International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-3 Issue-3, August 2013

Arbiter Design Using Verilog for Switching to Communicate in Between Multiple Resources

Tarun Kumar Gauttam, Rekha Agrawal, Sandhya Sharma

Abstract*— This project attempts to describe a special type of circuit called an arbiter to be used in a larger design called switch to communicate in between multiple resources. The design specification gives results according to suggested implementation for the circuit. Finally, possibilities for addition, revision and testing structure for an integrated circuit implementation of the arbiter will be proposed. The main contribution of this paper is the design and optimization of asynchronous arbiter circuit using CMOS, Bi-cmos and synchronous fast round- robin arbiters and the design of On-Chip Scheduler. Scheduler is expressed here in verilog RTL and simulation results are presented to indicate the performance. This paper will present design ideas for effectively interfacing to an arbiter and investigate coding styles for some common arbitration schemes. When Circuits need to be constructed out of several self-timed parts, the arbitration is often required for the asynchronous design. We consider the creation of the general purpose arbiter delegating M resources to N clients. Firstly, the task is done for the case of one to three resources being offered to one to three clients and preserving capability to allow one to all clients accessing resources simultaneously using verilog.

Index Terms*— Complementary metal oxide semiconductor (CMOS).Register transistor logic (RTL).

I. INTRODUCTION

The arbiters are an important piece of the scheduler design in which Grant and request signals are identically designed with the exception of the rules determining when the Priority State may be updated.

Many systems exist in which a large number of requesters must access a common resource. The common resource may be a shared memory, a networking switch fabric, a specialized state machine, or a complex computational element. An arbiter is required to share the resources among the many requesters. When putting an arbiter into a design, many factors must be considered. The interface between the requesters and the arbiter must be appropriate for the size and speed of the arbiter. Also, the coding style used will usually impact the synthesis results. The arbiter keeps away the system from metastability state of non arbitery system, when large number of request is sending from different clients to communicate with large number of resources. It works on three process request, grant and Accept.

Manuscript published on 30 August 2013.
***Correspondence Author(s)**
Tarun Kumar Gauttam, Electronic and Communication Department, Suresh Gyan Vihar University,Jaipur,India.
Rekha Agarwal,Electronic and Communication Department, Suresh Gyan ViharUniversity,Jaipur,India.
Prof. Shandhya Sharma, Electronic and Communication Department, Suresh Gyan Vihar University, Jaipur,India.

A. *Step1 Request*

Each unmatched input sends a request to every output which for which it has a queued cell.

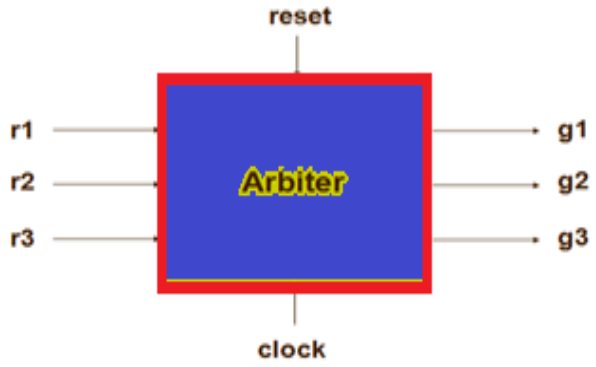

Fig.1: Block diagram synchronous arbiter.

B. *Step2 Grant*

In an unmatched output receives any requests, it chooses the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer to the highest Priority element of the round-robin schedule is incremented (modulo N) to one location Beyond the granted input if the grant is accepted in Step 3 of the first iteration.

C. *Step 3 Accept*

If an unmatched input receives a grant, it accepts the one that appears next in a fixed, round-robin schedule starting from the highest priority element.

II. ASSUMPTION FOR ARBITER DESIGN

There are three independent requests say r1, r2, r3. In the project three inputs is given as in detailed when the input is given through the r1, r2, r3 then after the processing it shows result in form of g1, g2, g3.

It is assumed that the priority in the order r1>r2> r3.In this project the highest priority among the input's is r1, then r2 and lowest priority is r3 . here the highest priority work given to highest priority input r1,then according to work .when the high priority work is given to the high priority input then when request is sent to the high priority input then acknowledge is send immediately and process on work start on the input. Now if input is given to r2 then first r1 input access then after complete the task then it moves to r2 or respond after the r1 execution. Duration of access time (Timeout period) is programmed through the independent processor and the data bus. Duration of execution or responding the programme access time (time out period) play important role for executing the input signals.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

If any signal is given to the input according to the priority but arbiter does not respond in given time then whole process repeat again in certain time period and same acknowledge is given by certain time and grant signal generate in timeout period then process further start, otherwise further process become stop and process of acknowledge and grant repeat again and again. Arbiter at any instant of time is in one of the following states:

A) Grant_g1
B) Grant_ g2
C) Grant_ g3
D) Idle.

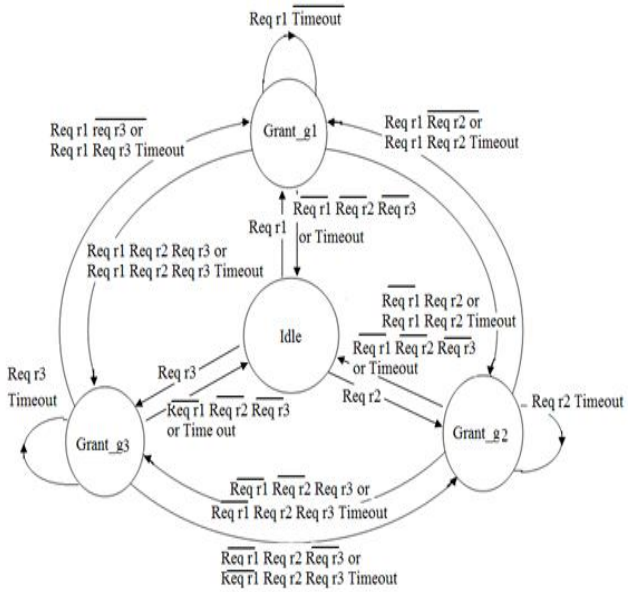

Fig.2: State diagram of synchronous arbiter.

III. DESIGN OF SYNCHRONOUS ARBITER USING VERILOG

In this chapter we discuss about to design the arbiter using verilog this chapter consist two parts in first part we implement the arbiter using Xilinx 9.2 ise tool using verilog in other remaining part we implement asynchronous arbiter circuit and their results using orcad 16.2 tool.

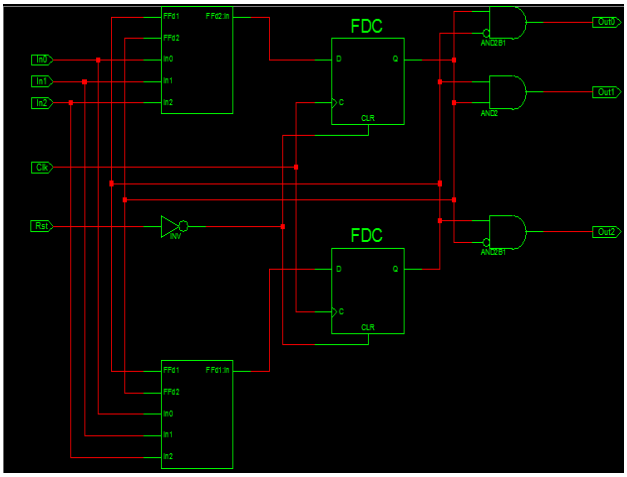

Fig. 3: RTL Schematic view of arbiter.

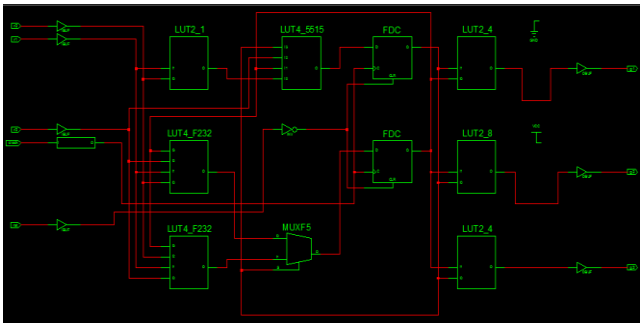

Fig. 4: Technology schematic view of arbiter.

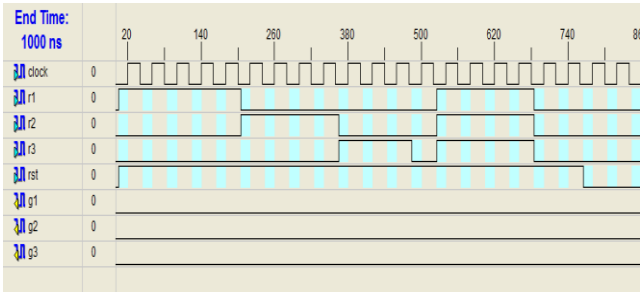

Fig. 5: Input to the arbiter built by verilog.

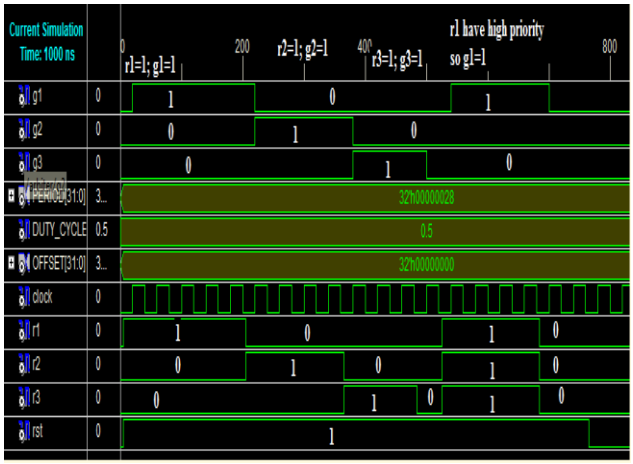

Fig. 6: Output of arbiter after simulation of verilog code.

IV. CIRCUIT IMPLEMENT OF ASYNCHRONOUS ARBITER CIRCUIT

A. *Circuit of arbiter*

Circuit of arbiter is shown below

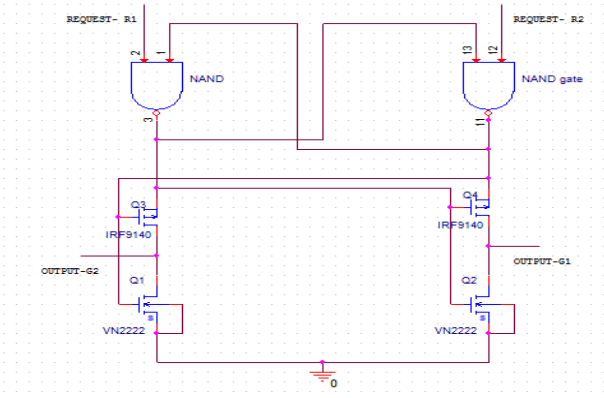

Fig. 7: Two bit input arbiter circuit

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

This circuit build using orcad16.5 tool. Above circuit shown of arbiter circuit in which two nand gate use for built and request is given to this circuit as form R1 and R2. This circuit works only two input (R1=0,and R2 =1) and (R1=1,and R2= 0) and gives result in form of G1 and G2. For the condition of R1=0 and R2=1 it gives G2 =1 it means when request (R2) is high than grant (G2=1) will be high and for condition of R2=0 and R1=1 it gives output as G1=1,it means when request (R1) is high than grant (G1=1)will be high . In this circuit (R1=0 and R2=0) or (R1=1 and R2=1) are two illegal state.

Table 1: Truth table of two Bit input asynchronous arbiter.

INPUT		OUTPUT	
Request (R1)	**Request (R2)**	**Grant (G1)**	**Grant(G2)**
0	0	illegal input	
0	1	0	1
1	0	1	0
1	1	illegal input	

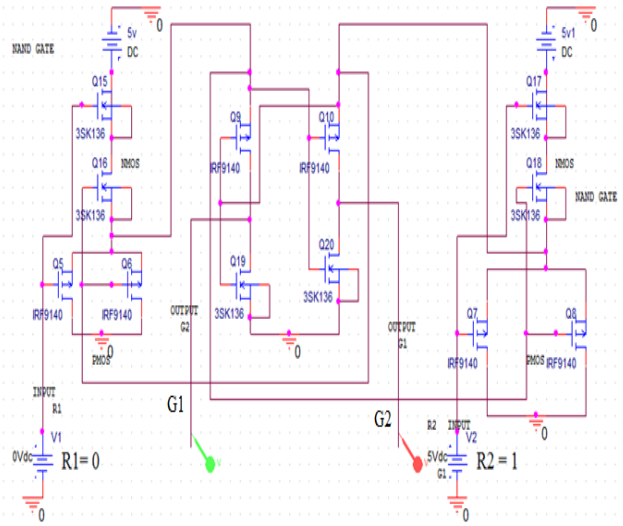

Fig. 8: Asynchronous arbiter circuit two bit request using cmos.

For avoiding these (R1=1, and R2=1) and (R1=0 and R2=0) situation use one bit input arbiter using inverter circuit . Circuit of one_bit operated arbiter is given below Fig.[9].

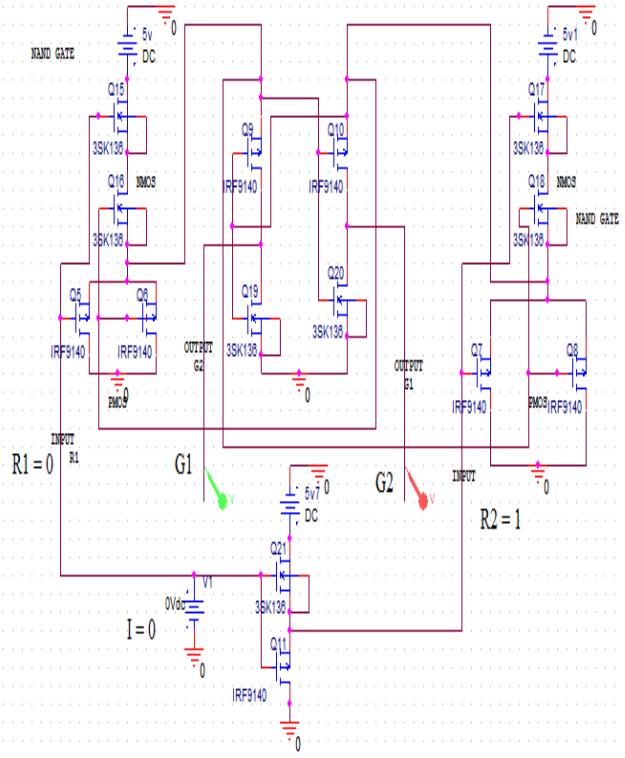

Fig. 9: One bit operate arbiter using CMOS.

In this circuit it works only in two condition (R1=1,R2=0) and (R1=0,R2=1) after using as inverter circuit in two_bit operate arbiter or asynchronous circuit.

B. Circuit Simulation and Results of two bit operate arbiter using cmos.

A. For (R1=0, R2=1)

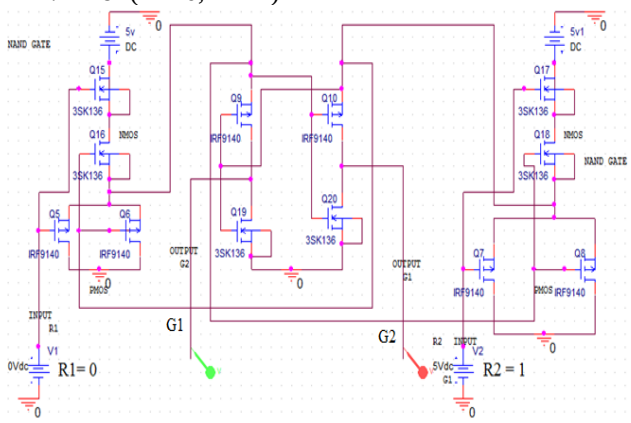

Fig. 10: Input (r1=0,r2=1) to arbiter using CMOS.

RESULT

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

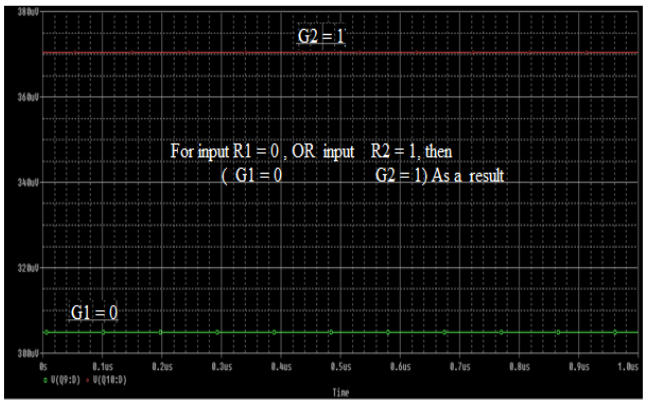

Fig. 11: Simulation result arbiter using CMOS.

B. For (R1 = 1 and R2 = 0)

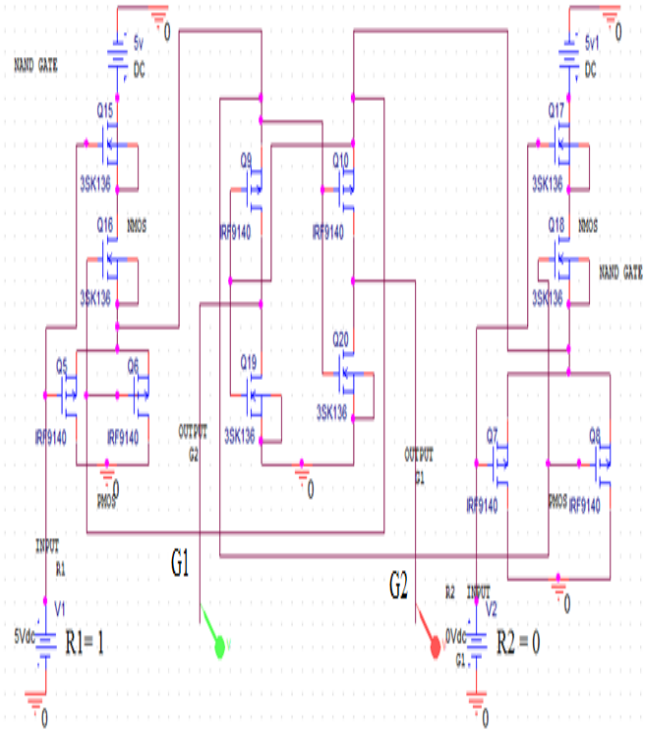

Fig. 12: Inputs (R1=1, R2=0) arbiter using CMOS.

RESULT

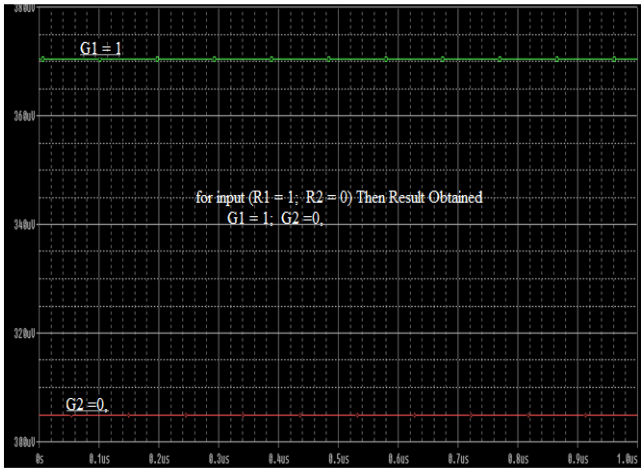

Fig. 13: Result after simulation of circuit of arbiter using CMOS.

C. For Input (R1 = 0, R2 = 0)

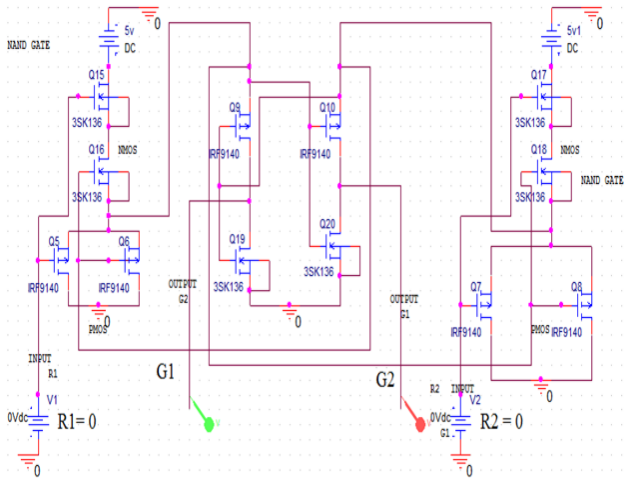

Fig. 14: Input (R1=0, R2=0) to arbiter using CMOS.

RESULT (Avoid this condition)

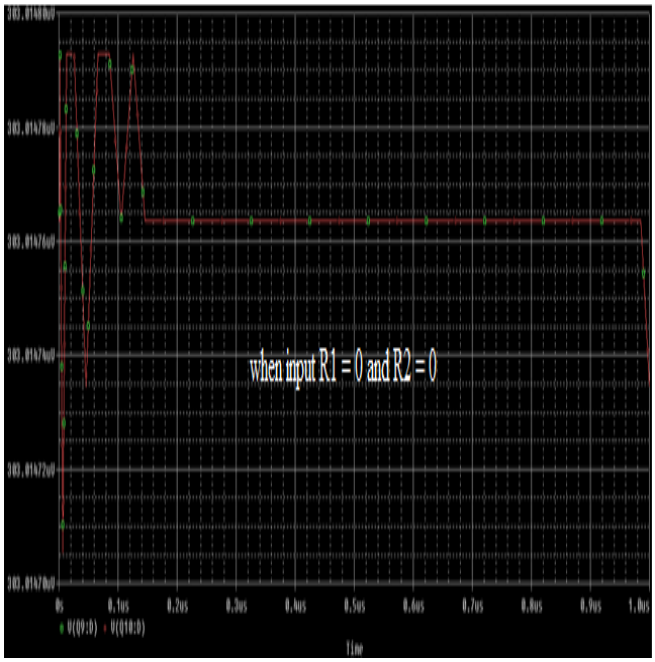

Fig. 15: Result after simulation of arbiter using CMOS.

D. For Input (R1 = 1, R2 = 1)

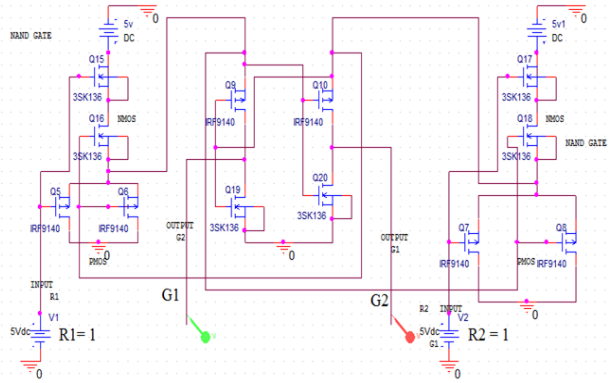

Fig. 16: Input (R1=1, R2=1) of arbiter using CMOS.

RESULT (Avoid this condition)

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

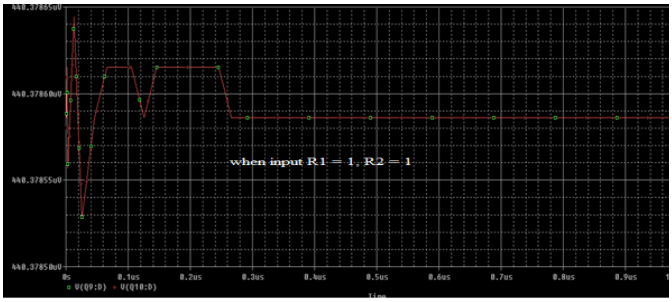

Fig. 17: Result after simulation of arbiter circuit using CMOS.

C. Circuit simulation and Result one_bit operate arbiter using CMOS

Here to avoid the condition of result as shown in Fig.15, Fig.17 then use the one bit arbiter circuit as shown below(Fig.18)

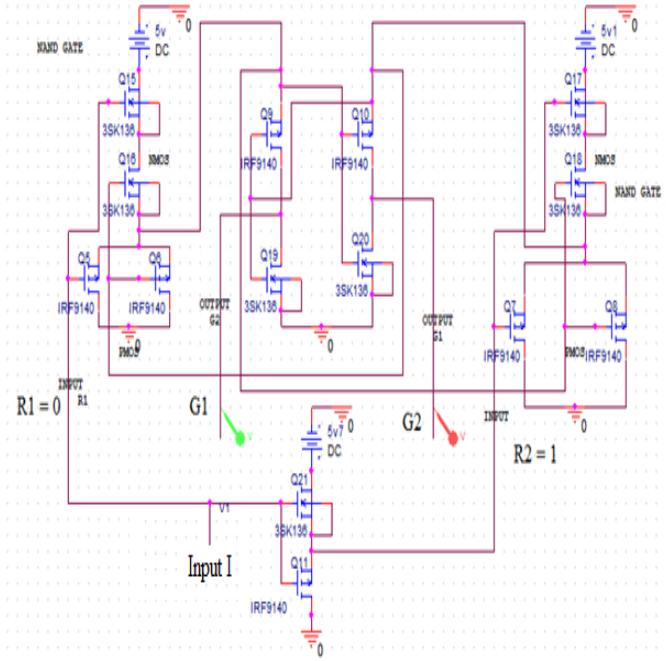

Fig. 18: circuit diagram one bit operate arbiter using CMOS.

This circuit is made by CMOS using inverter circuit. This circuit is made on orcad 1.6 tool to remove unwanted responses of (R1=1, R2=1 or R1=0, R2=0) .It can be understand by table which is given below.

Table 2: Truth table of one bit arbiter using cmos.

INPUT			OUTPUT	
INPUT I	**Request (R1)**	**Request (R2)**	**Grant (G1)**	**Grant(G2)**
0	0	1	0	1
1	1	0	1	0

A. For Input (I=0)

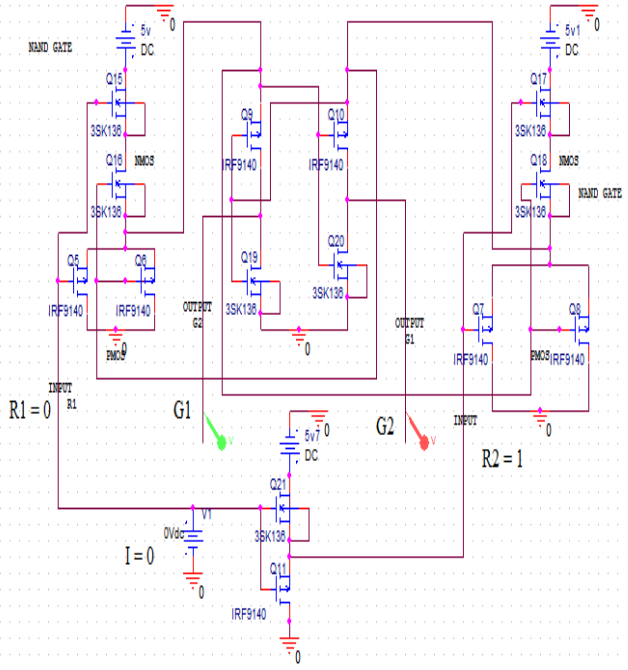

Fig. 19: Input (I = 0) one bit operate arbiter using CMOS.

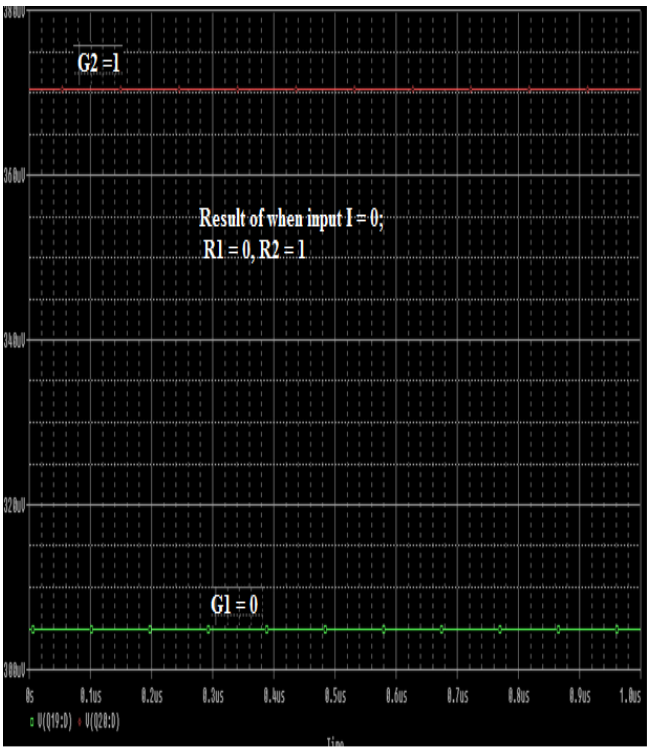

Fig. 20: Result after simulation (I = 0) one_bit operate arbiter using CMOS.

B. For input (I = 1)

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

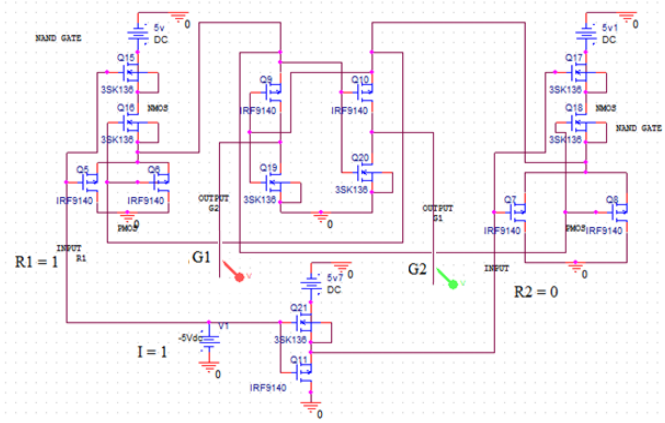

Fig. 21: input (I = 1) one_bit operate arbiter using cmos.

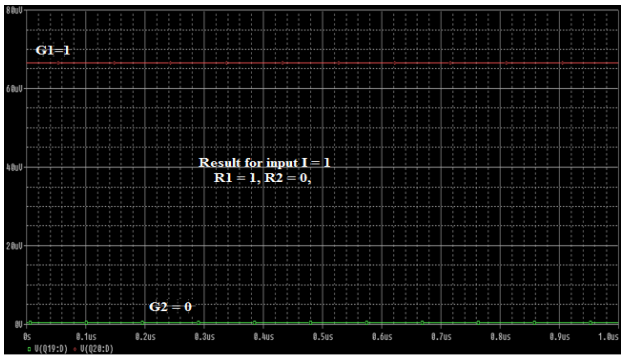

Fig. 21: result after simulation (I = 1) one bit operate arbiter using CMOS.

D. Circuit simulation and Result one_bit operate arbiter using Bi-cmos.

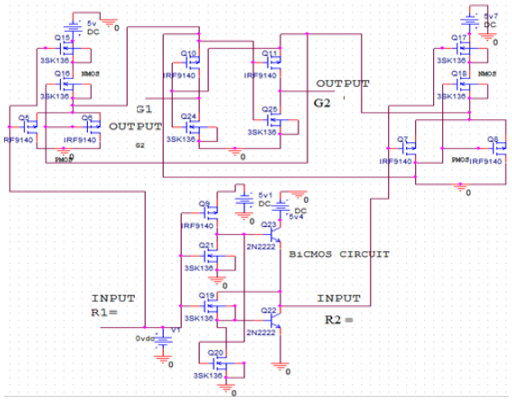

Fig. 22: one bit arbiter controller circuit using bi-cmos .

A. Circuit when input applied (I=0)

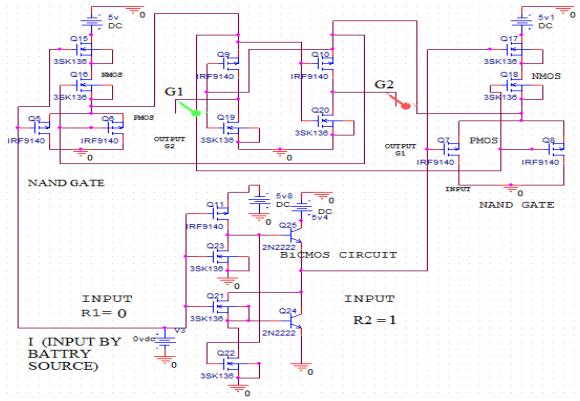

Fig. 23: Input applied to Bi-cmos arbiter circuit when input is applied (I=0).

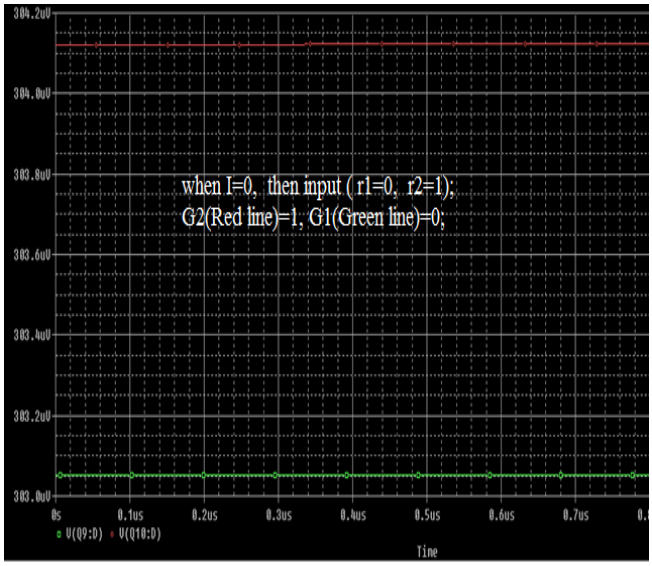

Fig. 24: Result of Bi-cmos arbiter circuit when input is applied (I=0).

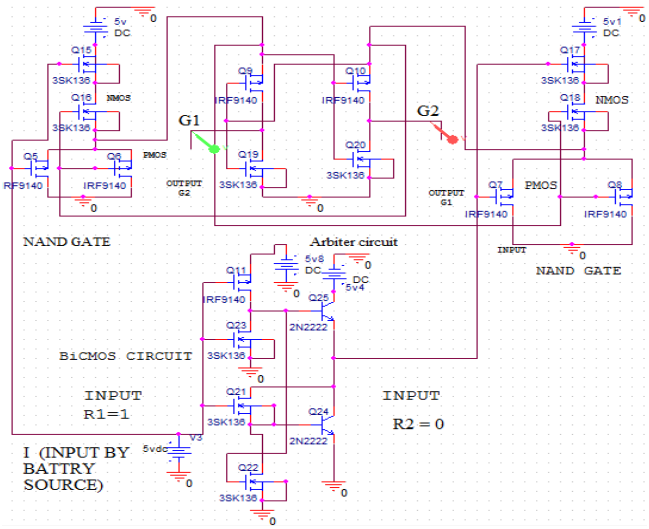

Fig. 25: Input applied to Bi-cmos arbiter circuit when input is applied (I=1)

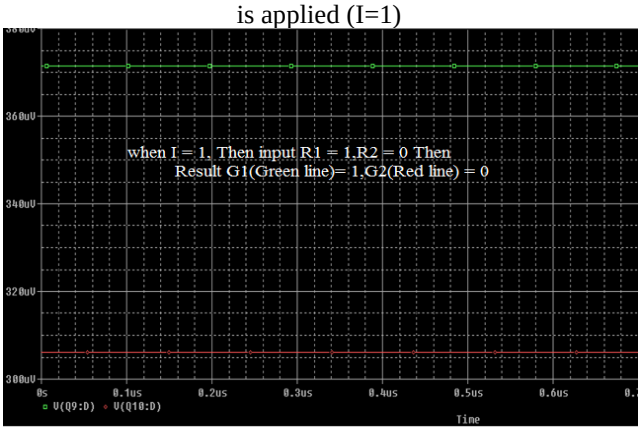

Fig. 26: Result of Bi-cmos arbiter circuit when input is applied (I=0).

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

V. APPLICATION OF ARBITER FOR ACCESS MULTISOURCE USING VERILOG.

An arbiter has a capability to access one or several resource in between many clients. The communication with an arbiter is do --ne by two way communication channels. A channels is typically formed of the request or (grant) acknowledgement pair of signals All clients accessing the same resource can be completely asyn- chronous and unrelated to each other, this means that request can arrivals at any time, possibly simultaneously which could create the metastability problem in non-arbiter circuit. There are a num -ber of arbitration phenomena as (a) many to one (b) many to many (c) one to many; these phenomena can be found at all computing system. This phenomena can be understand these given Fig.27 below

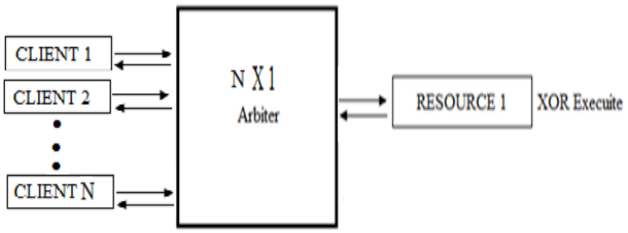

Fig. 27: Arbitration phenomena of many (client) to one (resource)

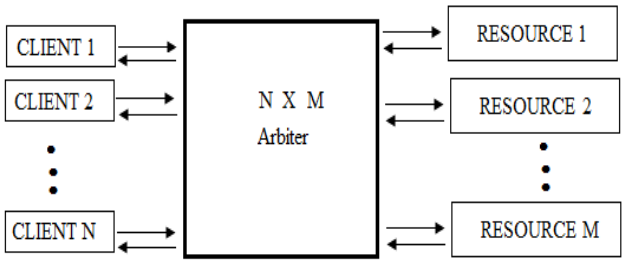

Fig. 28: Arbitration phenomena of many (client) to many (resources).

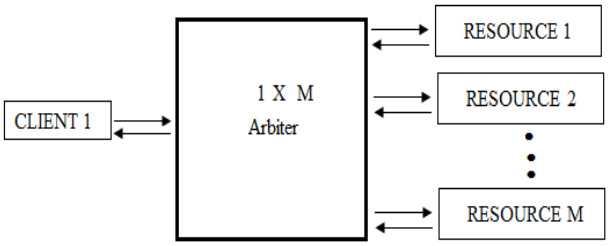

Fig. 28: Arbitration phenomena of one client to many resources.

A. *Design of arbiter using verilog*

Here for design some assumption is that resource 1 act as a xor gate execution , and resource 2 act as a and gate execution, and resource 3 act as a xnor gate means when resource 1 is access by any client of n then xor operation will execute ,similar when resource 2 access by any client of n then and gate is execute, and last resource 3 execute then xnor operation will execute. Here arbiter design for one client access three(3) resources using many client to one resource phenomena.

A. Arbitration phenomena of one resource to many clients

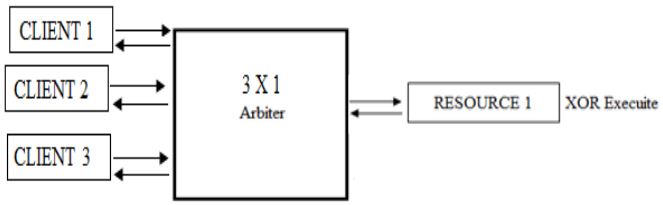

Fig. 29: block diagram of arbitration phenomena of one resource to many(3) client.

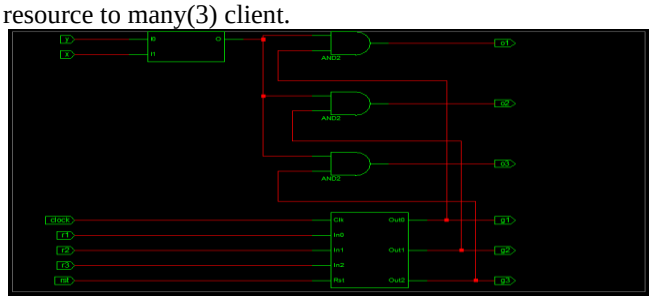

Fig. 30:RTL Schematic view of one resource to many client phenomena arbiter.

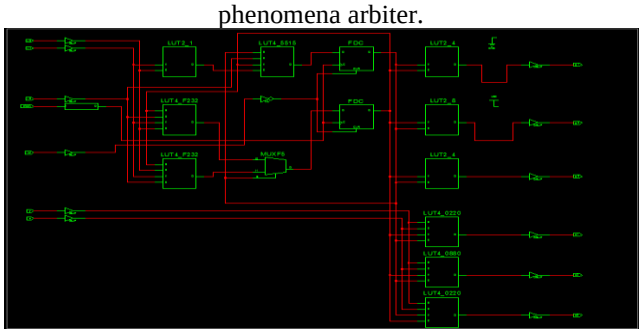

Fig. 31: Technology Schematic view of one to many resources arbiter phenomena.

Inputs of one to many resources arbiter

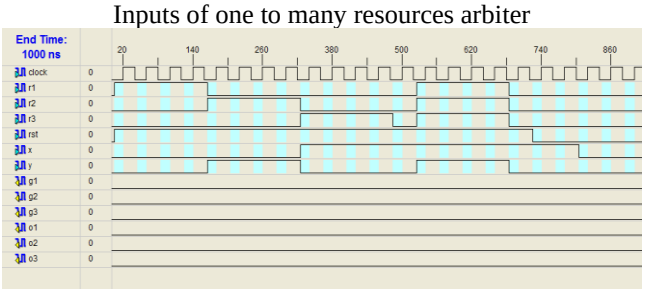

Fig. 32: Input of arbitration phenomena of one resources to many client.

Result of one resource to many client resources

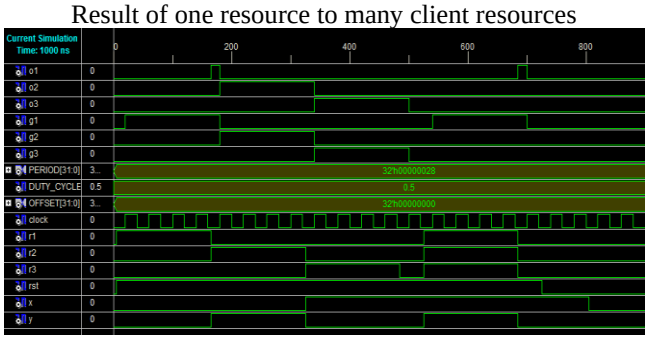

Fig. 33:Result of the arbiter for many client to one resource phenomena arbiter.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

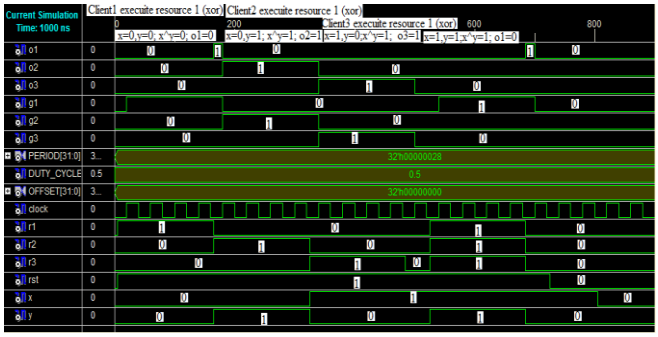

Fig. 34: explanation of result after simulation of one resource to many clients.

A. Arbiter design for one to many resource phenomena

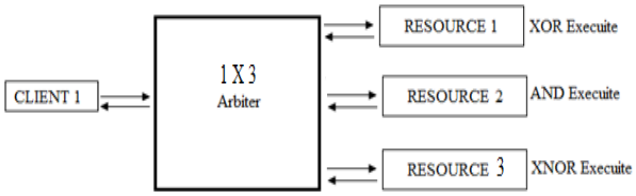

Fig. 35: block diagram of arbitration phenomena of one client to many resources(3).

Inputs of one to many resource phenomena

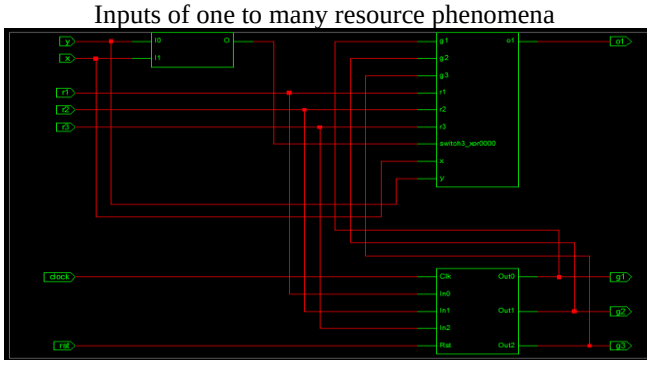

Fig. 36: RTL Schematic view of many resource to one client phenomena arbiter.

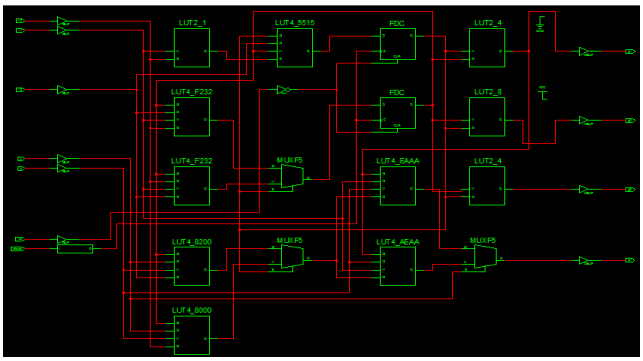

Fig. 37: Technology Schematic view of one client to many resources arbiter.

Inputs of the arbiter

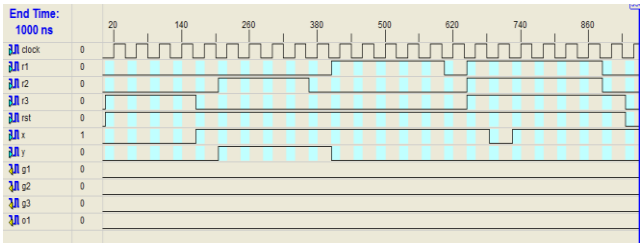

Fig. 38: Inputs of the one client to many resources (3) arbiter

Result of arbiter.

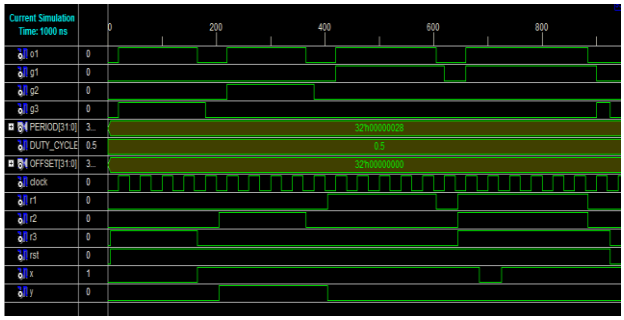

Fig. 39: Result of design of arbiter of one client to many resources phenomena.

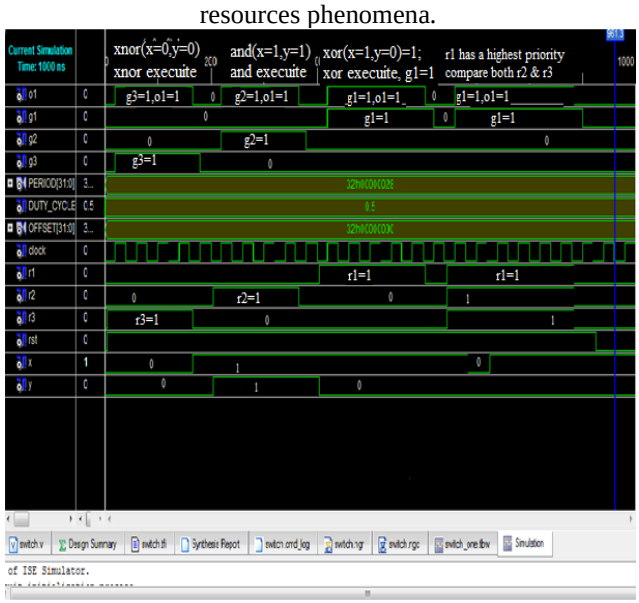

Fig. 40: Result of design of arbiter of one client to many (3) resources phenomena.

B. Arbiter design for many clients to many resource phenomena

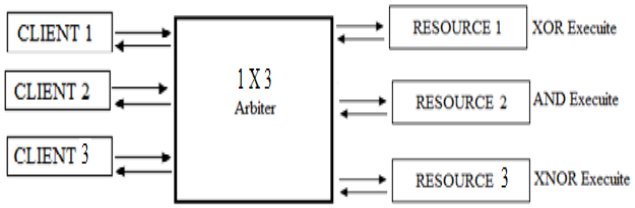

Fig. 41: Design of Arbiter of many client to many resource phenomena.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

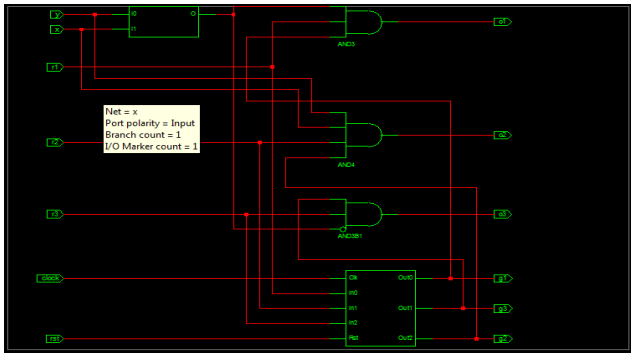

Fig. 42: RTL Schematic view of many resources to many client arbiter design.

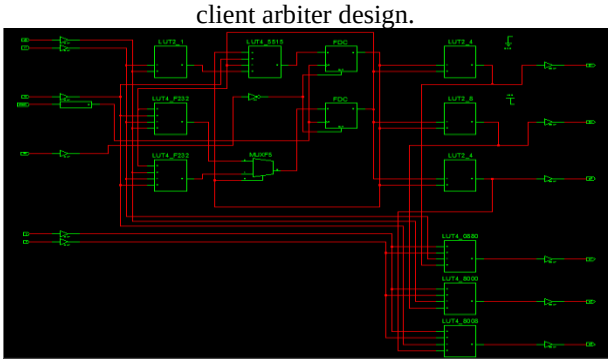

Fig. 43: Technology schematic view of many to many clients arbiter design.

Input of the arbiter (many to many resources)

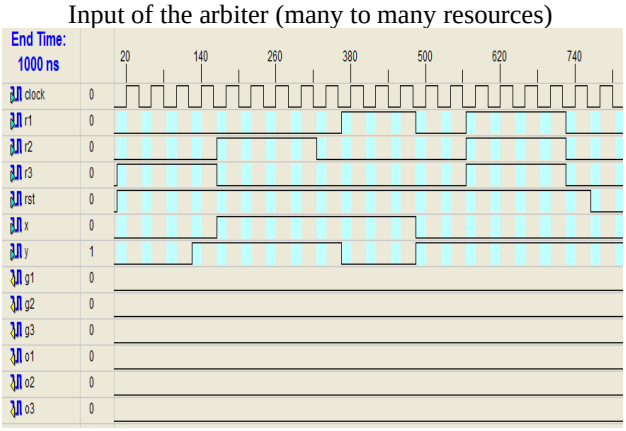

Fig. 44: Arbiters input for many client accesses many Resources.

Result of arbiter (many client to many resource)

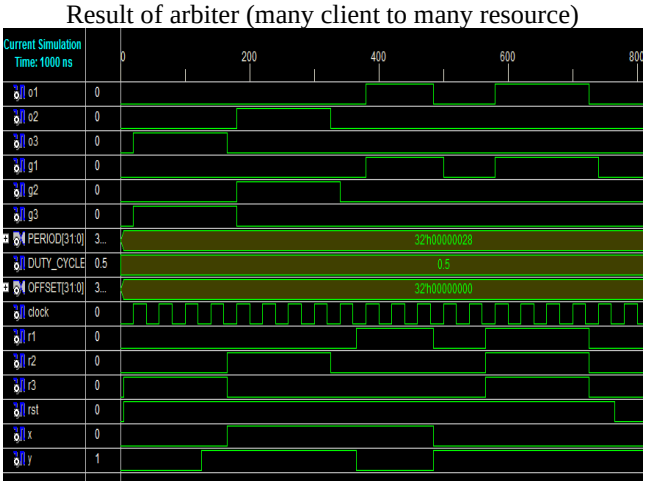

Fig. 45:Arbiter result for many client access many Resource.

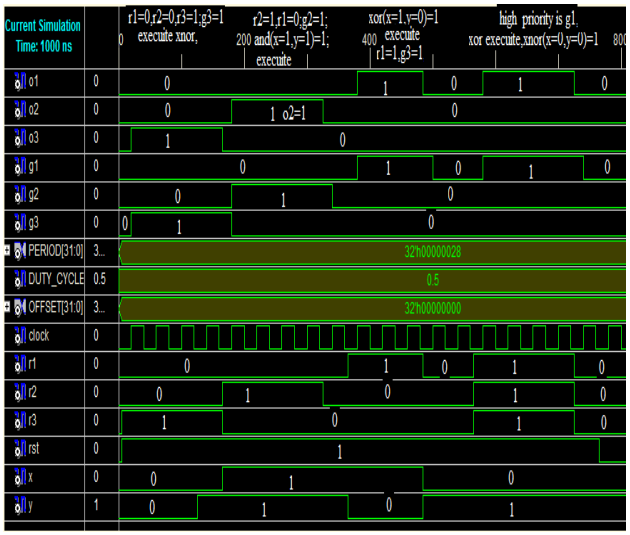

Fig. 46: Result shown after simulation according to many output to many resources.

VI. CONCLUSION

This paper presented several different methods for interfacing to an arbiter. Which method is the best one to use for a particular design depends on the size and speed of the chip that is being built .options were presented which cover a wide range of potential applications. Two common arbitration schemes were then investigated. For priority arbiters, it is shown that the primary consideration for coding style is ease of writing, reading, and maintaining the code. A simple three line implementation was shown to accomplish these goals for round-robin arbiters. In this paper we simulate the circuit with cmos and bi-cmos and obtained result of arbiter circuit according to design.This paper describes the design of an arbiter managing handshake between clients and resources. Each resource is actively reporting of its availability and can be connected to any of the clients. The work is part of a larger project that analyses the realization of a segmented bus, starting from high levels of abstraction down to implementation. This project is verified by Xilinx, model sim simulator and Orcad 16.5 Tool.

REFERENCES

1. Pankaj Gupta, “On the Design of Fast Arbiters”, Oct2, 1997.
2. Jonathan Chao, “Saturn: A Terabit packet Switch Using Dual Round-Robin”, IEEE Communications Magazine, vol. 38, no. 12 December 2000, pp78-84.
3. Arbiter: Style and Coding Style, Matt Weber Silicon Logic Engineering.inc.
4. Multi –Resources arbiter design; Stanislavs Golubcovs, Andrey, Mokhov, Yakovlev {Stanislavs, Golubcovs, Andrey, Mokhov, Alex. Yakovlev}@ncl.ac.uk.
5. Arbiter for an Asynchronous Counter flow Pipeline by James Copus.
6. Synopsys, inc., “Design ware Foundation Library Data book, Volume 2”, v2000.11, pp 111-126

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)