TEACHING FUNDAMENTALS OF DIGITAL LOGIC DESIGN AND VLSI DESIGN USING COMPUTATIONAL TEXTILES

Sivateja Inampudi

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

August 2014

Inampudi, Sivateja. Teaching Fundamentals of Digital Logic Design and VLSI Design Using Computational Textiles. Master of Science (Electrical Engineering), August 2014, 48 pp., 6 tables, 56 figures, 10 numbered references.

This thesis presents teaching fundamentals of digital logic design and VLSI design for freshmen and even for high school students using e-textiles. This easily grabs attention of students as it is creative and interesting. Using e-textiles to project these concepts would be easily understood by students at young age. This involves stitching electronic circuits on a fabric using basic components like LEDs, push buttons and so on. The functioning of these circuits is programmed in Lilypad Arduino. By using this method, students get exposed to basic electronic concepts at early stage which eventually develops interest towards engineering field.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude towards Dr. Gayatri Mehta for her unwavering support, excellent guidance, unrelenting patience, and for providing me with a professional atmosphere for doing research. I would like to thank Dr. Hyoung Soo Kim and Dr. Kamesh Namaduri for supporting my research and being willing to participate in my defense committee. I would also like to thank Dr. Shengli Fu for supporting me financially throughout my M.S. degree.

I would like to thank the Graduate Writing Support Center, especially Noah Geisert, for helping me with put up my ideas clearly in the writing. Many thanks to my friends: Amrutha Chalasani, Madhavi Guturu, Charu Sneha Reddy. Lastly, Wholehearted thanks to my family for supporting me spiritually throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This research deals with introducing digital electronics to freshmen or even high school students to make them much closer to electronics. Signals that represent with discrete band of analog values are defined as digital circuits or digital electronics. Importance of digital systems, ‘0’ and ‘1’s, is transmission of signals without any degradation due to noise and more accurate restoration.

This technology made easier to design digital circuits without having complex calculations of some digital components like transistors, capacitors, only. There is only need to know their operation. Digital electronics also has many advantages like reliability, cheap, high flexibility, less noise affective and easy to store even lengthier information. These advantages of digital electronics has made its own room for digital logic design as a core course in electrical engineering.

Digital logic design is a subject of designing electronic components with simple logical operations. These logical operations are technically defined as logic gates. Based its importance, There is need to teach fundamentals in an interesting way and to make them learn thoroughly with more insight. Although digital logic design is being main stream in electronics, it can be introduced to even high school students by educating its importance and concepts. In order to grab their attention, these technical concepts are to be converted into entertaining lessons. By this method, students get comfortable with these concepts ever since high school. This fruitful interest would probably help in motivating students to choose the stream of electrical engineering and attract towards research areas, thereby technology improvement. The advancement of digital logic design subject goes with VLSI (very large scale integration) design which is the process of creating integrated circuits by combining thousands of transistors in a single chip.

This research work aims to communicate both digital logic design and VLSI concepts to high school students in fun way. The process of extending digital knowledge to students includes various methods. One such method, that my work deals with, is utilizing e-textiles to teach basic concepts of digital logic design to students. E-textiles, which are also called electronic textiles, is the discipline of combining the concepts of electronics and fabrics which include electronic components like LEDs, push buttons, switches, and so forth. E-textiles has its space in various fields like fashion technology, interior designing, and even more applications. Employing e-textiles as a teaching aid is a unique way to introduce digital electronics to students.

Using e-textiles in teaching digital logic concepts involves fun and initiates creative thought in children, and is also an amicable way to present these concepts to students from other engineering fields rather than electrical engineering. Based on this idea, I have used method of stitching the circuits in textiles which includes a concept hidden on the fabric. This thesis mainly focuses on basic concepts of digital logic design and VLSI design in EE curriculum.

Lilypad Arduino is used as a media to project the concepts. Lilypad Arduino is a sewable Microcontroller that can operate digital components and can program by using Arduino software. In this thesis, I have used e-textiles for conveying such concepts as logic gates, multiplexer, decoder, counter, shift register, and ring counter, all of which include the basics of digital logic design, and for conveying concepts of MOS transistor, power gating, and clock gating in VLSI design.

1.2 Related Work

Previously, some workshops were conducted related to teaching methods like textile oriented, graphical oriented, and robotic oriented representations. All different interactive platforms are introduced where students can learn different circuit's behaviors. These different methods have made technology more approachable and accessible to students.

An interactive learning environment (ILE) is designed to combine traditional resources of a text book to hands-on design experiences [3]. It is a JAVA based computer aided design where tools can be accessed through interactive figures where students can investigate the behavior of different circuits. Based on this web based learning environment with zero administration costs, students are getting valuable insight with equal fun [3].

A series of workshops were conducted by Cambridge, MA [1] students in high schools as "a curriculum for teaching computer science through computational textiles" where they have showed the improvement in their learning capability. In these workshops, they have introduced students to new technologies like MODKIT, and Lilypad Protosnap board. They assigned students to stitch circuits and to make them work by programming with Mod kit software.

More recent works have been done by Buechley, Eisenburg in high schools [2], where these workshops also helped students acquire exposure to Lilypad Arduino tool kit with basic circuits and programming languages. Other works have been done by people kafai, Serale and Fields [3] where they explored how e-textiles can increase the legibility and visibility of electronics and programming concepts. They have also explored how technology learning can be increased when students are deeply involved in developing their own design projects

1.3 Organization of Thesis

The thesis report is organized as follows: Chapter 1 the Introduction which includes total overview of the thesis, its advantages and also other related works. Chapter 2 presents the implementation of basic concepts of digital logic design. Chapter 3 presents implementation of VLSI design fundamentals. Chapter 4 provides the conclusion and future work.

CHAPTER 2

DIGITAL LOGIC DESIGN

2.1 Logic Gates

A logic gate is an elementary building block of a digital circuit. It is a physical device implementing a Boolean function, and performs a logical operation on one or more logical inputs, and produces a single logical output. There are seven basic logic gates: AND, OR, XOR, NOT, NAND, NOR, and XNOR.

2.1.1 AND Gate

AND gate implements the logical conjunction. The logic HIGH output results only, when all the inputs are HIGH. If either or both the inputs of AND gate are LOW, then logic LOW output results. The symbol and truth table of AND gate are shown in figure 1 and table 1, respectively.

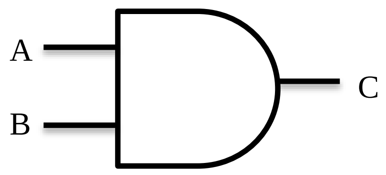


Figure 1: AND gate

Table 1: Truth table of AND gate

INPUTS		OUTPUT
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

2.1.1.1 Implementation

Implementation of a 2-input NAND gate can be attained in a simple way. Consider a road having two bridges (like switches) in between from source point to destination point. The ability of reaching the destination point is only feasible, when both the bridges are closed. If any one of the bridges is open or both the bridges are open, then no one can reach the destination. The relation of this concept to AND gate logic is described as follows.

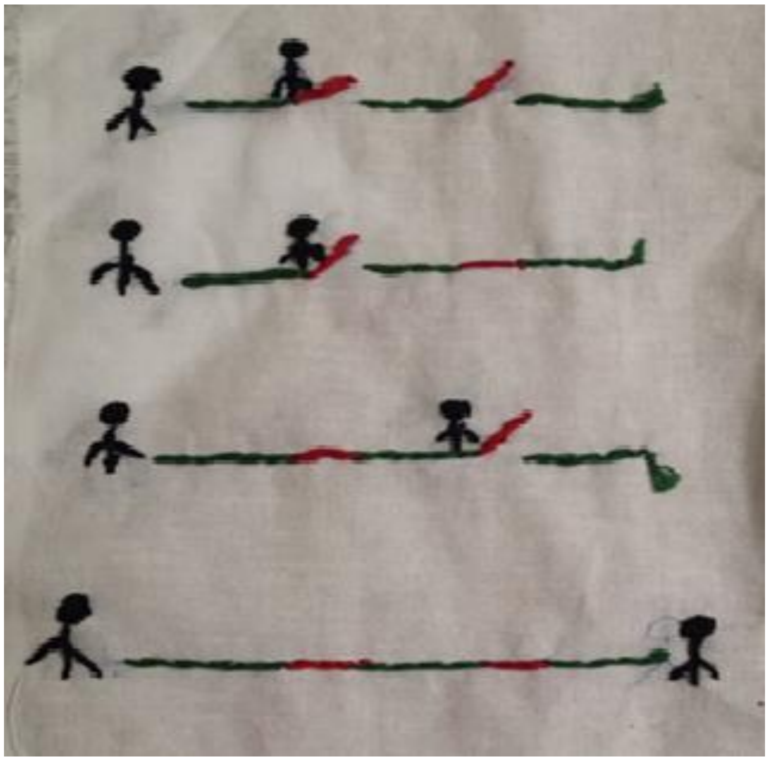


Figure 2: Representation of the AND gate

From figure 2, let the two inputs be the bridges (A and B) in the road, represented as red color. In the first path, the boy unable to reach the other side (output = 0) as both the bridges are open, (A = 0 and B = 0). In the second and third paths, even one of the bridges are open (A = 0, B = 1 and A = 1, B = 0 respectively), the boy unable to cross the road (output = 0). Finally, in the last path, there is a possibility of reaching the destination point (output = 1) as both the bridges are closed (A = 1 and B = 1).

2.1.2 OR Gate

OR gate is the logic gate that implements the logical disjunction. The logic HIGH output results when any one of the input or both the inputs are HIGH and logic LOW output results only when all the input are LOW. In another way, the function of OR effectively finds the maximum between two binary digits. The 2-input OR gate symbol and its truth table are shown in figure 3 and table 2, respectively.

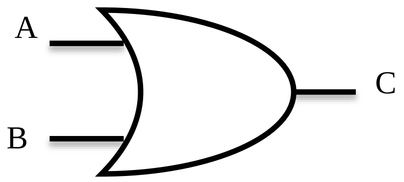


Figure 3: OR gate

Table 2: Truth Table

INPUTS		OUTPUT
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

2.1.2.1 Implementation:

The 2-input OR gate is implemented by considering two parallel roads each having a bridge with same source point and destination point. One can reach the destination point by choosing the particular road that is the one that happens to be bridge closed. When both the bridges are closed, both the routes have viability to reach the destination. But, reaching the destination is unachievable, when both the bridges are open. This concept has related to AND gate as follows.

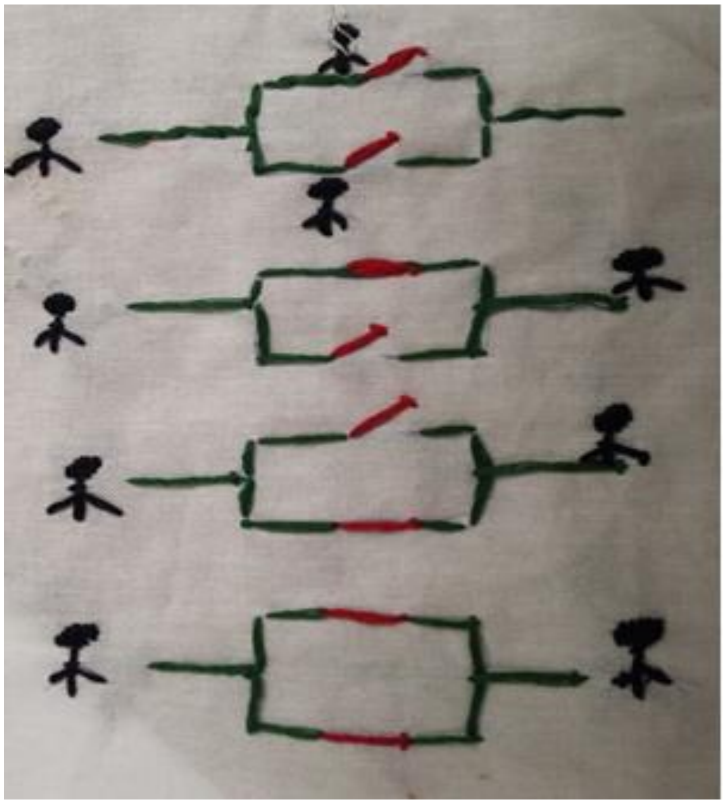


Figure 4: Representation of the OR gate

From figure 4, consider the bridges of both the parallel roads as inputs (A and B), represented in red. In the first path, when both the bridges are open, (A = 0 and B = 0), the boy is unable to reach the other end (output = 0). Whereas, in the second and third paths, when one of the bridges is closed (A = 0, B = 1 and A = 1, B = 0 respectively), then the boy can pick that particular road and manages to reach the end point (output = 1). Lastly, when both the bridges are closed (A = 1 and B = 1), he can choose any which way to reach his destination.

2.1.3 NOT Gate

NOT gate is an inverter which implements the logical negation. It is a single input and single output gate. When the input is HIGH, then the output will be LOW and vice versa. The NOT gate symbol and its truth table are shown in figure 5 and table 3, respectively.

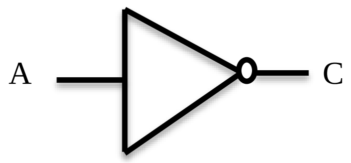


Figure 5: NOT gate

Table 3: Truth Table

INPUT A	OUTPUT C
0	1
1	0

2.1.3.1 Implementation

The NOT gate is portrayed as shown in figure by employing LEDs as input and output. The logic LOW level and logic HIGH level are represented by LED OFF and LED ON respectively. Push button is used to control the input signal.

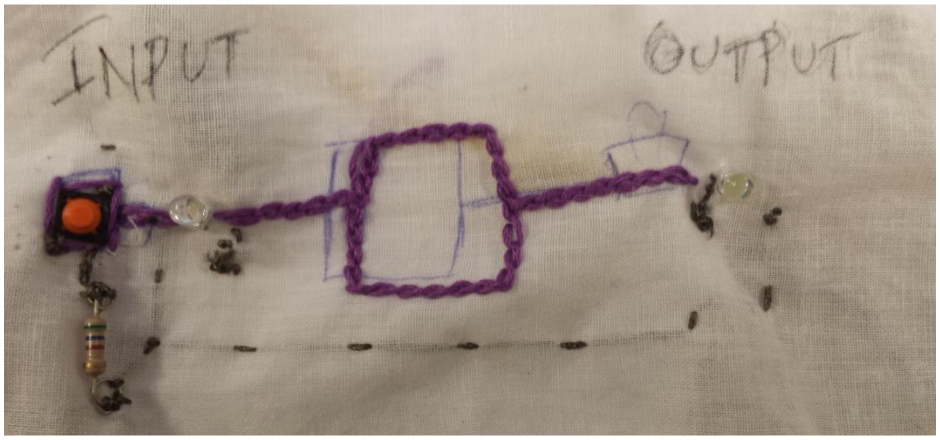


Figure 6: NOT gate representation

On operation of push button, the state change of input LED in the e-textile presentation depends on its previous state. The input changes to HIGH when it was LOW previously and changes to LOW, when it was HIGH. Finally, according to the input, the output will change. The figure 7 represents when the INPUT being HIGH (LED in ON state), output being LOW (LED in OFF state).

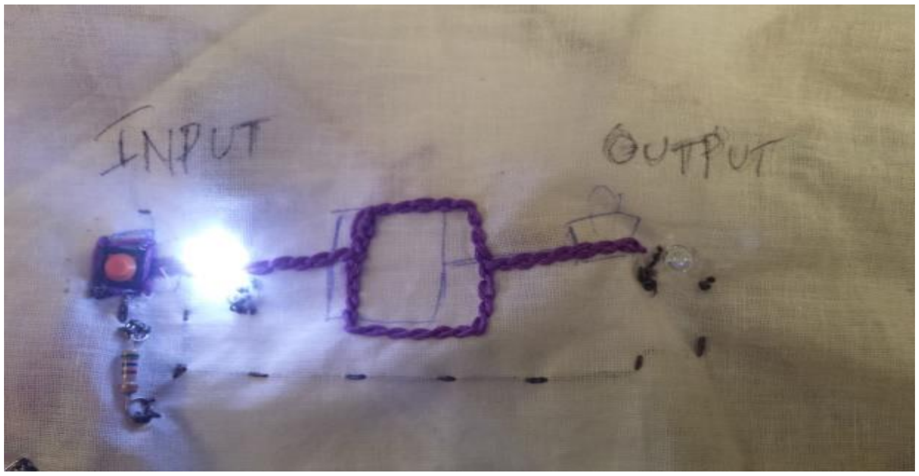


Figure 7: Represents NOT gate operation when the input is HIGH

The figure 8 represents when the INPUT is LOW (LED in OFF state), the output is HIGH (LED in OFF state).

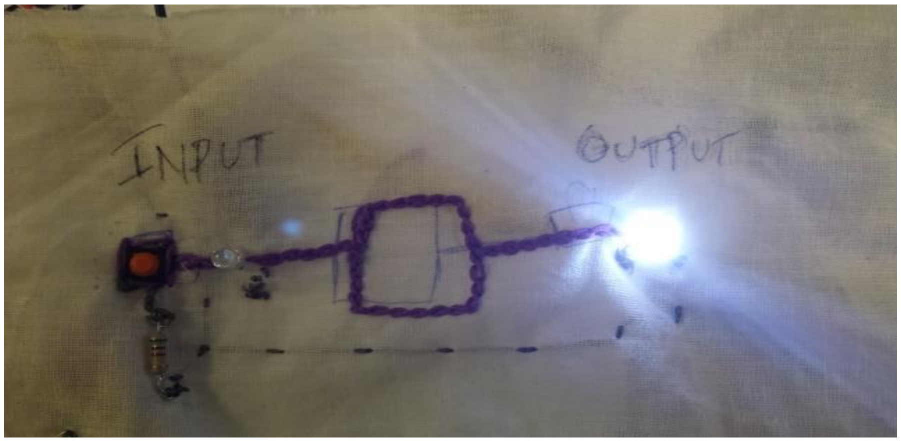


Figure 8: Represents NOT operation when the input is LOW

2.1.4 NAND Gate

The NAND gate (Negated AND or NOT AND) produces the output of logic LOW, only when all the inputs are HIGH, thereby, compliment to that of AND gate output. In 2-input NAND gate, LOW output results when both the inputs are HIGH, and HIGH output results when any one of the inputs or both the inputs are LOW.

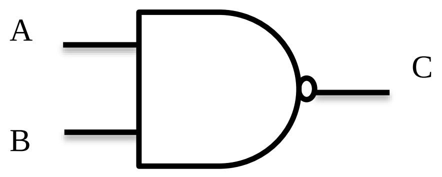


Figure 9: NAND gate

Table 4: Truth Table

INPUTS		OUTPUT
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

2.1.4.1 Implementation

Consider a professor who wants students to do their homework. The Professor gets happy even if one of the students done with their work. The probability of getting unhappy is only when no student did their work. This concept of students doing work and professor happiness can be connected with the operation of NAND gate. Let students be the inputs and professor be the output of the NAND gate. The e-textile representation of NAND gate is shown in figure 10.

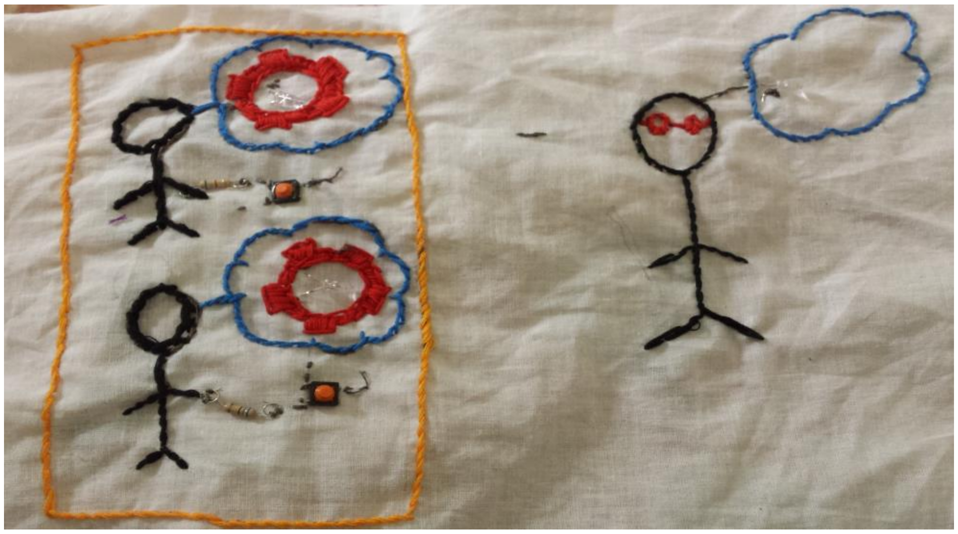


Figure 10: NAND gate representation

Let the students doing work is represented as input ‘0’ and not doing work is represented as ‘1’. Consider two students, when both the students doing work (inputs be ‘00’), the professor gets happy (output =1). If any one of the students or both the students not done with work (inputs be 01, 10 or 11), then professor gets disappointed (output = 0).From figure 11, LEDs of both the students are OFF showing that they have done their work and finally professor is happy(ON)

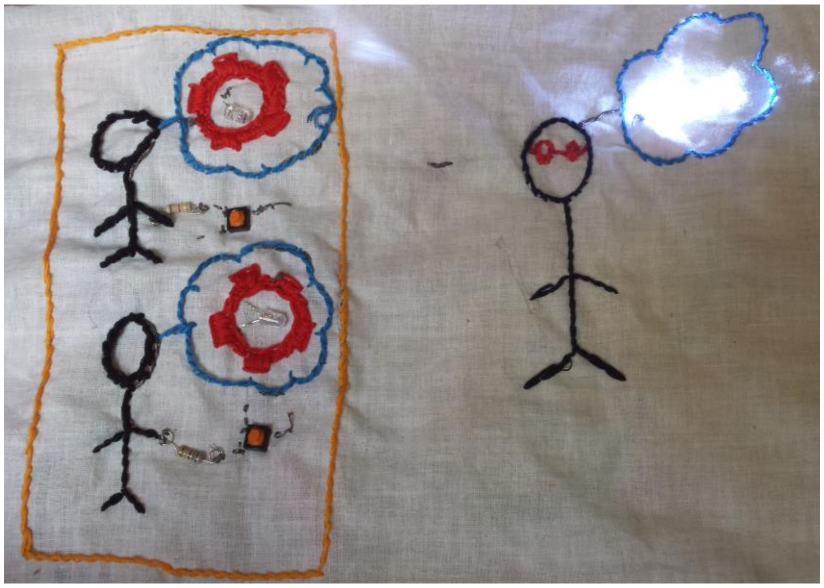


Figure 11: NAND gate when both the inputs are LOW

From respective figures 12, 13 one of the students has done the work. Even then the professor is happy.

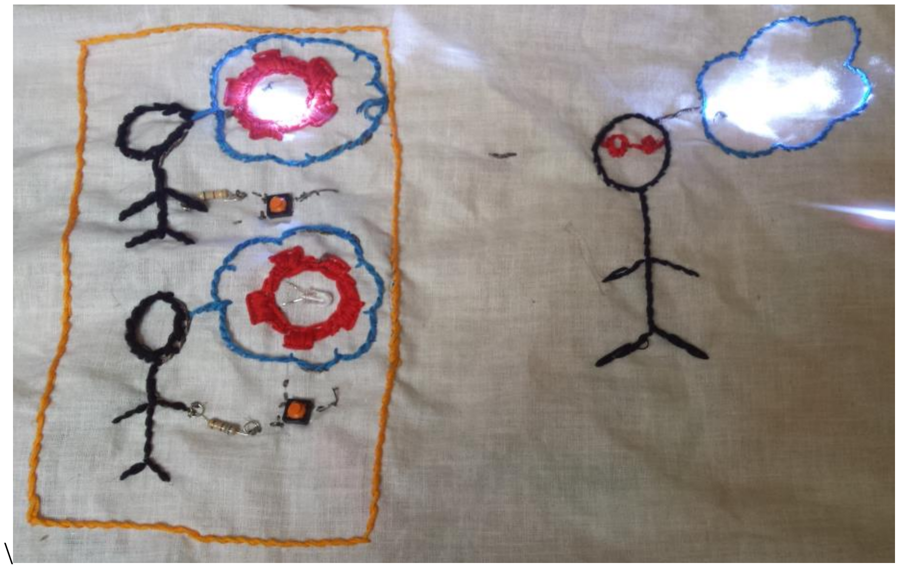

\

Figure 12: NAND gate when one of the input is LW and the other HIGH as ‘10’

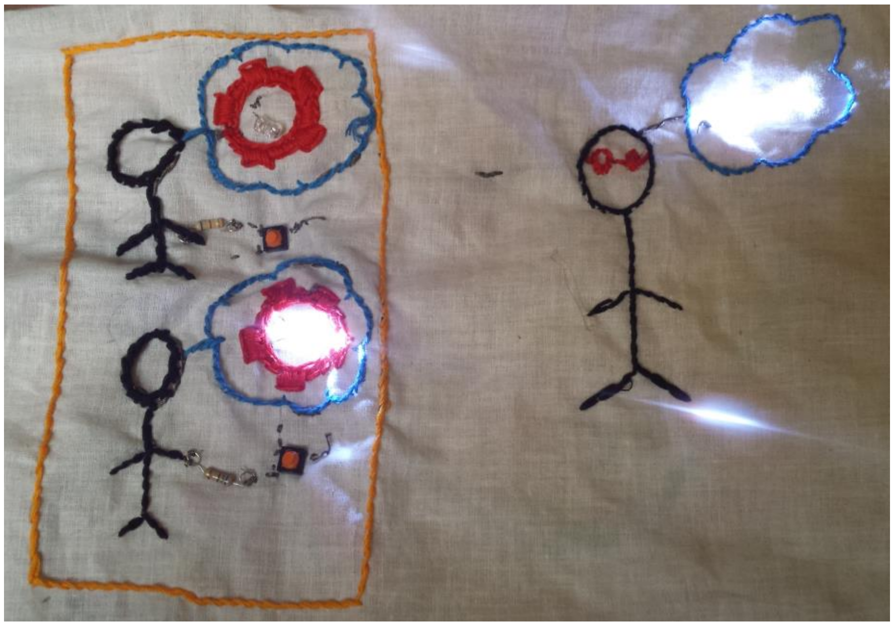


Figure 13: NAND gate when one of the input is LOW and the other HIGH as ‘01’

If both the students did not do their work (from figure 14, both the LEDs are ON), the professor gets unhappy (LED OFF).

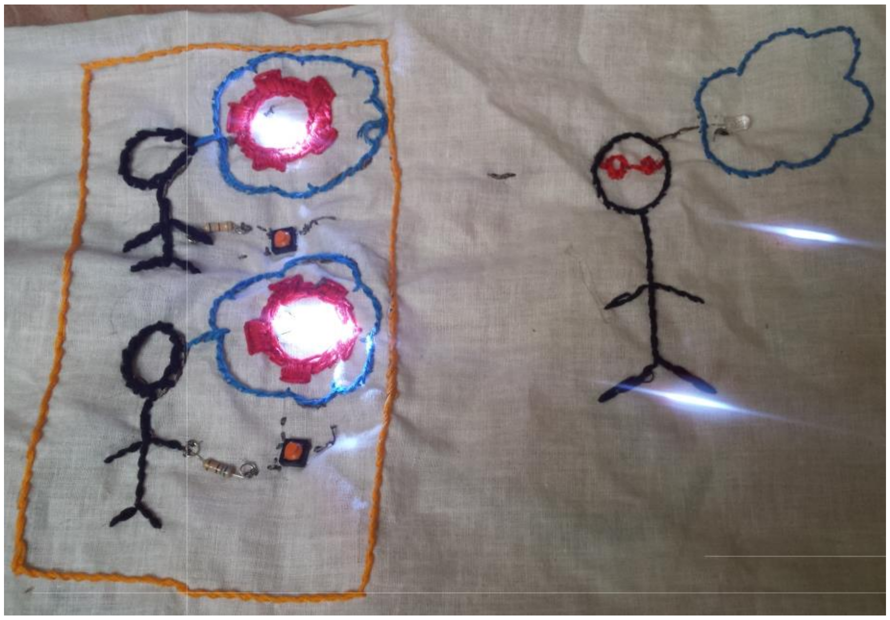


Figure 14: NAND gate when both the inputs are HIGH

2.1.5 NOR Gate

NOR gate (OR NOT) produces logic HIGH output when all the inputs are LOW. Consequently, its output compliment to that of OR gate output. In 2-input NOR gate, the output will be LOW, when either both the inputs are HIGH or any one of the inputs is HIGH.

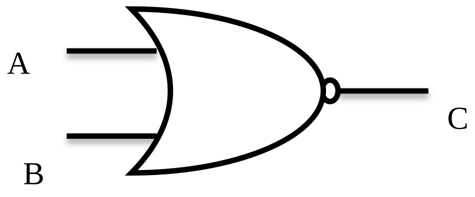


Figure 15: NOR gate

Table 5: Truth Table

INPUTS		OUTPUT
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

2.1.5.1 Implementation

The projection of 2-input NAND gate is also achieved by considering students and professors as inputs and output, but of different activity. Let’s consider the professor expects each and every student to be silent in a classroom. The professor gets unhappy even one of the students makes noise. This concept of the professor and students with NOR gate operation is related as follows. Consider input ‘0’ as student to be silent and vice versa. Similar to the NAND gate, professor gets happy, if every student is silent (output = 1). The e-textile representation of NOR gate is as shown in figure 16.

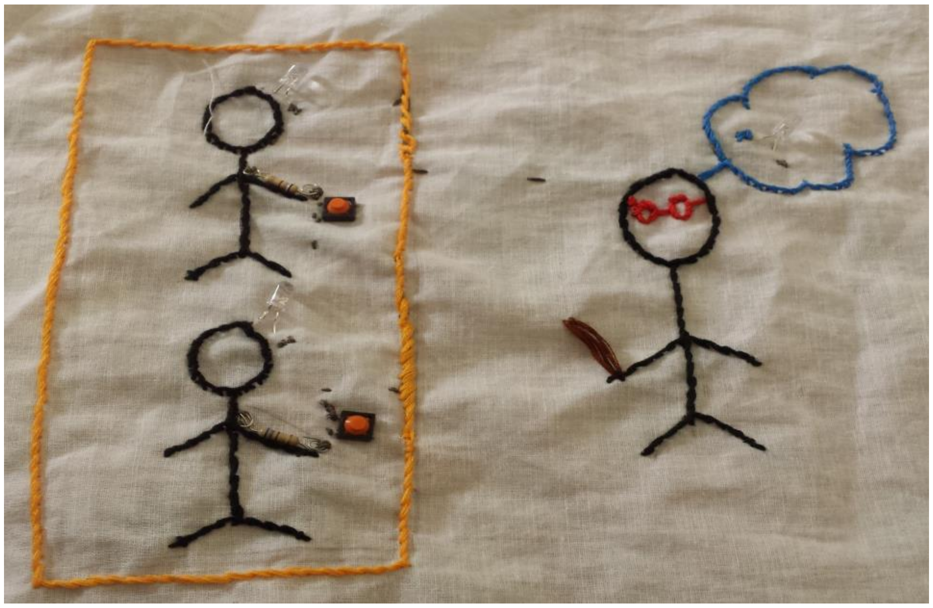


Fig 16: NOR gate representation

From figure 17, the output activity is same as of NAND gate, where LEDs ON on professor happiness. Whereas the input side, LEDs OFF on silent nature of students. This represents, when both the inputs are ‘00’, NOR gate output is ‘1’.

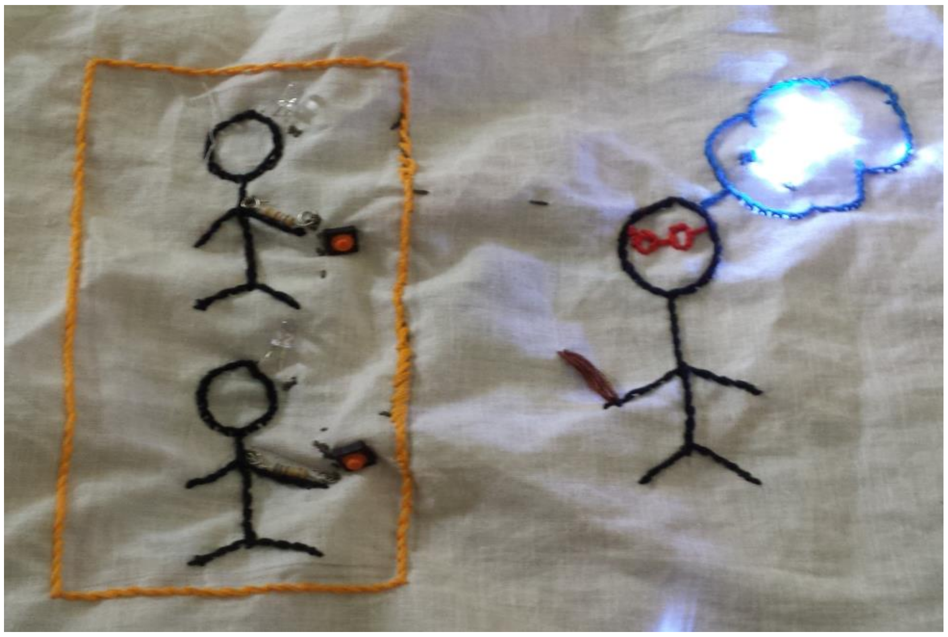


Figure 17: NOR gate with both the inputs are LOW

From figure 18 and 19, even one of the student is not silent (input = ‘01’ or ‘10’), Professor gets unhappy (output = 0).

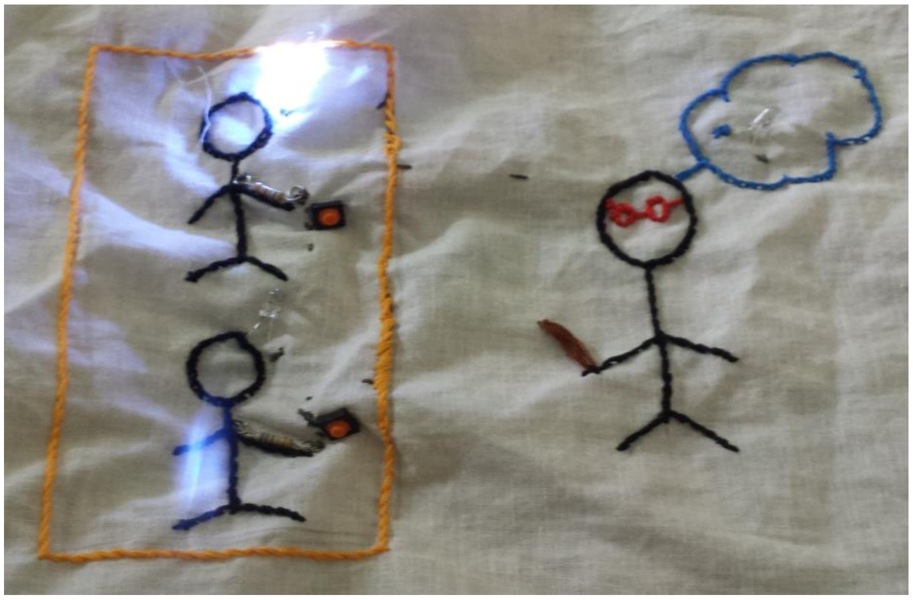


Figure 18: NOR gate with one of the input is HIGH and the other LOW as ‘10’

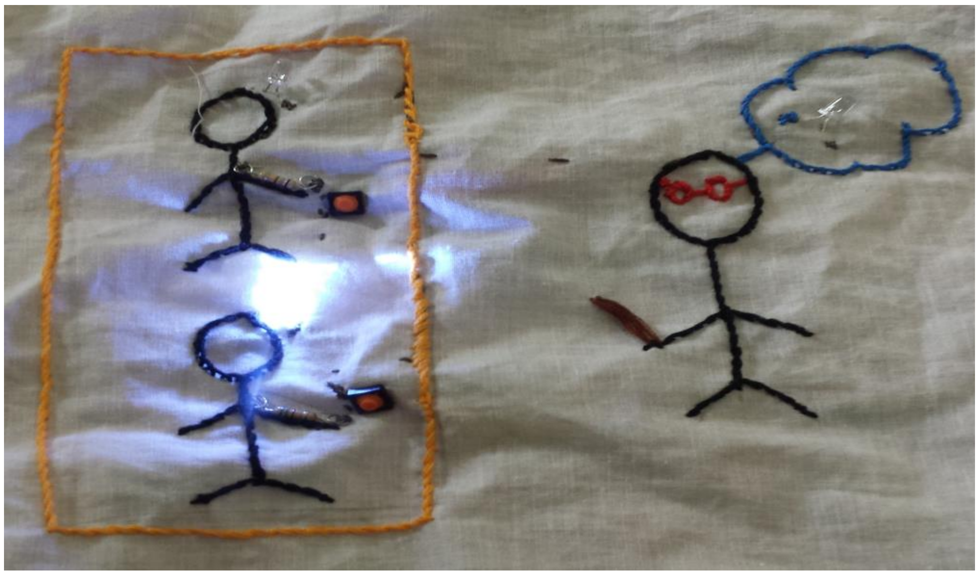


Figure 19: NOR gate with one of the inputs is LOW and the other HIGH as ‘01’

Similarly, from figure 20, even professor gets unhappy (output = 0) when both are not silent (input = ‘11’ and LEDs ON).

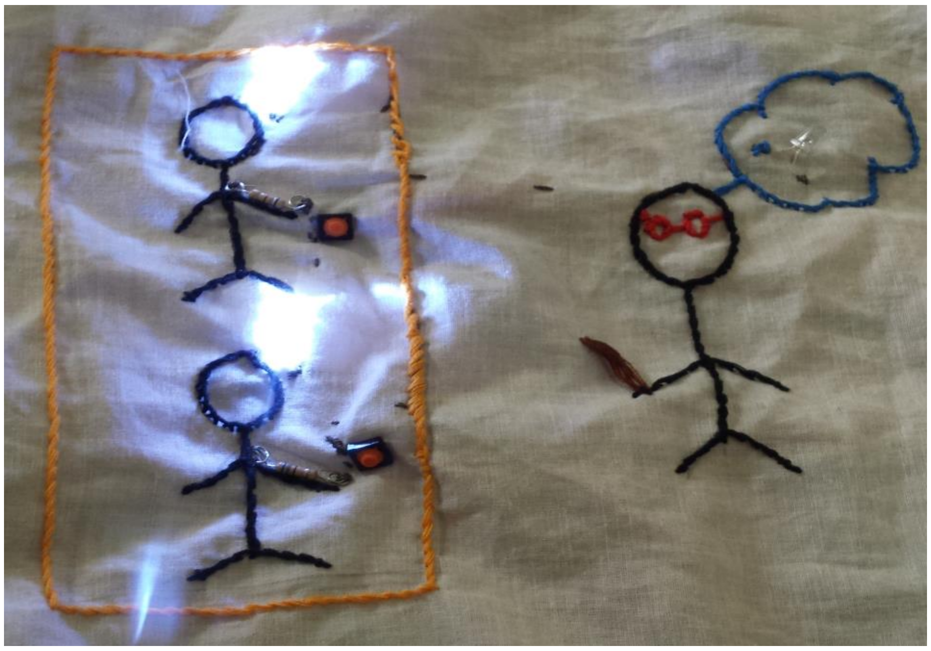


Figure 20: NOR gate with both the inputs are HIGH

2.2 Multiplexer

A multiplexer is a device that chooses one of several input signals and forwards the selected input into a single line. A multiplexer of 2^n inputs has n select lines, which are used to pick which input line to send to the output.

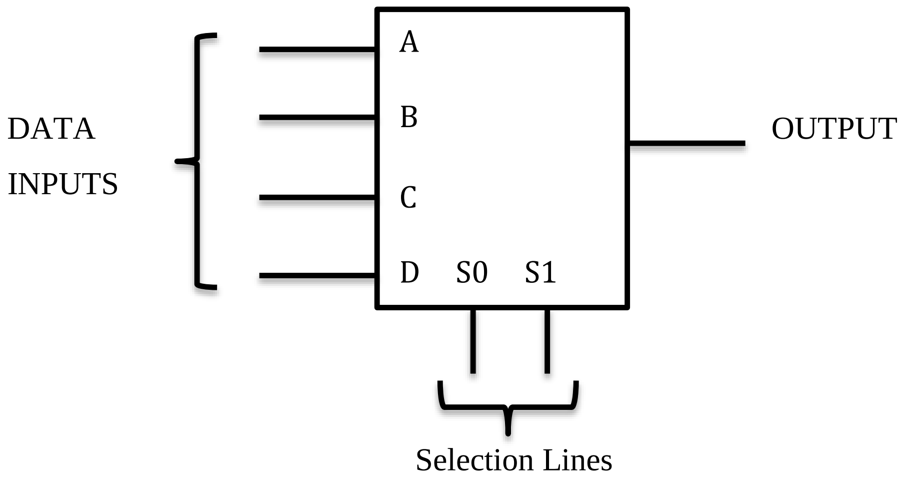


Figure 21: 4X1 Multiplexer

Figure 21 represents a 4X1 multiplexer which has 2 selection lines, 4 data inputs, and 1 output. Based on the selection lines value, particular data input will be redirected to output. The data at node A is forwarded to output, when selection lines are ‘00’ and data at node B is forwarded, when selection lines are ‘01’. Similarly, it goes for other nodes at C and D when selection lines are ‘10’ and ‘11’ respectively.

2.2.1 Implementation

The 4X1 multiplexer is implemented by considering the LEDS of red, blue, green, and yellow, respectively, that are used as inputs, and by considering the dip switch as the selection line/lines and the RGB LEDs as the outputs (RGB means red green blue, and when the green and blue mix, the LEDs make yellow). The electronic textile representation of 4X1 multiplexer is shown in figure 22.

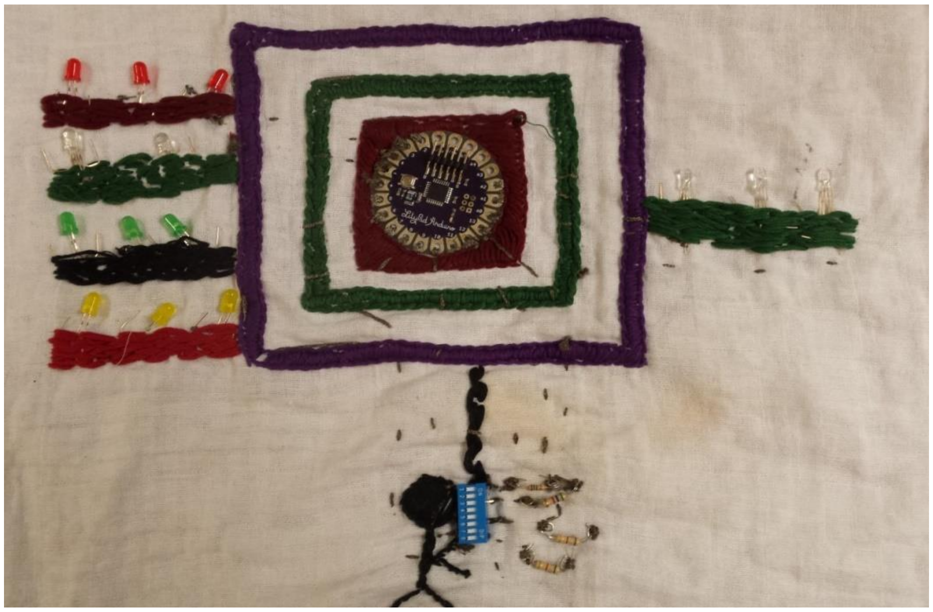


Figure 22: The Multiplexer representation

From figure 23, On selection of first switch (means selection of 00th line), the red LEDs (of Node A) gets activated (turns ON) and redirected to output by turning RGB LEDs into red color.

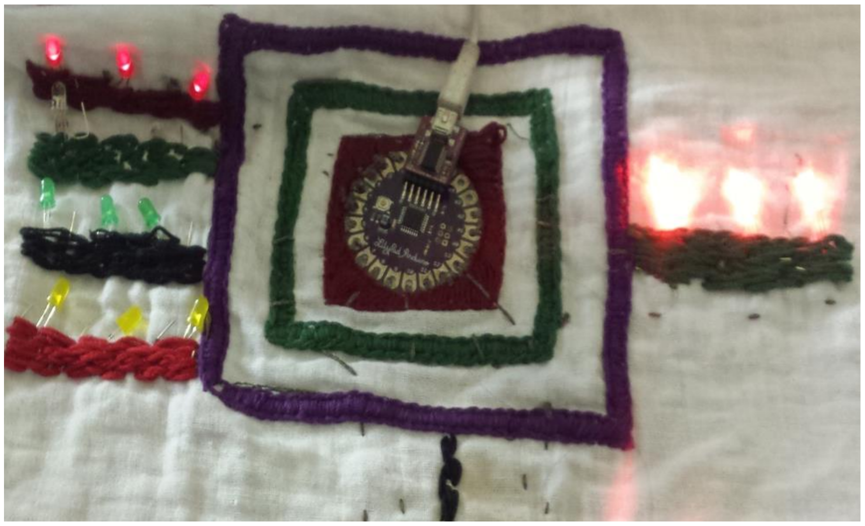


Figure 23: Multiplexer when the selection is ‘00’- red color forwards

Similarly, by selecting second switch (01th selection line), RGB LEDs are turned blue color as (node 2) Blue color LEDs at input side is forwarded as shown in figure 24.

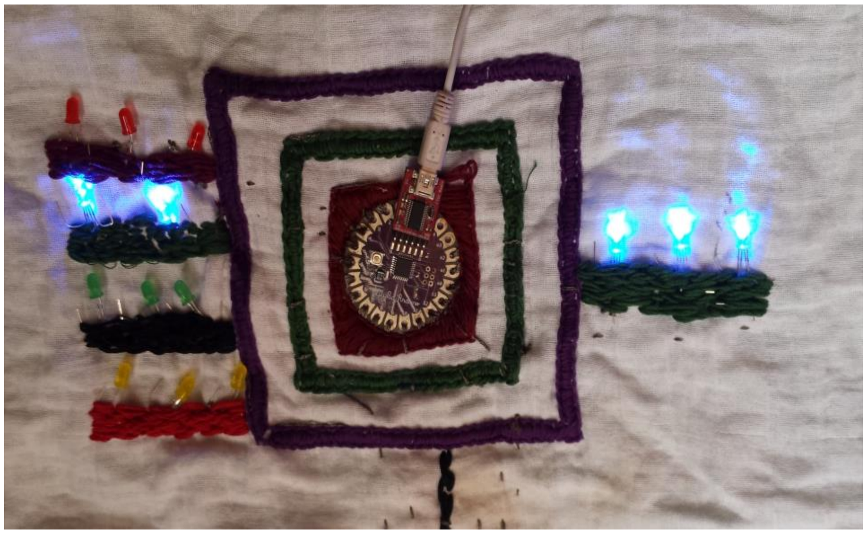


Figure 24: Multiplexer when the selection is ‘01’ – blue color forwards

When third switch (01th selection lines) is selected, green color at node 3 gets forwarded and displayed by turning RGB LEDs into green.

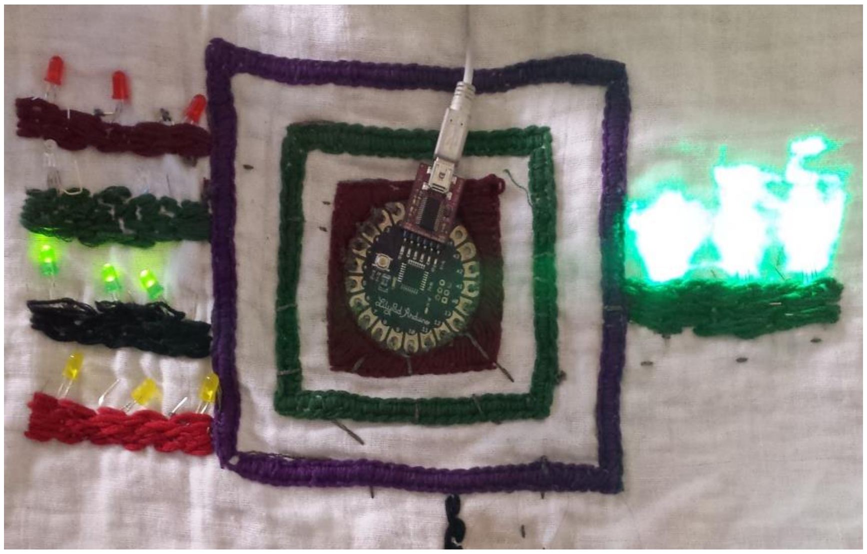


Figure 25: Multiplexer when the selection is ‘10’ – green color forwards

If the fourth switch is selected, then the input of the 11th line is selected and forwarded to the output. This is represented as yellow LEDs glowing at the input side and RGB LEDs turning into yellow on the output side (figure 26).

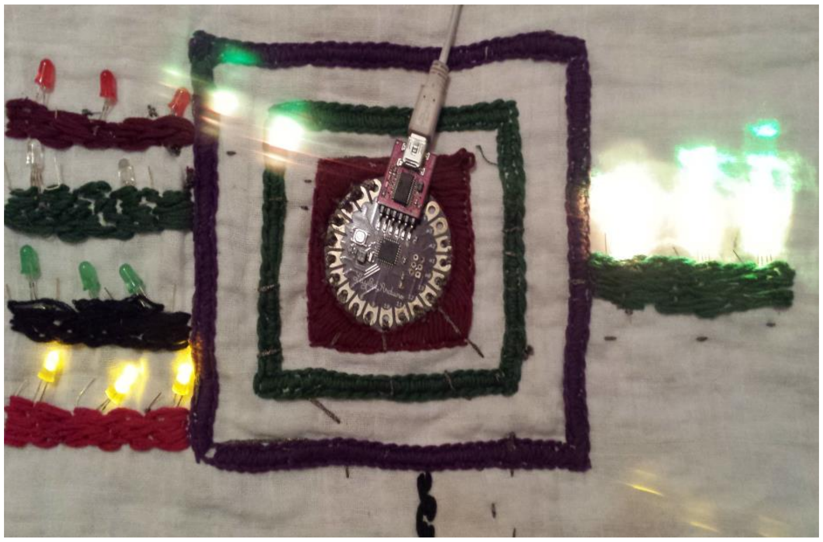


Figure 26: Multiplexer when the selection is ‘11’ – yellow color forwards

2.3 Decoder

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. It decodes the encoded information, so that the original information can be retrieved. A decoder can take the form of a multiple-input, multiple-output logic circuit that converts coded inputs into coded outputs, where the input and output codes are different. The enabled inputs must be ON for the decoder to function; otherwise its outputs will assume those inputs as disabled and will have different output code word. Decoding is necessary in different applications such as data multiplexing, 7-segment display, and memory address decoding.

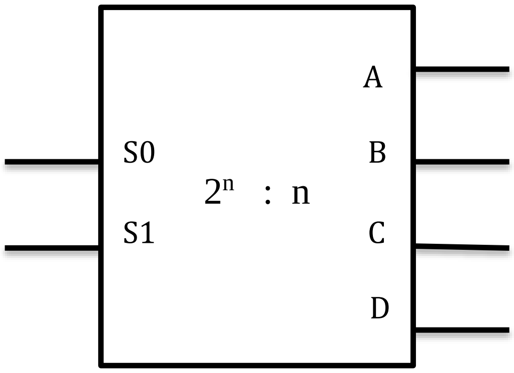


Figure 27: Decoder

Table 6: Truth Table

S0	S1	A	B	C	D
0	0	1	0	0	0
0	1	0	1	0	0
1	0	1	0	1	0
1	1	0	0	0	1

2.3.1 Implementation

The functioning of Decoder is illustrated by considering non-interconnected rows and columns. Switches are connected at the final points of rows and columns. LEDs are used as a contact point of that particular row and column. When a particular row and column is connected, then the LED connected to that row and column will glow.

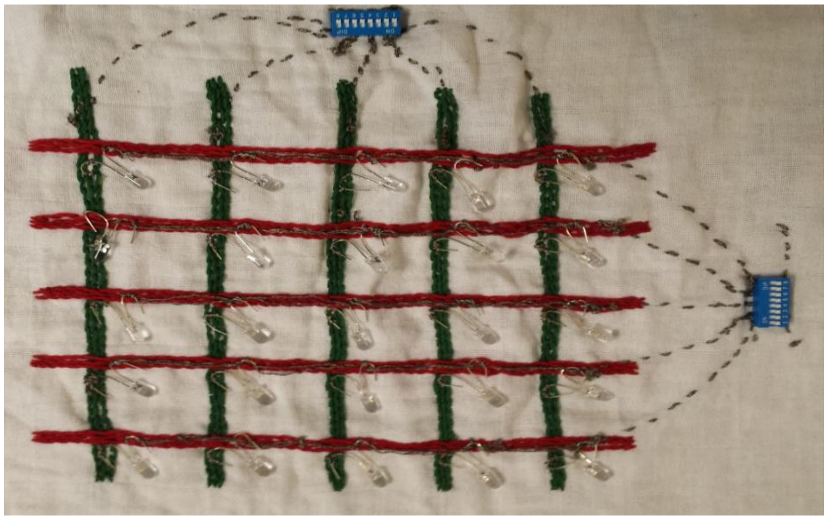
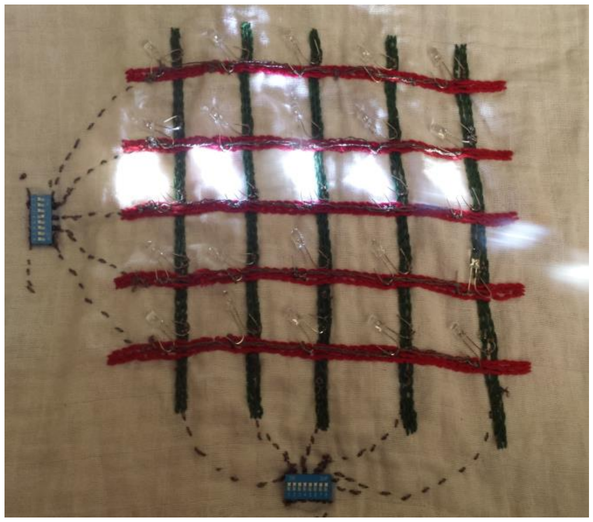


Figure 28: Decoder representation (Matrix form)

The LEDs are connected in the form matrix where all the positives of one row are connected and all the negatives of one column are connected.


(a)

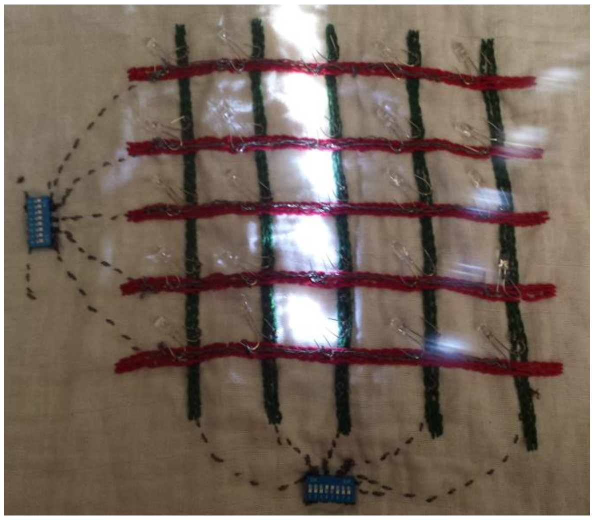

(b)

Figure 29: (a) Decoder showing one entire row connected their particular columns (LEDs of third row ON). (b) Decoder showing one entire column connected their particular rows (LEDs of third column ON)

Figure 30, represents the selection of row 1 and column 1 by using dip switches and the glowing of the first LED.

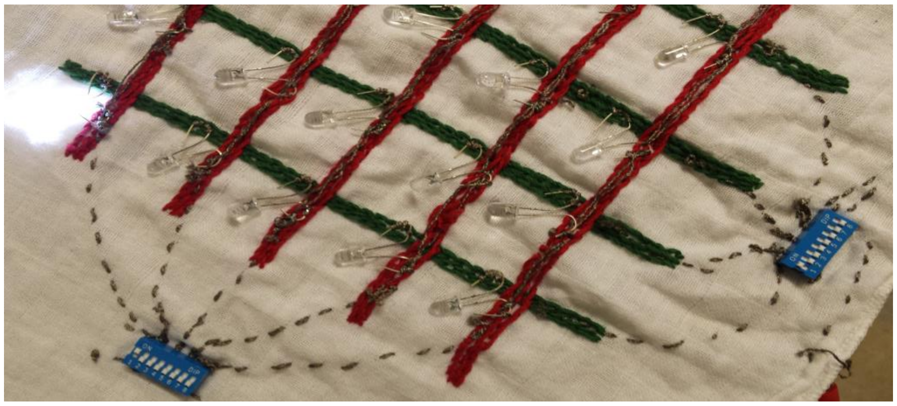


Figure 30: The dip switch selection to glow first LED – row 1 and column 1

2.4 Shift Register

A shift register is a cascade of flip-flops sharing the same clock, in which the output of each flip-flop is connected to the data input of the next flip-flop in the chain, resulting in a circuit that shifts by one position. Shift register is a multidimensional where it can be implemented simply by running several shift registers of the same bit-length parallel. A shift register can be parallel and serial. Also, there are different types of shift registers based on whether the input is serial or parallel and whether the output is serial or parallel.

If input is appended serially and output is extracted as parallel, then it is a serial-in parallel-out shift register (SIPO). If input is appended serially and output also extracted in serial, then it is called as serial-in serial-out (SISO). If input is appended in parallel and output is extracted in serial, then it is called as parallel-in serial-out (PISO). If both input and output are appended and extracted in parallel respectively, then it is called as parallel-in parallel-out (PIPO). There are also bi-directional, in addition to universal, shift registers which also allow shifting in both directions, from left to right and from right to left.

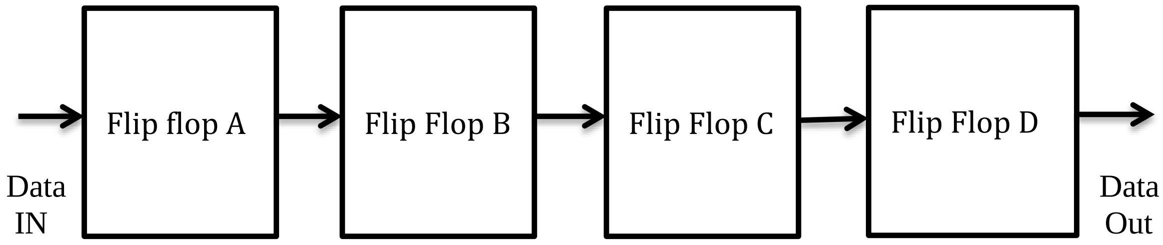


Figure 31: Serial shift register

2.4.1 Implementation

In shifter register implementation, I have considered the serial-in serial-out shift register. In this SISO, the data can be applied sequentially (bit by bit) either from left to right or from right to left.

N-Bit shift register contains n flip-flops. The register is first cleared by forcing all zeros of n-bit length to flip-flops. In first clock cycle, the ‘nth bit’ of new data is transferred to first flip-flop (from left to right). Then, in the second clock cycle, data ‘n-1th bit’ is shifted to second block where ‘nth bit’ is transferred to first flip-flop. In the third clock cycle, ‘n-2th’ is shifted to 3rd flip-flop, ‘n-1th bit’ and ‘nth bit’ to second and third flip-flops respectively. This process goes on. After nth clock cycle, the nth bit results as output. In order to get total data out, 2N cycles are required. Consider an example of a 4-bit register having 4 flip flops with data 0001.

$$\begin{tikzcd}
\text{4-bit data 0001} \arrow[r] & \text{shift Register} \\
0\ 0\ 0\ 1 \arrow[r] & 0 & 0 & 0 & 0 \\
\text{1st clock cycle} \arrow[r] & 1 \arrow[dr] & 0 \arrow[dr] & 0 \arrow[dr] & 0 \\
\text{2nd clock cycle} \arrow[r] & 0 \arrow[dr] & 1 \arrow[dr] & 0 \arrow[dr] & 0 \\
\text{3rd clock cycle} \arrow[r] & 0 \arrow[dr] & 0 \arrow[dr] & 1 \arrow[dr] & 0 \\
\text{4th clock cycle} \arrow[r] & 0 \arrow[dr] & 0 \arrow[dr] & 0 \arrow[dr] & 1 \\
\text{5th clock cycle} \arrow[r] & 0 & 0 & 0 & 0 & \text{and data ‘1’ results as output.}
\end{tikzcd}$$

The e-textile representation of 4-bit register is shown in figure 32. Each block (as register) is connected to LEDs (represents data). Continuous glowing of LEDs one by one represents shifting of data from one block to another block.

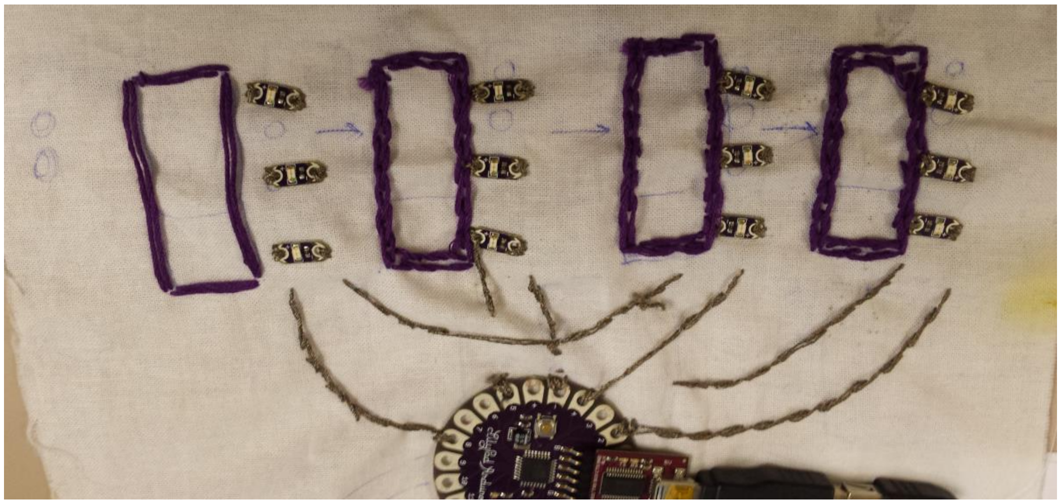


Figure 32: The serial shift register representation

Consider, the 4-bit register (blocks) has data 0000 (all LEDs are LOW) before the first clock cycle. When the data ‘1’(LED HIGH) is applied to first register in first clock cycle, the data in 4 bit register would be 1000. This is represented as blue LEDs HIGH (data ‘1’), and all the other block LEDs LOW (data ‘0’) in figure 33.

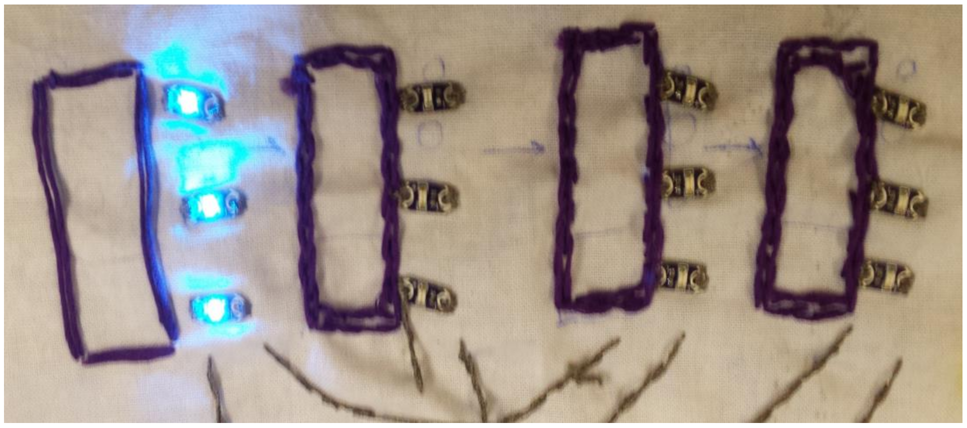


Figure 33: The shift register when the data is at first flip-flop in the first clock cycle

In the second clock cycle, when the data ‘0’ is applied to the first register, then 4-bit register has data 0100. This is represented as red LEDs of 2nd block HIGH (data ‘1’), and all the other block LEDs LOW (data ‘0’) in figure 34.

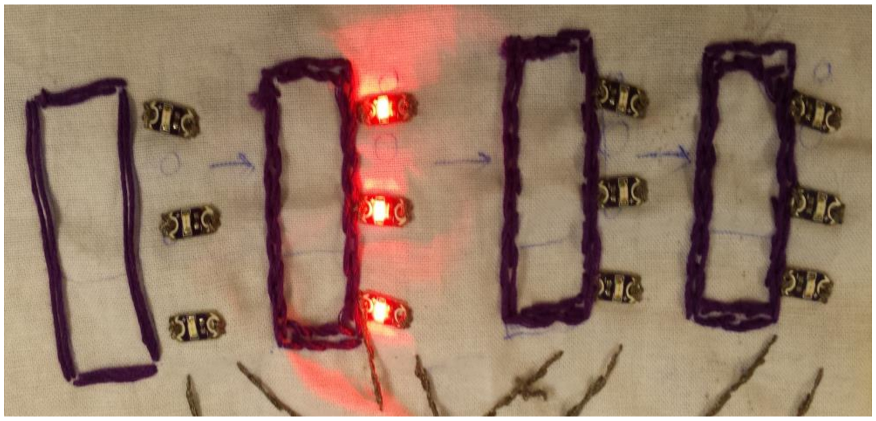


Figure 34: The 4-bit shift register when the data at second flip-flop in the second clock cycle

In the third clock cycle, when data ‘0’ is applied to the first register, then 4 bit register has data 0010. This is represented as blue LEDs of 3rd block HIGH (data ‘1’), and all other block LEDs LOW (data ‘0’) in figure 35.

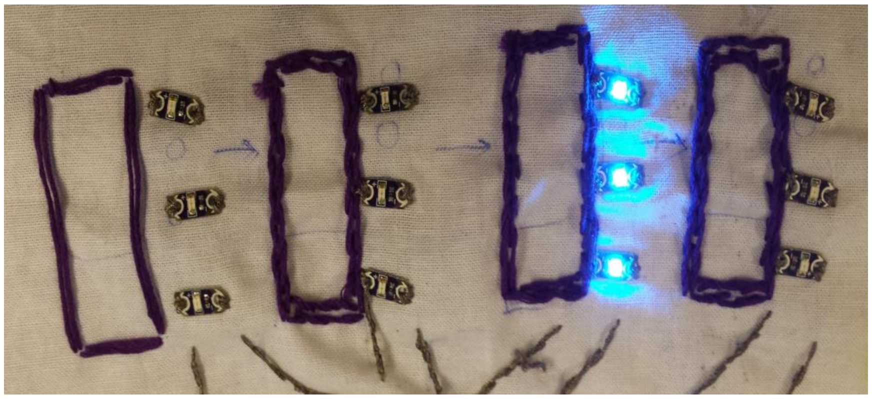


Figure 35: The 4- bit shift register when the data at third flip-flop in the third clock cycle

Similarly, in the fourth clock cycle, when data ‘0’ is applied to the first register, then the 4 bit register has data 0001. This is represented as red LEDs of 4th block HIGH (data ‘1’), and all the other block LEDs LOW (data ‘0’) in figure 36.

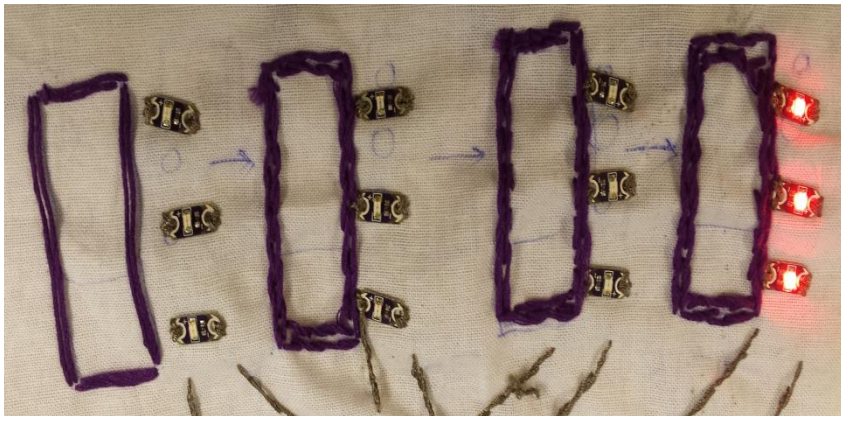


Figure 36: The 4-bit shift register when the data at final flip-flop in the final clock cycle

2.5 Counter

A counter is a device which counts the number of times a particular event has occurred. It also count values in related to a clock signal. Register type circuits like flip-flops are used to implement counters. Different classification of counters exists. They are: Asynchronous counter, synchronous counter, up/down counter, Decade counter, ring counter. In asynchronous counter, the inverted output of former flip-flop is fed as a clock to latter flip-flop which means the working of flip-flops is driven by their own outputs. In synchronous counter, all state outputs are controlled by using single clock. In up/down counter, based on control input it counts up and down number of times of particular event. These counters are digital in nature and count in binary form with wide applications like digital clocks, timers, and so on.

2.5.1 Implementation

Digital clock of counting minutes is implemented by using 7 segment displays and shift registers (74HC595). Shift registers are used to drive the data from 7 segment display to other.

The change timing (minutes) can be done by only re- programming it. When it gets started, it counts from 00 to 99. After 99th minute, it can again turn to 00. This process repeats. The figure shows minute clock. In this clock, it also consists of minutes adjustments with set button which acts as more interactive design for counters.

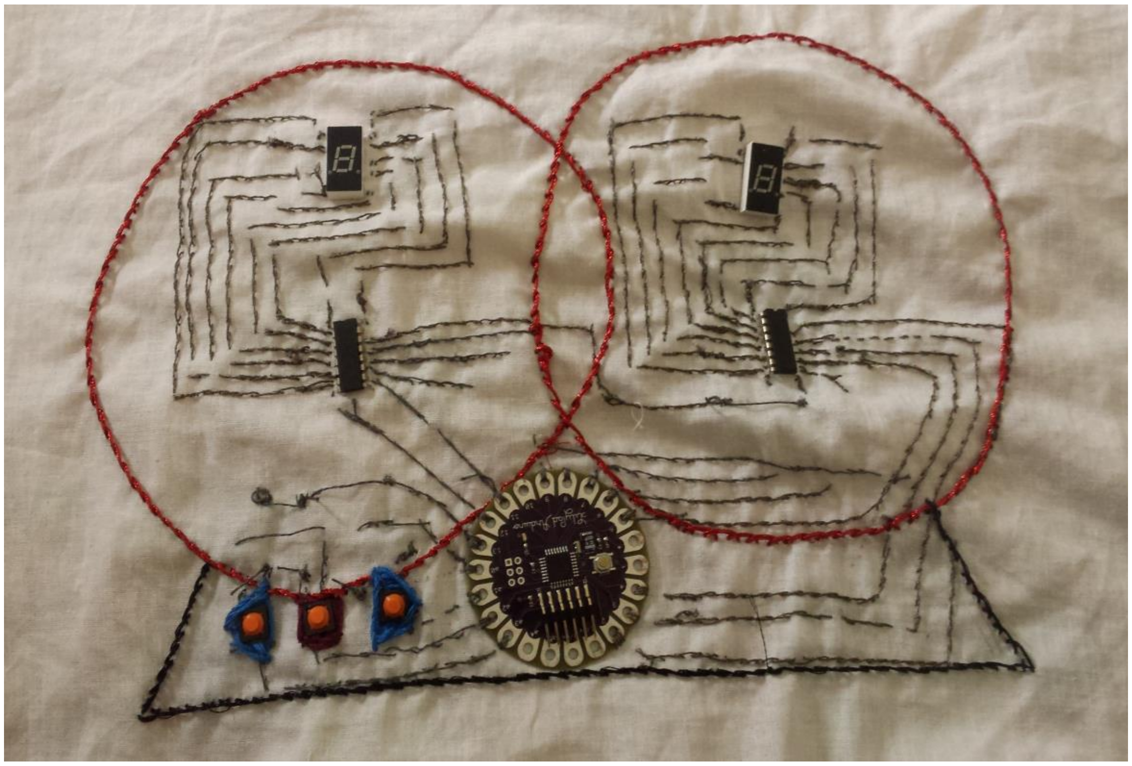


Figure 37: Counter design

2.6 Ring Counter

Ring counter is a type of counter similar to a circular shift register. The output of the last shift register is fed to the input of the first register. Ring counters are used in hardware logic design such as ASIC and FPGA to create simple finite state machines. An adder circuit is more complex than ring counters, as a binary counter requires an adder circuit.

There are two types of ring counters, which are straight ring counter and twisted ring counter. In straight ring counter, the output of the last flip-flop is connected to the first flip-flop input and circulates a single 1-bit around the ring; whereas in twisted ring counter, the complement of the output of the last flip-flop is connected to the input of the first flip-flop and circulates a stream of ones followed by zeros around the ring.

Ring counters with hamming distance of 2 allow the detection of single bit upsets that can occur in hazardous environments. The main disadvantage of ring counters is that they have lower density codes.

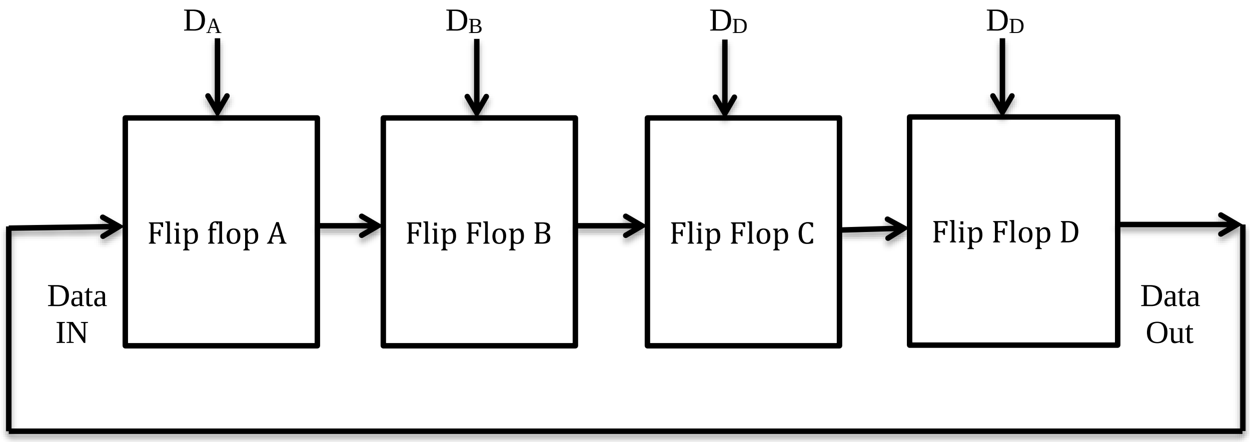


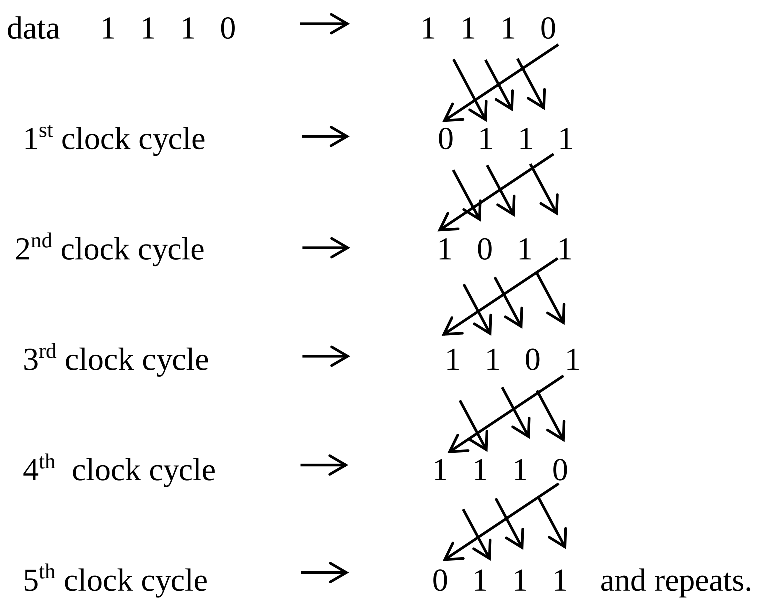
Figure 38: Ring counter

2.6.1 Implementation

In ring counter, the input is appended in parallel to all the flip-flops for the first clock cycle. The appended data is circulated from second clock cycle onwards. Consider a 4-bit register where 4-bit data 1110 is transferred. After 1^{st} clock cycle, all the flip- flops contains the data of 1110. Then, for 2^{nd} clock cycle, the data -out at the fourth flip-flop is fed to data-in in which final data in all flip-flops is 0111. After third and fourth clock cycles, the data will be

1011 and 1101, respectively. In the next clock cycle, the final data output is 1110, which identically matches the input.

The data in the 4-bit register cycles for every clock cycle and does not change other than among the combinations of 1110, 0111, 1011, and 1101. Total data should be applied parallel for all the four blocks in order to change the data. Then that data cycles for every clock cycle.


data 1 1 1 0 → 1 1 1 0
1st clock cycle → 0 1 1 1
2nd clock cycle → 1 0 1 1
3rd clock cycle → 1 1 0 1
4th clock cycle → 1 1 1 0
5th clock cycle → 0 1 1 1 and repeats.

The e-textile representation of 4-bit register is as follows. Each block (as register) is connected to LEDs (represents data). Feedback data is also represented by LEDs. Here also, ‘1’ is considered as LEDs ON, 0 is considered as LEDs OFF. Continuous glowing of LEDs one by one represents shifting of data from one block to another block.

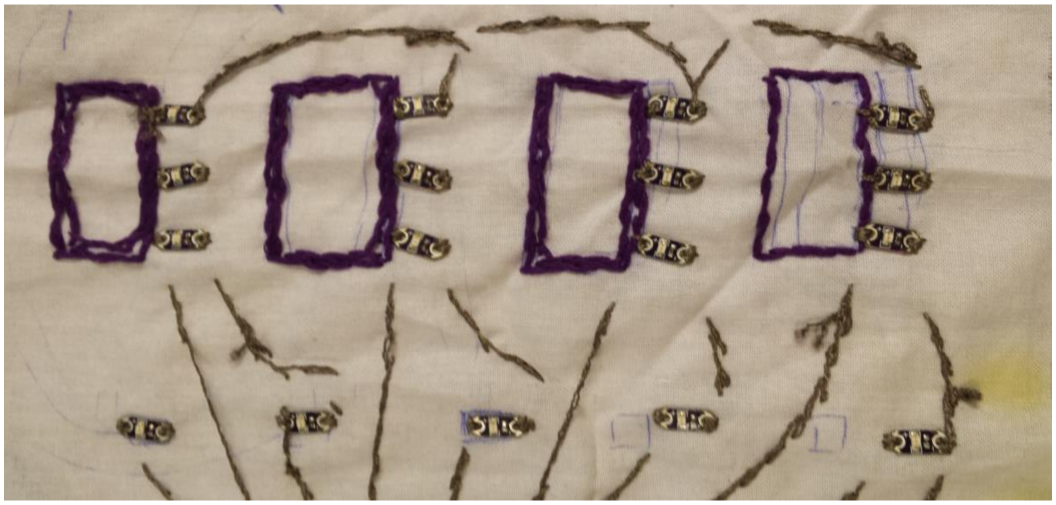


Figure 39: Representation of the ring counter along with back LEDs

Before the first clock cycle, all the LEDs are OFF (represents data 0000). Then 1110 data is applied in parallel for all the blocks in first clock cycle. This is presented as first three blocks LEDs ON (data ‘1’) and last block LEDs OFF (data ‘0’) in figure 40.

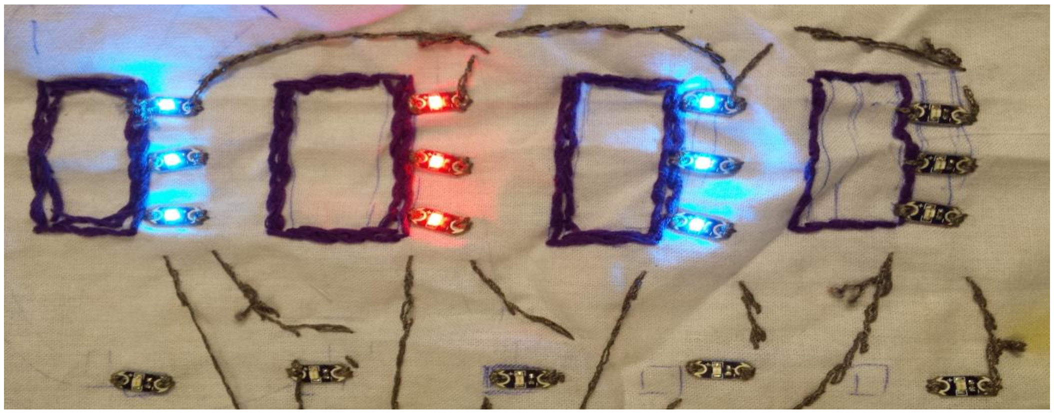


Figure 40: The ring counter representing 1110

In the second clock cycle, the data gets circulated as last bit to first, second to first, and so on. The final data is 01111. This is presented as three blocks from last block LEDs ON (data ‘1’) and first one LEDs OFF (data ‘0’) in figure 41.

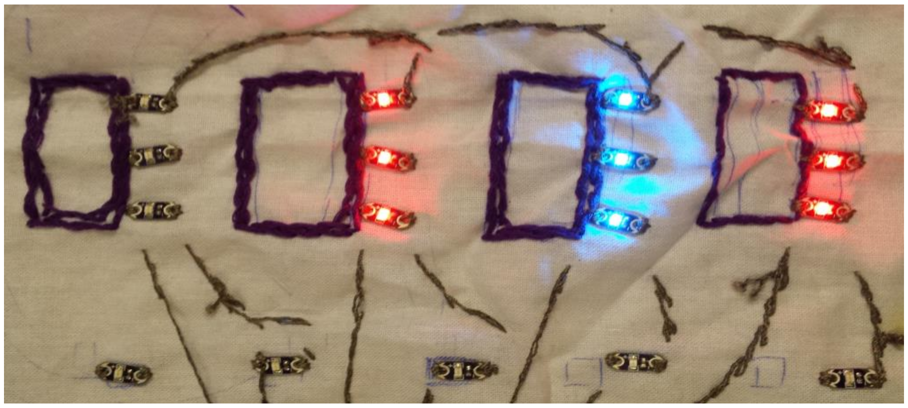


Figure 41: The ring counter representing 0111

In the third clock cycle, data will be 1011. This is represented as first, third and last block LEDs HIGH (data ‘1’) and second blocks LEDS LOW (data ‘0’).

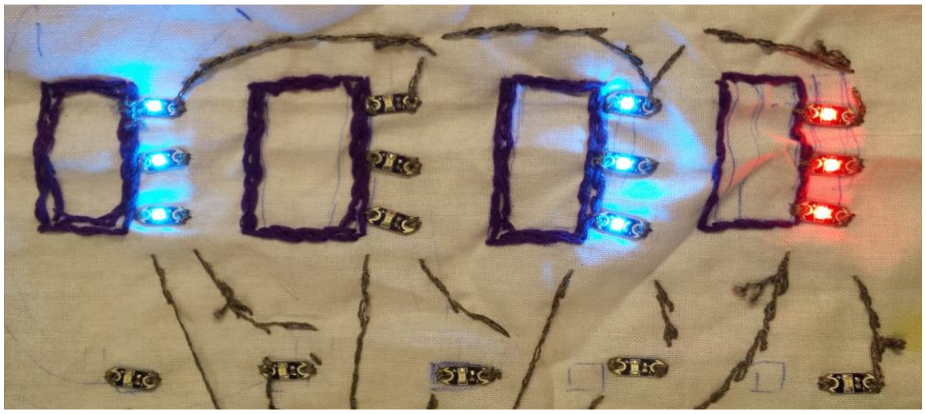


Figure 42: The ring counter representing 1011

In the fourth clock cycle, data will be 1101. This is represented as first, second and last block LEDs HIGH (data ‘1’) and third blocks LEDS LOW (data ‘0’). Then, the cycle repeats.

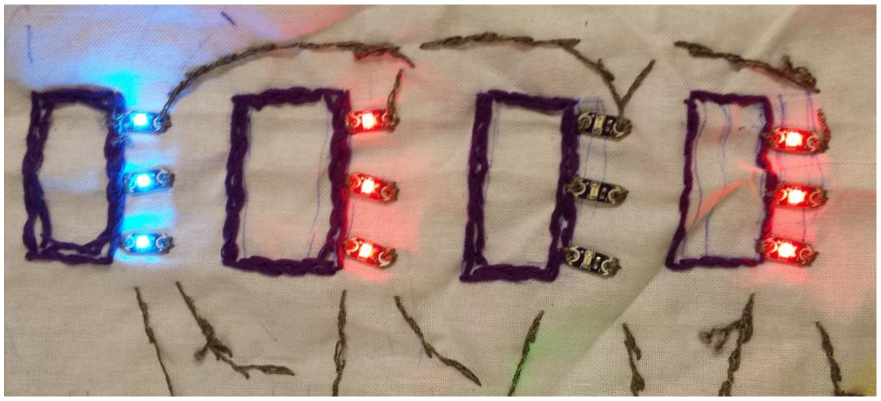


Figure 43: The ring counter representing 1101

CHAPTER 3

VLSI DESIGN

3.1 MOS Transistors

The metal oxide semiconductor field-effect transistor (MOSFET) is a special type of field effect transistor that works by electronically varying the width of channel with charge carriers. This transistor is used for amplifying or switching electronic signals. It is device of four terminals: source (S), gate (G), drain (D), and body (B). The body is connected to ground. This is the most common transistor used in both analog and digital circuits. Generally, most transistors are majority carrier devices. There are two types of MOS transistors – NMOS and PMOS. The PMOS transistors are made by adding P impurities in N substrate and NMOS transistors made by adding N impurities in P substrate. Holes are majority carriers in PMOS transistors where electrons are majority carriers in NMOS. In NMOS transistors, the source and drain have free electrons whereas the controlling gate has free holes.

NMOS has different regions of operation: cut off, linear, and saturation regions. If input voltage which is gate to source voltage V_{gs} is less than threshold voltage V_t, then body- source and body- drain will be reverse biased and there will be no conducting path between source to drain. This mode of operation is called as cut-off region.

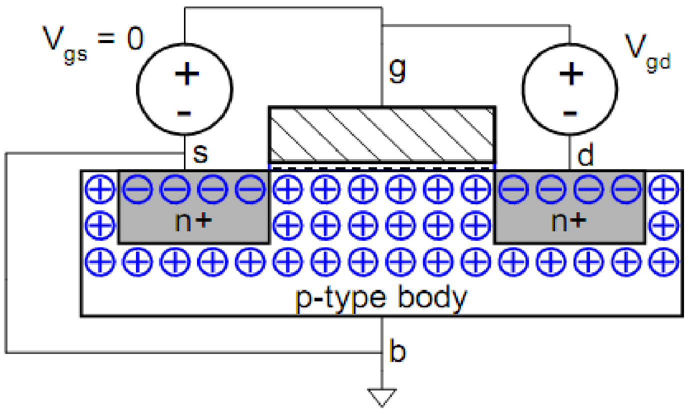


Figure 44: NMOS cutoff region

If input voltage V_{ds} is greater than threshold voltage, then the free flow of inversion electrons form a conducting path between drain and source channel. Hence, current I_{ds} start flowing from drain to source. The current flowing will increase as input voltage increases. This region is called as linear region.

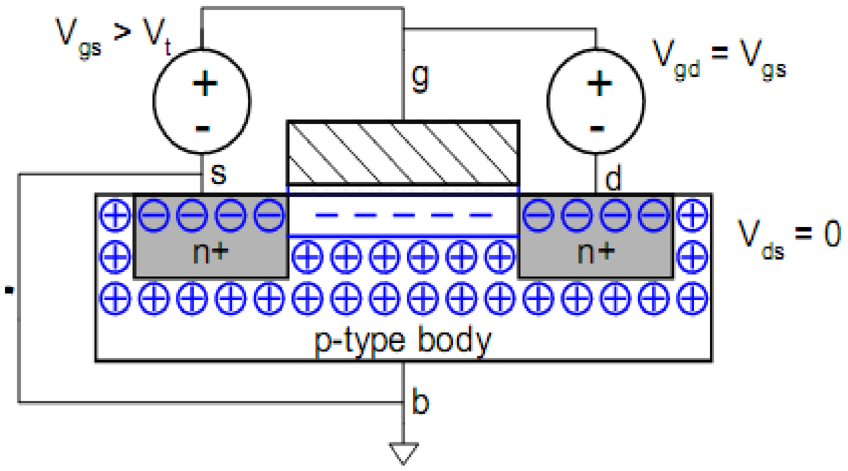


Figure 45: NMOS linear region

If input voltage is increased furthermore, then the channel is no longer inverted near the drain region and is pinched off. This region is called as saturation region. Then, current gets saturated and I_{ds} is independent of input voltage. The threshold voltage is based on usage of material, number of dopants in the body, and thickness of oxide.

3.1.1 Implementation

The flow of electrons in NMOS transistor when voltage applied is greater that threshold voltage is implemented as shown in figure 46. The LEDs represents electrons and continuous glowing of electrons represents its flow. A push button acts as gate to source voltage (Vgs). On pushing the button, then LEDs starts glowing representing the Vgs voltage greater that threshold and showing free flow of electrons.

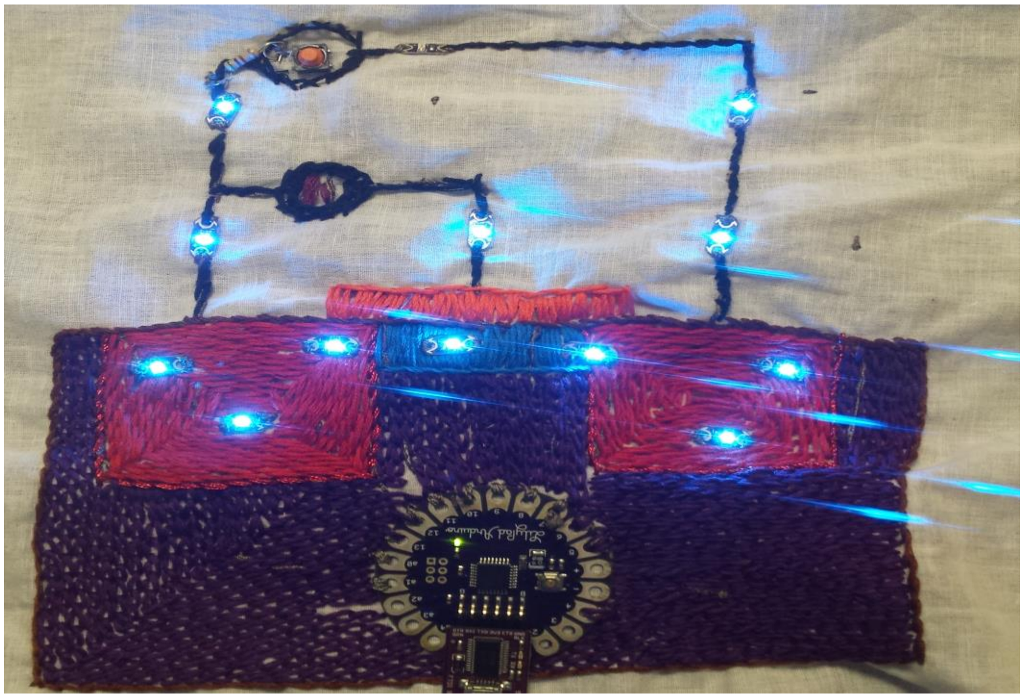


Figure 46: NMOS representation in linear region

3.2 Clock Gating

Clock gating is defined as a technique used in synchronous circuits for reducing the dynamic power dissipation. This is done by turning off the clock to registers in unused blocks so that flip-flops in them do not have switching states. Switching states will consume power. This switching power is calculated by using activity factor. Activity factor is the probability that the circuit will switch from one node to another. This clock gating requires determination whether block will be used or not. This is done by adding more logic to the circuit. Clock gating works by taking the enable conditions attached to registers, and uses them to gate the clocks. This clock gating process can also save significant die area as well as power, since it removes large numbers of MUX and replaces them with clock gating logic.

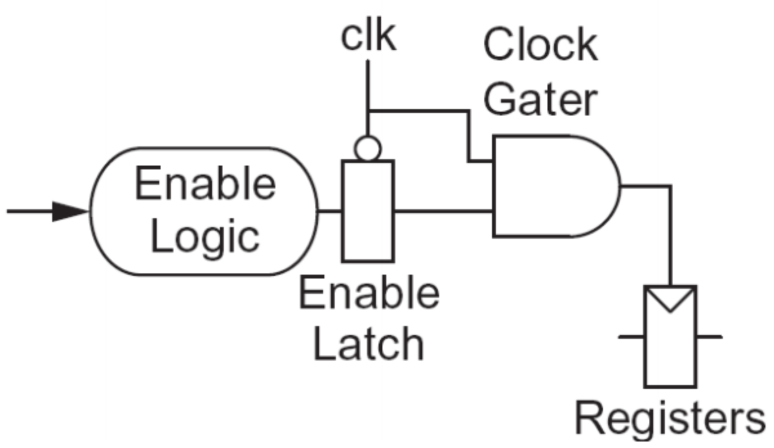


Figure 47: Clock gating

The clock gating logic can be added into the design in different ways. Addition of logic code into the RTL code as an enabling condition automatically translates itself into clock gating logic by synthesis tools. Inserting into the design manually by the RTL designers is also known as module level clock gating, and can be done by initiating the library. Any RTL modifications to improve clock gating will result in functional changes to the design which needs to be verified.

3.2.1 Implementation

Let’s consider different blocks represented as 3x3 matrix form as shown in the figure 48. Firstly, assume that all blocks are fed with clock of figure 49. When someone wants to represent with these matrix oriented blocks in different patterns, one can turn off the clock by using push button. On assuming, push button as enable logic, this whole concept represent clock gating(unused blocks turning off -> blocks of different patterns). LEDs of each and every block are used to represent their clock. LEDs ON denotes clock is applied and vice versa.

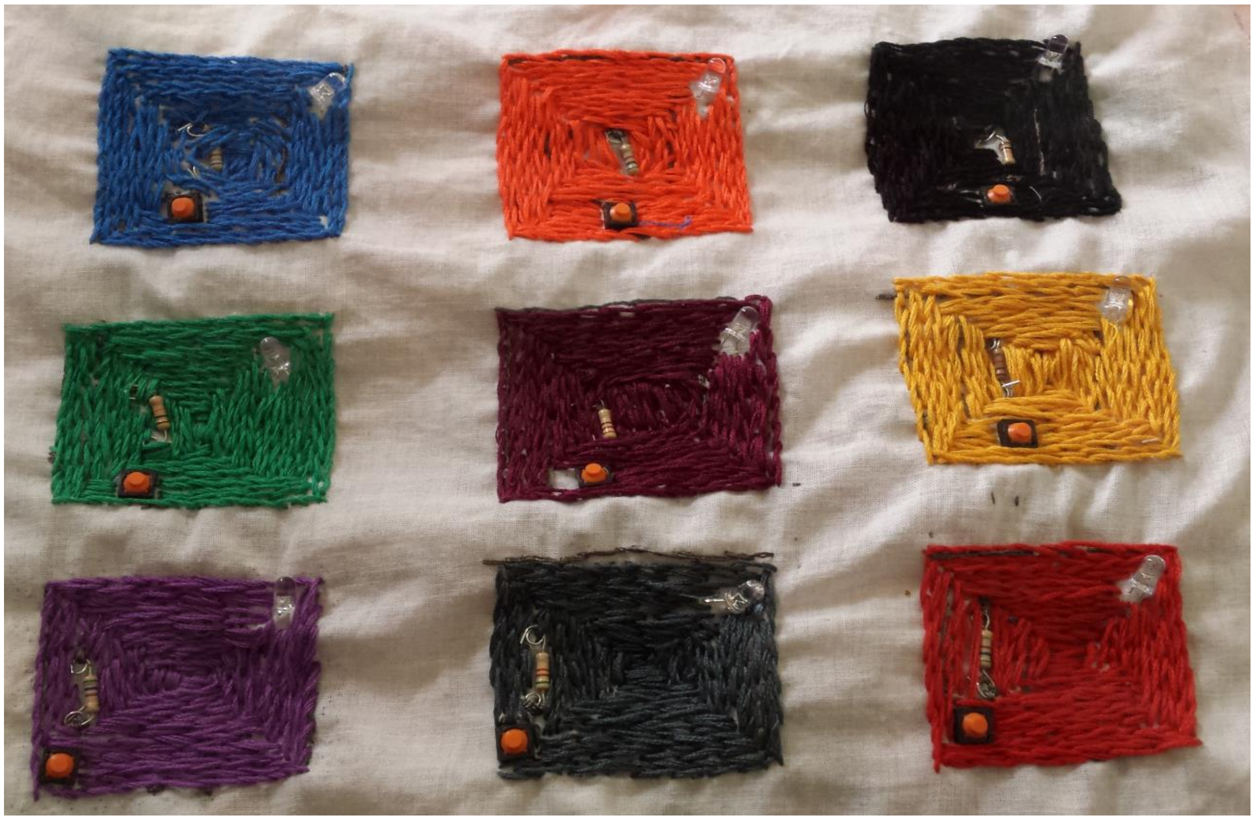


Figure 48: Clock gating representation

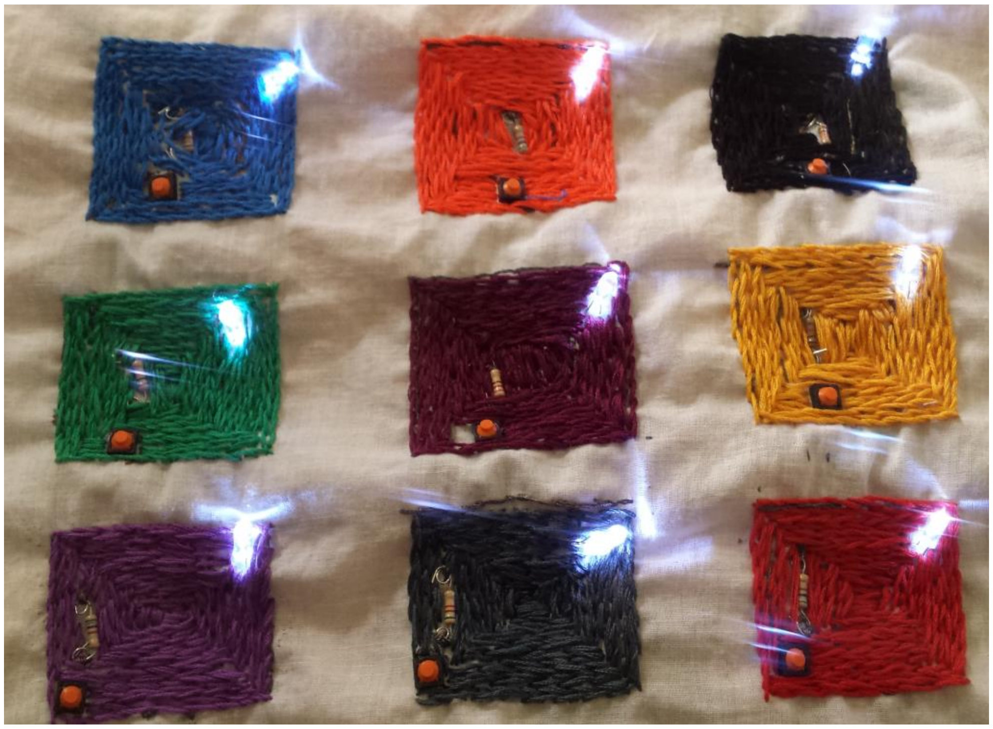


Figure 49: Clock is fed to all the blocks

By operating the pushbuttons, there is possibility of achieving different types of patterns (turning of unused block clocks).

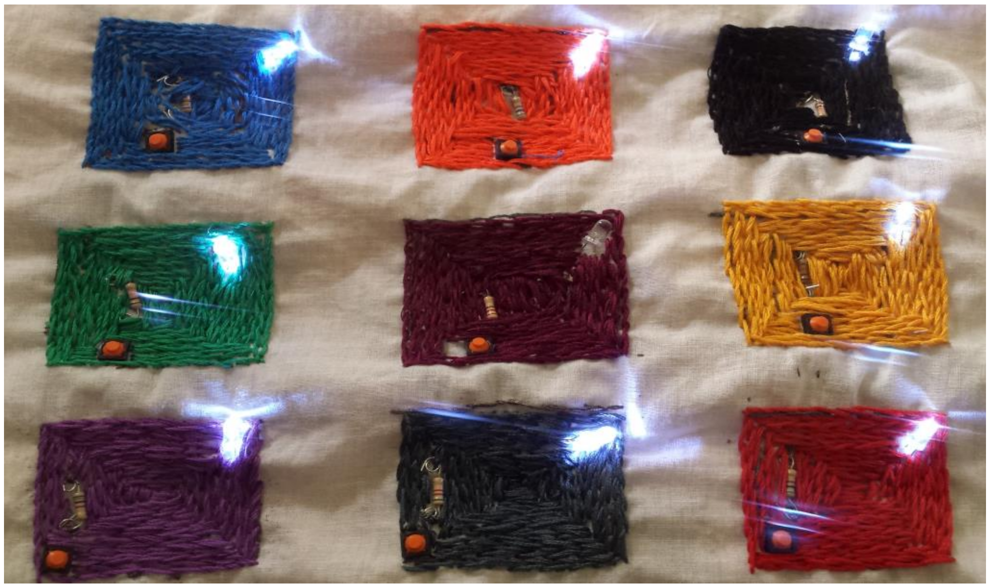


Figure 50(a): Square pattern (turned off clock unused middle block)

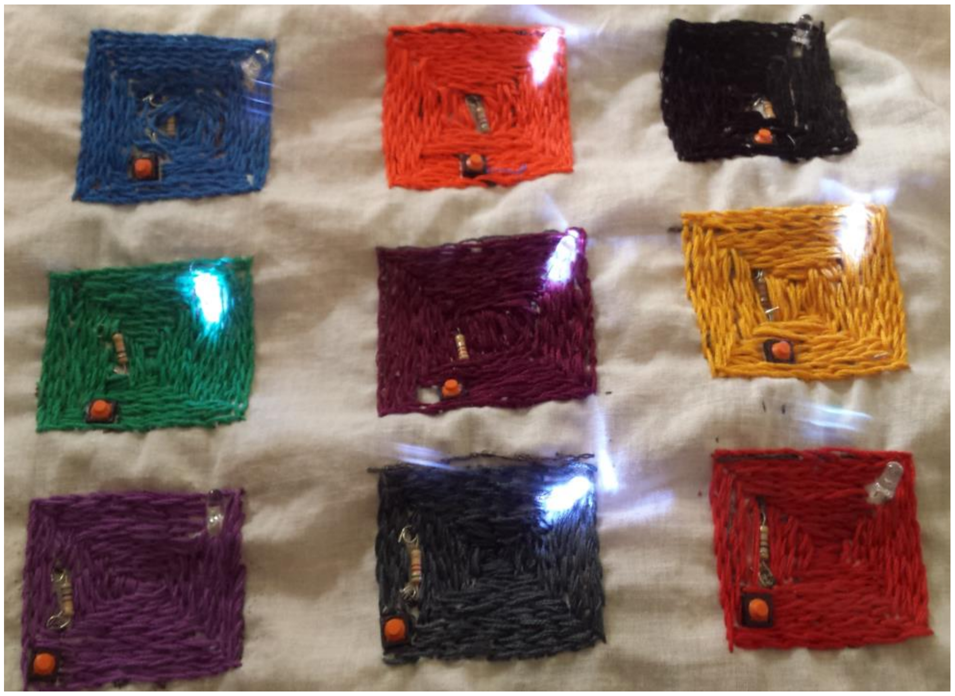


Figure 50(b): Plus pattern

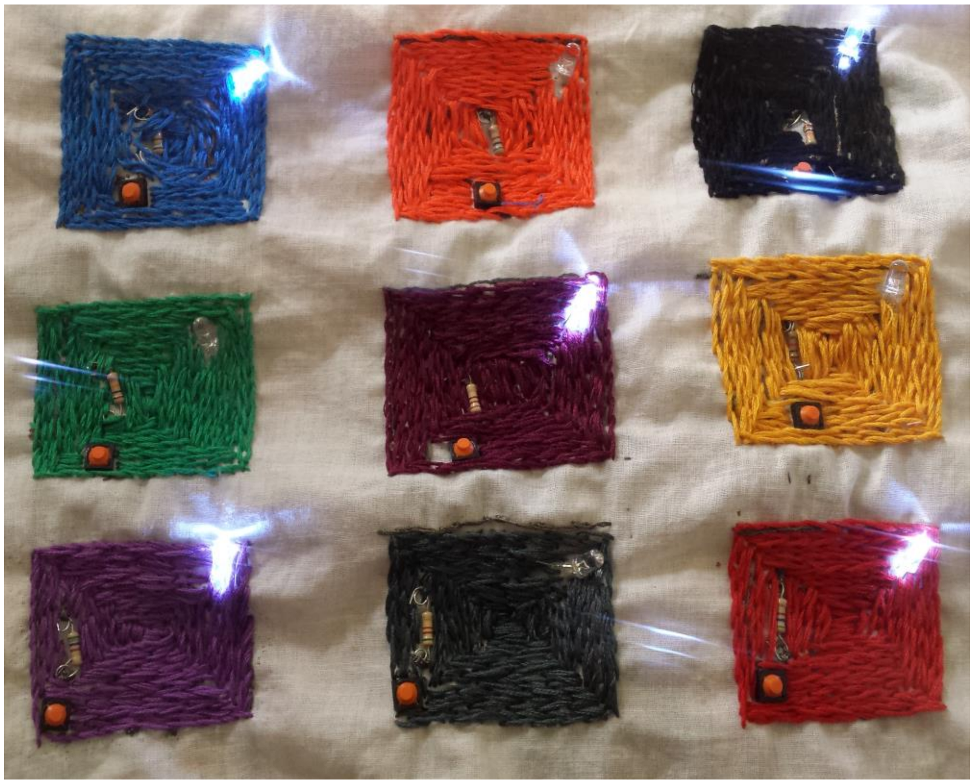


Figure 50(c): Cross pattern

3.3 Power Gating

Power gating is a technique used in integrated circuits in order to reduce power consumption by applying virtual V_{dd} to blocks of the circuit that are unused. This also helps in reducing stand-by or leakage power. The introduction of virtual V_{dd} in the circuit is operated by header switch transistors. This virtual V_{dd} power is applied to these blocks in sleep mode. The output of the blocks should be gated to prevent the invalid logic to the next blocks. Power gating effects design architecture more than does clock gating. Power gating increases time delays as power gated modes should be safely entered and exited. Shutting down the blocks can be accomplished either by software or hardware.

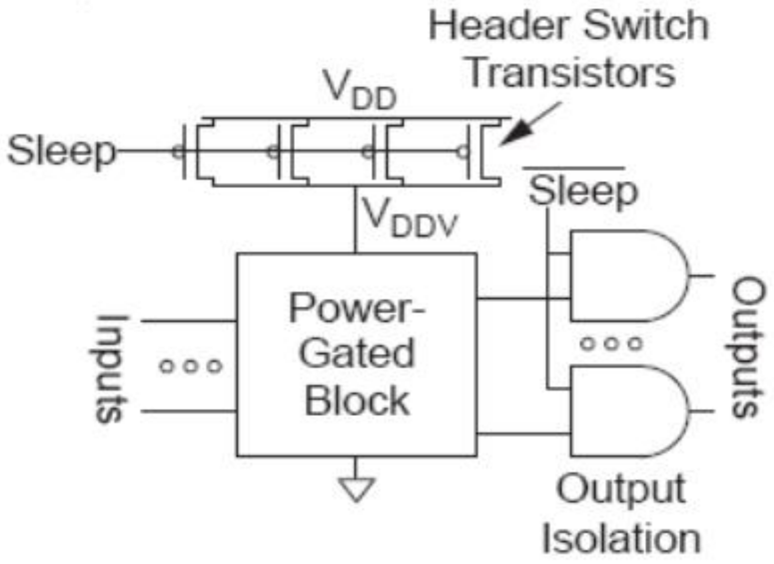


Figure 51: Power gating

Driver software can schedule the power down operations. Generally, an externally switched supper supply is used for power gating logic. This is done by shutting down the power at small intervals of time. CMOS switches that provide power to the circuits are controlled by power gating controllers. But, the outputs are discharged very slowly. Hence, output voltage levels may spend more than threshold voltage levels. So, PMOS transistors as header switches are most helpful in power gating. Header switching also requires careful sizing as it should add minimum delay to the circuit during active operation and low leakage during sleep.

3.3.1 Implementation

From figure 52, red LEDs represent Vdd, blue LEDs represent virtual Vdd and green LEDs represent power to registers. When the registers are in active stage, the virtual Vdd is connected original Vdd, making connect to actual power to register blocks.

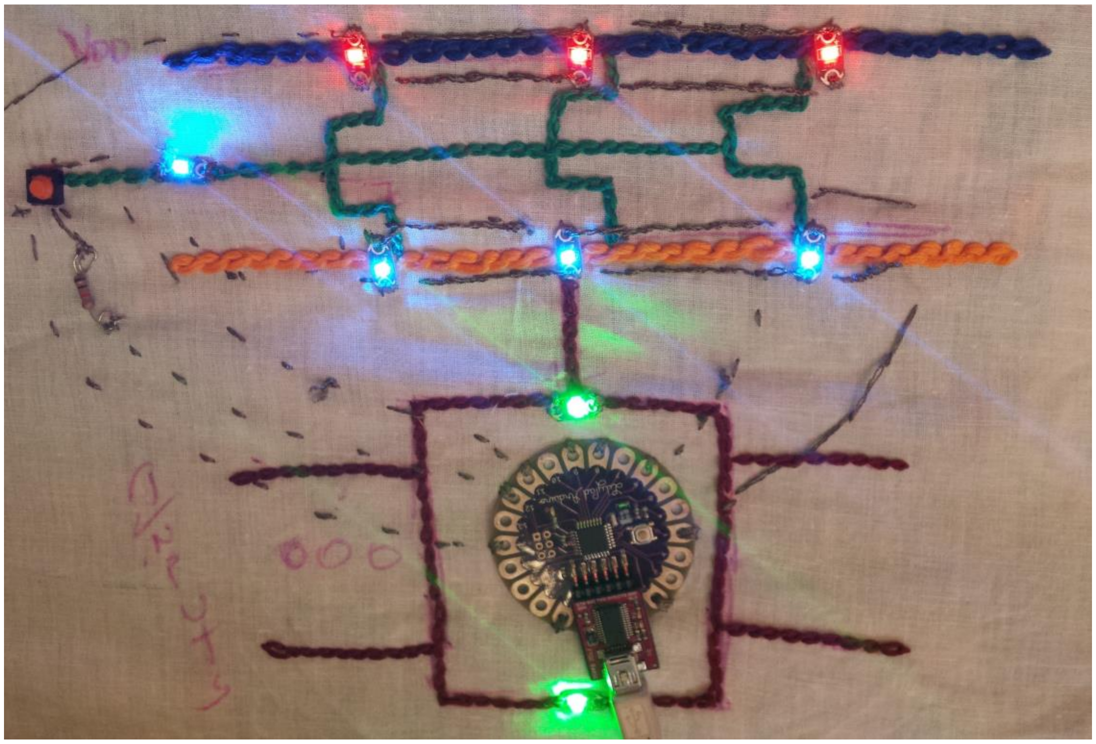


Figure 52: Original Vdd to registers

Push button is used to control the connection of original Vdd to the virtual Vdd. On switching, the vritual Vdd gets disconnected and slowly goes to zero. From figures – 53, 54, 55, the blue LEDs (virtual Vdd representation) getting less bright on turning off the power to blocks whereas the LEDs are totally OFF (grounded) on figure 56.

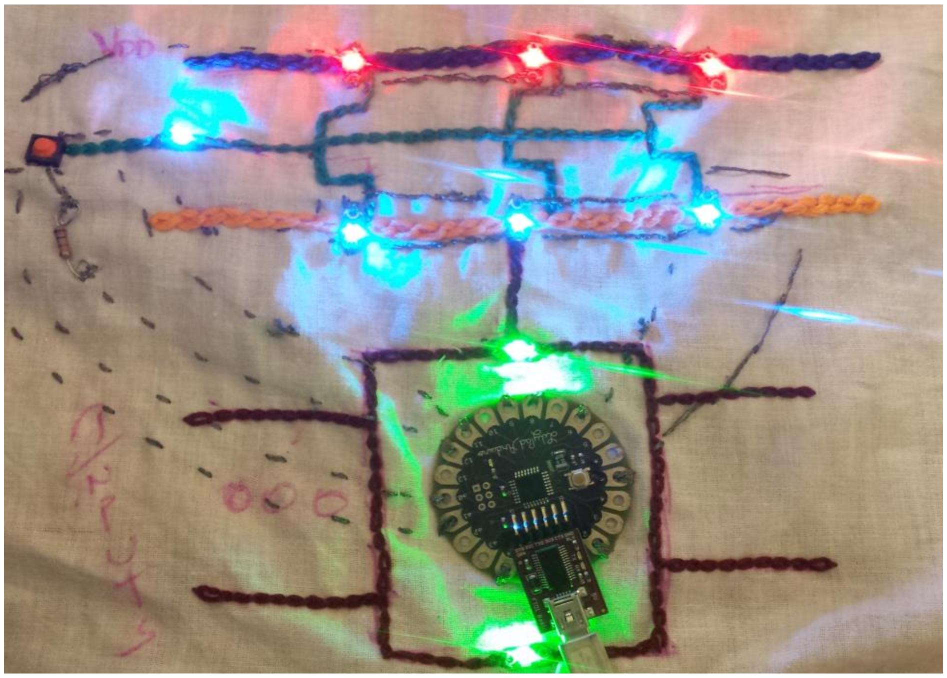


Figure 53: Turning OFF power to registers – block LEDs OFF and virtual LEDs less brightness

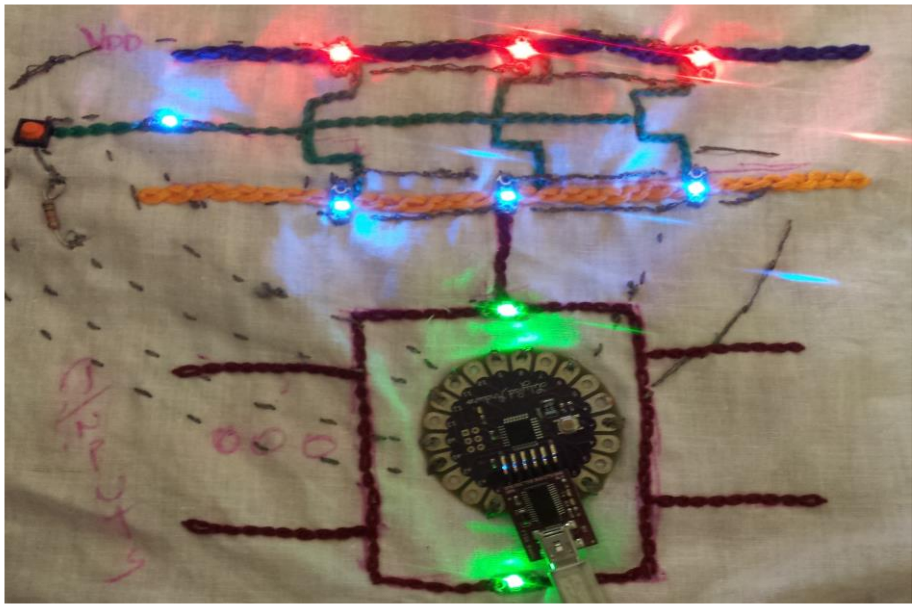


Figure 54: Less brightness of LEDs on non- connection with original Vdd

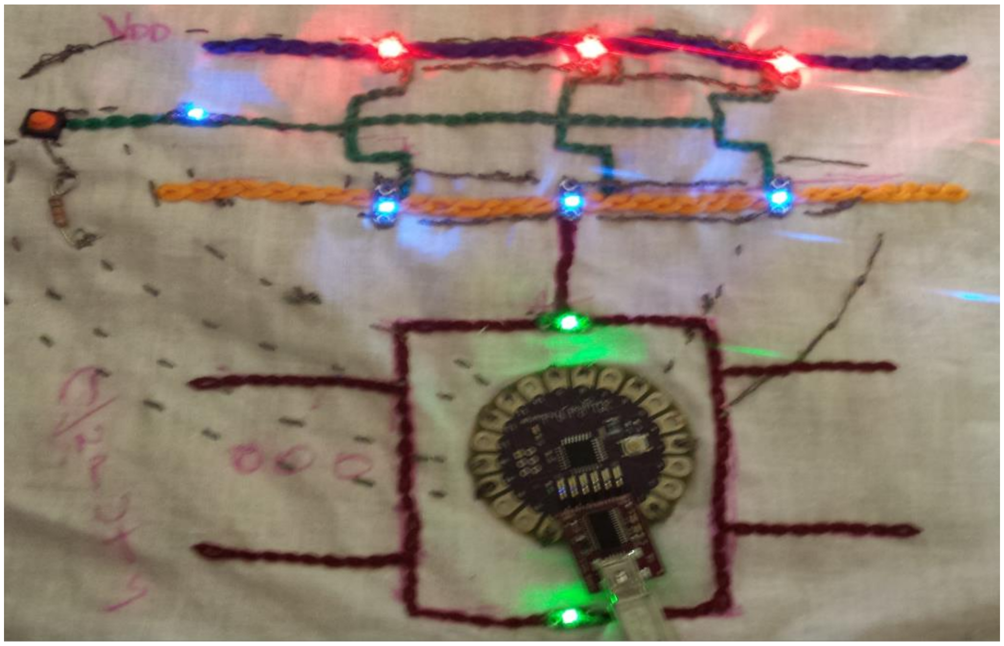


Figure 55: Little brightness of LEDs on non- connection with original Vdd

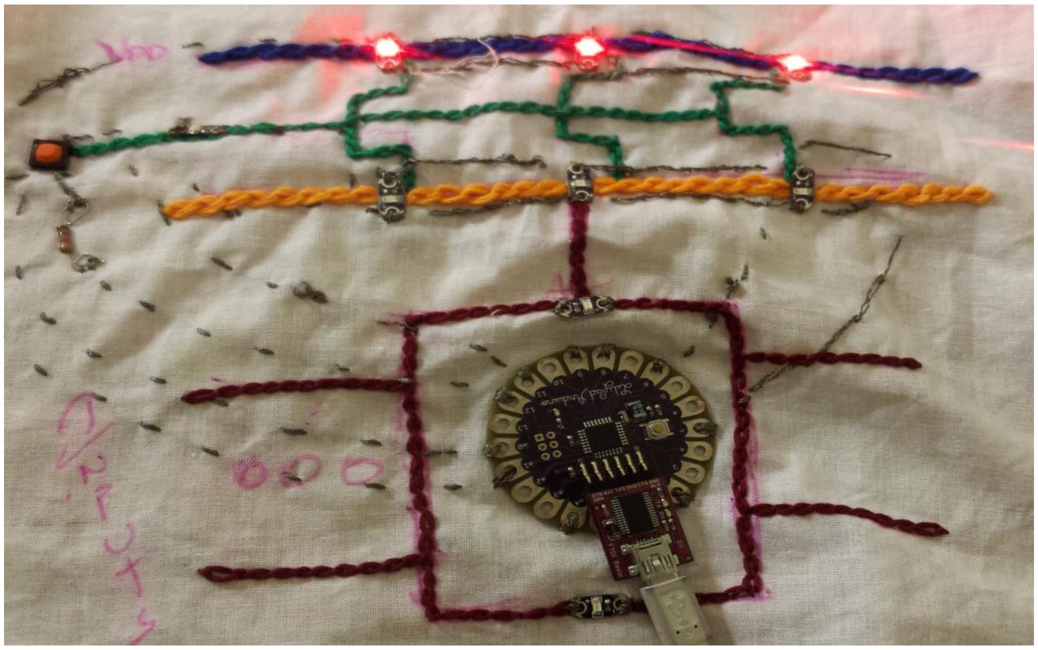


Figure 56: NO connection b/w original Vdd to Virtual Vdd

CHAPTER 4

CONCLUSION AND FUTURE WORK

This thesis is mainly focused on the fundamentals of digital logic design and VLSI design. This is an interesting way of introducing to freshmen and high school students. This would help in enhancing enthusiasm towards electrical engineering in students. This method of introduction gives a strong hold on these concepts. This can also be extended by introducing advanced topics of VLSI design and can be made more interesting by using illustrative examples.

REFERENCES

[1]Kanjun Qiu., Leah Buechley., Edward Baafi., Wendy Dubow : A Cirriculum for teaching computer science through computational textiles, ACM (2010), 423–432.

[2]Buechley, L., Eisenberg, M., and Elumeze, N. Towards a curriculum for electronic textiles in the high school classroom. SIGCSE Bull. 39, 3 (2007), 28–32.

[3] Charles Hacker and Renate Sitte: "Interactive Teaching of Elementary Digital Logic Design with WinLogiLab". IEEE transaction on education. Vol 47, No 2. May 2004.

[4] BG Prusty, O Ho, and S Ho: Adaptive Tutorials using eLearning Platform for Solid Mechanics Course in Engineering. 20th Australasian Association for Engineering Education Conference University of Adelaide, 6-9 December 2009. ISBN 1 876346 59 0 © 2009.

[5] Yumei Kang, Gengye Chen, Yunhong Cheng, Lin Xu: Design and Realization of Network Teaching Platform Based on E-Learning. 2008 International Conference on Computer Science and Software Engineering. pp 218.

[6] Zhan Chen, Israel Koren: Techniques for Yield Enhancement of VLSI Adders. International Conference on Application-Specific Array Processors, IEEE 1995.

[7] Yuan Taur, Douglas A. Buchanan, Wei Chen, David J. Frank, Khalid E. Ismail, Shih-Hsien Lo, George A. Sai-Halasz, Raman G. Viswanathan, Hsing-Jen C. Wann, Shalom J. Wind, and Hon-Sum Wong: CMOS Scaling into the Nanometer Regime. Proceedings of the IEEE, vol. 85, NO. 4, APRIL 1997.

[8] Course: Digital Electronics. IISC Bangalore, India.

[9] Clark T. Merkel, Mechanical Engineering :A Matlab-Based Teaching Tool for Digital Logic; 2006.

[10] Bolton, W., "Mechatronics, Electronic Control Systems in Mechanical and Electrical Engineering", pub. by Addison Wesley Longman Limited.