ECE593 Group Project: AMBA 3 AXI Protocol

- Keerthi Chowdary Yerragangu
- Erik Fox

Functional Specification:

- https://developer.arm.com/documentation/ihi0022/b

Verification Plan:

Technical Requirements:

- **Description of verification levels**
    - ~~Designer level~~
        - ~~Because we have no idea about the amount of designer level verification done by the designer, it is important that we establish baseline functionality ourselves.~~
        - ~~Master protocol~~
            - ~~Confirm functionality of internal FSM's:~~
                - ~~Write address channel~~
                - ~~Write data channel~~
                - ~~Write response channel~~
                - ~~Read address channel~~
                - ~~Read data channel~~
        - ~~Slave protocol~~
            - ~~Confirm functionality of internal FSM's:~~
                - ~~Write address channel~~
                - ~~Write data channel~~
                - ~~Write response channel~~
                - ~~Read address channel~~
                - ~~Read data channel~~
    - ~~Unit level~~
        - ~~Master protocol~~
            - ~~Test that master modules can perform required functions, and follow correct protocol.~~
        - ~~Slave protocol~~
            - ~~Test that slave modules can perform required functions, and follow correct protocol.~~
    - System level
        - Transactions/Interface
            - Execute functions, at transaction level, between master and slave to confirm interconnection and protocol between units are correct.

- **Functions to be verified**

- Basic Transactions
    - Read burst: Various burst lengths
    - Overlapping read burst: Master can initiate second read after slave accepts first, and process in parallel.
    - Write burst: Various burst lengths
    - Out of Order transactions: The protocol requires that transactions with the same ID tag are completed in order, but transactions with different ID tags can be completed out of order.

- Additional Features:
    - Burst types: The AXI protocol supports three different burst types that are suitable for:
        - Normal memory accesses
        - Wrapping cache line bursts
        - Streaming data to peripheral FIFO locations
        - Types:
            - Fixed
            - Incrementing
            - Wrapping
    - System Cache support: The cache-support signal of the AXI protocol enables a master to provide to a system-level cache the bufferable, cacheable, and allocate attributes of a transaction
        - Bufferable
        - Cacheable
        - Allocate
    - Protection Unit Support: To enable both privileged and secure accesses, the AXI protocol provides three levels of protection unit support:
        - Normal or privileged
        - Secure or nonsecure
        - Instruction or data
    - Atomic Operations: The AXI protocol defines mechanisms for both exclusive and locked accesses.
        - Normal access
        - Exclusive access
        - Locked access
    - Error Support: The AXI protocol provides error support for both address decode errors and slave-generated errors.
        - OKAY: Normal access successful
        - EXOKAY: Exclusive access successful
        - SLVERR: Slave error
        - DECERR: Decode error
    - Unaligned address: To enhance the performance of the initial accesses within a burst, the AXI protocol supports unaligned burst start addresses.

- **Specific tests and methods**
  - Type of verification:
    - Greybox. This type of verification is appropriate for this design because it is helpful to know information about internal state transitions.
  - Verification Strategy:
    - ~~Designer level: Deterministic testing designed to follow all paths through the FSM's.~~
    - Unit and System Level's: Constrained randomization with "unrandomizing" controls.
  - Abstraction Level:
    - ~~Designer Level: Bit level~~
    - Unit and System level: Command level
  - Checking Strategy:
    - ~~Designer Level: Golden vector~~
    - Unit and System level: Transaction-based checking.

- **Coverage requirements**
  We try to obtain code coverage and functional coverage. Code coverage helps to find which part of the code is not getting executed. While functional coverage checks for functionality of design .
  We will write different covergroups which cover signals in read address channel, read data channel, write address channel , write data channel , write response channel ,low power interface signals.
  We will write coverpoints for handshake dependencies in read and write transactions.

- **Test scenarios**
  - -Write address channel
    - -when ARESETn is low, corresponding write channel control signals should go low.
    - -when AWVALID is valid , the address and control information must remain stable until AWREADY goes high.
    - -once asserted AWValid,AWREADY should be high throughout the transaction.
  - -Write data channel
    - -when reset is low, corresponding write data control signals should be low.
    - -when WVALID is high, WDATA should be valid and stable.
    - -once asserted WVALID,WREADY should be high throughout the operation.
  - -Write response channel
    - -when reset is low, corresponding write response control signals should be low.
    - -once asserted BVALID,BREADY they should remain the same throughout the operation.

-Read address channel

        -when reset is low, corresponding read address control signals should be low.

        -when ARVALID is high, ARADDR should be valid and stable.

        -once asserted ARVALID,ARREADY should be high throughout the operation.

-Read data channel

        -when reset is low, RVALID,RDATA,RREADY should be low.

        -when RVALID is asserted , RDATA should be high and stable.

        -once asserted RVALID,RREADY should be high throughout operation.

-Interesting Scenarios:-

        - Alternate write and read operations to the same memory location.

        -Simultaneous read,write to the same memory address.

        -Simultaneous write operations to different memory locations.

        -multiple writes to the same memory location and perform read at the end to check for latest written data.

        - Try to access an out-of-bound address.

Project Management:

- **Required tools :**
    1) EDA playground
    2)QuestaSim for simulation
    3) Notepad++ for writing code
    4) GITHub
- **Risks and dependencies**

    License for tools used i.e; Questasim in our project.

    Timely availability of resources to complete the project.
- **Resources requirements**

    1)Comprehensive Functional Verification Verification :The complete industry cycle by Bruce Wile,John C.Gross, Wolfgang Roesner.

    2)Lectures materials of Prof.Tom Schubert.

    3)AMBA AXI protocol Specification.

    4)SystemVerilog Tutorials on SystemVerilog Tutorial (chipverify.com)

    5)Design source code: taken from GITHUB yvnr4you/AMBA_AXI3: System Verilog and Emulation. Written all the five channels. (github.com)

    6)People: Erik fox

            Keerthi chowdary yerragangu

- **Schedule details**

    Week 6 - submit Verification plan.

Week 7 - Submit waveform along with Test bench which is driven by deterministic test cases checking all the operations of the design.

Week 8 -Testbench written in system verilog classes. To build a bus functional model having a testbench interface,coverage,generator.

Week 9-Build a scoreboard.Collect and improve coverage by constrained random tests.Use of Assertions. Run tests and debug them.

Week 10 - Submit final code , report .