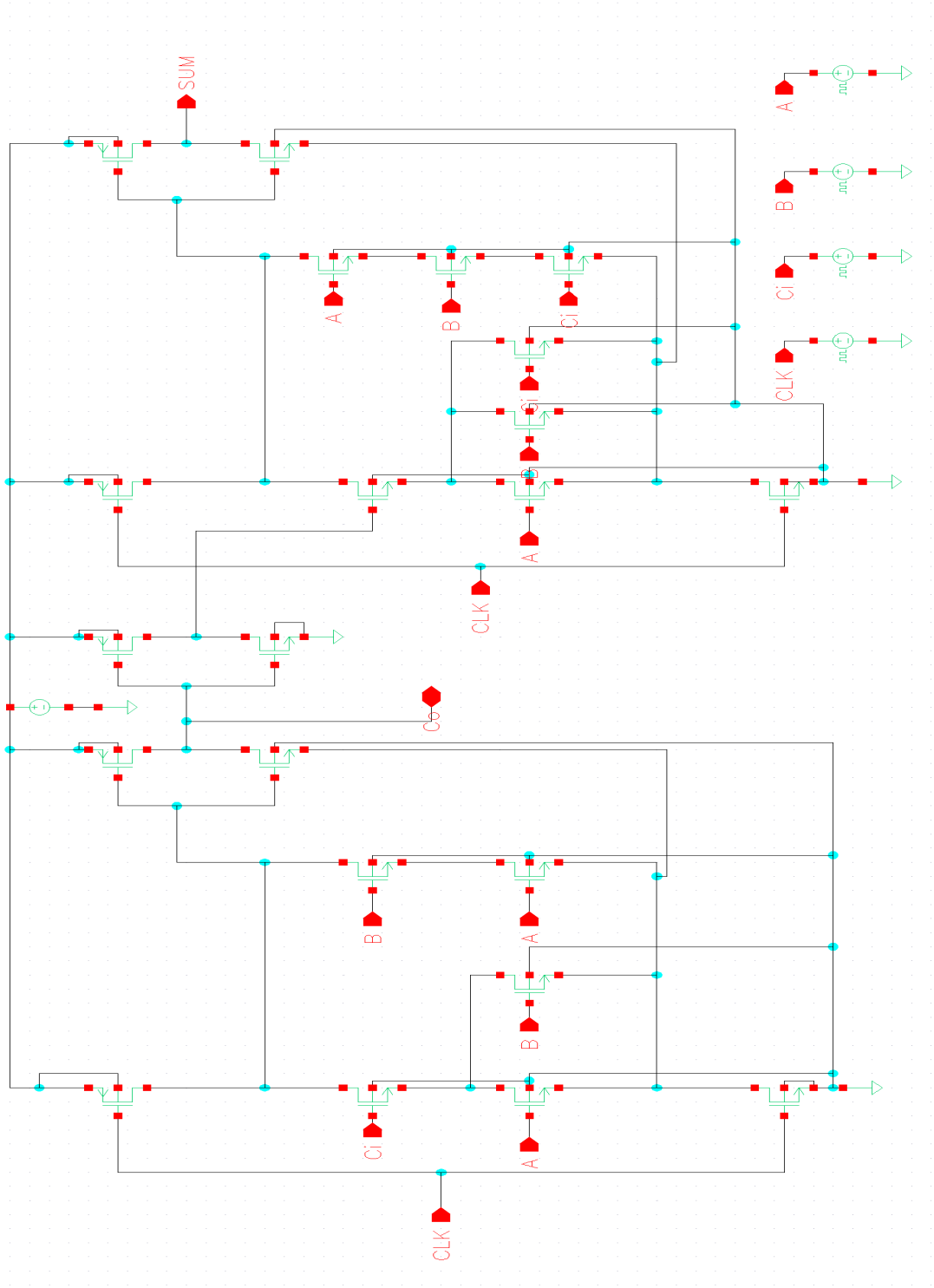

Figure 3.45 VIRTUOSO Schematic of 1-bit full adder using PDB logic

The VIRTUOSO schematic of 1-bit full adder using PDB logic is illustrated in Fig. 3.45.

3.12.3 1-bit Full Adder using FQR logic

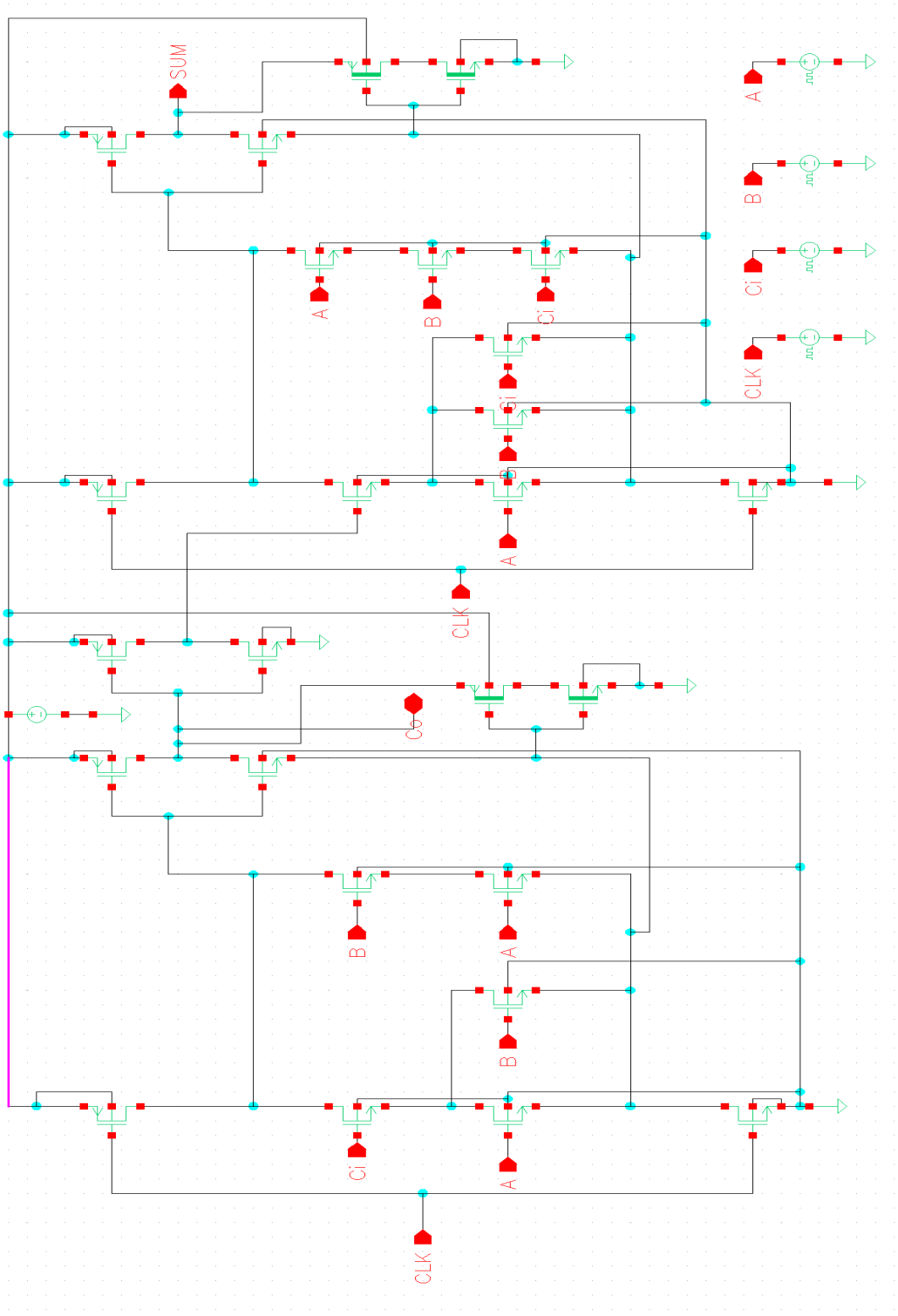

Figure 3.46 VIRTUOSO Schematic of 1-bit full adder using FQR logic

The VIRTUOSO schematic of 1-bit full adder using FQR logic is illustrated in Fig. 3.46.

3.13 CHARACTERIZATION OF 2-BIT FULL ADDER

3.13.1 2-bit Full Adder using domino logic

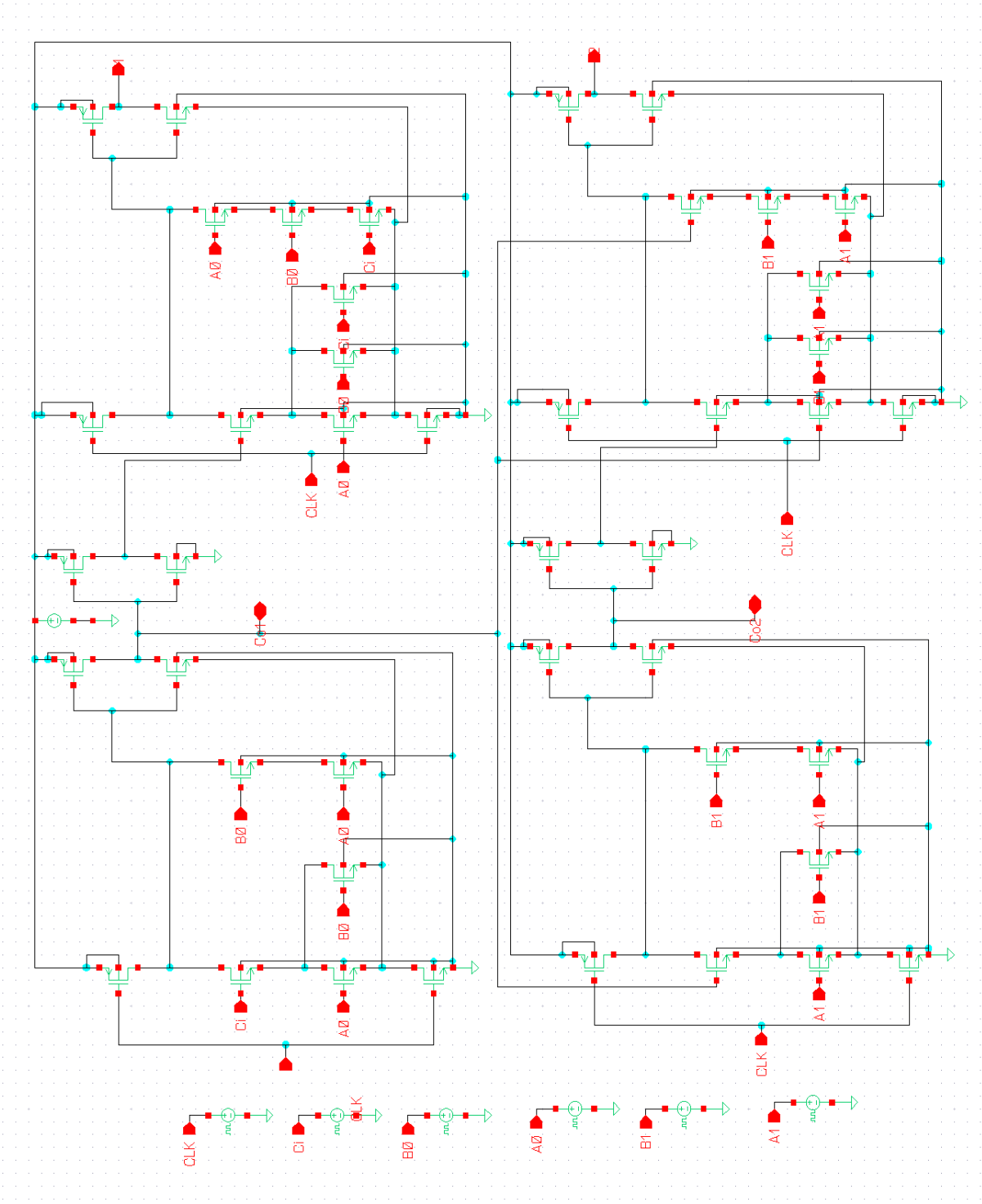

Figure 3.47 VIRTUOSO Schematic of 2-bit full adder using domino logic

The VIRTUOSO schematic of 2-bit full adder using domino logic is illustrated in Fig. 3.47. Timing Waveform of 2-bit full adder, is illustrated in Fig. 3.48.

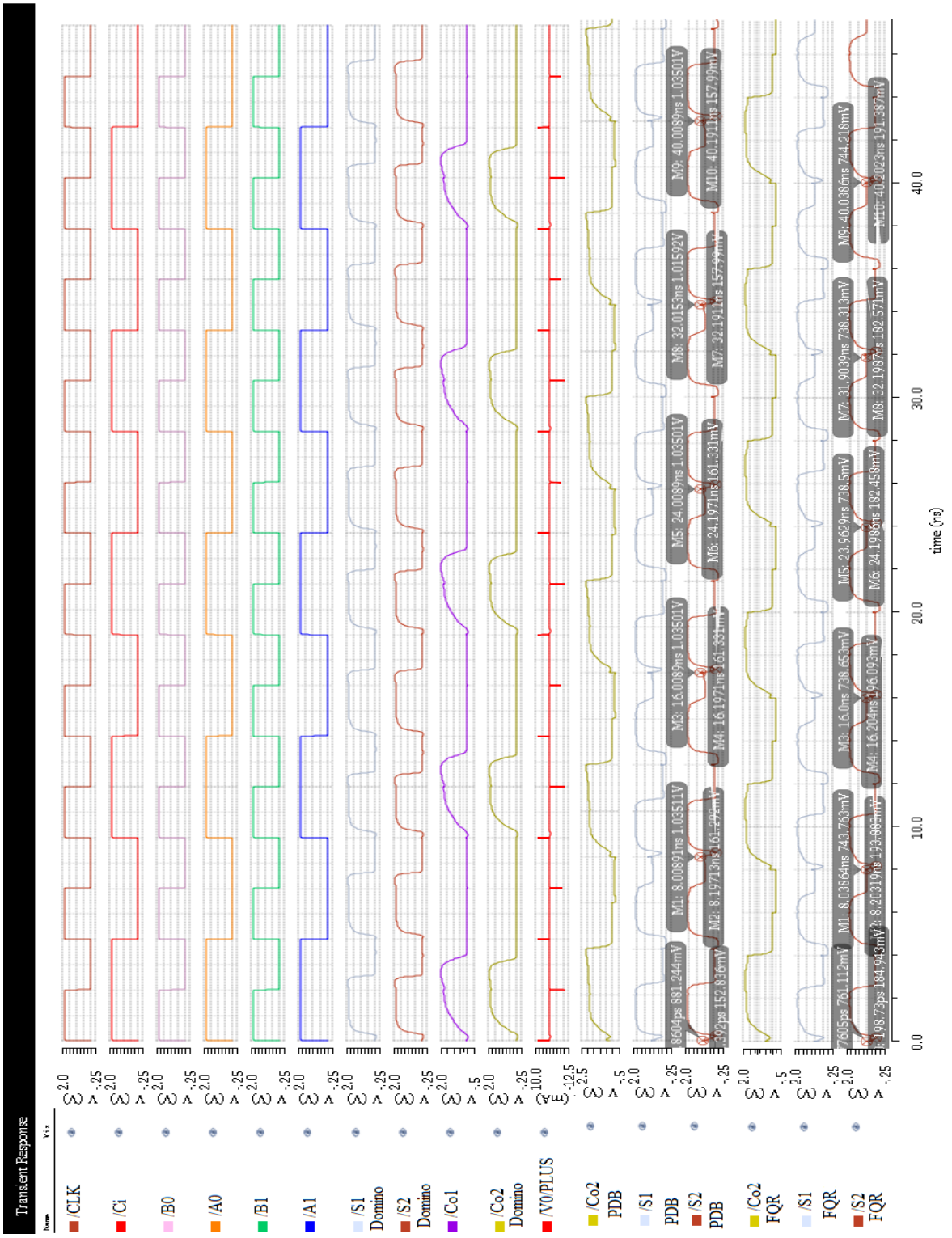

Figure 3.48 Timing Waveform of 2-bit full adder

3.13.2 2-bit Full Adder using PDB logic

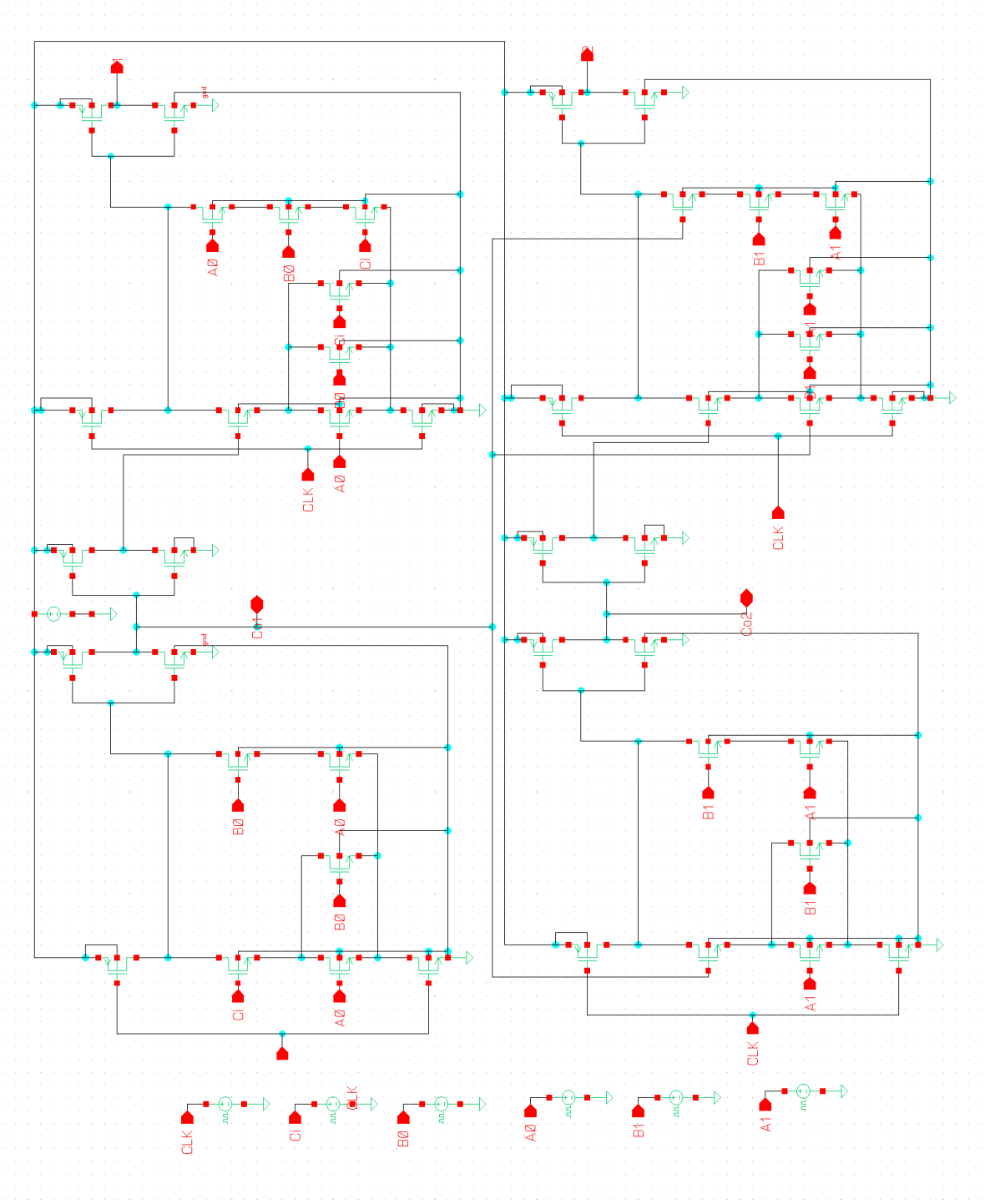

Figure 3.49 VIRTUOSO Schematic of 2-bit full adder using PDB logic

The VIRTUOSO schematic of 2-bit full adder using PDB logic is illustrated in Fig. 3.49.

3.13.3 2-bit Full Adder using FQR logic

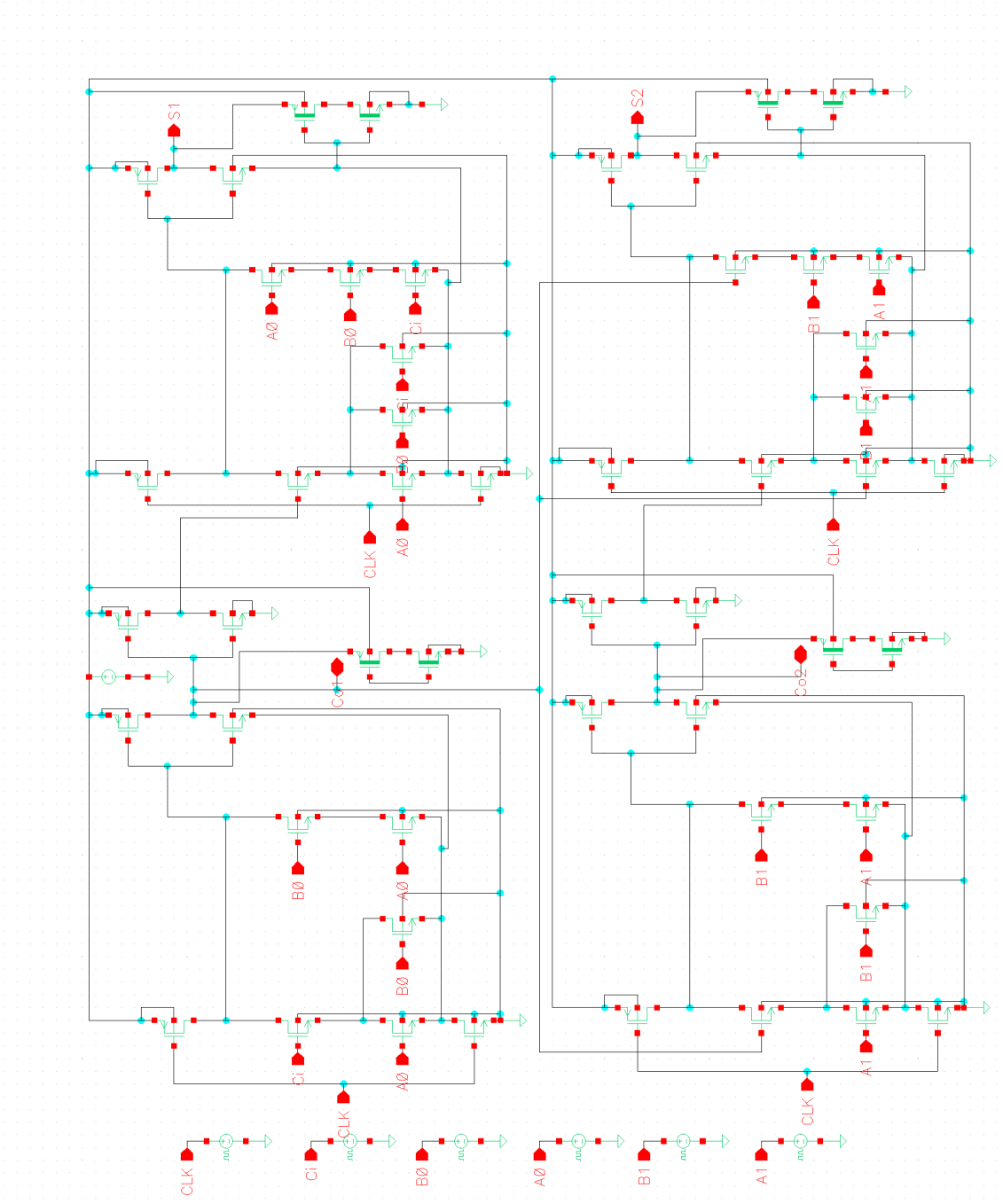

Figure 3.50 VIRTUOSO Schematic of 2-bit full adder using FQR logic

The VIRTUOSO schematic of 2-bit full adder using FQR logic is illustrated in Fig. 3.50.

3.14 CHARACTERIZATION OF 1-BIT FULL SUBTRACTOR

3.14.1 1-bit Full Subtractor using domino logic

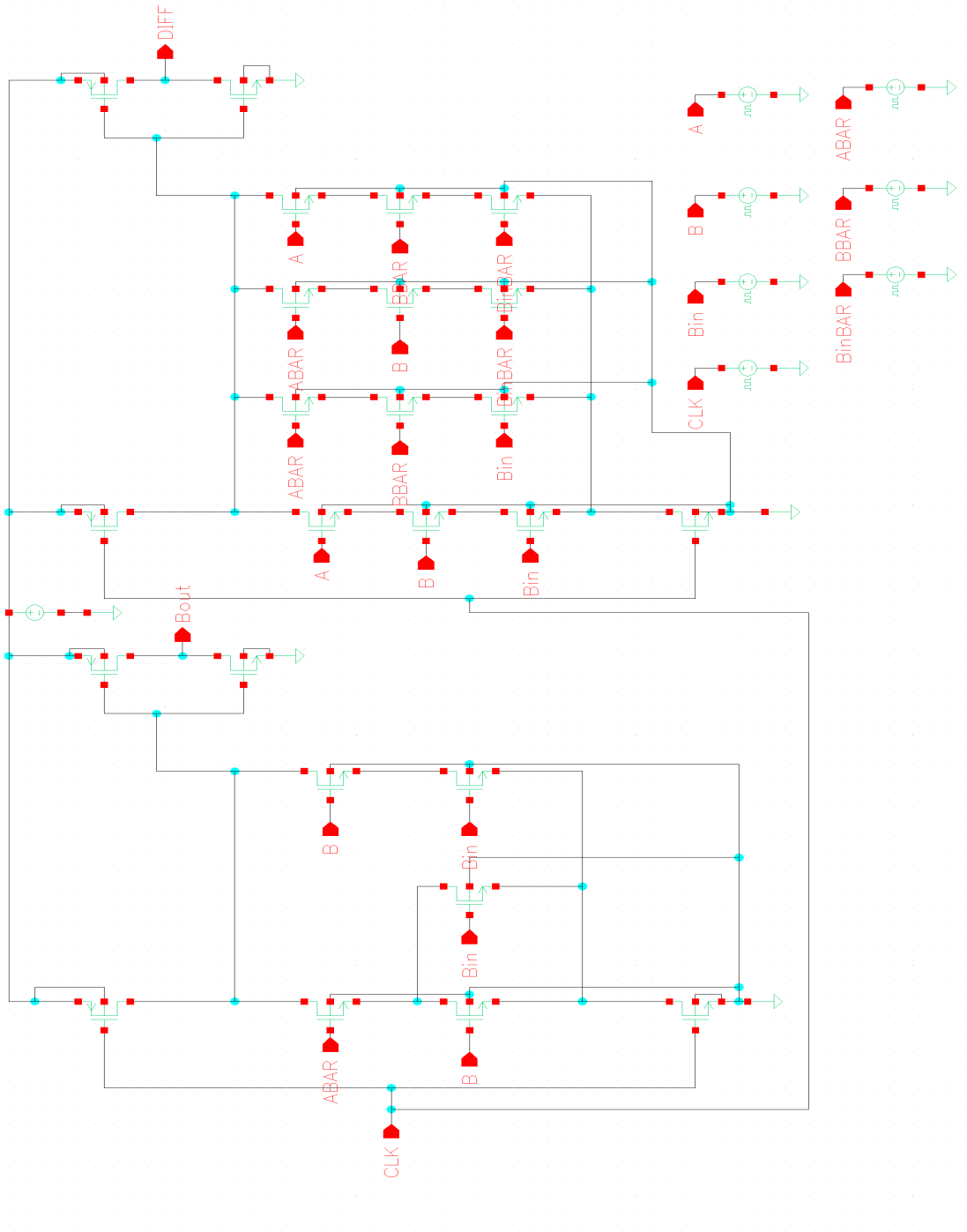

Figure 3.51 VIRTUOSO Schematic of 1-bit full subtractor adder using domino logic

The VIRTUOSO schematic of 1-bit full subtractor using domino 1ogic is illustrated in Fig. 3.51. Timing Waveform of 1-bit full subtractor, is illustrated in Fig. 3.52.

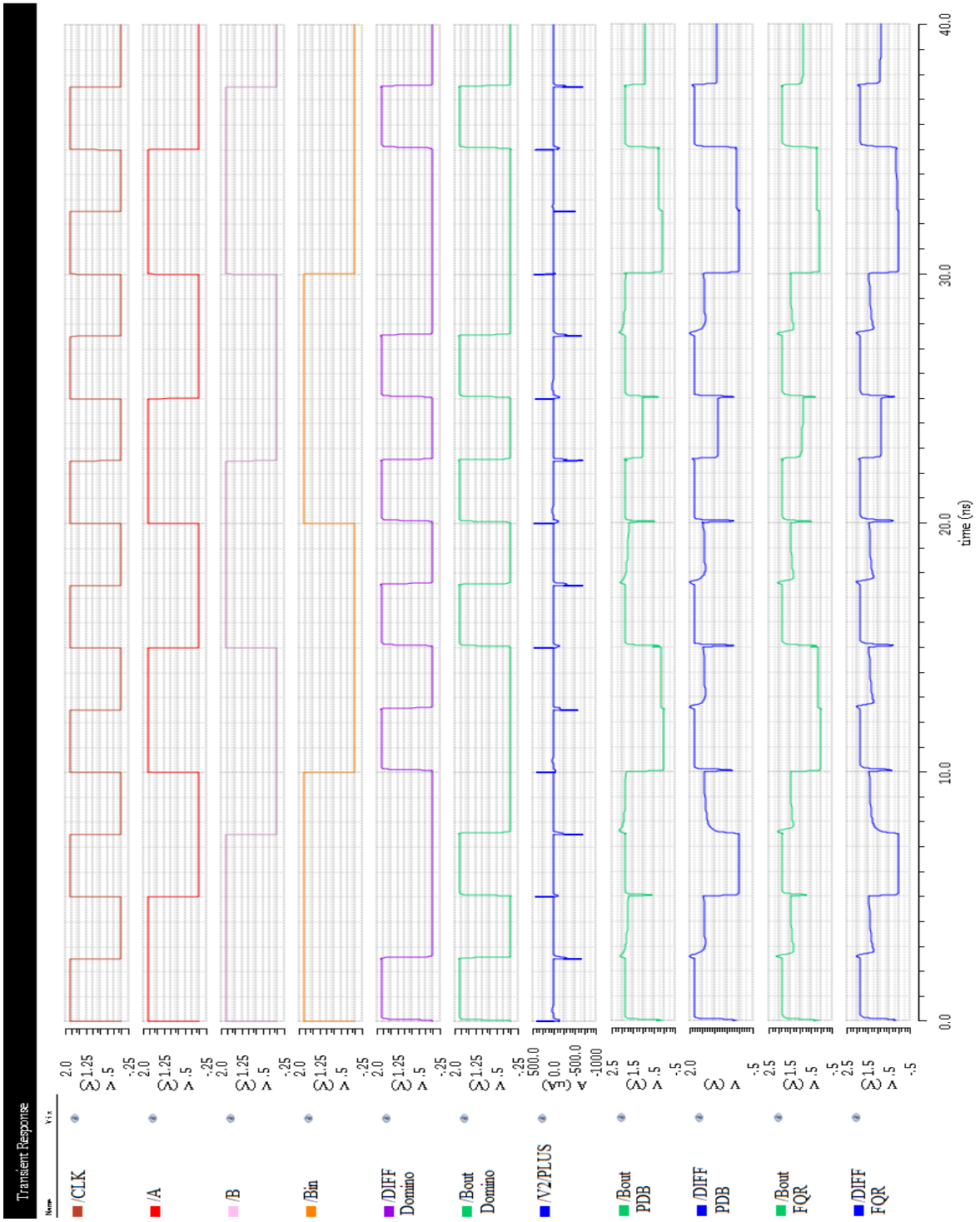

Figure 3.52 Timing Waveform of 1-bit full subtractor

3.14.2 1-bit Full Subtractor using PDB logic

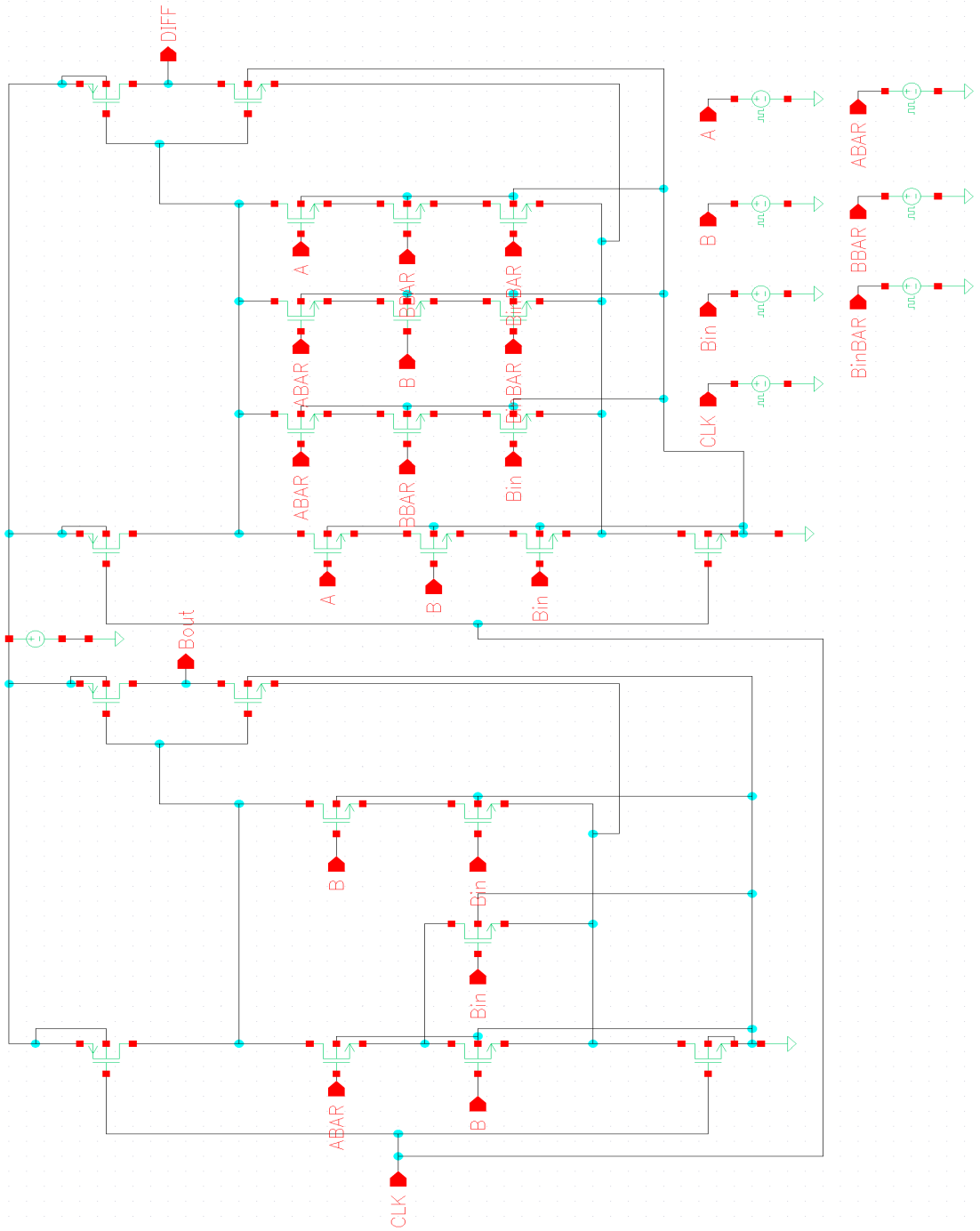

Figure 3.53 VIRTUOSO Schematic of 1-bit full subtractor using PDB logic

The VIRTUOSO schematic of 1-bit full subtractor using PDB logic is illustrated in Fig. 3.53.

3.14.3 1-bit Full Subtractor using FQR logic

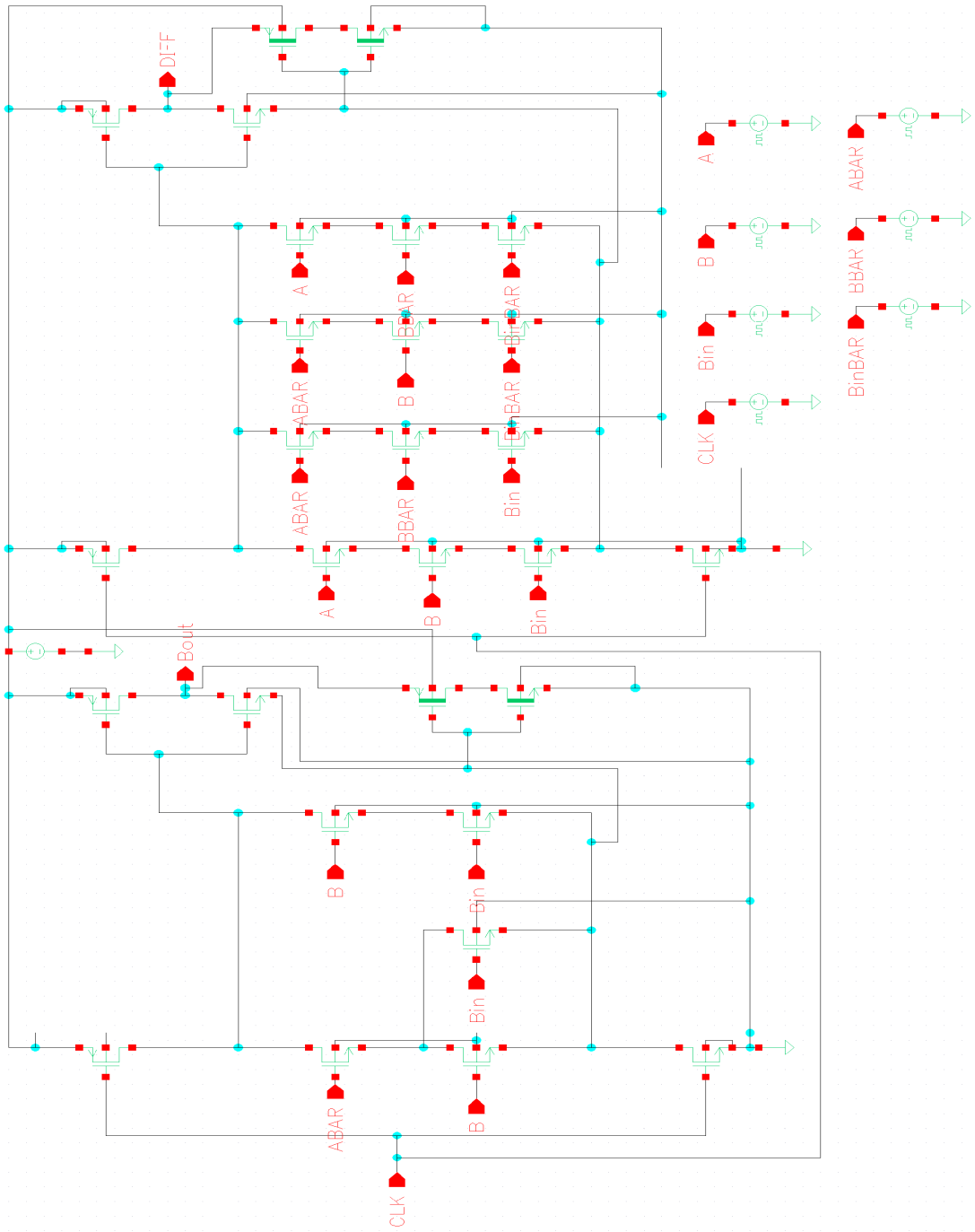

Figure 3.54 VIRTUOSO Schematic of 1-bit full subtractor using FQR logic

The VIRTUOSO schematic of 1-bit full subtractor using FQR logic is illustrated in Fig. 3.54.

3.15 CHARACTERIZAION OF INCREMENTER

3.15.1 Incrementer using domino logic

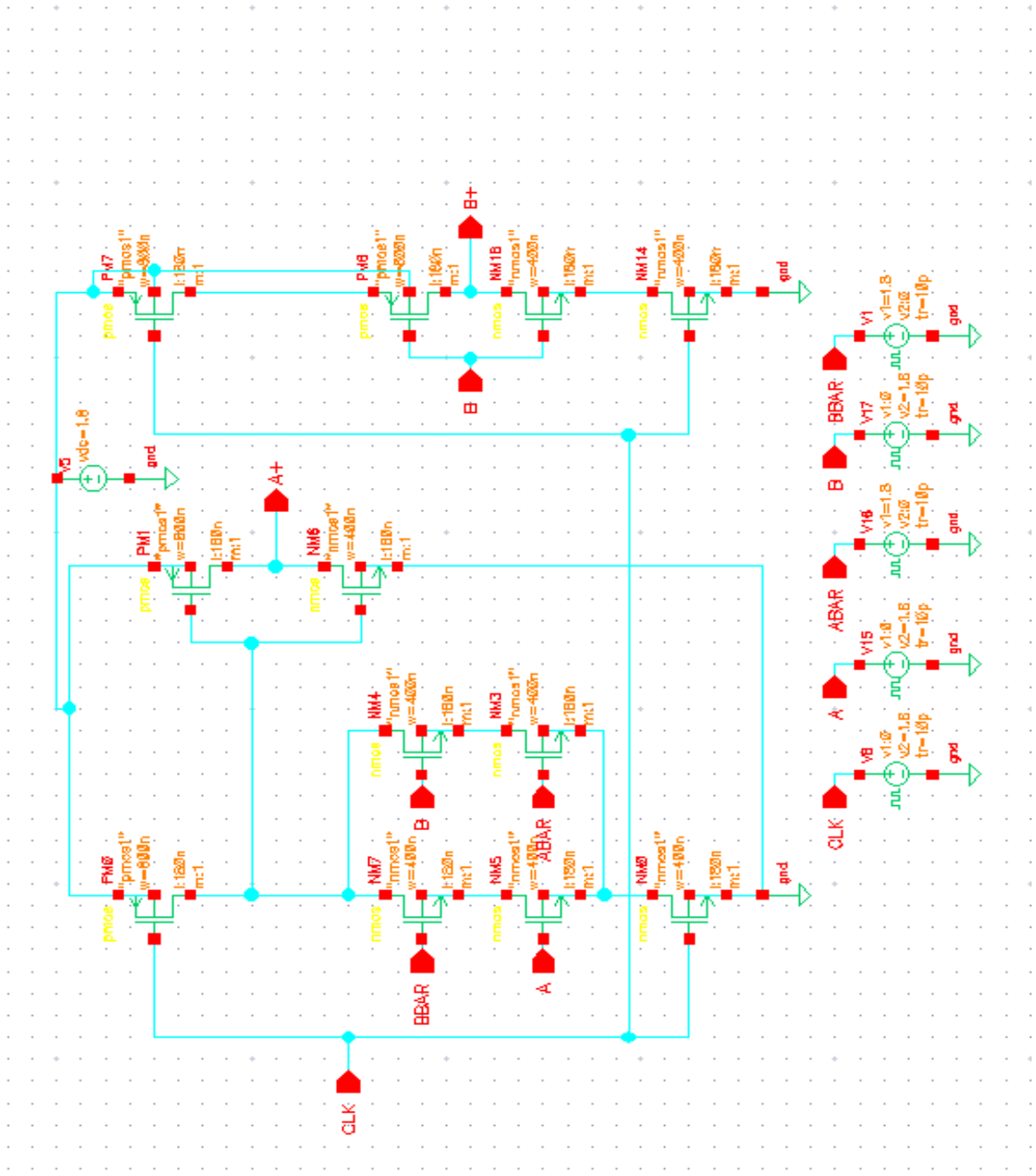

Figure 3.55 VIRTUOSO Schematic of incrementer using domino logic

The VIRTUOSO schematic of incrementer using domino logic is illustrated in Fig. 3.55. Timing Waveform of incrementer, is illustrated in Fig. 3.56.

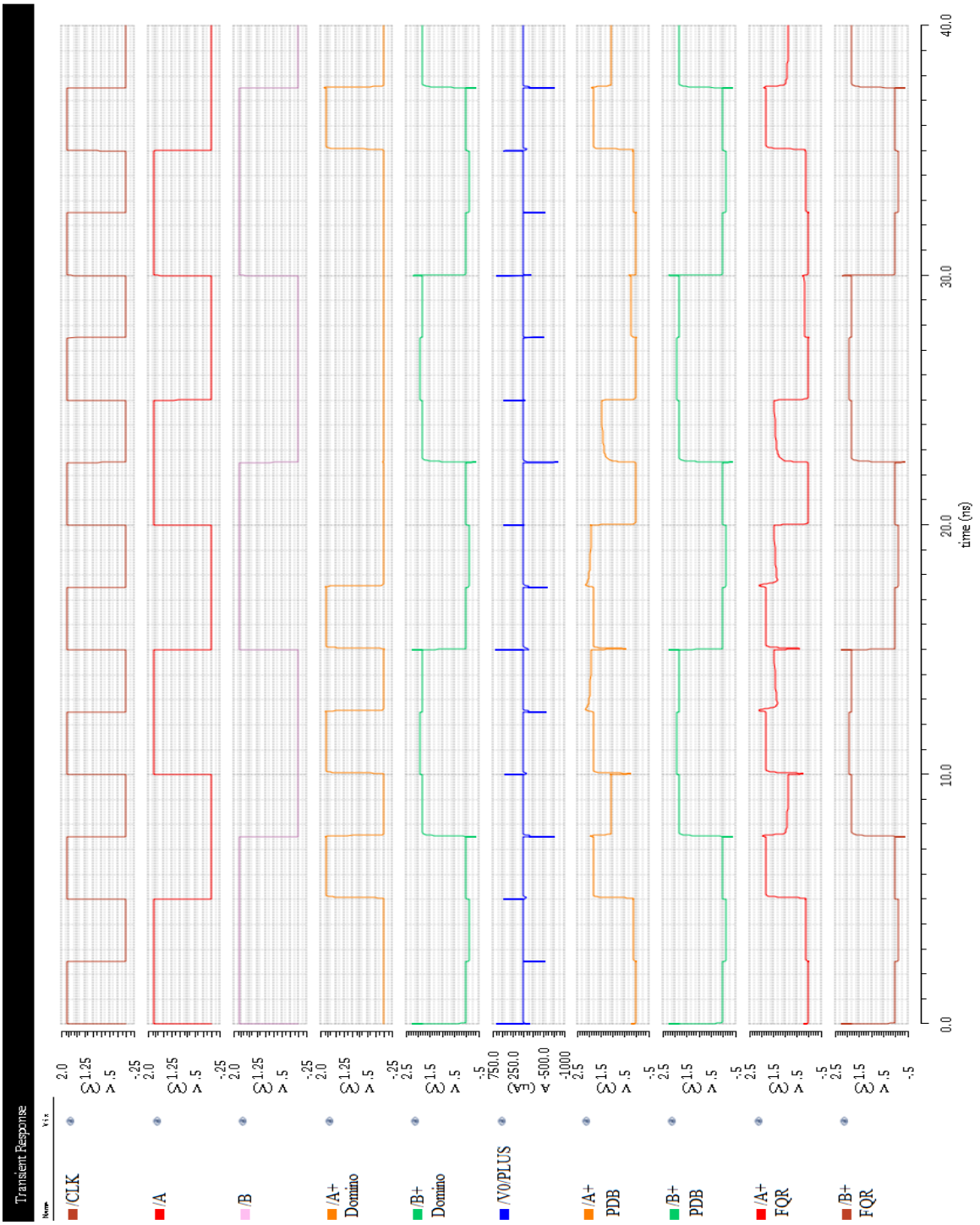

Figure 3.56 Timing Waveform of incrementer

3.15.2 Incrementer using PDB logic

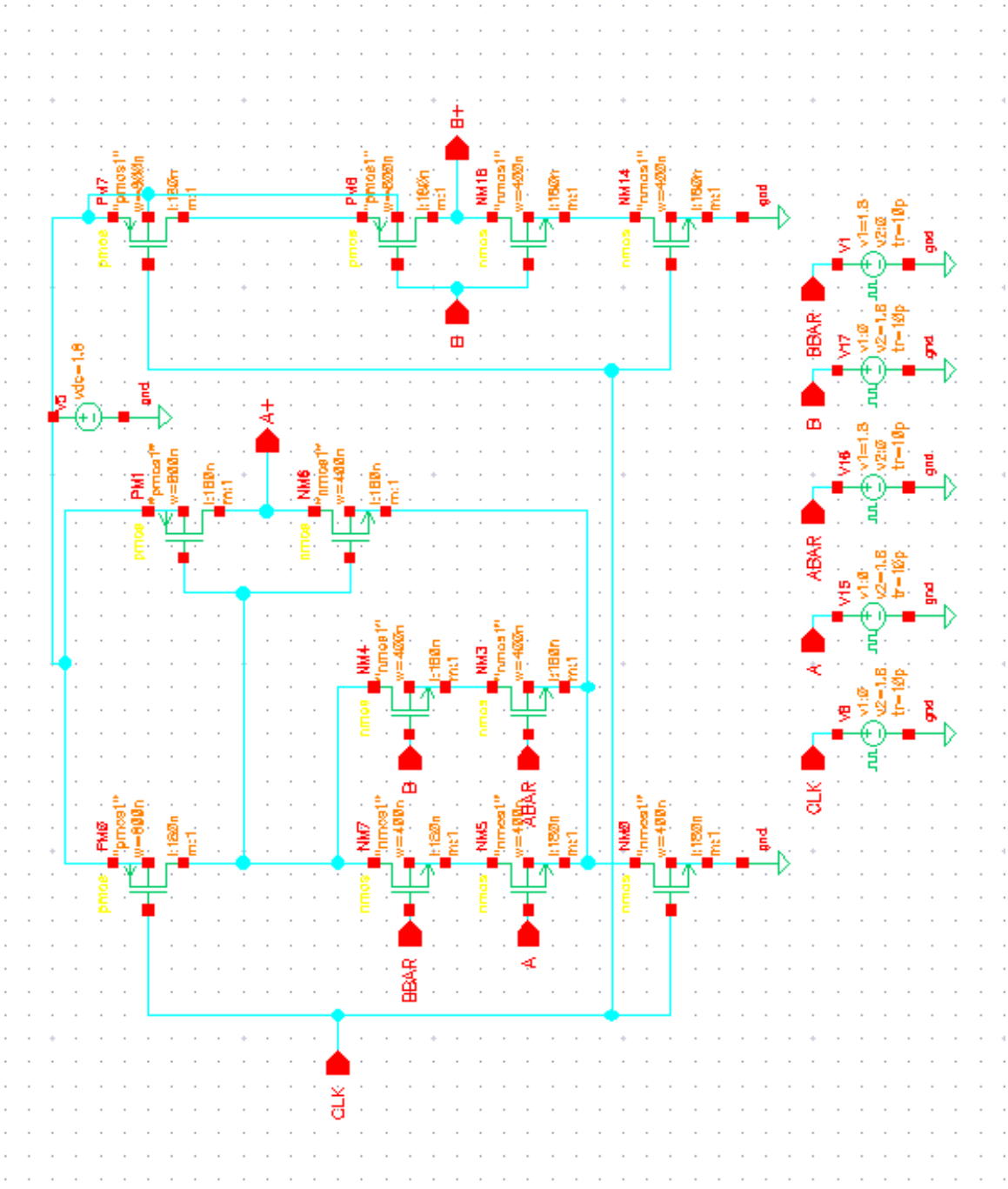

Figure 3.57 VIRTUOSO Schematic of incrementer using PDB logic

The VIRTUOSO schematic of incrementer using PDB logic is illustrated in Fig. 3.57.

3.15.3 Incrementer using FQR logic

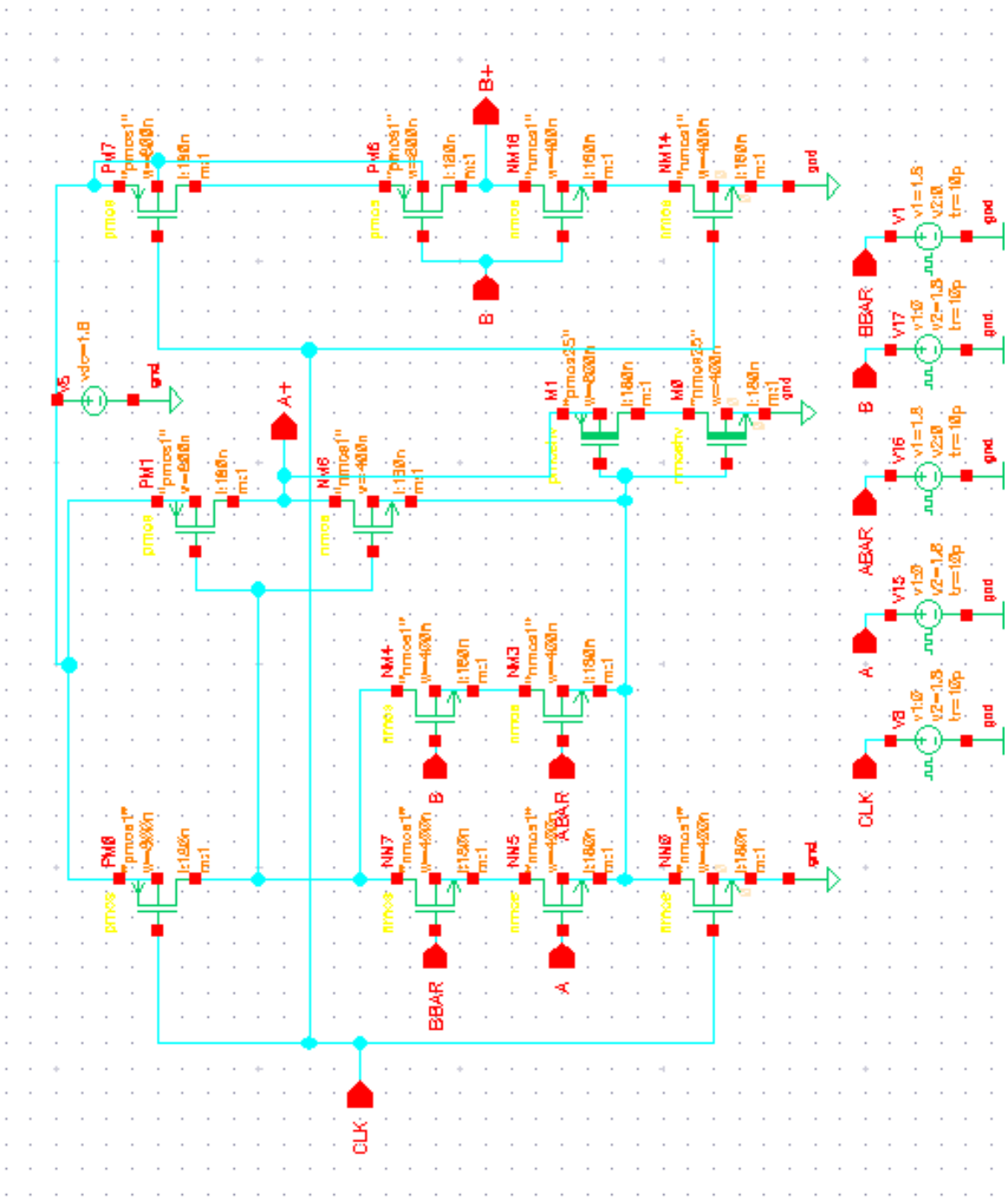

Figure 3.58 VIRTUOSO Schematic of incrementer using FQR logic

The VIRTUOSO schematic of incrementer using FQR logic is illustrated in Fig. 3.58.

3.16 CHARACTERIZATION OF DECREMENTER

3.16.1 Decrementer using domino logic

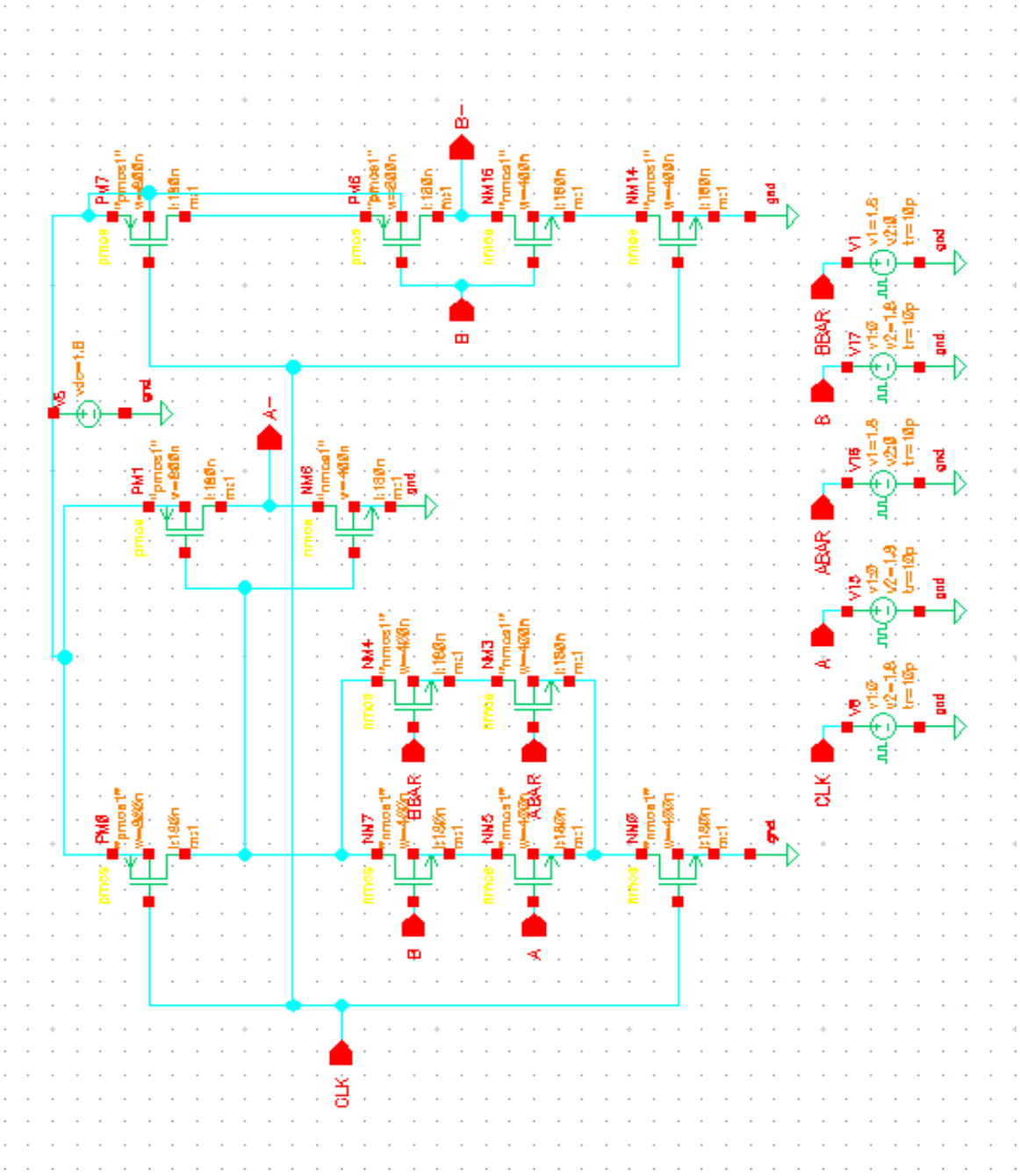

Figure 3.59 VIRTUOSO Schematic of decrementer adder using domino logic

The VIRTUOSO schematic of decrementer using domino logic is illustrated in Fig. 3.59. Timing Waveform of decrementer, is illustrated in Fig. 3.60.

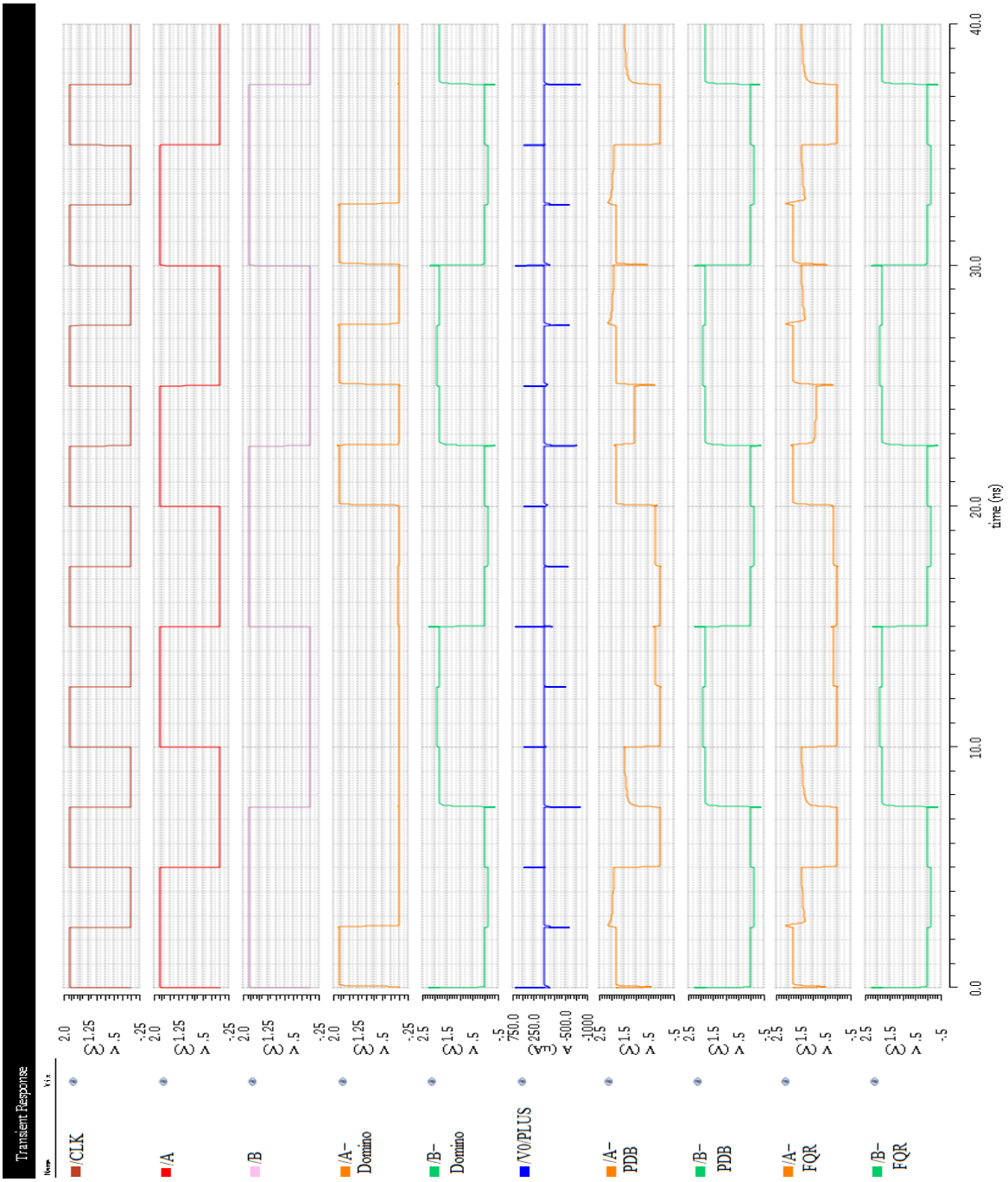

Figure 3.60 Timing Waveform of decrementer

3.16.2 Decrementer using PDB logic

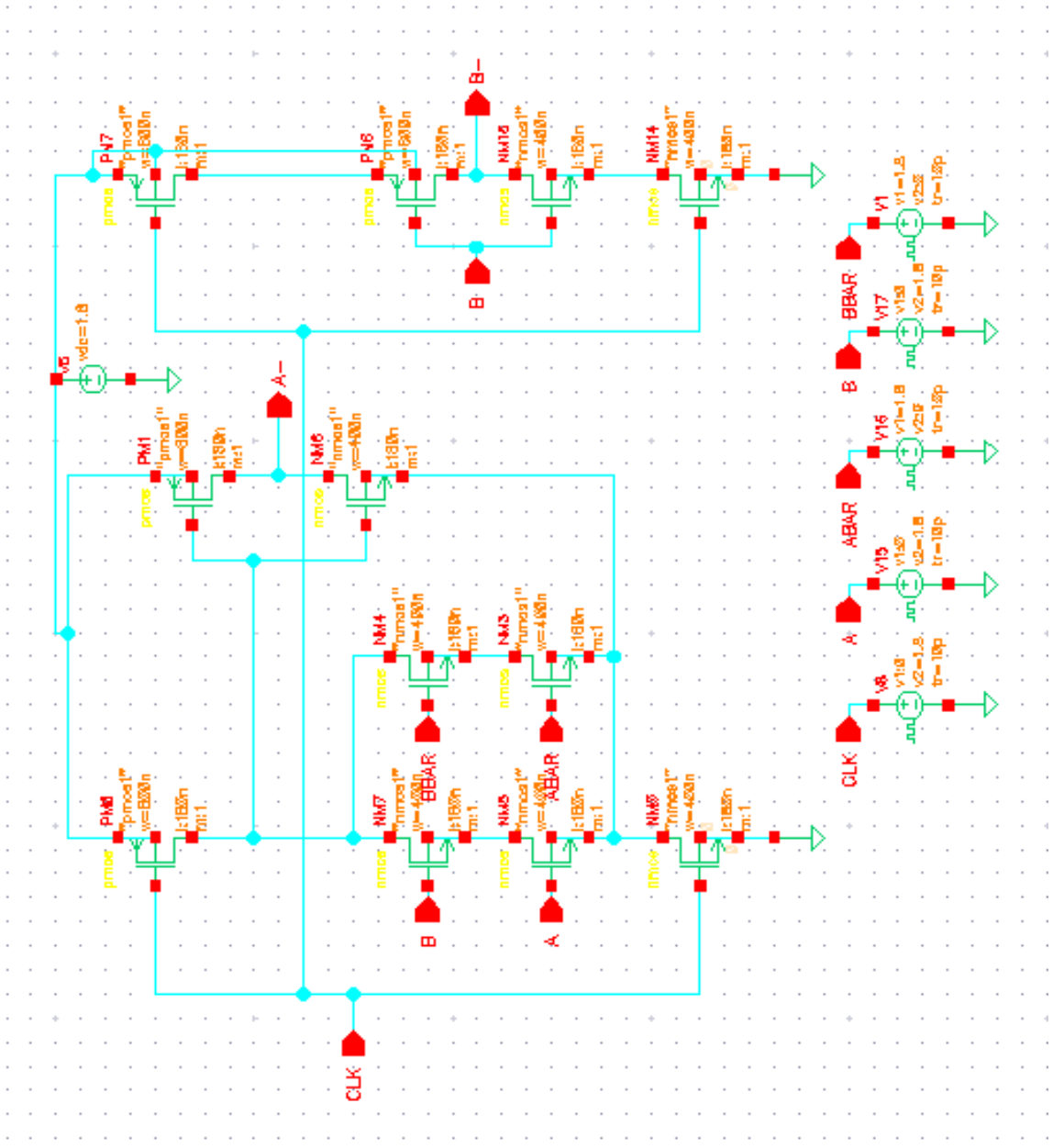

Figure 3.61 VIRTUOSO Schematic of decrementer using PDB logic

The VIRTUOSO schematic of decrementer using PDB logic is illustrated in Fig. 3.61.

3.16.3 Decrementer using FQR logic

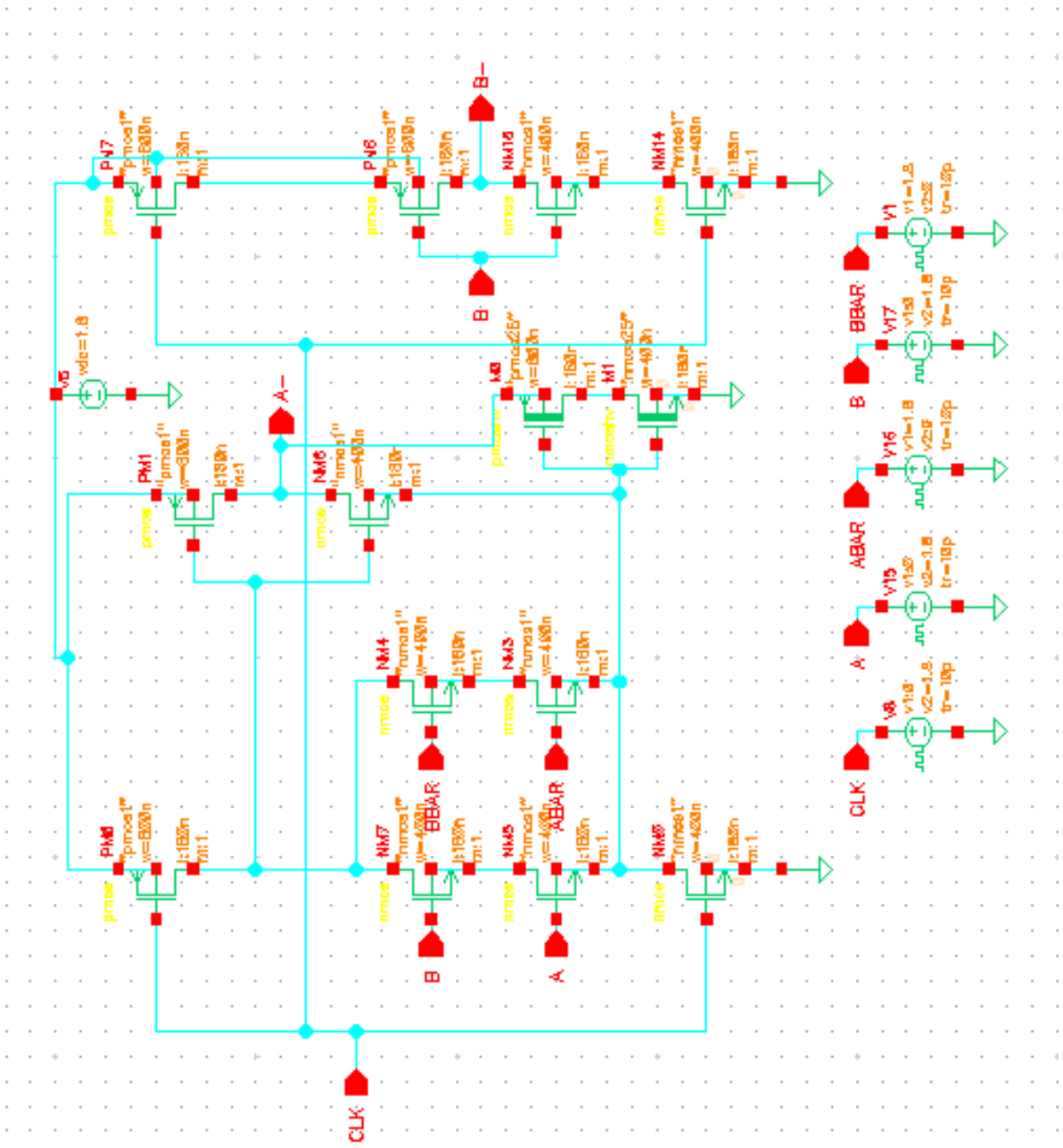

Figure 3.62 VIRTUOSO Schematic of decrementer using FQR logic

The VIRTUOSO schematic of decrementer using FQR logic is illustrated in Fig. 3.62.

3.17 ARITHMETIC LOGIC FUNCTIONS IMPLEMENTED

The following table represents the arithmetic logic functions implemented in this work.

ARITHMETIC LOGIC FUNCTION(S)	OUTPUT
Transfer Input A	A
Logical AND Gate	A.B
Logical OR Gate	A+B
Logical NAND Gate	(A.B)'
Logical NOR Gate	(A+B)'
Logical XOR Gate	(A'.B)(AB')
Logical XNOR Gate	(A'.B')(A.B)
Half Adder	SUM = (A⊕B) = A.B' + A'.B CARRY = A.B
Half Subtractor	DIFFERENCE = (A⊕B) = A.B'+ A'.B BORROW = A'.B
Full Adder	SUM = (A⊕B⊕Cin) Cout = (A.B)+(B.Cin)+(A.Cin)
Full Subtractor	DIFF = (A⊕B⊕Bin) Bout = A'Bin + A'B + BBin
Incrementer	(Ai + Bi) +1
Decrementer	(Ai + Bi) -1

Table 3.1 Arithmetic Logic Functions Implemented

CHAPTER 4

SIMULATION RESULTS AND ANALYSIS

To evaluate and analyse the performance of all the arithmetic logic functions implemented using domino logic, pseudo dynamic buffer logic and footed quasi resistance model, we have used 180nm technology node in VIRTUOSO, Cadence.

4.1 DELAY EVALUATION AT DIFFERENT CLOCK FREQUENCIES

Clock Frequency (MHz)	**Delay(ns)**		
	Domino	**PDB**	**FQR**
1000	0.12	0.61	0.89
500	0.21	0.86	1.15
250	1.86	1.89	2.16
100	4.27	4.9	5.11

Table 4.1 Delay evaluation at different clock frequencies

4.2 POWER DISSIPATION AT DIFFERENT CLOCK FREQUENCIES

Clock Frequency (MHz)	**1000**	**500**	**250**	**100**
Power in Domino (µW)	33.68	16.93	8.50	3.41
Power in PDB (µW)	18.83	9.45	4.76	1.92
Power in FQR (µW)	23.18	11.62	5.84	2.34
Power Saving (%) (PDB Vs FQR)	-23.10	-22.96	-22.68	-21.87
Power Saving (%) (Domino Vs FQR)	31.17	31.36	31.29	31.38

Table 4.2 Power dissipation at different clock frequencies

4.3 DELAY EVALUATION AT 180nm TECHNOLOGY

Arithmetic Functions	Delay (ns)		
	Domino	PDB	FQR
Buffer	1.62	2.06	2.32
Logica1 AND Gate	3.08	3.12	3.32
Logica1 OR Gate	0.11	0.18	0.23
Logica1 NAND Gate	0.19	0.25	0.34
Logical NOR Gate	5.13	5.38	5.6
Logica1 XOR Gate	1.11	1.23	1.55
Logica1 XNOR Gate	5.01	5.16	5.47
Ha1f Adder	5.18	5.21	5.92
Ha1f Subtractor	5.03	5.19	5.45
1-Bit Fu11 Adder	2.57	3.59	4.17
2-Bit Fu11 Adder	3.45	3.99	4.29
Fu11 Subtractor	2.98	4.01	5.09
Incrementer	1.79	2.01	2.25
Decrementer	1.01	1.44	1.68

Table 4.3 Delay Analysis at 180nm

4.4 POWER DISSIPATION AT 180nm TECHNOLOGY

Arithmetic Functions	Power(uW)			Power Saving (%)	
	Domino	PDB	FQR	Domino Vs FQR	PDB Vs FQR
Buffer	15.63	8.74	10.75	31.22	-22.99
Logica1 AND Gate	3.93	2.88	3.38	13.99	-17.36
Logica1 OR Gate	4.57	1.88	2.15	42.95	-14.36
Logica1 NAND Gate	4.45	2.09	2.43	45.39	-16.27
Logical NOR Gate	3.18	1.88	2.05	35.53	-9.04
Logica1 XOR Gate	5.75	3.47	3.97	30.96	-14.40
Logica1 XNOR Gate	4.64	4.05	4.14	10.78	-2.22
Ha1f Adder	11.66	9.51	9.94	14.75	-4.52
Ha1f Subtractor	11.32	9.75	10.10	12.19	-3.59
1-Bit Fu11 Adder	34.69	20.3	21.06	39.29	-3.74
2-Bit Fu11 Adder	112.6	86.8	89.6	20.42	-3.22
Fu11 Subtractor	39.68	21.22	23.04	41.93	-8.57
Incrementer	7.07	5.18	5.45	22.91	-5.21
Decrementer	10.56	5.85	6.05	42.71	-3.42

Table 4.4 Power Dissipation Analysis at 180 nm

4.5 POWER-DELAY PRODUCT AT 180nm TECHNOLOGY

Arithmetic Functions	Power-Delay Product (PDP)		
	Domino	**PDB**	**FQR**
Buffer	25.24	18.01	24.91
Logical AND Gate	11.92	8.96	11.22
Logical OR Gate	0.51	0.35	0.47
Logical NAND Gate	0.85	0.50	0.82
Logical NOR Gate	16.31	10.11	11.48
Logical XOR Gate	6.38	4.26	6.14
Logical XNOR Gate	23.25	20.89	22.64
Half Adder	60.39	49.50	58.84
Half Subtractor	56.93	50.61	55.04
1-Bit Full Adder	89.15	72.87	87.50
2-Bit Fu11 Adder	699.25	623.39	691.89
Full Subtractor	118.35	85.09	117.17
Incrementer	12.65	10.41	12.26
Decrementer	10.55	8.42	10.16

Table 4.5 Power-Delay Product at 180 nm

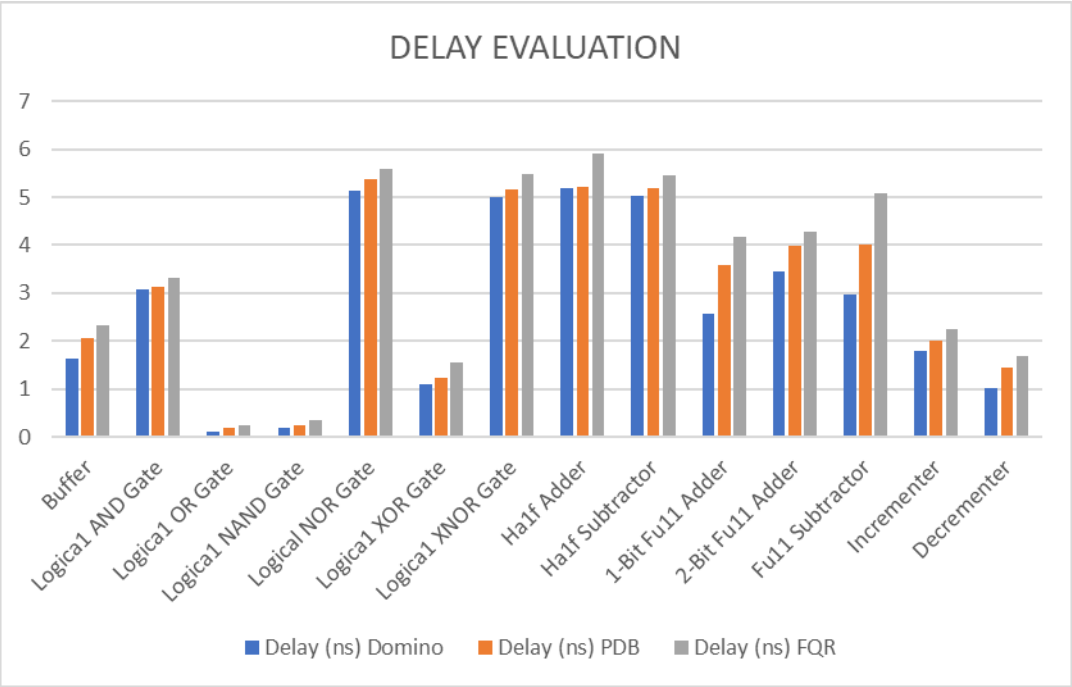

Figure 4.1 Bar graph for delay evaluation

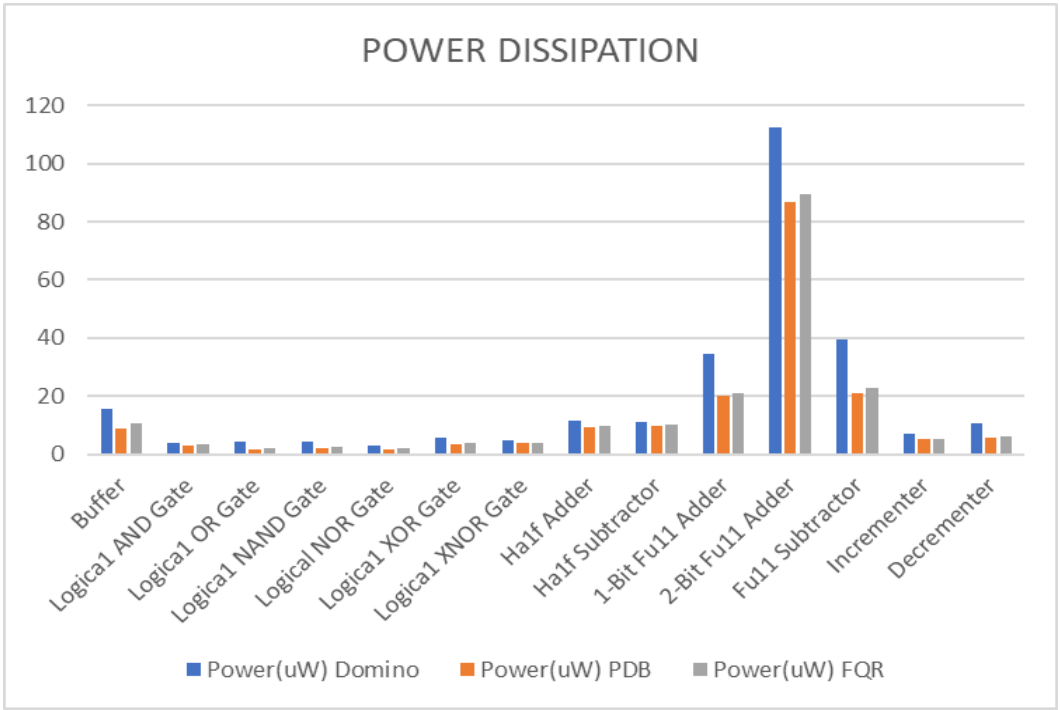

Figure 4.2 Bar graph for power dissipation

4.6 AVERAGE AND STANDARD DEVIATION EVALUATION OF FULL ADDER

1-Bit Full Adder		**2-Bit Full Adder**	
PDB	**FQR**	**PDB**	**FQR**
0.778	0.535	0.878	0.577
0.722	0.536	0.874	0.550
0.761	0.563	0.854	0.542
0.760	0.505	0.843	0.556
0.793	0.544	0.860	0.553
0.822	0.532	0.873	0.562
0.784	0.534	0.876	0.561
0.765	0.521	0.838	0.549
0.746	0.541	0.856	0.544
0.758	0.556	0.842	0.539
MEAN = 0.7502	**MEAN** = 0.5398	**MEAN** = 0.8496	**MEAN** = 0.5518
SD = 0.0302	**SD** = 0.0223	**SD** = 0.03261	**SD** = 0.0113

Table 4.6 Average and Standard Deviation Evaluation

ARITHMETIC FUNCTIONS	**CORRELATION COEFFICIENT**		**% SAVING**
	Domino Vs PDB	Domino Vs FQR	
Buffer	0.653	0.721	10.41
Logical AND	0.774	0.866	11.88
1-Bit Full Adder	0.884	0.995	12.56
2-Bit Full Adder	0.823	0.954	15.91

Table 4.7 Correlation Analysis

As depicted in the Fig. 3.40 and Fig. 3.44, it can be inferred from the timing waveforms of 1-bit full adder and 2-bit full adder that FQR implementation results in ‘good 1’ and ‘poor 0’ when the output makes a transition from logic 1 to logic 0. Noise Margin for logic ‘0’ should be high. Due to that, the noise margin for logic ‘1’ in PDB implementation becomes significantly less, i.e., almost negligible. This results in faulty output. Though, for FQR implementation the noise margin for logic ‘1’ has significant scope to detect the input signal. Therefore, the functionality of a cascaded logic circuit is better for FQR model.

From the Table 4.6, it can be observed that a sample data is taken for the PDB and FQR implementation of 1-bit full adder and 2-bit full adder. For the taken data values, Mean and Standard Deviation has been evaluated. From the results, it can be observed that the standard deviation for the PDB logic increases significantly for the cascading logic function. This signifies that for the PDB logic, the ripples in the circuit enhances when complex logic circuit is designed. Hence, it can be deduced that noise margin is negatively affected which may further lead to faulty output.

For FQR implementation, the standard deviation reduces for the logic designs with multiple cascading stages. This causes less voltage transitions to propagate through the logic circuit and hence provides reduced variation in the voltage level at the output of multiple cascaded stages. Hence, even if the power consumed in FQR logic is higher than that of PDB logic, FQR logic provides better results.

From Table 4.7, the correlation of the FQR and PDB logic waveforms with respect to the domino logic has been evaluated for buffer, logical AND gate, 1-bit full adder, 2-bit full adder. The output of domino logic implementation of a function is approximately ideal. The implementation of buffer, logical AND gate, 1-bit full adder and 2-bit full adder depicts increasing percentage of correlation by 15% which signifies that the FQR waveform for a logic function is significantly identical to domino logic. Thus, from the evaluated values it can be derived that the output of FQR topology is closer to the domino logic model. Hence, FQR logic produces better output as compared to the PDB logic for a complex logic function.

In this work, two performances of the above circuits are analyzed: Delay and Power dissipation. These performances and metrics are measured for all arithmetic logic functions based on domino logic buffer model, pseudo dynamic buffer model and footed quasi resistance model. Each of the designs is implemented to determine the optimal trade-off between delay-power dissipation.

From above evaluation, we can observe that Power Saving of Footed Quasi Resistance Model Footed Quasi Resistance model (FQR) is less compared with PDB. However, it is not beneficial to implement cascaded circuits using PDB if we want to avoid compromising on power saving. The evaluation of the simulation results obtained, leads to an observation that as the clock frequency is increased, the power dissipation as well as the delay of the FQR model correspondingly increases as well. This implies that power saving (%) for domino logic is increased with the rise in clock frequency. By above simulation results we can observe that the FQR logic has less power dissipation of 29.58% as compared to domino logic.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

The domino logic buffer suffers from the issue of precharge pulse propagation. This problem is overcome by using PDB topology. Implementation of a logic function using PDB topology eliminates precharge pulse propagation. However, PDB technique poses a problem while designing of cascaded circuits to execute a logic function. In this work, we used footed quasi resistance technique for implementing logic functions since it is capable of resolving the problems due to cascading. However, there is one drawback of this implementation: Increase in power dissipation. Even with increased power dissipation, this technique allows the designer to implement cascaded designs. This observation is drawn by analyzing the power dissipation and delay of domino logic, PDB, and FQR logic models for implementation of different logic functions. To determine the accurate results, the power consumption of domino logic is compared with that of PDB topology and FQR topology. Thus, the Footed Quasi Resistance model exhibits power trade-off to overcome the cascading issues of PDB domino logic.

Footed quasi resistance topology can be further utilized to design an Arithmetic Logic Unit (ALU) for implementing all the logic functions described in this thesis. Experimental verification of the obtained results can be performed for logic circuits having higher degree of cascading and complex logic designs.

REFERENCES

[1] Bokare, U. M., & Gaidhani, Y. A. (2017). Design of CMOS dynamic logic circuits to improve noise immunity. 2017 International Conference on Communication and Signal Processing (ICCSP).

[2] R.G.D.Jeyasingh, N.Bhat and B.Amrutur, "Adaptive Keeper Design for Dynamic Logic Circuits Using Rate Sensing Teehnique," IEEE Trans. Very Large Seale Integr.(VLSI) Syst., Feb. 2011.

[3] H. F. Dadgour and K. Banerjee, “A novel variation tolerant keeper architecture for high-performance low-power wide fan in dynamic or gates,” IEEE Trans. Very Large Scale (VLSI) Syst., vol. 18, Nov. 2010.

[4] F.Mendoza-Hernandez, M.Linares-Aranda, V. Champac, Noisetolerance improvement in dynamic CMOS logic circuits. In: proceedings of the lEE circuits, devices and systems, Dec 2006.

[5] L. Ding and P. Mazumder, “On circuit techniques to improve noise immunity of CMOS dynamic logic,” IEEE Trans. Very Large Scale Integer. Syst., Sep. 2004.

[6] Tyler J. Thorp, Member, IEEE, Gin S. Yee, Member, IEEE, and Carl M. Sechen, Fellow, IEEE, “Design and Synthesis of Dynamic Circuits”, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 11, NO.1, FEBRUARY 2003.

[7] R.H. Krambeck, C. M. Lee, H.-F.S. "Law,High-speed compact circuits with CMOS",IEEE Journal of Solid-State Circuits, SC-17 (1982).

[8] G.Yee and C. Sechen, “Dynamic logic synthesis,” in Proc. IEEE Custom Integrated Circuits Conf., May 1997.

[9] P. Larsson, and C. Svensson, "Noise in digital dynamic CMOS circuits," IEEE Journal of Solid-State Circuits, vol. 29, June. 1994.

[10] V. Friedman and S. Liu, “Dynamic logic CMOS circuits,” IEEE J. Solid- State Circuits, vol. SC-19, pp. 263–266, Apr. 1984. [8] G.Yee and C.Sechen, “Dynamic logic synthesis,” in Proc. IEEE Custom Integrated Circuits Conf., May 1997.

[11] M.W. Allah, M.H. Anis and M.l. Elmasry, "High speed dynamic logic circuits for scaled-down CMOS and MTCMOS technologies" in Proc. IEEE Inter. Symp. Low Power Electronics Design, July 2000.

[12] N. F. Goncalves and H. J. De Man, "NORA: A dynamic CMOS technique for pipelined logic structures, IEEE J . Solid State Circuits, vol. SC-18, June 1983.

[13] J. Kuo, J. Lou, "Low-Voltage CMOS VLSI Circuits", Wiley interscience, NewYork, 1999.

[14] Y. Lih, N. Tzartzanis and W. W. Walker, "A Leakage Current Replica Keeper for Dynamic Circuits," in IEEE Journal of Solid-State Circuits, vol. 42, Jan. 2007.

[15] Y.Ji-Ren, I.Karlsson, C.Svensson, A true single-phase-clock dynamic CMOS circuits technique, IEEE journal of solid state circuits 22, October1987.

[16] R. H. Krambeck, C. M. Lee and H. F. S. Law, "High-speed compact circuits with CMOS,"IEEE Journal of Solid State Circuits, vol. 17, Jun. 1982.

[17] A. Alvandpour, R. K. Krishnamurthy, K. Soumyanath, and S. Y. Borkar, "A conditional keeper technique for sub-130nm wide dynamic gates," in Proceedings of Intertional Symposium on VLSI Circuits, 2001.

[18] M. Anis, Mohab H., Mohamed W. Allam, and Mohamed I. Elmasry. "Energy-efficient noise-tolerant dynamic styles for scaled-down CMOS and MTCMOS technologies." IEEE Transactions on Very Large Scale Integration (VLSI) Systems

10.2 (2002).

[19] M. H. Anis, M.W. Allam, and M. I. Elmasry, "High-speed dynamic logic styles for scaled-down CMOS and MTCMOS technologies," in Proceedings of International Symposium on Low- Power Electronics and Design, 2000.

[20] Atila Alvandpour,, Ram K. Krishnamurthy, K. Soumyanath and Shekhar Y. Borkar, “A Sub-130-nm Conditional Keeper Technique”, IEEE Journal of solidstate circuits, vol. 37, May 2002.

[21] Ankita Sharma , Divyanshu Rao and Ravi Mohan, “Design and Implementation of Domino Logic Circuit in CMOS”, Journal of Network Communications and Emerging Technologies (JNCET) Volume 6, Issue 12, December (2016) .

[22] A. Peiravi and M. Asyaei, “Robust low leakage controlled keeper by current-comparison domino for wide fan-in gates, integration,” VLSI J., 2012.

[23] Shiksha and K. K. Kashyap, "High speed domino logic circuit for improved performance," 2014 Students Conference on Engineering and Systems, Allahabad, 2014.

[24] Roy, K., Mukhopadhyay, S. and Mahmoodi-Meimand, H., 2003. Leakage current mechanisms and leakage reduction techniques in deep sub-micrometer CMOS circuits. Proceedings of the IEEE.

[25] Ravikumar, R., “Double Stage Domino Technique: Low-Power HighSpeed Noise-tolerant Domino Circuit for Wide Fan-In Gates”. International Journal of Engineering and Technology (IJET) Vol 8 No 3 Jun-Jul 2016 (2016).

[26] A. Peiravi and M. Asyaei, "Current-Comparison-Based Domino: New Low-Leakage High-Speed Domino Circuit for Wide Fan-In Gates," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, May 2013.

[27] Rangari, A. V., & Gaidhani, Y. A. (2016). Design of comparator using Domino Logic and CMOS Logic. 2016 Online International Conference on Green Engineering and Technologies (IC-GET).

[28] A. Manikandan and J. Ajayan, “High Speed Low Power 64-Bit comparator designed Using Current Comparison Based Domino Logic,” IEEE sponsored 2nd international conference on electronics and communication systems (icecs ‘2015).

[29] D.Y.Ponomarev, G. Kueuk, O.Ergin, K. Ghose, "Energy Effieient Comparators for Supersealar Datapaths", IEEE Trans. Computers, vol. 53, July 2004.

[30] Satwik Patnaik, Shruti Mehrotra “A Low-Power, Area Efficient Design Technique for Wide Fan-in Domino Logic based Comparators” 2013 International Conference on Circuits, Power and Computing Technologies [ICCPCT2013].

[31] F.Moradi, APeiravi and H.Mahmoodi, "A High Speed and Leakage Tolerant Domino Logie for High Fan-in Gates," Proc. 15th ACM Great Lakes Symp. on VLSI (GLSVLSI'05), 2005.

[32] H. Mahmoodi and K. Roy, “Diode-footed domino: A leakage-tolerant high fan-in dynamic circuit design style,” IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 51, Mar. 2004.

[33] Sah, N., & Mittal, E. (2017). An improved domino logic. 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS).

[34] Neha Vaish, Sampath Kumar V, “Energy efficient and high speed domino logic circuits ” Int. Journal of Engineering Research and Applications, Vol. 5, Issue 4, (Part -1) April 2015.

[35] David Van Campenhout, Trevor Mudge, Karem A. Sakallah, “Timing verification of sequential domino circuits”, Proceeding of the 1996 IEEE/ACM International Conference on Computer-Aided Design.

[36] Harris, D., & Horowitz, M. A. (1997). Skew-tolerant domino circuits. IEEE Journal of Solid-State Circuits.

[37] Fang Tang, Amine Bermark,Zhouye Gu,"Low power dynamic logic design using a pseudo dynamic buffer, INTEGRATION", the VLSI journal 45(2012) .

[38] Deepika Bansal, B P Singh, Ajay Kumar, “Comparative analysis of improved Pseudo Domino logic”, 31st National convention of Electronics and Telecommunication Engineers, October 2015.

[39] Thota Ravi Sankar, Sankit R. Kassa, R.K. Nagaria and Ankur Kumar, “Performance Analysis of Footed Quasi Resistance Scheme for Low Power VLSI Circuits”, 1st IEEE International Conference on Power Electronics. Intelligent Control and Energy Systems (ICPEICES-2016).

[40] Wairya. S, Nagaria. R. K and Tiwari. S, 2012, “Comparative performance analysis of XOR-XNOR function based high speed CMOS full adder circuits for low voltage VLSI design”, International Journal of VLSI design and Communication Systems (VLSICS), AIRCC Publication.

[41] Amit Kumar Pandey, Jayant Kumar Tiwari, Ram Awadh Mishra, “Design of New Low Leakage Power Domino XOR Circuit”, International Journal of Computer Applications (0975 – 8887) Volume 65– No.1, March 2013.

[42] Chowdhary. S. R, Banerjee. A, Roy. A and Saha. H, 2008, “A high speed 8 transistor full adder design using novel 3 transistor XOR gates”, International Journal of Electrical and Computer Engineering,vol.3,no.12.

[43] S. Wairya, Himanshu Pandey, R. K. Nagaria and S. Tiwari, Member, IEEE, "UItra low voItage high speed I-bit CMOS adder", Power, Control and Embedded Systems (lCPCES), 20 I 0 International Conference.

[44] Navi, K., Kavehie, O., Rouholamini, M., Sahafi, A., & Mehrabi, S. (2007). A Novel CMOS Full Adder. 20th International Conference on VLSI Design Held Jointly with 6th International Conference on Embedded Systems (VLSID’07).

[45] J.M. Rabey , A Chandrakasan, and B. Nicolic, Digital Integrated Circuits, 2nd ed., Prentice Hall, 2003.

[46] H. Mahmoodi-Meimand and K. Roy, "Diode-footed domino: a leakage tolerant high fan-in dynamic circuit design style," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 3, March2004.

[47] Kumar, S., Singhal, S., Pandey, A. K., & Nagaria, R. K. (2013). Design and simulation of low power dynamic logic circuit using footed diode domino logic. 2013 Students Conference on Engineering and Systems (SCES).

[48] V. Kursun, E.G. Friedman, Domino logic with variable threshold voltage keeper, IEEE Transactions on VLSI Systems, 11 (6) (2003).

[49] Jinn-ShyanWang, Ching-RongChang, ChingweiYeh, Analysis and design of high-speed and low-power CMOS PLAs, IEEE Journal of Solid-State Circuits 36 (8) (2001).

[50] Meher, P., & Mahapatra, K. K. (2011). A high-performance circuit technique for CMOS dynamic logic. 2011 IEEE Recent Advances in Intelligent Computational Systems.

[51] Adarsh, C. S. D., Lakshmi, T. V., & Kamaraju, M. (2017). Implementation and comparative analysis of double gate low power multiplexers using dynamic logic styles. 2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA).

[52] Dadashi, A., Mirmotahari, O., & Berg, Y. (2016). Domino dual-rail, high-speed, NOR logic, with 300mV supply in 90 nm CMOS technology. 2016 IEEE International Symposium on Consumer Electronics (ISCE).

[53] Shinde, J. R., Salankar, S. S., & Shinde, S. J. (2016). Multi-objective optimization domino techniques for VLSI circuit. 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI).

[54] Salendra.Govindarajulu, Dr.T.Jayachandra Prasad, P.Rangappa, “Low Power, Reduced Dynamic Voltage Swing Domino Logic Circuits”, Indian Journal of Computer Science and Engineering Vol 1 No 2, 74-81, 2011.

[55] Vojin G. Oklobdziza and Robert K. Montoye “Design Performance Trad-Offs in CMOS Domino Logic” IEEE Journal solid state circuiit VOL Sc12 No-2 April, 1986.

[56] Rajeev Kumar, Maneesh Kumar Singh, Vimal Kant Pandey, “Performance of low power Domino Circuits using pseudo dynamic buffer”, IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 4, Issue 6, Ver. I (Nov - Dec. 2014).

[57] Akurati, S. K., Angeline, A. A., & Bhaaskaran, V. S. K. (2017). ALU design using Pseudo Dynamic Buffer based domino logic. 2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2).