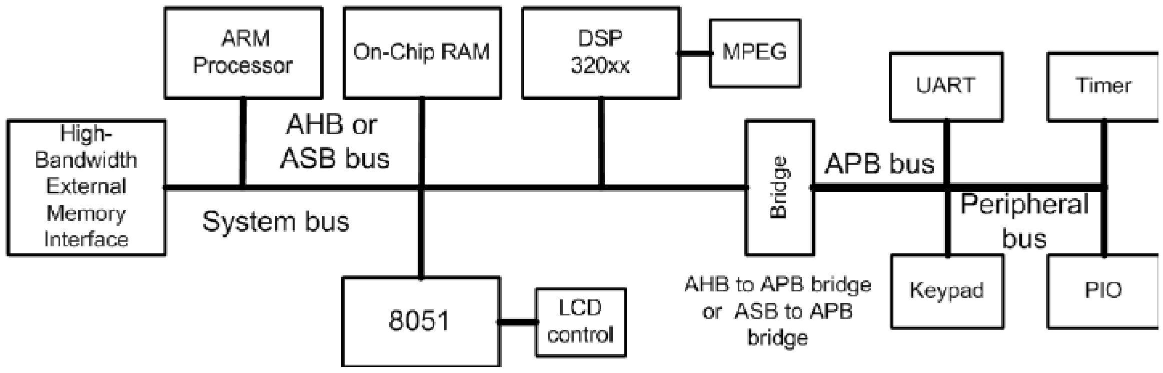
AMBA APB Protocol

Introduction

- AMBA

The AMBA (Advanced Microcontroller Bus Architecture) is an open standard bus protocol developed by ARM for designing complex System-on-Chip (SoC) architectures. It provides a framework and specifications for efficient communication and integration of IP components within a SoC. The architecture includes buses like AHB, ASB, APB, and AXI, each catering to specific requirements in terms of performance and complexity. AMBA promotes modularity and reusability, simplifying the development of embedded systems in various domains.



- ABP Protocol

Developed by ARM, the APB protocol provides a low-power, low-complexity interface for connecting peripheral devices to a microprocessor or microcontroller.

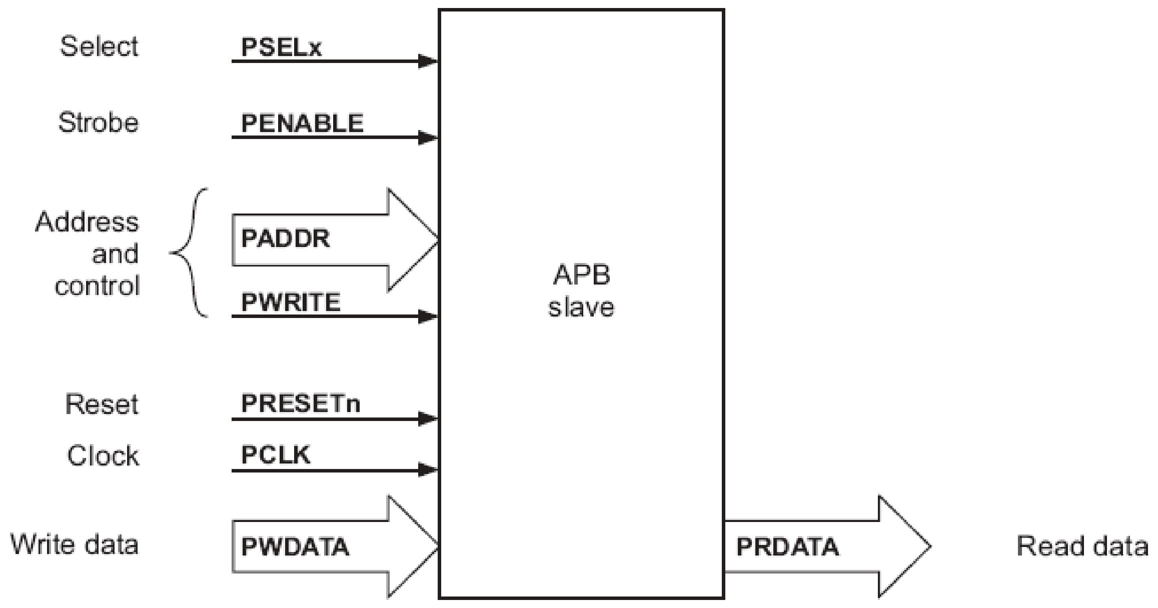
The AMBA APB protocol is designed to address the specific requirements of peripheral devices that operate at slower speeds and do not require high bandwidth. It is commonly used to connect peripheral devices such as timers, interrupt controllers, UARTs (Universal Asynchronous Receiver-Transmitters), and other low-speed peripherals to the system bus.

The AMBA APB protocol serves as a crucial interface protocol for connecting peripheral devices in SoC designs. Its simplicity, low power consumption, and widespread adoption have made it a popular choice in the industry.

Characteristics of APB

1. **Simplicity:** APB is designed to be simple, making it easier to implement and integrate into SoC designs. Its simplicity is especially suitable for low-power and low-complexity peripheral devices.
2. **Low Power Consumption:** APB is optimized for low-power operation, making it ideal for connecting low-speed peripheral devices that don't require high bandwidth.
3. **Single Clock:** APB operates using a single clock, which simplifies the timing requirements and reduces complexity in the design.
4. **Master-Slave Architecture:** APB follows a master-slave architecture, where a single master initiates all data transfers to connected slave devices. This architecture ensures orderly and controlled communication between the master and slaves.
5. **Control and Data Transfers:** The protocol defines rules and guidelines for address and data transfers, read and write operations, arbitration, error handling, and clock synchronization. This ensures efficient and reliable communication between the master and slave devices.
6. **Suitable for Low-Speed Peripherals:** APB is commonly used for connecting low-speed peripheral devices such as timers, interrupt controllers, UARTs, and other similar peripherals to the SoC. It provides a cost-effective and efficient solution for these types of devices.

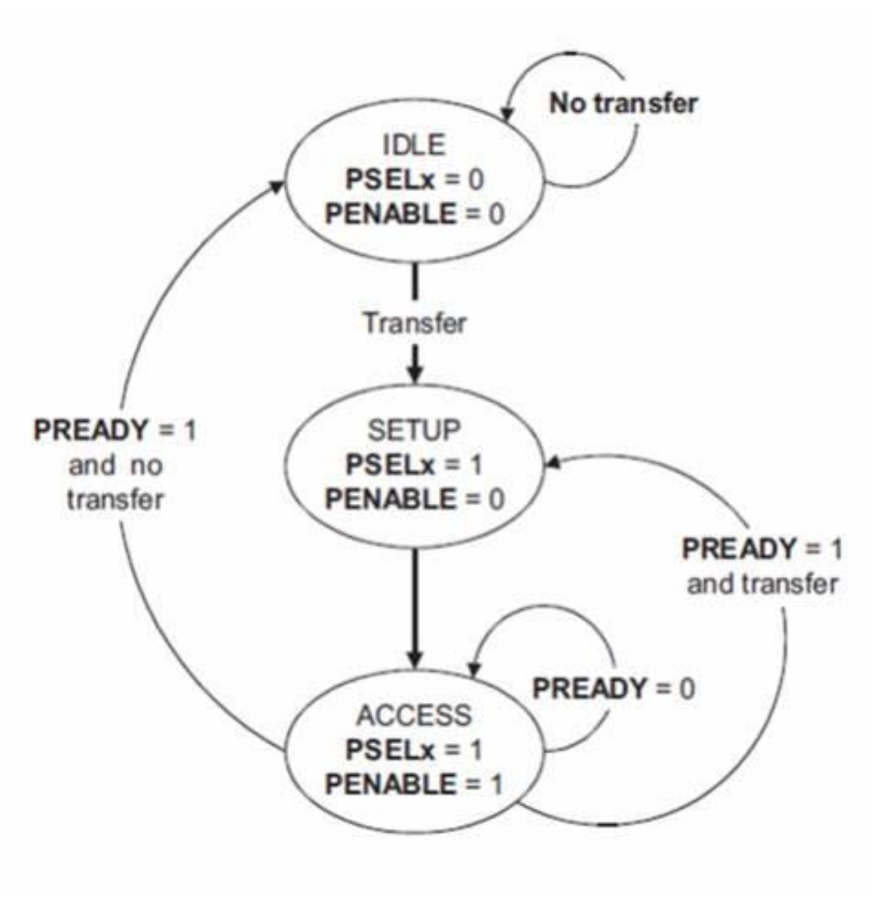
Block Diagram



Signal Description

- **PCLK Clock:** The rising edge of PCLK times all transfers on the APB.
- **PRESET:** System bus equivalent Reset. The APB reset signal is active LOW.
- **PADDR:** 32-bit address bus
- **PSEL:** The slave device is selected and that a data transfer is required.
- **PENABLE:** Enable, this signal indicates the second and subsequent cycles of an APB transfer.
- **PWRITE:** control bit to dictate read or write operation.
- **PWDATA:** 32 bits Write data if PWRITE is HIGH.
- **PREADY**: Ready To extend an APB transfer.
- **PRDATA:** 32 bits Read data if PWRITE is LOW.
- **PSLAVERR:** Slave error, this signal indicates a transfer failure.

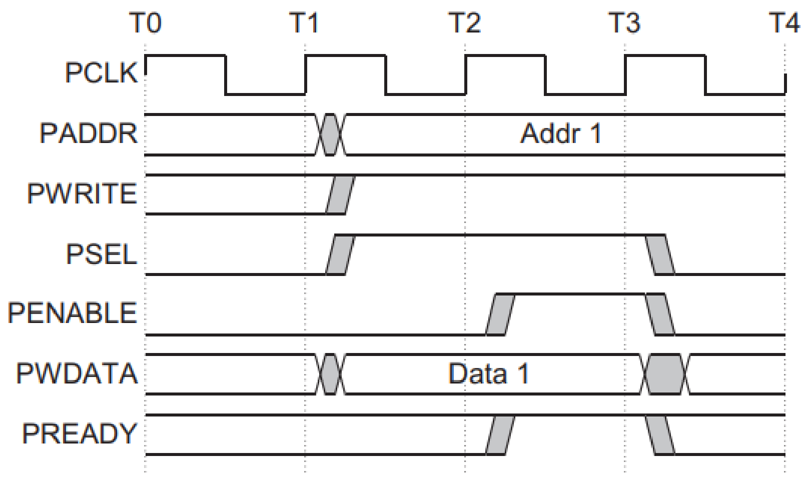
Operating States



- **IDLE**: This is the default state of the APB.
- **SETUP**: When a transfer is required the bus moves into the SETUP state, where the appropriate select signal, PSELx, is asserted. The bus only remains in the SETUP state for one clock cycle and always moves to the ACCESS state on the next rising edge of the clock.
- **ACCESS**: The enable signal, PENABLE, is asserted in the ACCESS state. The address, write, select, and write data signals must remain stable during the transition from the SETUP to ACCESS state. Exit from the ACCESS state is controlled by the PREADY signal from the slave
- In summary, the APB protocol involves transitioning from the idle state to the setup phase, where the master device prepares for a transaction. The access phase follows, where the master initiates the transaction by asserting control signals and providing the address.

Expected Waveforms

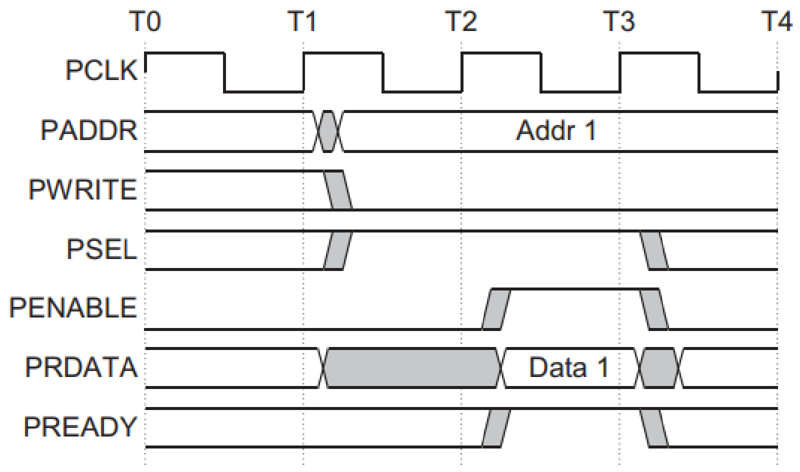
Write Operation



- At T1, a write transfer starts with PADDR, PWDATA, PWRITE, and PSEL, being registered at the rising edge of PCLK. It is called the SETUP cycle.
- At the next rising edge of the clock T2 it is called ACCESS cycle, PENABLE, and PREADY, are registered. When asserted, PENABLE indicates starting of Access phase of the transfer. When asserted, PREADY indicates that the slave can complete the transfer at the next rising edge of PCLK.
- The PADDR, PWDATA, and control signals all remain valid until the transfer completes at T3, the end of the Access phase.

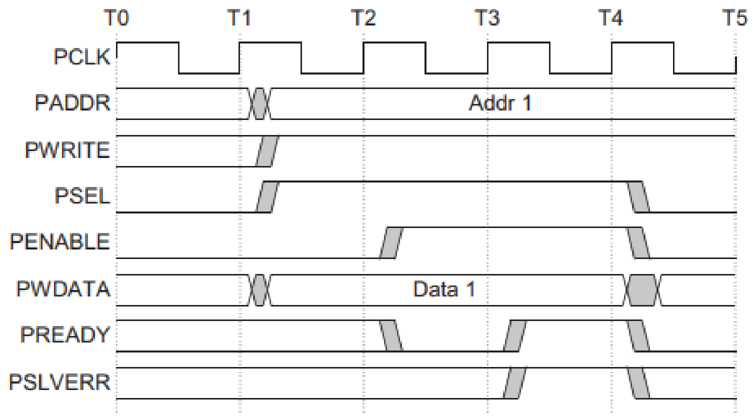
- The PENABLE, is disabled at the end of the transfer. The select signal PSEL is also disabled unless the transfer is to be followed immediately by another transfer to the same peripheral.

Read Operation



- During read operation the PENABLE, PSEL, PADDR PWRITE, signals are asserted at the clock edge T1 (**SETUP cycle**).
- At the clock edge T2, (**ACCESS cycle**), the PENABLE, PREADY are asserted and PRDATA is also read during this phase. The slave must provide the data before the end of the read transfer.

Slave Error



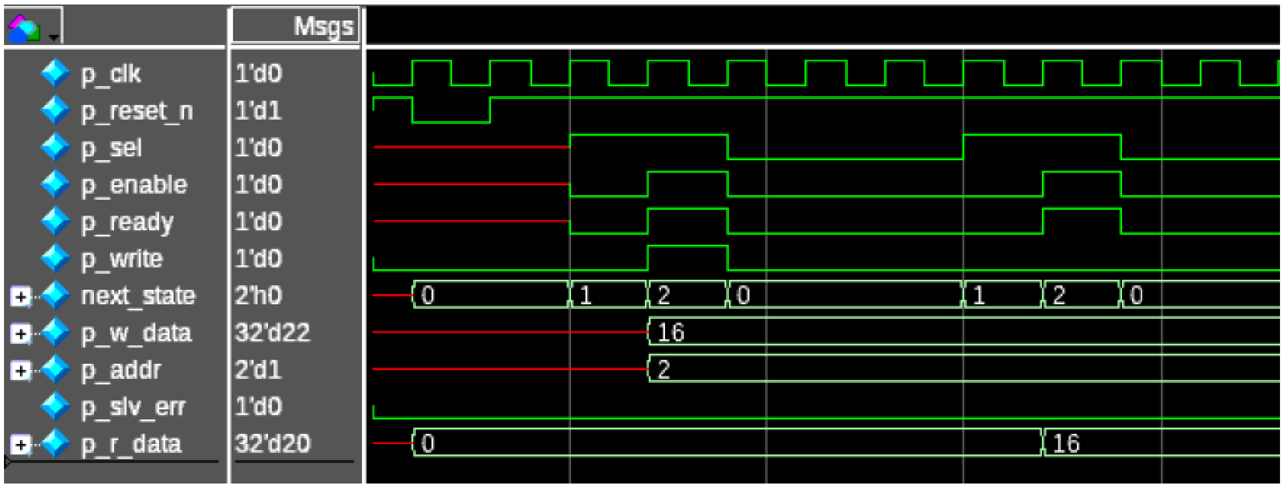
- We can use PSLVERR to indicate an error condition on an APB transfer. Error conditions can occur on both read and write transactions.
- PSLVERR is only considered valid during the last cycle of an APB transfer, when PSEL, PENABLE, and PREADY are all HIGH.
- There is no requirement for the peripheral to drive the data bus to all 0s for a read error.

- When a write transaction receives an error this does not mean that the register within the peripheral has not been updated. Read transactions that receive an error can return invalid data. There is no requirement for the peripheral to drive the data bus to all 0s for a read error.

Simulation Results

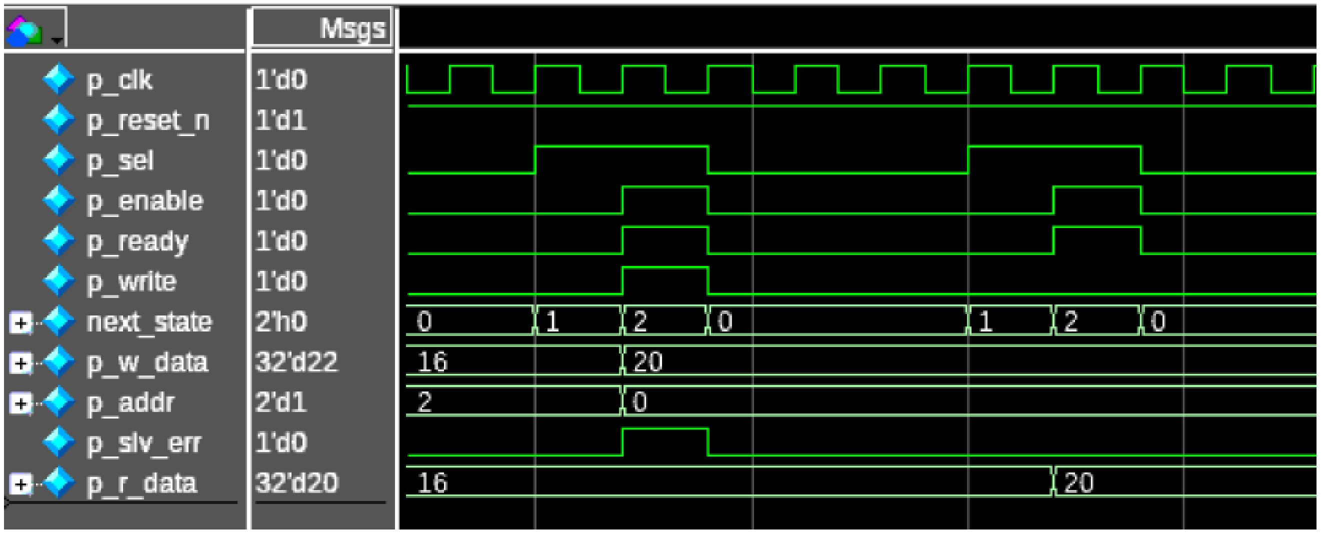
The following results are implementation of Verilog code in Questa-sim.

Read and Write Operation



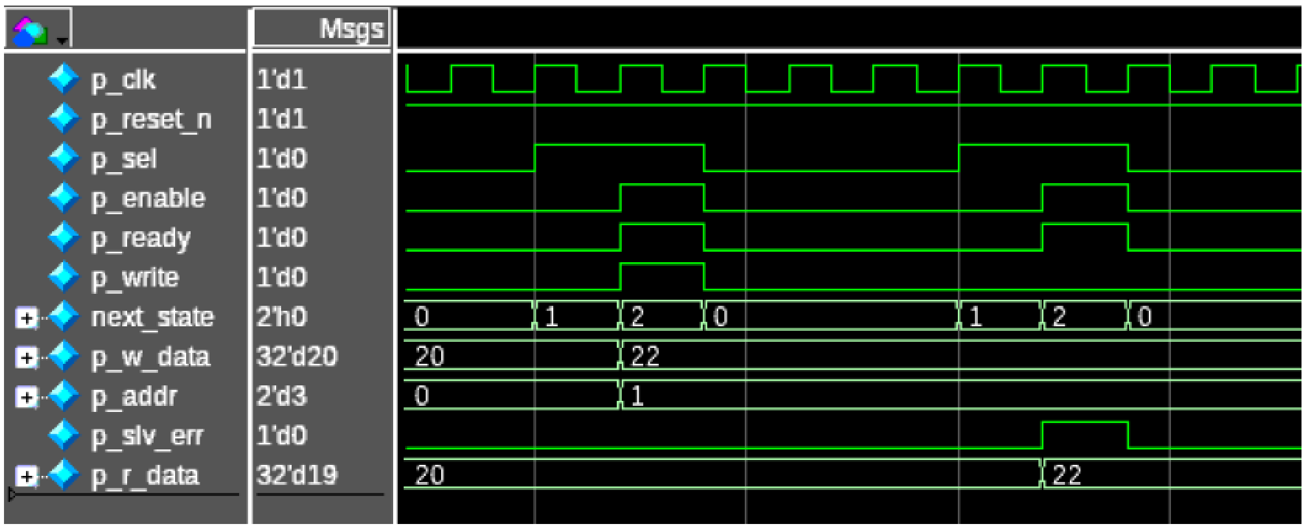
The data is being written into the memory and it is being read from the memory.

Only Read Operation



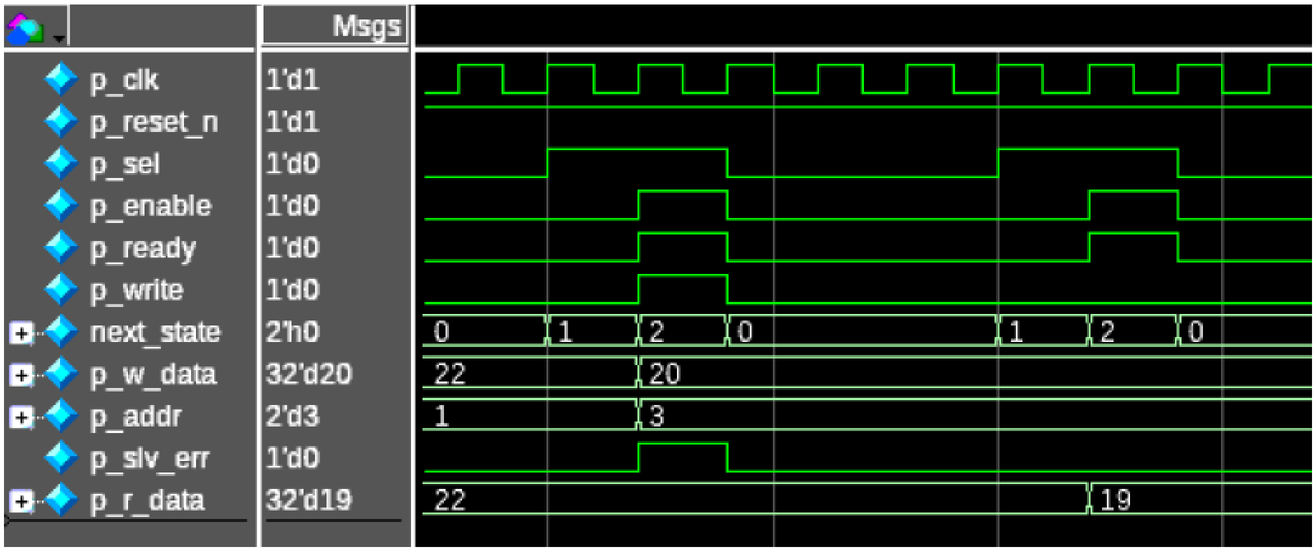
A slave error signal is observed when trying to write into this register and the data can be read normally from the register.

Only Write Operation



Data can be written normally into this register but a slave error signal is observed when data is to be read from it.

Fixed Value Operation



Data cannot be written into this register and a slave error signal is observed. Always a fixed value is read out of the register.

Conclusion

We have explored the key characteristics and operating states of the APB protocol. We have seen how its single-clock architecture, master-slave configuration, and well-defined phases enable efficient and controlled communication between the master and slave devices.

The APB protocol's straightforward two-phase protocol simplifies implementation and reduces complexity, making it particularly suitable for low-power and low-complexity peripheral devices. Its modularity and compatibility within the AMBA architecture also facilitate the reuse of intellectual property components, promoting efficiency and time-saving in SoC designs.

References

[1] ARM, “AMBA Specification Overview”: AMBA APB Protocol Specification (umich.edu)

[2] Design And Implementation Of Amba Apb Protocol, J Mukunthan *et al* 2021 *IOP Conf. Ser.: Mater. Sci. Eng.* **1084** 012050

[3] VERILOG Reference Manual, http://www.accellera.com

[4] Samir Palnitkar, “Verilog HDL: A guide to Digital Design and Synthesis (2nd Edition), Pearson, 2008.