

EEE 4762 PROJECT

COMPARATIVE STUDY OF D FLIP FLOP USING DIFFERENT TYPE OF LOGIC STYLES

Atiq Aziz Sadat 190021105

Nayeb Hasin 190021115

Mohammad Aman Ullah 190021304

Mainul Islam 190021312

TABLE OF CONTENTS

Theory:	3
D flip flop simulation Workflow:	5
D flip flop using basic gates	6
Circuit simulation in DSCH software:	6
Simulation in Microwind:	7
Verilog code:	8
Pass Transistor logic D flipflop:	8
Theory:	8
D flip-flop Implementation Using Pass Transistor Logic	10
Simulation in Microwind:	11
Verilog code:	13
Transmission gate logic D flipflop:	13
Theory:	13
D flip-flop Implementation Using Transmission Gate Logic	15
Simulation in Microwind:	16
Verilog code:	17
Logic method Comparison:	18
Software Limitations:	18

THEORY:

A flip-flop is an electronic component designed to store binary information. Specifically, a D flip-flop retains 2 bits of information at its outputs, Q and Q', where Q is the inverse of Q'.

Q and Q' always exhibit complementary logic states, meaning if Q is 1, then Q' is 0. The D flip-flop features a single input. When the input is at a LOW logic level (0), it sets Q to 0. Conversely, when the input is at a HIGH logic level (1), it sets Q to 1. If a 0 is applied at the input, it effectively resets or clears Q to 0, while a 1 at the input sets Q to 1.

In essence, a D flip-flop is essentially an SR (Set-Reset) flip-flop with an additional NOT gate in front of it. This NOT gate prevents the occurrence of both the hold and indeterminate conditions inherent in the SR flip-flop. The indeterminate condition is particularly problematic for the SR flip-flop as it can lead to unpredictable outcomes, making it crucial to avoid. To create a D flip-flop using NAND gates, we leverage their properties to construct a reliable circuit.

There are two kinds of D flip-flop circuit one of them is asynchronous, or non-clocked, D flip-flop. This type of flip-flop operates without a clock cycle, meaning it doesn't follow a scheduled timing sequence dictated by a clock signal. Instead, it immediately processes an instruction whenever there is data present on the data line.

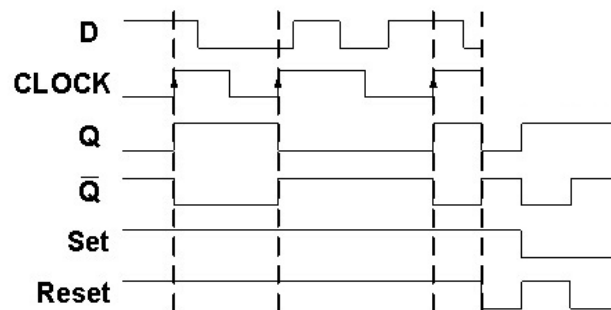
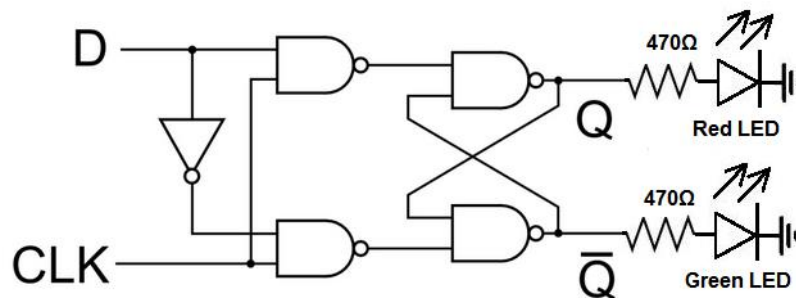
While this non-clocked D flip-flop suffices for straightforward applications, more advanced scenarios may benefit from a synchronous, or clocked, D flip-flop circuit. In synchronous designs, the input to the D flip-flop aligns with a clock signal, allowing for more precise control and coordination.

For the sake of a basic demonstration, we'll illustrate clocked D flip-flop circuit below.

D Flip Flop Logic Table		
Input	Q	Q
0	0	1
1	1	0

D Flip Flop from NAND Gates (Clocked)

In this illustration, we present a synchronous, or clocked, D flip-flop. It shares similarities with the asynchronous D flip-flop, with the key distinction that it relies on a clock signal. The input of the D flip-flop operates in coordination with the clock signal. When the clock signal is in a HIGH state, the data input is capable of being registered. Conversely, when the clock signal is in a LOW state, the data line is inaccessible, preventing any entry of input. The clock signal serves as a timing mechanism, determining the periods during which the flip-flop is responsive to changes in the data input.



D FLIP FLOP SIMULATION WORKFLOW:

Initially, we simulated the D flip-flop circuit using DSCH software, which primarily generates an LTSpice model. We obtained the corresponding timing diagram. Subsequently, we translated the circuit into Verilog code and integrated it into Quartus, producing another timing diagram. Additionally, we implemented the same Verilog code in Microwind, generating yet another timing diagram. In parallel, a D flip-flop circuit was designed using pass transistor and transmission gate logic, simulated in DSCH software, and then translated into Verilog. The Verilog code was employed in Microwind to generate the timing diagram.

Software Descriptions:

- **DSCH:** DSCH is a design entry tool used for digital and analog circuit simulation. It generates LTSpice models for simulation purposes.
- **Quartus:** Quartus is a tool often used for integrating and simulating digital designs, particularly in Verilog or VHDL.
- **Verilog:** Verilog is a hardware description language (HDL) used for modeling electronic systems. It is commonly employed in digital circuit design and simulation.
- **Microwind:** Microwind is a microelectronics design tool that includes a variety of features for digital and analog circuit design, simulation, and layout. It supports the implementation and simulation of Verilog code.

D FLIP FLOP USING BASIC GATES

Circuit simulation in DSCH software:

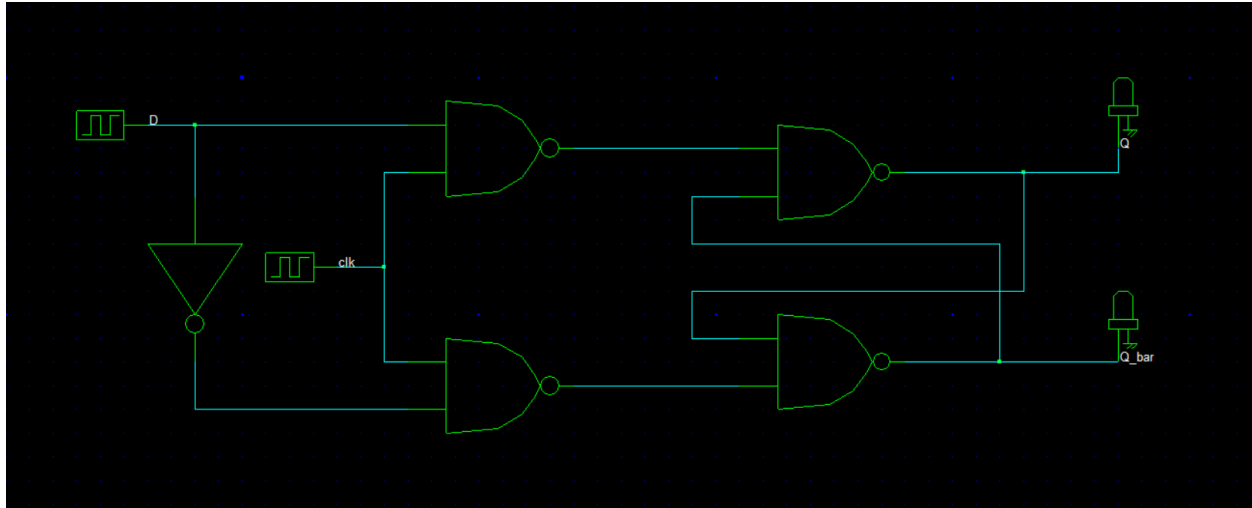


Figure 1. D flip flop(NAND) circuit in DSCH software

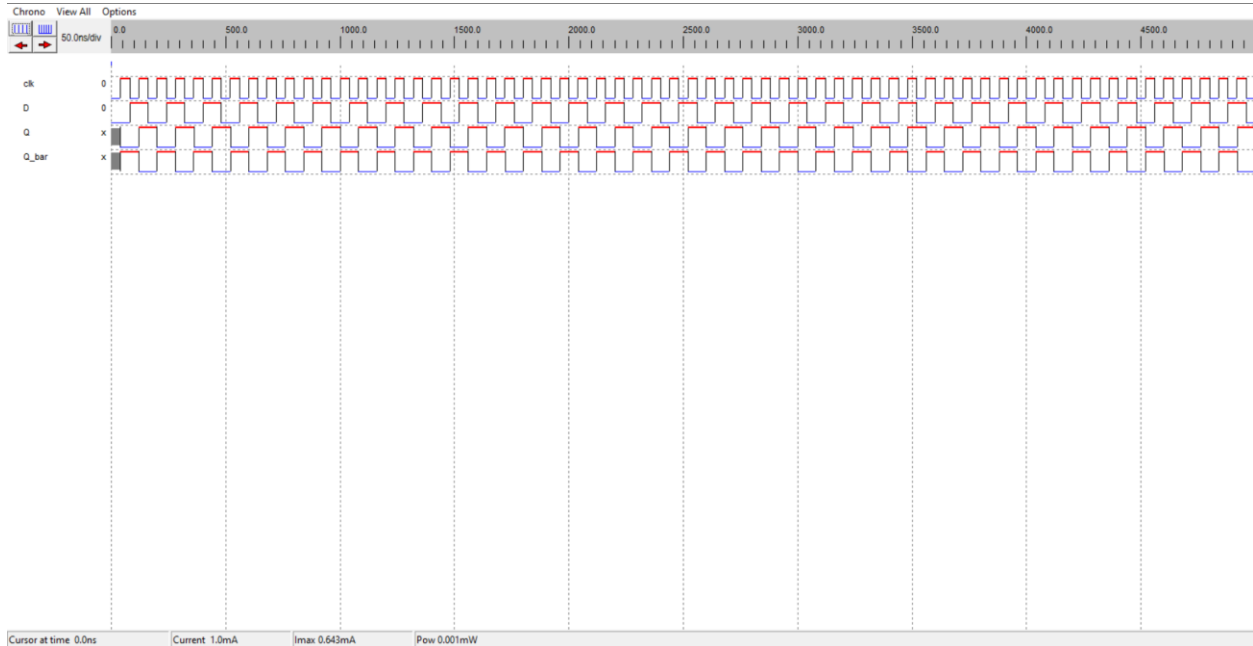


Figure 2. D Flip flop timing diagram in DSCH

Simulation in Microwind:

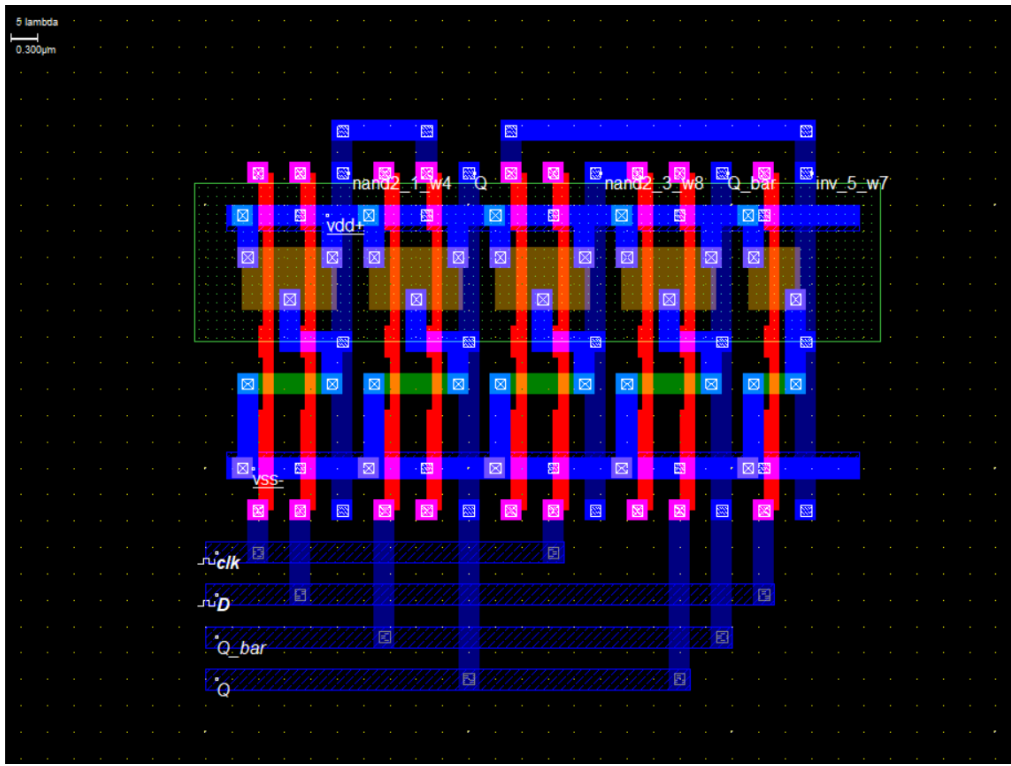


Figure 3. D flip flop layout in Microwind

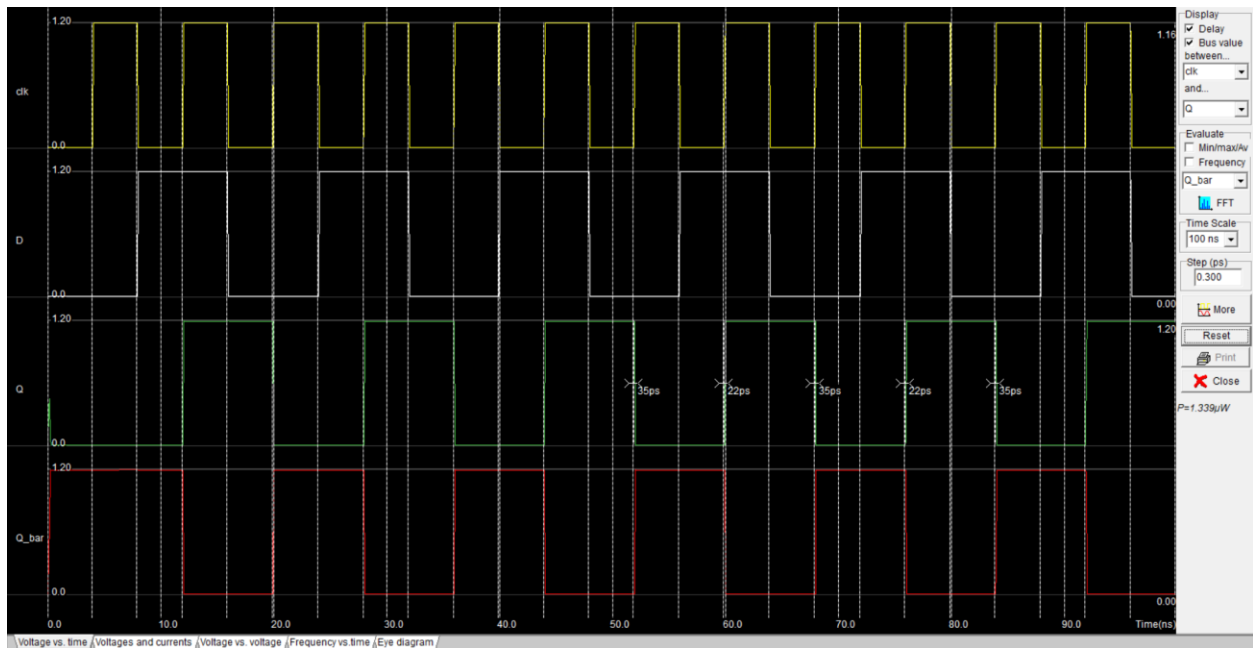
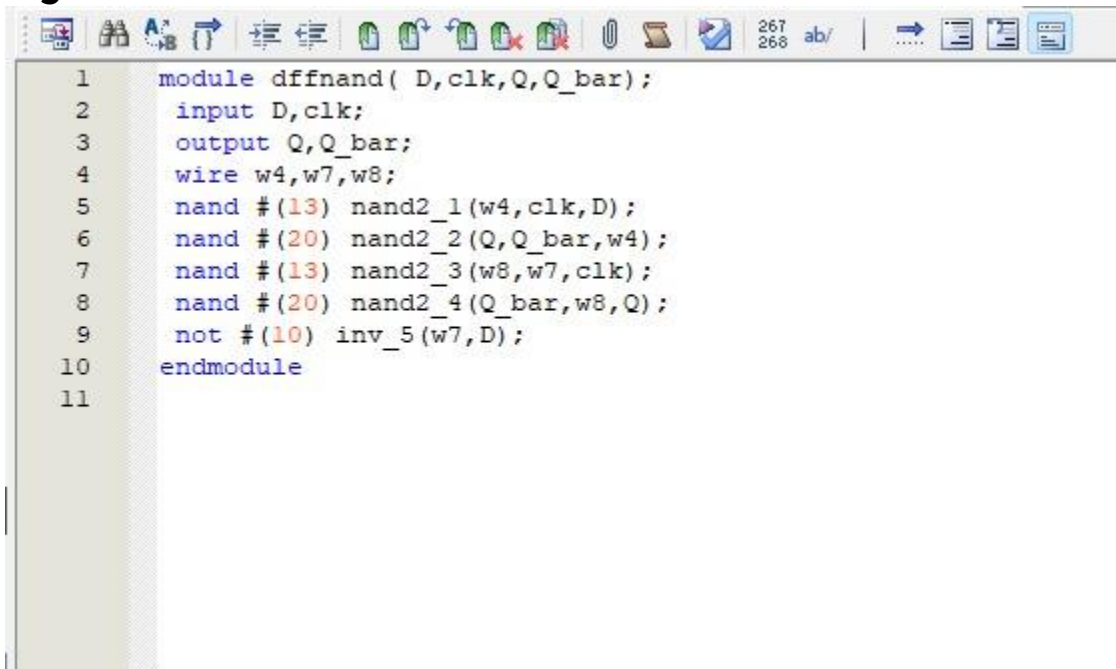


Figure 4. D Flip flop Microwind timing diagram

Verilog code:

A screenshot of a Verilog code editor window. The code defines a module named 'dffnand' with inputs 'D' and 'clk', and outputs 'Q' and 'Q_bar'. It uses four 2-input NAND gates and one 1-input NOT gate to implement a D flip-flop. The code is as follows:

```
1 module dffnand( D,clk,Q,Q_bar);
2   input D,clk;
3   output Q,Q_bar;
4   wire w4,w7,w8;
5   nand #(13) nand2_1(w4,clk,D);
6   nand #(20) nand2_2(Q,Q_bar,w4);
7   nand #(13) nand2_3(w8,w7,clk);
8   nand #(20) nand2_4(Q_bar,w8,Q);
9   not #(10) inv_5(w7,D);
10  endmodule
11
```

Figure 55. D flip flop Verilog code

Timing diagram in Quartus:

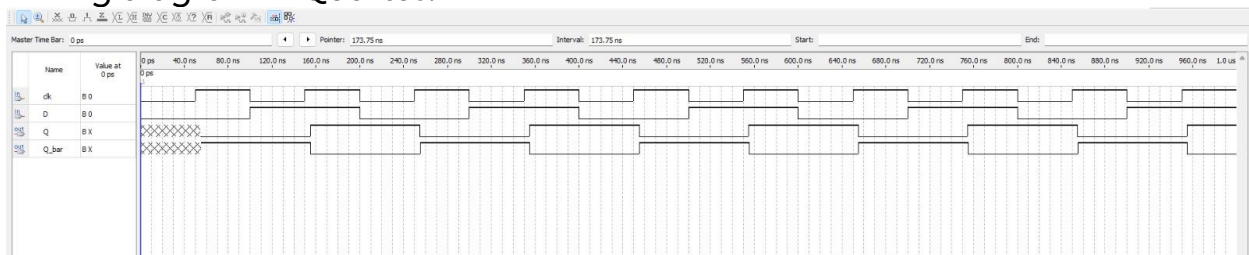


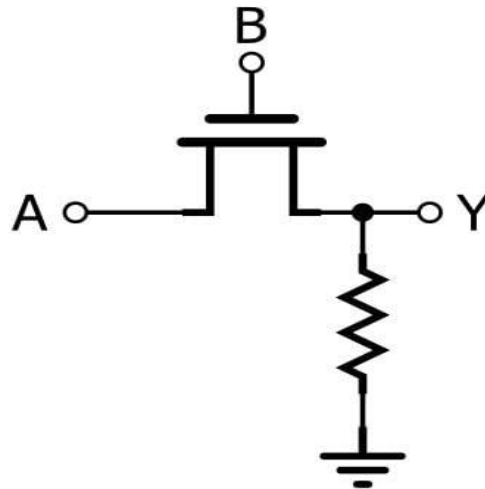
Figure 66. D Flip flop verilog timing diagram

PASS TRANSISTOR LOGIC D FLIPFLOP:

Theory:

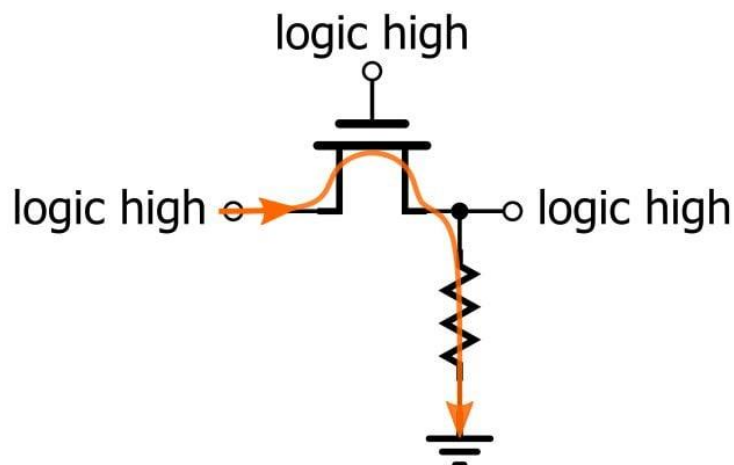
Pass transistors are used in electronic circuits to transfer signals. Commonly used in digital and analogue circuits. A pass transistor switches current between its source and drain terminals. Main pass transistors are N-type (NMOS) and P-type. Circuit requirements determine whether to use N- or P-type pass transistors. A basic pass transistor configuration switches the signal path. When "on" the transistor lets current through; when "off" it blocks it. Gate terminal voltage controls transistor on/off behavior. Pass transistors are used in

multiplexer circuits, transmission gates, and other signal routing and switching applications. They reduce power consumption and speed up certain circuits. One NMOS transistor can be used as a pass transistor logic switch. When the gate voltage is logic high, the switch is closed; when it is logic low, it is open. A single-transistor AND gate is shown in the diagram below.



A logic high input and a logic high switch-control signal (B) make the output (Y) logic high. Otherwise, it is not. That sounds like the AND truth table, but is it an AND gate? Your viewpoint matters. The circuit fails to drive a logic low when B is logic low. Simply disconnected, it floats. To establish a logic low, we need a pull-down resistor.

A standard CMOS-inverter-based AND gate requires six transistors, but we used only one transistor and one resistor to make a functional AND gate. However, the PTL circuit is not CMOS-compatible. First, it does not always provide a low-



resistance path to ground. Second, when the output is logic high, current flows from the input, NMOS, pull-down resistor, and ground, dissipating static power:

We lost an important benefit of inverter-based logic, namely that the power supply only delivers significant current during switching. (CMOS power dissipation is proportional to frequency because switching increases current and power.

D flip-flop Implementation Using Pass Transistor Logic

We have implemented pass transistor logic for building a D flip-flop in dsch software. This D FF is made up of a total of fourteen transistors. Pass-transistors and inverters make up the architecture of the D FF design. This architecture incorporates not one but two memory loop circuits. A D FF cell functions as a memory storage unit. One master memory cell is located on the left side of this structure, and one slave memory cell is located on the right side.

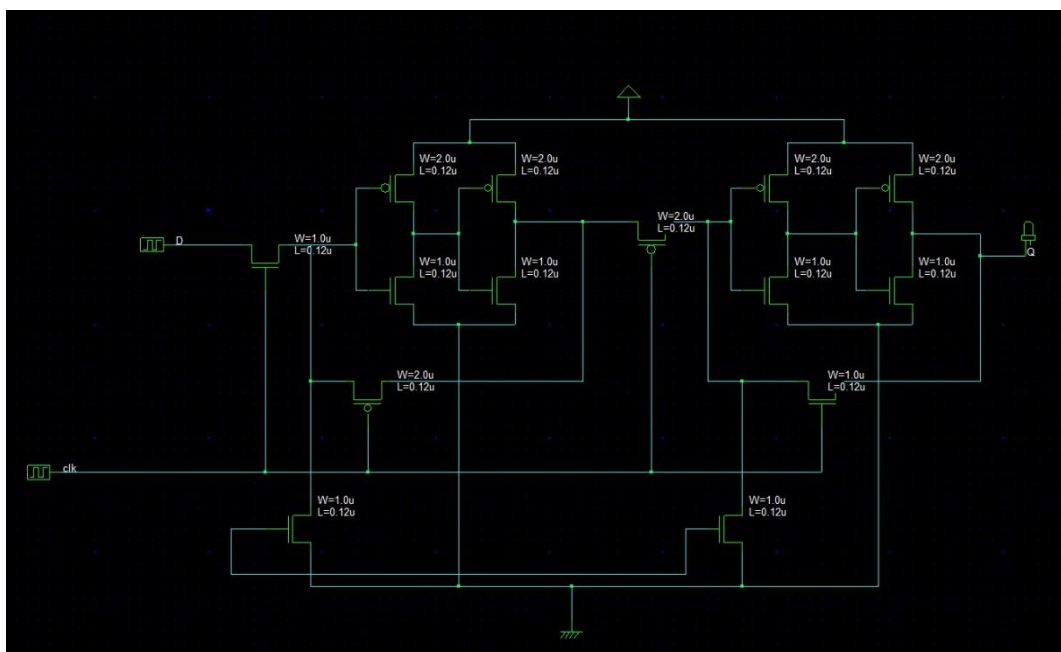


Fig. D flip-flop using pass transistor logic in DSCH software.

The value that is given near input D is updated whenever the clock is in the high state, as shown in the figure; this value is given near the master latch. Because

the master latch is operating properly, the slave latch can remember the previous state of the input D and generate the same result as the output Q. Now, if the clock is in a low state, then the slave circuit is updated with the new output, and in the meantime, the master latch maintains the value it was previously set to. The transition from one to zero (or from high to low) on the clock is the active edge of the clock. Therefore, it is a flip-flop with a negative edge triggered it.

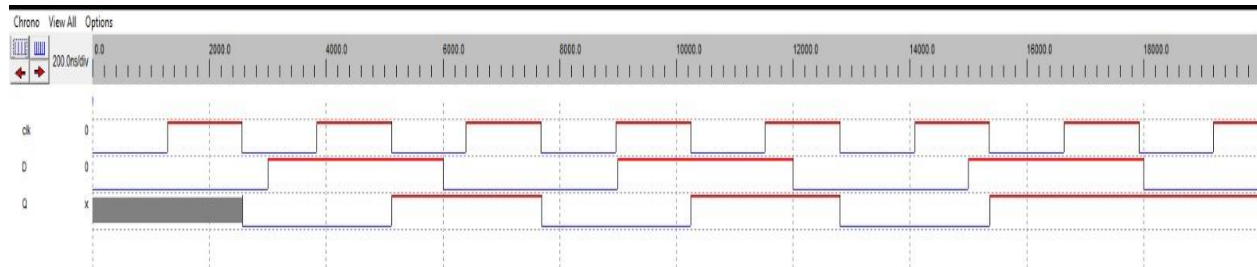


Fig. D flip-flop using pass transistor timing diagram in DSCH software.

Simulation in Microwind:

Now we will implement the D flip-flop transistor level design in microwind and observe the timing diagram.

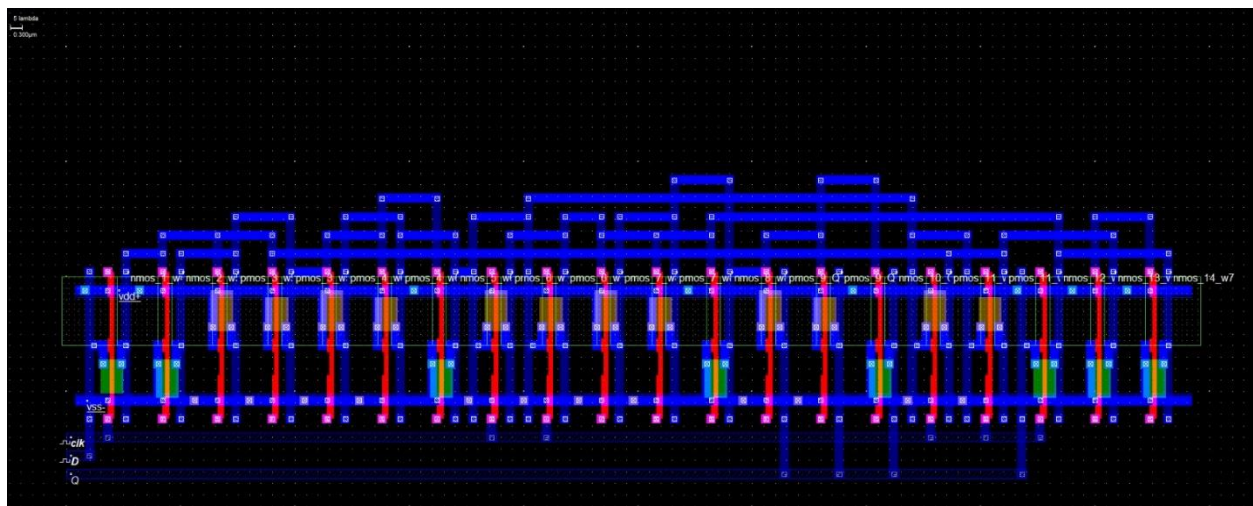


Fig. D flip-flop using pass transistor logic in Microwind software.

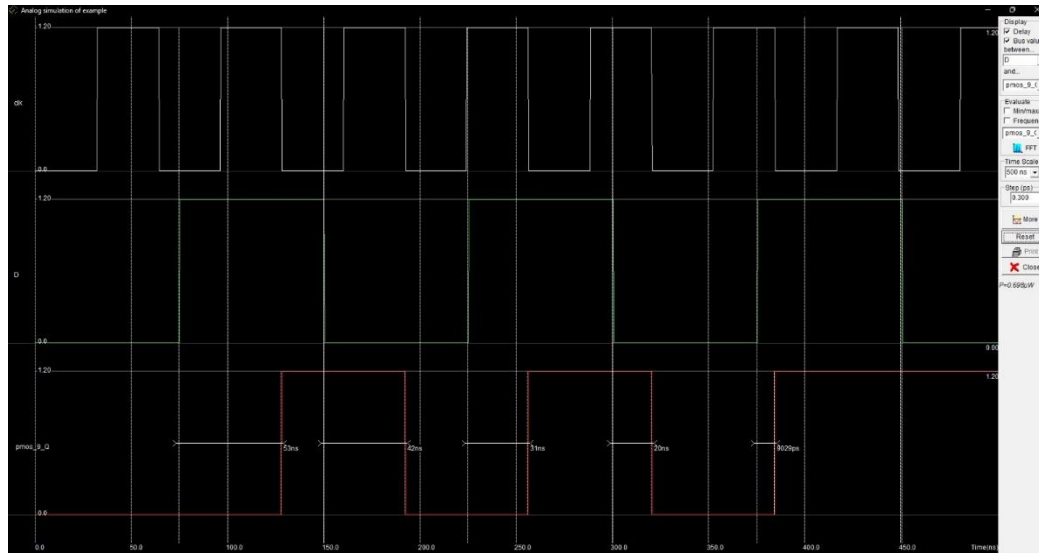


Fig. D flip-flop using pass transistor logic timing diagram in Microwind software.

From the timing diagram here we can see that pass transistor implementation of D flip-flop has reduced the propagation delay. The maximum delay we see in between some pulses is 29ps. But in most cases there is no delay. Also in this implementation the number of transistor is 14 which is lower than the other two implementations. The other two implementations using NAND and NOT gate also transmission gate and NOT gate requires 18 transistors. So, we were able to reduce the number of transistors which will surely reduce the cost of the flip-flop.

Verilog code:

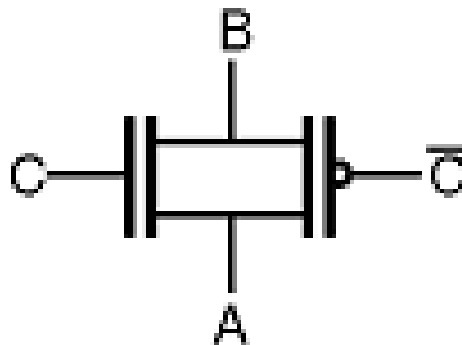
```
1 module dpass2( D,clk,Q);
2   input D,clk;
3   output Q;
4   wire w4,w5,w6,w7,w8,w10;
5   nmos #(1) nmos_1(w4,D,clk); // 1.0u 0.12u
6   nmos #(1) nmos_2(w5,vss,w4); // 1.0u 0.12u
7   pmos #(1) pmos_3(w5,vdd,w4); // 2.0u 0.12u
8   pmos #(1) pmos_4(w6,vdd,w5); // 2.0u 0.12u
9   nmos #(1) nmos_5(w6,vss,w5); // 1.0u 0.12u
10  pmos #(1) pmos_6(w7,w6,clk); // 2.0u 0.12u
11  pmos #(1) pmos_7(w8,vdd,w7); // 2.0u 0.12u
12  nmos #(1) nmos_8(w8,vss,w7); // 1.0u 0.12u
13  pmos #(1) pmos_9(Q,vdd,w8); // 2.0u 0.12u
14  nmos #(1) nmos_10(Q,vss,w8); // 1.0u 0.12u
15  pmos #(1) pmos_11(w4,w6,clk); // 2.0u 0.12u
16  nmos #(1) nmos_12(w7,Q,clk); // 1.0u 0.12u
17  nmos #(1) nmos_13(w4,vss,w10); // 1.0u 0.12u
18  nmos #(1) nmos_14(w7,vss,w10); // 1.0u 0.12u
19 endmodule
20
```

Fig. D flip-flop using pass transistor logic Verilog code.

TRANSMISSION GATE LOGIC D FLIPFLOP:

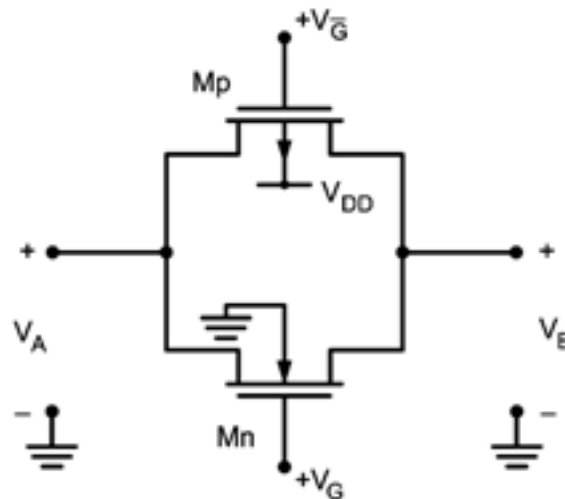
Theory:

The voltage drop issue with the pass transistor logic is resolved by using the transmission gate logic. The complimentary characteristics of PMOS and NMOS transistors are used in this method. For example, PMOS transistors pass a strong "1" but a weak "0," and NMOS devices pass a strong "0" but a weak "1." As seen in the figure below, the transmission gate combines the best features of the two



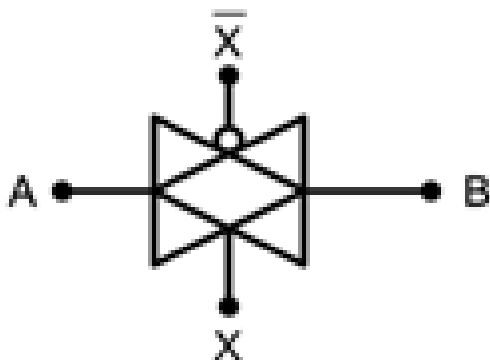
devices by connecting an NMOS and a PMOS transistor in parallel. The control signals directed towards the transmission gates C and \bar{C} exhibit mutual complementarity. The 'C' gate signal essentially acts as a bi-directional switch to activate the transmission gate. $A = B$ if $C = 1$, meaning that both MOSFETs are ON and the signal passes through the gate when $C = 1$. The MOSFETs switch off at $C = 0$, however, leaving an open circuit between nodes A and B. In this section CMOS logic circuits that are based on transmission gate are implemented. This indicates the use of transmission gate to implement logic circuits.

The basic structure of transmission gate is shown in Figure below which consists of NMOS and PMOS transistors. Here, V_G is applied to NMOS, and $(V_{DD} - V_G)$



applied to the PMOS.

The voltage-controlled switch used in transmission gates operates. Switch is closed when V_G is high because NMOS and PMOS are conducting. Consequently, there is a conduction line connecting the left and right sides. The MOSFETs are in cutoff and the switch is open when the voltage is low. As a result,



X	A	B
1	0	0
1	1	1
0	0	?
0	1	?

VA and VB do not directly relate to one another. The transmission gate symbol in the figure below is driven by the switching signals X and X*, which are applied to the NMOS and PMOS gates, respectively.

D flip-flop Implementation Using Transmission Gate Logic

In the implementation of D flip-flop using transmission gate logic we used a combination of negative level-sensitive latch and positive level-sensitive latch that giving an edge-sensitive device. Data is changed only at the active edge of the clock. This means we have used positive edge triggering here.

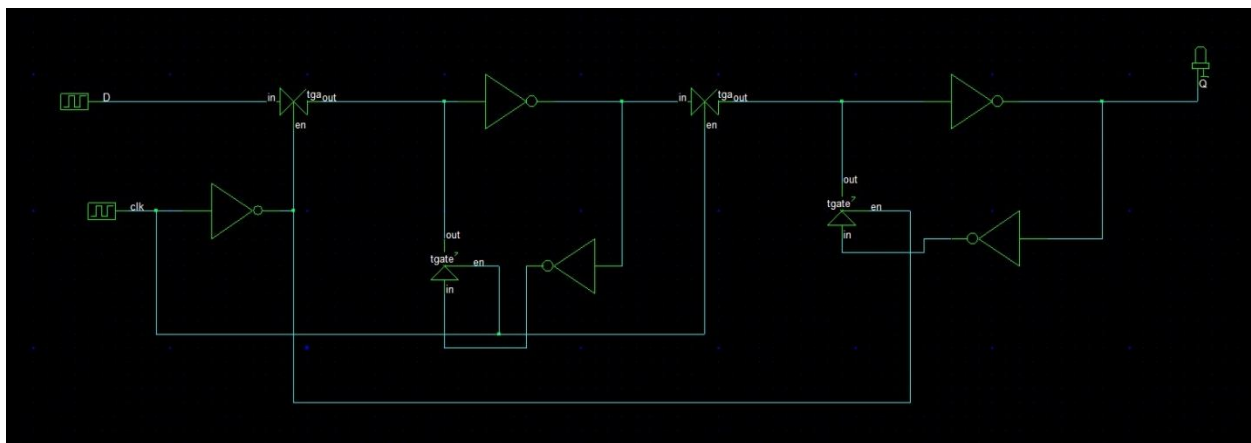


Fig. D flip-flop using pass transistor logic in DSCH software.

When **Clk= LOW (0)** T1, T4 is ON and T2, T3 is OFF. New data (D) is continuously entering through T1 and getting stored till the edge of T2, it cannot pass through T2 and T3 transmission gate because they are off. This operation is for the master latch. For slave latch it keeps retaining the previously stored value of output (Q).

When **Clk= HIGH (1)** T2, T3 are ON and T1, T4 are OFF. Now master latch did not allow new data to enter into the device because T1 is OFF and the previously stored data at point 4 is going through the path to Q and this same data is reflected at the output and this does not change until the next rising edge and this same data is also going to the transmission gate T4.

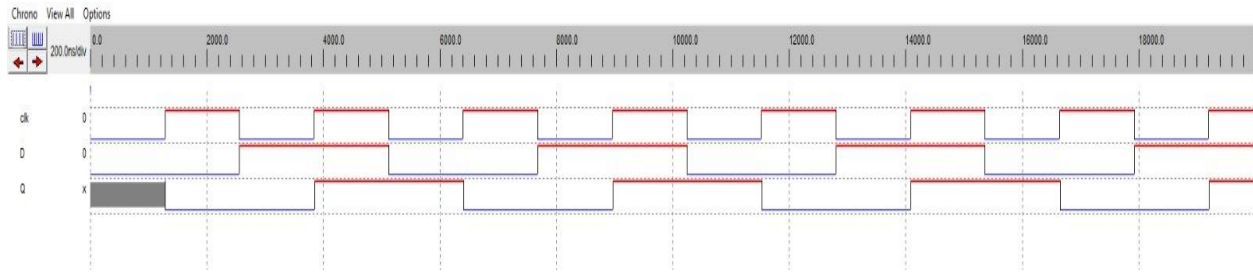


Fig. D flip-flop using transmission gate timing diagram in DSCH software.

Simulation in Microwind:

Now we will show the transistor level design of D flip-flop using the transmission gates in Microwind.

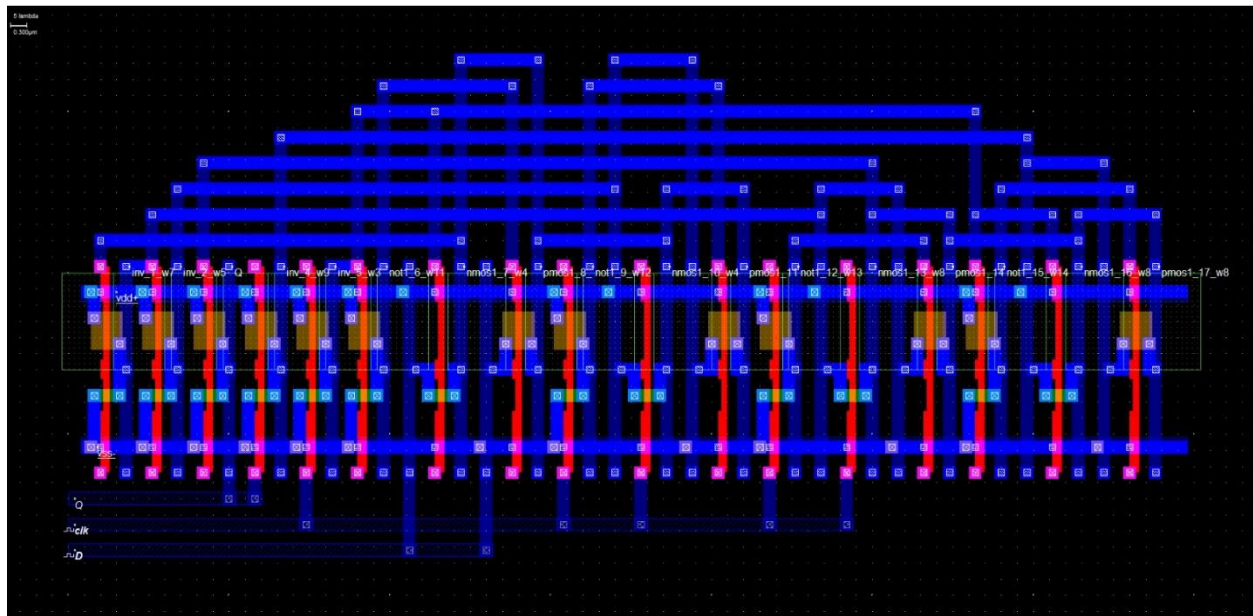


Fig. D flip-flop using transmission gate in Microwind software.

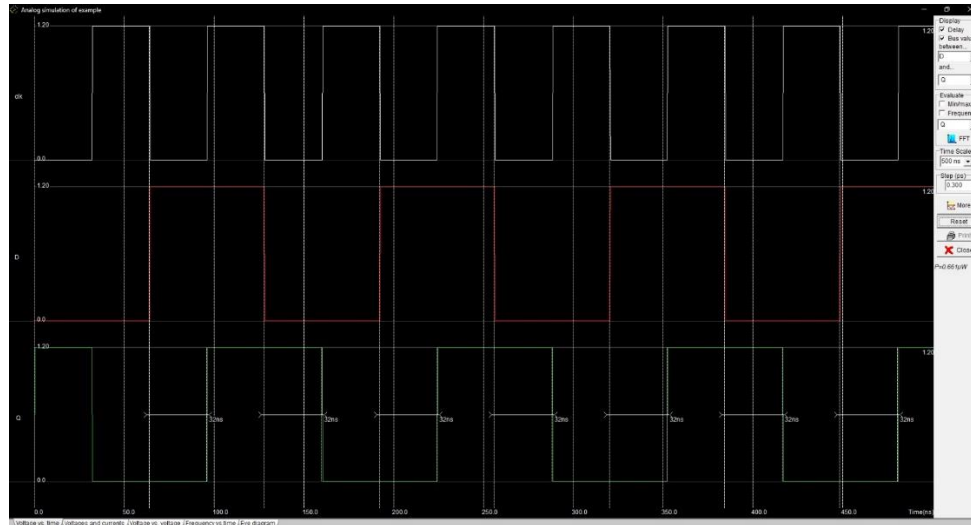


Fig. D flip-flop using transmission gate timing diagram in Microwind software.

From both timing diagram in both softwares we can see that the transmission gate implementation of D flip-flop has improved propagation delay than normal NAND gates. But the total transistor number remains the same.

Verilog code:

```

1  module dtrans( D,clk,Q);
2      input D,clk;
3      output Q;
4      wire w3,w4,w5,w7,w8,w9,w11,w12;
5      wire w13,w14;
6      not #(1) inv_1(w7,w4);
7      not #(1) inv_2(w5,w7);
8      not #(1) inv_3(Q,w8);
9      not #(1) inv_4(w9,Q);
10     not #(1) inv_5(w3,clk);
11     not #(1) not1_6(w11,w3);
12     nmos #(1) nmos1_7(w4,D,w3);
13     pmos #(1) pmos1_8(w4,D,w11);
14     not #(1) not1_9(w12,clk);
15     nmos #(1) nmos1_10(w4,w5,clk);
16     pmos #(1) pmos1_11(w4,w5,w12);
17     not #(1) not1_12(w13,clk);
18     nmos #(1) nmos1_13(w8,w7,clk);
19     pmos #(1) pmos1_14(w8,w7,w13);
20     not #(1) not1_15(w14,w3);
21     nmos #(1) nmos1_16(w8,w9,w3);
22     pmos #(1) pmos1_17(w8,w9,w14);
23 endmodule
24

```

Fig. D flip-flop using transmission gate Verilog code.

LOGIC METHOD COMPARISON:

	Basic gate	Pass Transistor	Transmission gate
Delay	22ps	29ps	No such delay observed
Edge triggering	Positive edge	Negative edge	Positive edge
No of transistors	18T	14T	18T
Power Consumption	High	lowest	Lower

SOFTWARE LIMITATIONS:

DSCH, QUATZ, and Microwind, offer valuable functionalities for digital circuit design and simulation, each has its limitations:

- **DSCH:**

Limited Component Library: DSCH may have a limited library of components, which can constrain the complexity and diversity of circuits that can be designed.

Analog Circuit Emphasis: It might be less suitable for intricate digital designs, as DSCH tends to be more focused on analog circuit simulation.

- **Quartus:**

Learning Curve: Quartus may have a steeper learning curve for beginners due to its comprehensive set of features, which might be overwhelming for those new to digital design.

- **Verilog:**

Steep Learning Curve: Verilog, being a hardware description language, might pose a steep learning curve for those new to digital design and programming.

Limited Abstraction: Depending on the level of abstraction, Verilog may require more manual intervention for detailed control, potentially making it less user-friendly for certain designers.

- **Microwind:**

Limited Analog Capabilities: While Microwind is powerful for digital design, its analog simulation capabilities might be more limited compared to specialized analog design tools.

Restricted Component Library: Similar to DSCH, Microwind may have limitations in terms of the variety of components available for design

