INCISIVE VERIFICATION ARTICLE
MAY 2006

SYSTEMATICALLY IMPLEMENTING LATE ENGINEERING CHANGES ON YOUR PROJECT: DOs AND DON'Ts

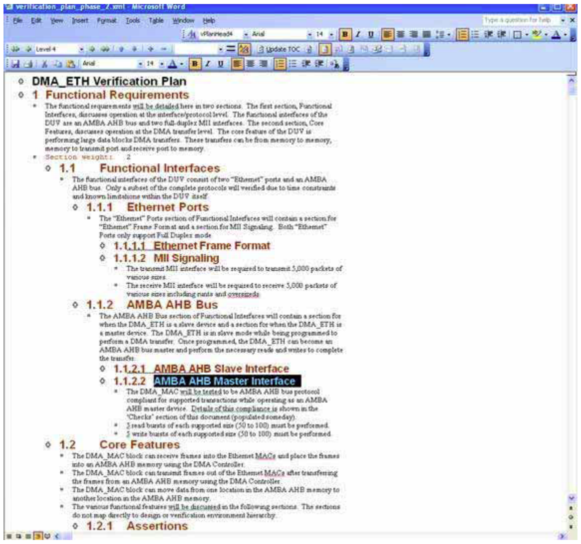
ILANA GOLAN, SENIOR PRODUCT ENGINEER, CADENCE DESIGN SYSTEMS

Project management and automation are quickly becoming the most critical elements in the overall design and verification process. The most effective verification management strategy requires that design teams focus on some key features for success.

First and foremost is the need to drive the entire project/design from a good overall specification. This spec needs to be developed with planning and metric-based checkpoints in mind. The mentality of "beginning with the end in mind" allows for optimal resource usage, much higher design quality, and realistic schedule estimates. Despite all the best planning, changes in requirements happen up to the last day.

Today's design teams typically do not systematically address the problems that cause slips or productivity and quality issues. They tend to focus solely on individual tasks, engine performance, or languages, rather than defining the entire verification challenge, independent of its solution. In fact, most verification plans are merely a set of incomplete discussion notes that atrophy as the project moves forward.

Change management of the design and verification process needs to start with the goals of what needs to be verified. From this, an experienced verification team will have the ability to develop changes to the plan that are complete and inclusive of the intended goals. Let's take a closer look at an executable verification plan (Figure 1).



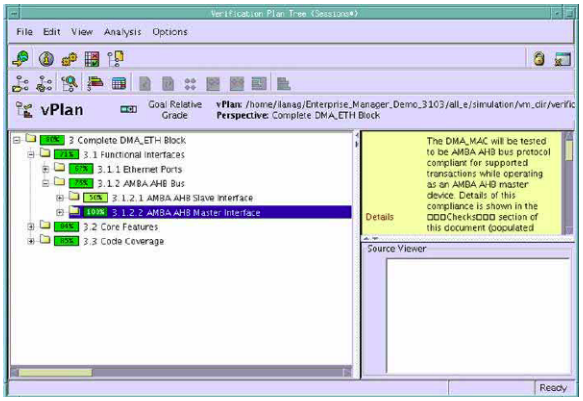
- DMA_ETH Verification Plan
- 1 Functional Requirements
  - The functional requirements will be detailed here in two sections. The first section, Functional Interfaces, discusses operation at the interface/protocol level. The functional interfaces of the DUV are an AMBA AHB bus and two full-duplex MII interfaces. The second section, Core Features, discusses operation at the DMA transfer level. The core feature of the DUV is performing large data blocks DMA transfers. These transfers can be from memory to memory, memory to transmit port and receive port to memory.
  - Section weight: 2
  - 1.1 Functional Interfaces
    - The functional interfaces of the DUV consist of two "Ethernet" ports and an AMBA AHB bus. Only a subset of the complete protocols will verified due to time constraints and known limitations within the DUV itself.
    - 1.1.1 Ethernet Ports
      - The "Ethernet" Ports section of Functional Interfaces will contain a section for "Ethernet" Frame Format and a section for MII Signaling. Both "Ethernet" Ports only support Full Duplex mode
      - 1.1.1.1 Ethernet Frame Format
      - 1.1.1.2 MII Signaling
        - The transmit MII interface will be required to transmit 5,000 packets of various sizes.
        - The receive MII interface will be required to receive 5,000 packets of various sizes including runts and oversizeds
    - 1.1.2 AMBA AHB Bus
      - The AMBA AHB Bus section of Functional Interfaces will contain a section for when the DMA_ETH is a slave device and a section for when the DMA_ETH is a master device. The DMA_ETH is in slave mode while being programmed to perform a DMA transfer. Once programmed, the DMA_ETH can become an AMBA AHB bus master and perform the necessary reads and writes to complete the transfer.
      - 1.1.2.1 AMBA AHB Slave Interface
      - 1.1.2.2 AMBA AHB Master Interface
        - The DMA_MAC will be tested to be AMBA AHB bus protocol compliant for supported transactions while operating as an AMBA AHB master device. Details of this compliance is shown in the "Checks" section of this document (populated someday).
        - 5 read bursts of each supported size (50 to 100) must be performed.
        - 5 write bursts of each supported size (50 to 100) must be performed.
  - 1.2 Core Features
    - The DMA_MAC block can receive frames into the Ethernet MACs and place the frames into an AMBA AHB memory using the DMA Controller.
    - The DMA_MAC block can transmit frames out of the Ethernet MACs after transferring the frames from an AMBA AHB memory using the DMA Controller.
    - The DMA_MAC block can move data from one location in the AMBA AHB memory to another location in the AMBA AHB memory.
    - The various functional features will be discussed in the following sections. The sections do not map directly to design or verification environment hierarchy.
    - 1.2.1 Assertions

Figure 1: An executable verification plan captures the metrics

Based on the executable verification plan above, here are some of the DOs and DON'Ts your team needs to be aware of to create a much more systematic and successful design process, and to introduce project changes effectively.

DOs

- Have a plan that is always kept up-to-date and is executable. Your total coverage results after regressions will be mapped directly into your plan. This is what your progress and completion will be measured against. An executable plan allows for coverage grades to be mapped to the features in the plan (Figure 2).


Verification Plan Tree (Sessions*)
File Edit View Analysis Options
vPlan
Goal Relative Grade
Details
Source Viewer
Ready

vPlan: /home/ilanag/Enterprise_Manager_Demo_3103/all_e/simulation/vm_dir/verific
Perspective: Complete DMA_ETH Block

80% 3 Complete DMA_ETH Block
71% 3.1 Functional Interfaces
67% 3.1.1 Ethernet Ports
75% 3.1.2 AMBA AHB Bus
50% 3.1.2.1 AMBA AHB Slave Interface
100% 3.1.2.2 AMBA AHB Master Interface
84% 3.2 Core Features
85% 3.3 Code Coverage

The DMA_MAC will be tested to be AMBA AHB bus protocol compliant for supported transactions while operating as an AMBA AHB master device. Details of this compliance is shown in the □□□Checks□□□ section of this document (populated

Figure 2: Coverage grades mapped to the features in the plan

- Identify all parts of the system that are impacted by each change and incorporate that into the overall plan before making any changes to the specification.
- Identify an owner of the change process who will be committed to following the process to completion. This person needs to have a good overall view of the project and maintain a high attention to detail.
- Measure the impact of changes by the scope of the change to the verification plan. Especially consider impact on various dependencies outside of your control, such as any third-party cores; designs that may use third-party soft/hard IP; or any dependencies on support from a semiconductor vendor.
- Always review the effectiveness and quality of your process on a regular basis and at the end of each project. Each change forced by errors in the past should be used as a learning experience and categorized for future projects.

DON'Ts

- Don't assume a change is completed until specific system behaviors exhibiting the change have been verified. Especially take care of changes up and down the entire process that reach other areas of specialization.
- Don't assume one person can implement a change to completion. Effectiveness in verification is a lot higher with redundancy and cross-checks.
- Don't assume that a simple change is the best solution. At times, a "simple change" may create a butterfly effect and morph the system being verified from relatively stable to utterly chaotic. Watch out for the fallacy of simple solutions by reviewing changes with all adjacent groups and the architecture team.
- Don't use shortcuts, like not updating plans and all necessary metrics of quality. Shortcuts may lesson quality, impact predictability, and decrease productivity.
- Don't change multiple parts of the system all at once without first assessing dependencies. In some cases, it may be prudent to verify the design in a few, well-defined iterations.

By following these simple guidelines, good planning based on verification management can be realized. Your organization will benefit as all stakeholders begin to leverage the ability to capture and review the verification plan as you drive your next design to closure.