Design, Verification and Physical Designing of FCFS Arbiter

Saurin Shah[1] PankajChavda[2] HarshilGajjar[3]
[1,2,3]M.E. Student
[1,2,3]VLSI and Embedded Systems Design
[1,2,3]Gujarat Technological University

Abstract— In an environment where multiple masters and multiples slaves are used only one master can access the bus at a time for transaction over bus. Due to this, a mechanism is required which allows a single master to access the bus. This is accomplished by an electronic devise named as 'Arbiter'. Arbiter allocates access to shared resources. Based on priory different arbitrations schemes are required like Fair scheme, Round Robin scheme, Priority Based scheme, First Come First Serve scheme etc. Here a complete ASIC flow of First Come First Serve Arbiter is explained. Designing is done using Verilog, verification is performed using SystemVerilog and physical designing is shown. For complete process Synopsys tools are used.

Keywords: Arbiter, Arbitration Scheme, FCFS, Verilog, SystemVerilog, Synopsys

I. INTRODUCTION

Bus is a group of wires using which address and data is transported from one device to another.Bus-master initiates the whole transfer.Bus-master has complete control over the bus.A bus-system could have more than one bus-masters attached to it. There may be more than one device, requesting to become the bus-master. However, only devices can act as a bus-master at a given time.An arbiter decides as to which of the requesting devices should become the bus-master at one particular time. An arbiter may follow different kinds of policies to choose the best-candidate for mastering the bus. [3]

Different arbitration schemes used in arbiter are Fairness Scheme, Priority Scheme, Biased Scheme, Round Robin Scheme, First Come First Serve Scheme, Dynamic Scheme etc. Fig. 1 shows the basic block diagram of Arbiter.

II. RTL DESIGN OF FCFS ARBITER

In FCFS scheme, each requesting device has a counter. By default one device is always granted if no request on the bus. But if there are requests on the bus requests are granted according to FCFS scheme. The device with the highest counter value is granted for the bus. All counters are set to zero on reset, and when first device request for bus, its counter is incremented. While continuing access for the same if another device requests for the bus its counter is incremented and previous granted device's, which is still active on the bus, counter is also incremented. After that if again another device request for the bus its counter is incremented and the counter of previous two requesting device are also incremented. So according to FCFS scheme the device which request first for the bus will always have the highest counter value and will be granted first. When device complete its task on the bus and leave the bus, its counter value is set to zero. So automatically next highest counter value device will be granted. This RTL is developed for the same logic using Verilog language.[1]

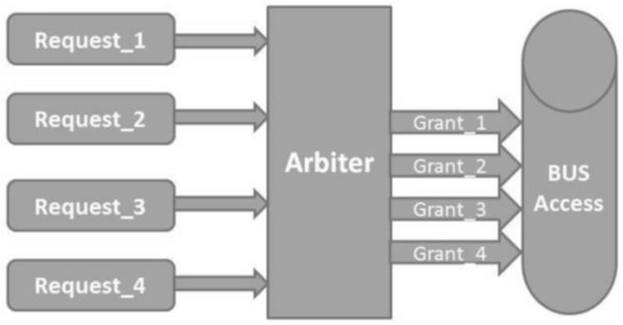

Fig.1: Block diagram of Arbiter

III. VERIFICATION USING SYSTEMVERILOG

Verification is one of the most important steps in ASIC flow as it consumes 70% of the time of flow. Verification actually validates the functionality of the design. Functional correctness is checked using all possible scenarios for inputs. Even error case scenarios are also checked to check design's functional correctness. Here a verification environment is developed using SystemVerilog language which is Hardware Verification Language (HVL) [2]. It is built over Verilog. It has all the advantages of Verilog and moreover added features like OOP, Interface, IPCs, Assertions, Constrained Randomization and Coverage makes it reasonably good verification language[2].

As saw in Fig. 2, SystemVerilog based verification environment use for Verify the RTC code who covered all

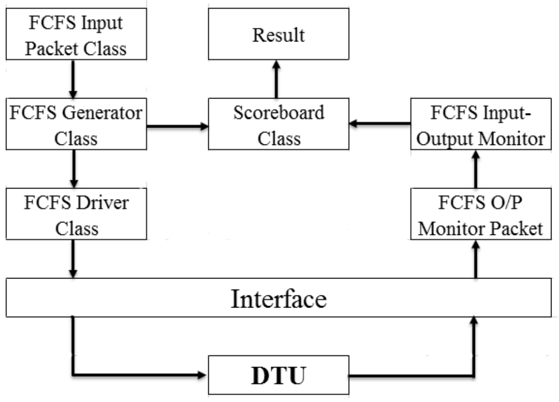

Fig.2: systemverilog Based FCFS Verification Environment

corner cases. It is layered based verification environment. There are different classes used in this environment.

(1) FCFS Packet class: - This class define all the input entity. It defines all the variables / fields used in the design.

(2) FCFS Generator class : - According todesired test cases, generator will generate all packets at zero time simulation. These packets covers all different type of test-cases, random test-cases and also it covers constrain random test-cases.
(3) FCFS Driver class : - The DUT's inputs are driven by the driver that execute single function, such as bus read or write. According to DUT require input driver send it.In driver block, it applies all input value to go to the local or global (main) interface.
(4) FCFS output Monitor : - DUT output signal drives onto the interface. This signals are then stored in this class. This class is independent of the value and functionality. It just monitors these signal and send to scoreboard class.
(5) FCFS Scoreboard : - Scoreboard class collect all generated signal from generator class and all DUT output signal from monitor class and compares both. Scoreboard class also check all output signal are working based on input or not.
(6) FCFS Result class : - Scoreboard class check the functionality of the RTL and it send this result in Result class. According to input output performance, result class generate the DUT correctness result.

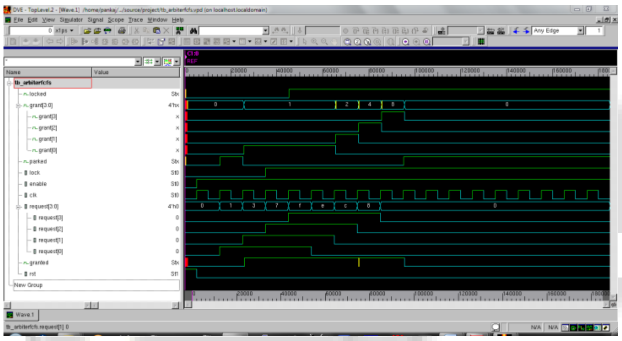
Fig.3: Fcfs arbiter simulation result

IV. SYNTHESIS

Synthesis is an automatic method of converting a higher level of abstraction to a lower level of abstraction. In other words the synthesis process converts Register Transfer Level (RTL) descriptions to gate-level net lists. For this process predefined technological libraries are required which maps RTL to gates. Design Compiler is very powerful tool from Synopsys which perform synthesis. According to ICs specification apply technological file. We also apply the wire-load model, clock rate, maximum fan out etc. and generate timing, power and area report. Fig. 4 shows the schematic diagram of FCFS arbiter after synthesis process.

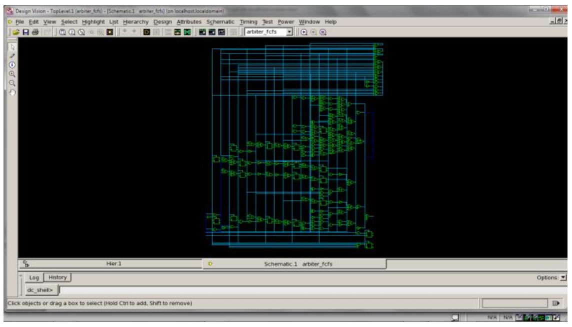

Fig.4: Schematic diagram of FCFS Arbiter

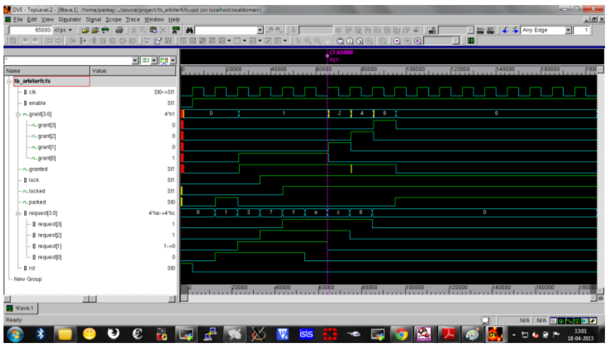
Fig.5: Gate level simulation

V. GATE LEVEL SIMULATION

We have verified the functionality of the RTL using SystemVerilog. It works fine. But after synthesis process the functionality of the design should be as same as RTL. Fig 5 shows the simulation results of the gate level simulation. And Fig. 6 shows the schematic diagram of FCFS arbiter after gate level simulation.

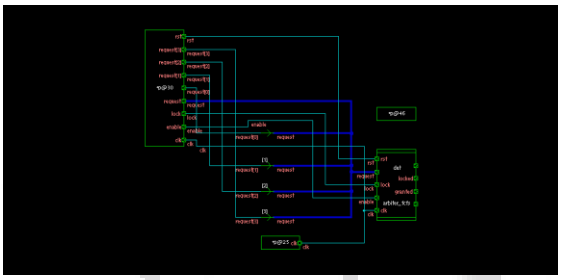
Fig.6: Schematic diagram of gate level simulation

VI. PHYSICAL DESIGN

Physical designing is process in ASIC flow, which is known as back end, where actual design is formed which can be forwarded for manufacturing in industry. For that synthesis output file which is net list is taken as input. Using net list we put all gates at proper place by floor planning. We perform routing to provide connection between all gates. Clocks are routed first as it is critical in every design. We manage all connection and optimize the design in such a way that all timing related functions work in proper manner, means no setup and hold violations. Fig. 7 shows the final schematic diagram after physical designing operation.

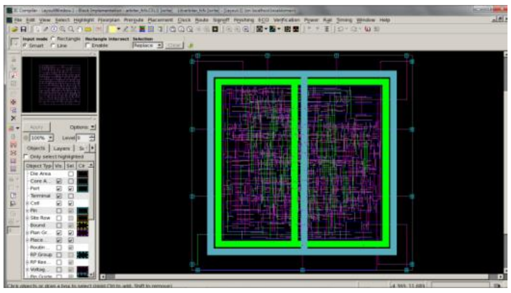
Fig.7: FCFS Arbiter ics Design

VII. CONCLUSION

Here complete ASIC flow for FCFS arbiter is performed. RTL designing is done using Verilog language. Verification is done using SystemVerilog language which validates the functionality of the design. Physical designing is performed up to design exchange file which is a digital file ready to be manufacture. For all above processes standard Synopsys galaxy series tools are used.

REFERENCE

[1] Samir H Palnitkar, Verilog HDL - A digital design and synthesis. Second Edition.
[2] Chris Spear, SystemVerilog for Verification, Second Edition, Springer.
[3] Arbiter information http://enpub.fulton.asu.edu/cse517/arbiter.htmlhttp:// www.asic - world.com / verilog / verilog _one_day1.html#Introduction.