Abstract

Everyday new technology which is faster, smaller and more complex than its predecessor is being developed. The increase in clock frequency to achieve greater speed and increase in number of transistors packed onto a chip to achieve complexity of a conventional system results in increased power consumption. Almost all the millions of gates used to perform logical operations in a conventional computer are irreversible. Reversible logic is gaining interest in the recent past due to its less heat dissipating characteristics. It has been proved that any Boolean function can be implemented using reversible gates. That is, every time a logical operation is performed some information about the input is erased or lost and is dissipated as heat.

Reversible logic has shown potential to have extensive applications in future emerging technologies such as quantum computing, optical computing, quantum dot cellular automata as well as ultra low power VLSI circuits, DNA computing to produce zero power dissipation under ideal conditions. Reversible logic is very essential for the construction of low power, low loss computational structures which are very essential for the construction of arithmetic circuits used in quantum computation, nano technology and other low power digital circuits. Recently, several researchers have focused their efforts on the design and synthesis of efficient reversible logic circuits. The important reversible gates used for reversible logic synthesis are Feynman Gate, New Gate and Fredkin gate.

Reversible implementations are also found in thermodynamics and adiabatic CMOS. Power dissipation in modern technologies is an important issue, and overheating is a serious concern for both manufacturer (impossibility of introducing new, smaller scale technologies, limited temperature range for operating the product) and customer (power supply, which is especially important for mobile systems).

This paper presents the combinational circuit of all basic reversible logic gates and also have done VHDL CODE of these circuits. Every combinational circuit of all basic reversible logic gates can be verified through simulations using VHDL and Verilog HDL.

Keywords: Reversible Logic, Reversible Gate, Garbage output, VHDL Code.

1. Introduction:

Everyday new technology which is faster, smaller and more complex than its predecessor is being developed. The increase in clock frequency to achieve greater speed and increase in number of transistors packed onto a chip to achieve complexity of a conventional system results in increased power consumption. Almost all the millions of gates used to perform logical operations in a conventional computer are irreversible. That is, every time a logical operation is performed some information about the input is erased or lost and is dissipated as heat.

In digital design energy loss is considered as an important performance parameter. Higher levels of integration and new fabrication processes have dramatically reduced the heat loss over the last decades. The power dissipation in a circuit can be reduced by the use of Reversible logic.

As per Landauer [1], for irreversible logic, each bit of information lost generates kTln2 Joules of heat energy, where k is Boltzmann’s constant and T is absolute temperature at which the computation is performed. For room temperature T, the amount of heat dissipated for one bit is small i.e. 2.9×10^-21 J. The current processors, first of all dissipate 500 times this amount of heat every time a bit is lost. Secondly, assuming every transistor out of more than 4×10^17 dissipates heat at the processor frequency of 2GHz, the figure becomes 4×10^19*kTln2 J/sec. Although, it is around 0.1W at 400 degree Kelvin.

Moore’s law predicts exponential growth of heat generated due to information loss which will be an intolerable amount in the next decade. This heat dissipation dramatically reduces the performance and lifetime of the circuits. The solution is to use revolutionary technology which enables extremely low power consumption and heat dissipation in computing.

Bennett [2] showed that if a computation is carried out in Reversible logic zero energy dissipation is possible, as the amount of energy dissipated in a system is directly related to the number of bits erased during computation. The design that does not result in information loss is irreversible. A set of reversible gates are needed to design reversible circuit. Several such gates are proposed over the past decades.

Reversible circuit/gate can generate unique output vector from each input vector, and vice versa, i.e., there is a one to one correspondence between the input and output vectors. Thus, the number of outputs in a reversible gate or circuit has the same as the number of inputs, and commonly used traditional NOT gate is the only reversible gate. More formally, a reversible logic gate is a k-input, k-output (denoted k|k) device that maps each possible input pattern into a unique output pattern. While constructing reversible circuits with the
help of reversible gates, some restrictions should be strictly maintained [3, 4]:

- Fan-out is not permitted.
- Loops are not permitted.

In reversible logic we have one more factor, which is more important than the number of gates used, namely the number of garbage outputs. The unutilized outputs from a reversible gate/circuit are called "garbage". Though every synthesis method engages them producing less number of garbage outputs, but sometimes garbage outputs are unavoidable. For example, a single output function of \( n \) variables will require at least \( n-1 \) garbage outputs, since the reversibility necessitates an equal number of outputs and inputs.

Reversible logic imposes many design constraints that need to be either ensured or optimized for implementing any particular Boolean functions.

**Firstly**, in reversible logic circuit the number of inputs must be equal to the number of outputs.

**Secondly**, for each input pattern there must be a unique output pattern.

**Thirdly**, each output will be used only once, that is, no fan out is allowed.

**Finally**, the resulting circuit must be acyclic.

2. Reversible Logic Gates:

In this section, we describe all about reversible logic and reversible logic gates. Though it is already briefly described about garbage outputs, in this section we will define it with more appropriate examples and figures.

**Definition 2.1:** Garbage is the number of outputs added to make an \( n \)-input \( k \)-output Boolean function \((n, k)\) function) reversible. In other sense, a reversible logic gate has an equal number of inputs and outputs \((k=k)\) and all the outputs are not expected. Some of the outputs should be considered to make the circuit reversible and those unwanted outputs are known as garbage outputs. A heavy price is paid for every garbage outputs.

**Example 2.1:** If we want to find the Exclusive-OR between two variables in reversible computation, the circuit will look like Fig. 2.1.

One extra output should be produced to make the circuit reversible and that unwanted output \((P=A,\text{ marked as } *)\) is known as garbage. Now we will define some popular reversible gates where most of them will be used in our proposed design.

**Definition 2.2:** \(1*1 \text{ NOT gate}\) is the simplest among all the reversible gates where the gate has only one input \((A)\) and one output \((B)\) such that \(B = A'\)

![Figure 2.2: 1x1 NOT GATE](image)

**Definition 2.3:** Let \(I_v\) and \(O_v\) be the input and output vector of a \(2*2 \text{ Feynman gate (FG)\}[5,6]\) respectively, where \(I_v=(A,B)\) and \(O_v=(P=A, Q=A \oplus B)\). The block diagram for \(2*2 \text{ Feynman gate\) is shown in Fig.2.1.}

![Figure 2.4: 2x2 FEYNMAN GATE (CNOT GATE)](image)
Library ieee;
Use ieee std_logic.1164..all;
Entity feynmang is
Port(A, B: in std_logic;
P, Q: out std_logic);
end feynmang;
architecture ckt of feynmang is
begin
P<= A;
Q<= A xor B;
end ckt;

Definition 2.4: Let $Iv$ and $Ov$ be the input and output vector of a $3\times3$ Toffoli Gate [7,8] respectively, where $Iv=(A, B, C)$ and $Ov=(P=A, Q=B, R=AB \oplus C)$. Fig. 2.2 shows the $3\times3$ Toffoli gate.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{fig2.5}
\caption{3\times3 Toffoli Gate}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{fig2.6}
\caption{3\times3 Toffoli Gate}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{fig2.7}
\caption{3\times3 Toffoli Gate}
\end{figure}

Definition 2.5: Let $Iv$ and $Ov$ be the input and output vector of a $3\times3$ Fredkin Gate [7,9] respectively, where $Iv=(A, B, C)$ and $Ov=(P=A, Q=A'B \oplus AC, R=A'C \oplus AB)$. Fig. 2.3 shows the block diagram of $3\times3$ Fredkin gate.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{fig2.8}
\caption{3\times3 Fredkin Gate}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{fig2.9}
\caption{3\times3 Fredkin Gate}
\end{figure}

Definition 2.6: A $3\times3$ New Gate (NG) [10] can be defined as $Iv=(A, B, C)$ and $Ov=(P=A, Q=AB \oplus C, R=A'C \oplus B')$, where $I$, and $O$, are the input and output...
vector respectively. The block diagram of a 3*3 NEW GATE is shown in Fig. 2.10.

![Figure 2.10: 3*3 NEW GATE.](image)

Figure 2.11: combinational circuit diagram of 3x3 NEW GATE

Library ieee;
Use ieee std_logic.1164.all;
Entity newg is
Port(A, B, C : in std_logic;
P, Q, R : out std_logic);
end newg;
architecture ckt of newg is
signal Abar, Bbar, Cbar, S1, S2 : std_logic;
begin
P<= A;
Abar<= not A;
Bbar<= not B;
Cbar<= not C;
S1<= A and B;
S2<= Abar and Cbar;
Q<= S1 xor C;
R<= S2 xor Bbar;
End ckt;

Definition 2.7: 3*3 Peres Gate[11] is another important gate which has a low quantum cost as compared to other gates. A single Peres gate can give generate and propagate outputs when the third input C = 0. Two Peres gates can be combined to form a full adder.

![Figure 2.12: 3X3 PERES GATE](image)

Figure 2.13: combinational circuit diagram of 3x3 PERES GATE

Library ieee;
Use ieee std_logic.1164.all;
Entity peresg is
Port(A, B, C : in std_logic;
P, Q, R : out std_logic);
end peresg;
architecture ckt of peresg is
signal S1: std_logic;
begin
P<= A;
P1<= A;
P2<= B;
P3<= C;
P4<= A and B;
P5<= A and C;
P6<= B and C;
P7<= A xor B;
P8<= A xor C;
P9<= B xor C;
P10<= S1 xor P4;
P11<= S2 xor P6;
P12<= S3 xor P8;
P13<= S4 xor P10;
P14<= S5 xor P12;
P15<= S6 xor P14;
P16<= S7 xor P16;
P17<= S8 xor P18;
P18<= S9 xor P20;
P19<= S10 xor P22;
P20<= S11 xor P24;
P21<= S12 xor P26;
P22<= S13 xor P28;
P23<= S14 xor P30;
P24<= S15 xor P32;
P25<= S16 xor P34;
P26<= S17 xor P36;
P27<= S18 xor P38;
P28<= S19 xor P40;
P29<= S20 xor P42;
P30<= S21 xor P44;
P31<= S22 xor P46;
P32<= S23 xor P48;
P33<= S24 xor P50;
P34<= S25 xor P52;
P35<= S26 xor P54;
P36<= S27 xor P56;
P37<= S28 xor P58;
P38<= S29 xor P60;
P39<= S30 xor P62;
P40<= S31 xor P64;
P41<= S32 xor P66;
P42<= S33 xor P68;
P43<= S34 xor P70;
P44<= S35 xor P72;
P45<= S36 xor P74;
P46<= S37 xor P76;
P47<= S38 xor P78;
P48<= S39 xor P80;
P49<= S40 xor P82;
P50<= S41 xor P84;
P51<= S42 xor P86;
P52<= S43 xor P88;
P53<= S44 xor P90;
P54<= S45 xor P92;
P55<= S46 xor P94;
P56<= S47 xor P96;
P57<= S48 xor P98;
P58<= S49 xor P100;
P59<= S50 xor P102;
P60<= S51 xor P104;
P61<= S52 xor P106;
P62<= S53 xor P108;
P63<= S54 xor P110;
P64<= S55 xor P112;
P65<= S56 xor P114;
P66<= S57 xor P116;
P67<= S58 xor P118;
P68<= S59 xor P120;
P69<= S60 xor P122;
P70<= S61 xor P124;
P71<= S62 xor P126;
P72<= S63 xor P128;
P73<= S64 xor P130;
P74<= S65 xor P132;
P75<= S66 xor P134;
P76<= S67 xor P136;
P77<= S68 xor P138;
P78<= S69 xor P140;
P79<= S70 xor P142;
P80<= S71 xor P144;
P81<= S72 xor P146;
P82<= S73 xor P148;
P83<= S74 xor P150;
P84<= S75 xor P152;
P85<= S76 xor P154;
P86<= S77 xor P156;
P87<= S78 xor P158;
P88<= S79 xor P160;
P89<= S80 xor P162;
P90<= S81 xor P164;
P91<= S82 xor P166;
P92<= S83 xor P168;
P93<= S84 xor P170;
P94<= S85 xor P172;
P95<= S86 xor P174;
P96<= S87 xor P176;
P97<= S88 xor P178;
P98<= S89 xor P180;
P99<= S90 xor P182;
P100)<= S91 xor P184;
P101)<= S92 xor P186;
P102)<= S93 xor P188;
P103)<= S94 xor P190;
P104)<= S95 xor P192;
P105)<= S96 xor P194;
P106)<= S97 xor P196;
P107)<= S98 xor P198;
P108)<= S99 xor P200;
P109)<= S100 xor P202;
P110)<= S101 xor P204;
P111)<= S102 xor P206;
P112)<= S103 xor P208;
P113)<= S104 xor P210;
P114)<= S105 xor P212;
P115)<= S106 xor P214;
P116)<= S107 xor P216;
P117)<= S108 xor P218;
P118)<= S109 xor P220;
P119)<= S110 xor P222;
P120)<= S111 xor P224;
P121)<= S112 xor P226;
P122)<= S113 xor P228;
P123)<= S114 xor P230;
P124)<= S115 xor P232;
P125)<= S116 xor P234;
P126)<= S117 xor P236;
P127)<= S118 xor P238;
P128)<= S119 xor P240;
P129)<= S120 xor P242;
P130)<= S121 xor P244;
P131)<= S122 xor P246;
P132)<= S123 xor P248;
P133)<= S124 xor P250;
P134)<= S125 xor P252;
P135)<= S126 xor P254;
P136)<= S127 xor P256;
P137)<= S128 xor P258;
P138)<= S129 xor P260;
P139)<= S130 xor P262;
P140)<= S131 xor P264;
P141)<= S132 xor P266;
P142)<= S133 xor P268;
P143)<= S134 xor P270;
P144)<= S135 xor P272;
P145)<= S136 xor P274;
P146)<= S137 xor P276;
P147)<= S138 xor P278;
P148)<= S139 xor P280;
P149)<= S140 xor P282;
P150)<= S141 xor P284;
P151)<= S142 xor P286;
P152)<= S143 xor P288;
P153)<= S144 xor P290;
P154)<= S145 xor P292;
P155)<= S146 xor P294;
P156)<= S147 xor P296;
P157)<= S148 xor P298;
P158)<=
Library ieee;
Use ieee std_logic.1164.all;
Entity trg is
  Port(A, B, C : in std_logic;
       P, Q, R : out std_logic);
end trg;
architecture ckt of trg is
signal Bbar, S1: std_logic;
beg
  P<= A;
  Bbar<= not B;
  S1<=A and Bbar;
  Q<= A xor B;
  R<= S1 xor C;
end ckt;

3. Advanced Reversible Logic Gates:
In this section, we describe all about reversible logic and advanced reversible logic gates. Though it is briefly described about garbage outputs and these advanced reversible logic gates can singly work as FULL ADDER, FULL SUBTRACTOR and more combinational circuits. In this section we will define it with more appropriate examples and figures.

Definition 3.1: 4*4 DPG GATE[12,13,14] is a combination of two 3x3 PERES GATE. As we know that the Peres Gate has a low quantum cost as compared to other gates, so the combination of it provide more useful Double Peres Gate. This gate can singly work as many combinational circuits as FULL ADDER & FULL SUBTRACTOR. This gate requires only one clock cycle and produces no extra garbage outputs, that is, it adheres to the theoretical minimum as established.

![Figure 3.1: 4*4 DPG GATE](image)

Figure 3.2: combinational circuit diagram of 4x4 DPG GATE

Library ieee;
Use ieee std_logic.1164.all;
Entity dpgg is
  Port(A, B, C, D : in std_logic;
       P, Q, R, S : out std_logic);
end dpgg;
architecture ckt of dpgg is
signal S1, S2: std_logic;
beg
  P<= A;
  Q<= A xor B;
  R<=(A xor B) xor D;
  S1<=Q and D;
  S2<=A and B;
  S<=S1 xor S2 xor C;
end ckt;

Definition 3.2: 4*4 DPG GATE[15] can also singly work as many combinational circuits as FULL ADDER & FULL SUBTRACTOR. This gate requires only one clock cycle and produces no extra garbage outputs, that is, it adheres to the theoretical minimum as established.

![Figure 3.3: 4*4 DKG GATE](image)

Figure 3.4: combinational circuit diagram of 4x4 DKG GATE

Library ieee;
Use ieee std_logic.1164.all;
Entity dkgg is
  Port(A, B, C, D : in std_logic;
       P, Q, R, S : out std_logic);
end dkgg;
architecture ckt of dkgg is
signal Abar, Dbar, S1, S2, S3, S4, S5, S6: std_logic;
beg
  Abar<= not A;
  Dbar<= not D;
  P<= B;
  S1<= Abar and C;

```
S2<= A and Dbar;
Q<= S1 or S2 ;
S3<= A xor B;
S4<= C xor D;
S5<=S3 and S4;
S6<=C and D;
R<= S5 xor S6 ;
S<=B xor S4;
End ckt;

4. PROBLEMS WITH REVERSIBILITY:

- However, in order to attain the supposed benefits of reversible computation, the reversible machine must actually be run backwards to attain its original state. If this step is not taken, then typically the machine becomes clogged up with digital heat i.e. entropy, and is thus rendered unable to perform further useful work.

- Another problem is that you must make sure your computation was performed with no errors - otherwise chaos (and not the original starting condition) may result when the machine is run backwards.

- Without running a reversible machine backwards, the main benefits of using reversible logic - namely dramatically reduced heat dissipation and power consumption - cannot be realized in a sustainable fashion.

So: is reversible logic a waste of time?

No. Reversible logic is of substantial significance.

5. BENEFITS OF REVERSIBILITY:

Reversible logic allows in the digital domain.

1) Power Management
2) Heat Management

What do digital power management and digital heat management even mean?

- Digital power refers to ordered bit patterns, which can be used to do digital work.
- Digital heat refers to disordered bit patterns that are no good to anyone.
- Management of digital power involves moving it to where it is needed.
- Management of digital heat involves moving it to where it can be dumped.

6. APPLICATION:

The potential application areas of reversible computing include the following [16,17,18]

- Nanocomputing
- Bio Molecular Computations
- Laptop/Handheld/Wearable Computers
- Spacecraft
- Implanted Medical Devices
- Wallet “smart cards”
- “Smart tags” on inventory

7. CONCLUSION:

This paper presents VHDL CODE of all Reversible Logic Gate, which provide us to design VHDL CODE of any complex combinational circuit. Here we have tried to make the VHDL code as much as possible. We can simulate and synthesis it using Xilinx software and also calculate the power consumption and compare it with the irreversible Full Adder.

REFERENCES


Biographies:

Devendra Goyal (devagoyal87@gmail.com, devagoyal@yahoo.co.in) has received his B.E. Degree in 2009 Electronics & Communication Engineering from MIT, KOTA & now pursuing M. Tech in VLSI DESIGN from Poornima College Of Engineering Jaipur. He has presented a paper In International Conference held in Janardan Rai Nagar Rajasthan Vidyapeeth (D) University Udaipur, April 2012 on “4-BIT REVERSILE FULLADDER USING DKG GATE IMPLEMENTATION IN VHDL.” And a paper presented in National Conference held in MITRC Alwar, April 2012 on “Reversible Full Adder Gate using Nano-Technology” His current area of research includes REVERSIBLE TECHNOLOGY and currently working on the project related to it.

Vidhi Sharma (sharmavidhif7@gmail.com) has received his B.E. Degree in 2009 Electronics & Communication Engineering from R.N.MODI, KOTA & now pursuing M. Tech in Electronics & Communication Engineering from Rajasthan College Of Engineering for Women, Jaipur. She has presented a paper In International Conference held in Janardan Rai Nagar Rajasthan Vidyapeeth (D) University Udaipur, April 2012 on “4-BIT REVERSILE FULLADDER USING DKG GATE IMPLEMENTATION IN VHDL.” And a paper presented in National Conference held in MITRC Alwar, April 2012 on “Reversible Full Adder Gate using Nano-Technology”. Her current area of research includes REVERSIBLE TECHNOLOGY & Data Encryption and currently working on the project related to it.