white paper

IDesignSpec^{TM}

Don't fear change, embrace it.

Anupam Bakshi, MS, MBA, President, Agnisys, Inc.

May 1st, 2008

Abstract

This white paper discusses the inefficiencies in the current digital design process, and how IDesignSpec^{TM} eliminates them, thereby reducing the design cost and improving quality and time to market. IDesignSpec^{TM} is a new methodology that increases the productivity of FPGA/ASIC, SoC and System Designs. It automates the generation of downstream data that typically had required manual creation from the original design specification. Its specialized editor enables designers to create correct-by-construction, reusable designs.

Introduction

Improving the design process is vital.

Making a chip and system design-process efficient is vital as millions of dollars are riding on their timely entry into the market. Design Architects and Engineering Managers are constantly on the lookout for improvements to the overall design process that would improve the bottom line. Any improvement in the design process which can ensure better quality and save time is desirable.

Managing Configuration and Register information is central to hardware design.

Today's typical digital designs are highly configurable with a lot of functionality crammed into them. This configurability makes the systems versatile and cost effective. Configurability is implemented as register settings in the chips. There could be hundreds if not thousands of such configuration and status registers in a typical hardware design. These registers play a central roll in the digital design process as they are the interface between hardware and software. The configuration and status registers constitute the "Programmer's Interface" of the hardware.

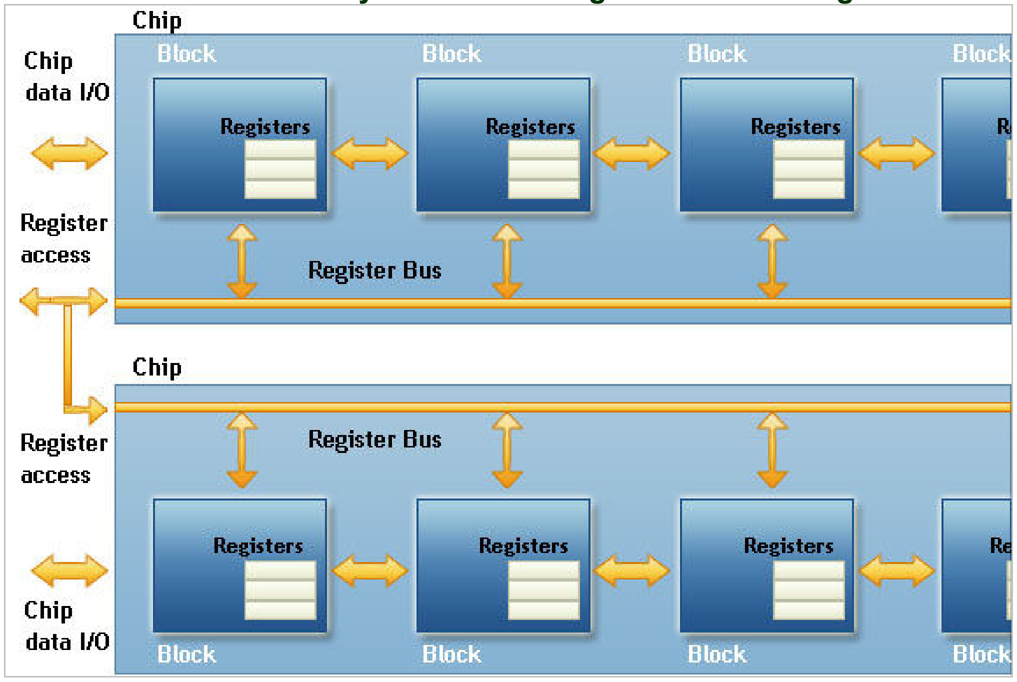
Carefully managing the register information and its changes thru the design process is essential to yield significant improvements in the quality of the design.

Background*(Skip this section if you are familiar with digital designs)*

Typical Hardware system contains many registers.

Figure 1, shows a typical hardware system with a couple of chips with a data I/O bus and a bus for register access. The chips contain several blocks, each with their own sets of registers. The register bus conforms to some standard protocol. Popular among the protocols are: AMBA, OCP-IP, Wishbone, Avalon and sometimes even proprietary buses. Factors like the frequency of operation, requirement for burst writes and reads, total bandwidth requirements are some of the factors that come into play in choosing the appropriate bus. It is possible that the choice of the register bus changes over time from one generation of product to the next.

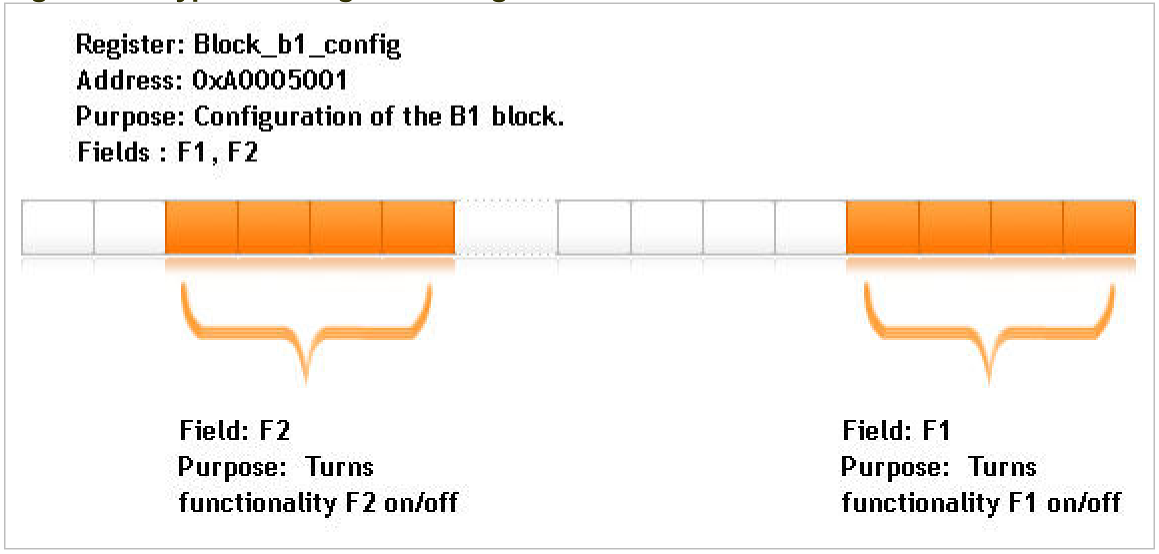
Figure 1: Canonical hardware system with configuration/status registers.



Typical register contains various fields.

A configuration register may be thought to be divided into several fields of arbitrary widths, each associated with certain functionality of the design (see Figure 2). The fields could individually be read writable, read-only, write-one-to-clear or contain a fixed value. These fields are logically in the same register, but could be implemented separately.

Figure 2: A typical configuration register with two fields.


Register: Block_b1_config
Address: 0xA0005001
Purpose: Configuration of the B1 block.
Fields : F1, F2
Field: F2
Purpose: Turns functionality F2 on/off
Field: F1
Purpose: Turns functionality F1 on/off

Information Flow

Let's take a closer look at the flow of information for a digital design in general, and more specifically the flow of register information through a design process.

Everyone in the design team depends on the correct Register information.

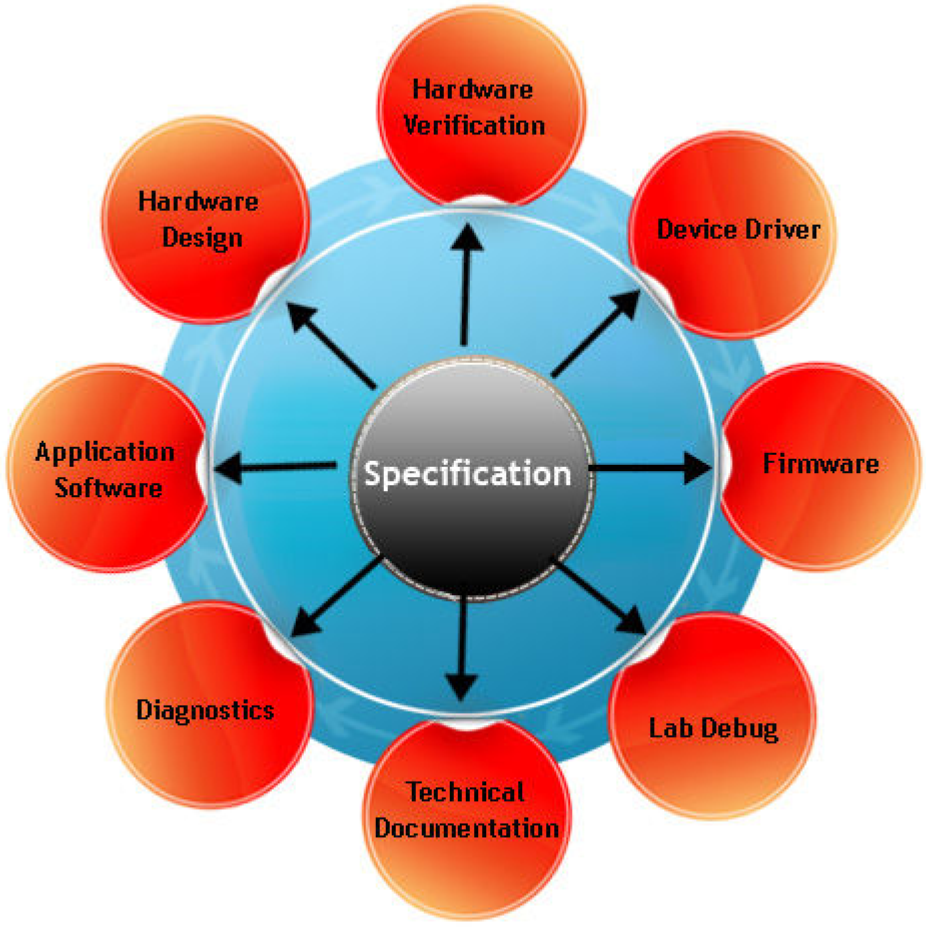
The Architect or hardware engineer typically creates a specification that includes the functionality of the design and the register details. All groups involved in the design process need correct, up-to-date, unambiguous information about the location of these registers by way of their address and their purpose. The design group needs this information for proper design implementation, the Verification team needs it for verify the intended purpose, the Device Driver, Diagnostics groups, lab technicians and the Software Application developers need it in order to properly use the hardware. These groups also need the latest, up to date documentation that goes with the register description. This information is typically captured in some text editor or spreadsheet.

After the specification is written the registers are hand crafted in RTL. The same information is then re-entered in the form of C/C++ header files for use in Application Software, Device Driver, Diagnostics and Firmware.

As the design process progresses, and implementation is started, new functionality is introduced and some of the old one is modified or eliminated. Each one potentially has the effect of modifying the register location, adding/modifying/removing the registers, its constituent fields and their purpose.

Changes to the specification leads to widespread changes in the various "views" of these registers as RTL needs to change, C/C++ code needs to change, documentation and even Lab debug instructions need to be updated.

Figure 3: Groups dependent on the specification.



A process where engineers manually read the specification and code the RTL or the C header files is highly undesirable as it is tedious, mundane, and costly both in terms of resources and time used.

Design Reuse

Design reuse is more than RTL reuse.

According to industry estimates, 80% of new ASIC/FPGA designs have components that are reused from earlier designs. Reuse within the company is vital for reducing the overall costs. However, reuse is not just RTL duplication. In order to be effective, components from all downstream views such as verification, firmware, drivers, and software must be effectively reused. Typically, engineers in various groups do their own duplication of code, if they are alerted to the fact that the design or a block has been reused from past design.

When reusing a design it is tedious to make sure that bugs removed from the original design also get communicated to the teams reusing that design.

When reusing a design, is the documentation reused too? If the design and documentation are generated from the same source, the problem of keeping them together goes way, because the single source contains both.

At times designs are locked to the hardware register bus used, or the registers are spread all over the design without clear boundaries. This is a hindrance to design reuse since a subsequent generation of the design may use a different register bus or may want to change the registers in some way.

Design Documentation

Managing Design Documentation can be a big challenge.

Often the documentation of the design is separate from the register definition. For example Microsoft Word may be used for design documentation and Microsoft Excel may be used to describe the registers. This approach suffers from the basic problem that it keeps the documentation of the design separate from the implementation. The documentation, the register definition and the design have to be manually and tediously kept in sync.

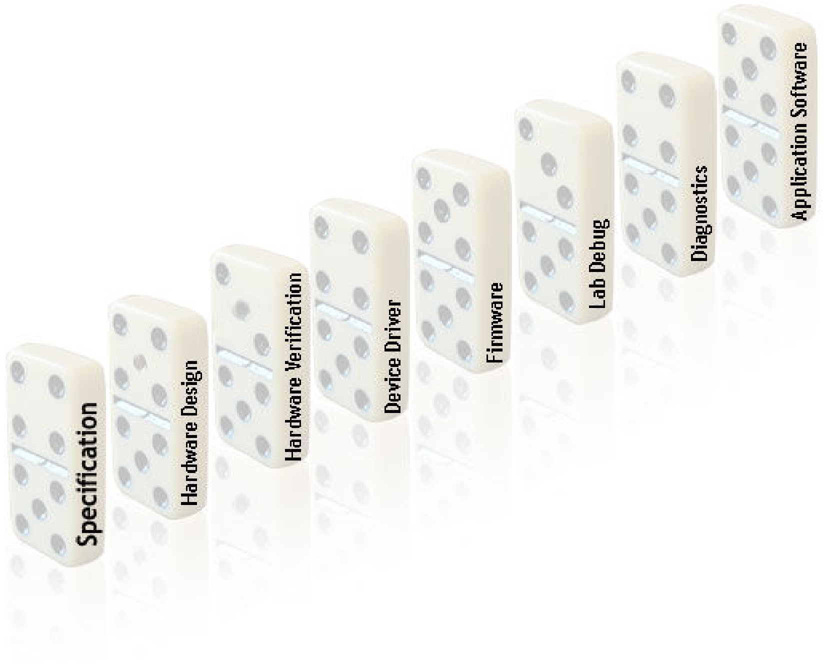
Problem Summary

Manually creating "views" leads to several disconnects.

In a digital design process, when a single source of design information is not maintained and when design team members have to enter the same data in different ways and in different formats, it leads to inefficiencies. This not just makes the team less productive, it's a huge time sink.

Change in the specification/register description leads to changes all over. If the change is not handled in an automated fashion, chaos reigns supreme and quality suffers.

Figure 4: The Domino Effect in Hardware Design.



Often the design teams overlook the importance of proper register specification and associated documentation. This leads to non-uniform specifications that are spread out all over the design environment. The specification is far removed from the real code that is implemented from them. This also leads to widespread issues with reusability and efficiency.

Current solutions

A couple of EDA companies have introduced tools to solve the problem described above. However, they are inadequate since they require designers to either learn a new language or a new GUI. In addition, they do not solve the problem of keeping the design documentation in sync with the register description.

A single system is required that enables the designer to "naturally" describe the specification of the design. This should be human and machine readable so that all downstream views can be automatically generated from it.

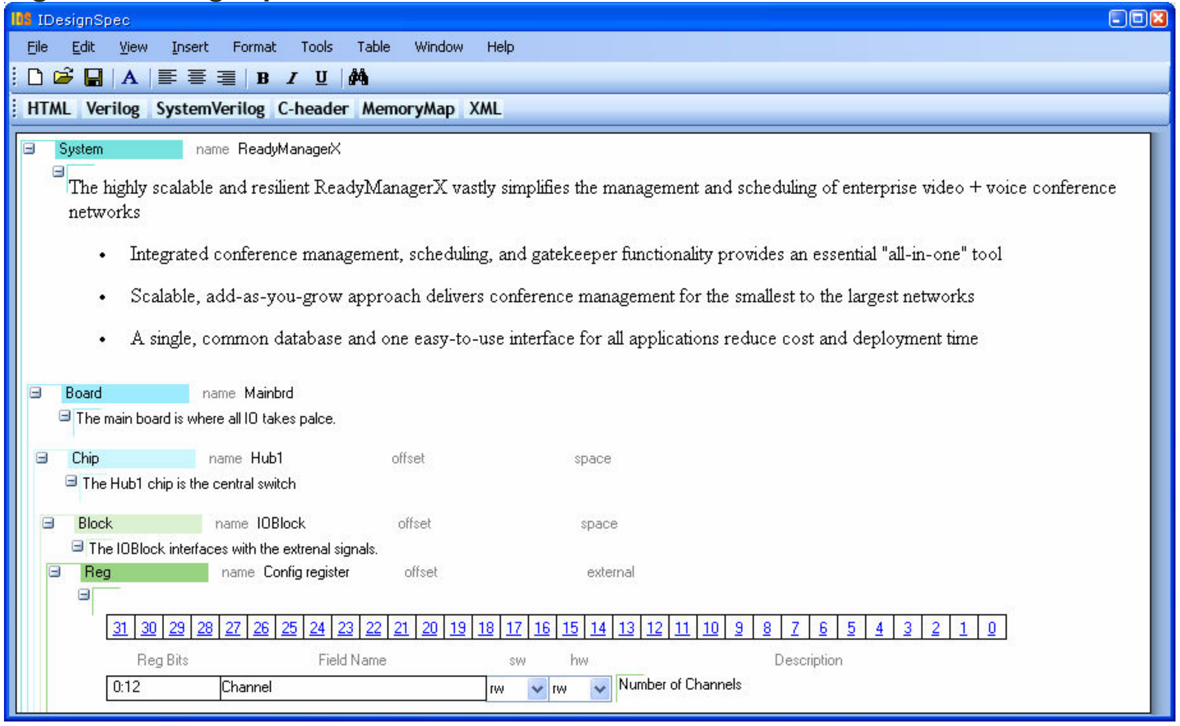
IDesignSpec™

IDesignSpec™ enables users to create all downstream views automatically.

IDesignSpec™ aims to solve the problems listed above. It brings the ease of a text editor to the system architect or designer to create an "executable specification". This specification contains everything that's needed to fully describe the design including documentation and register descriptions. All downstream views are generated from this single specification.

IDesignSpec™ is the world's first Specialized Editor™ for Systems or SoCs (System on Chips) with intelligence about the design process. Users are spared the botheration of learning of a new language or a new GUI with their own idiosyncrasies. IDesignSpec™ has been purposefully designed from the ground up to be lightweight add-on to any existing design process.

Figure 5: IDesignSpec™ screenshot.


IDesignSpec
File Edit View Insert Format Tools Table Window Help
HTML Verilog SystemVerilog C-header MemoryMap XML

System name ReadyManagerX

The highly scalable and resilient ReadyManagerX vastly simplifies the management and scheduling of enterprise video + voice conference networks

- Integrated conference management, scheduling, and gatekeeper functionality provides an essential "all-in-one" tool
- Scalable, add-as-you-grow approach delivers conference management for the smallest to the largest networks
- A single, common database and one easy-to-use interface for all applications reduce cost and deployment time

Board name Mainbrd

The main board is where all IO takes palce.

Chip name Hub1 offset space

The Hub1 chip is the central switch

Block name IOBlock offset space

The IOBlock interfaces with the extrenal signals.

Reg name Config register offset external

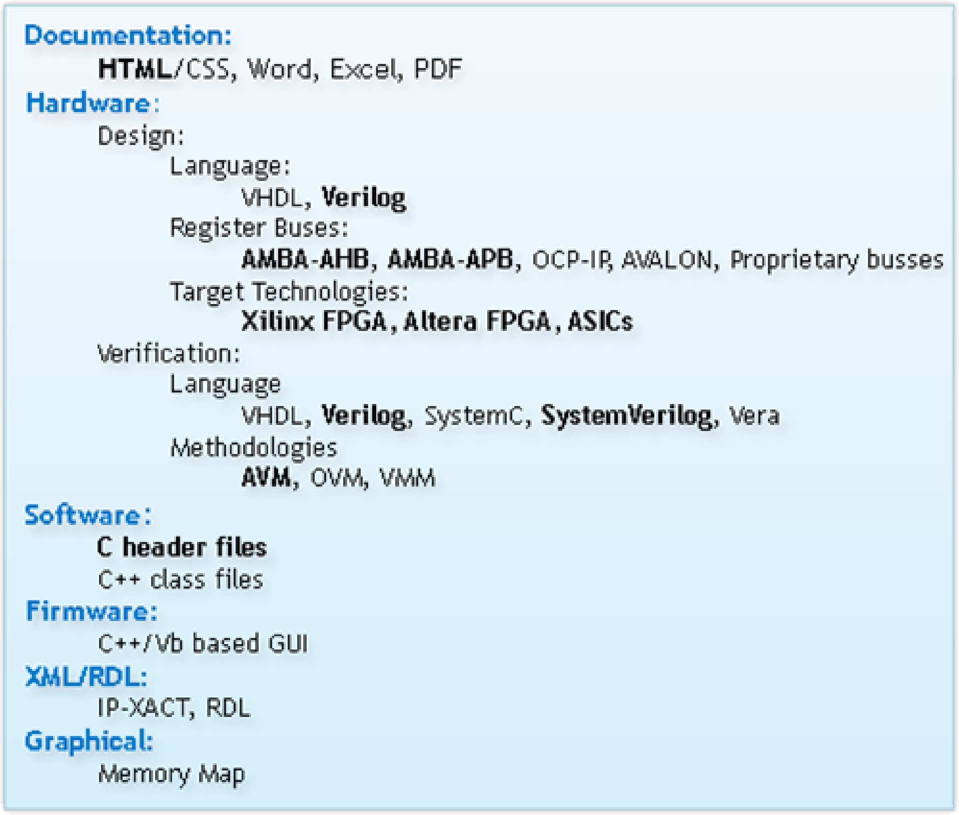
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Reg Bits	Field Name	sw	hw	Description
0:12	Channel	rw	rw	Number of Channels

IDesignSpec™ saves the design data in an XML format. This single source is then converted into the various views using XML Transformations. Changes made to the specification are reflected automatically in the various downstream views.

A single user or a team would use IDesignSpec™ to describe the design in terms of its functionality and register details. They would do so in a single editor and would not have to maintain multiple files in different formats. It stores its data in a human/computer readable XML format, so you always have direct access to your data. It uses the build in transformations to create a variety of outputs as shown in Figure 6 below:

Figure 6: Various views generated by IDesignSpec™.



Documentation:
- **HTML**/CSS, Word, Excel, PDF

Hardware:
- Design:
  - Language:
    - VHDL, **Verilog**
  - Register Buses:
    - **AMBA-AHB**, **AMBA-APB**, OCP-IP, AVALON, Proprietary busses
  - Target Technologies:
    - **Xilinx FPGA**, **Altera FPGA**, **ASICs**
- Verification:
  - Language
    - VHDL, **Verilog**, SystemC, **SystemVerilog**, Vera
  - Methodologies
    - **AVM**, OVM, VMM

Software:
- **C header files**
- C++ class files

Firmware:
- C++/Vb based GUI

XML/RDL:
- IP-XACT, RDL

Graphical:
- Memory Map

IDesignSpec™ is the first tool in the industry that not just supports but encourages design reuse. It enables you to reach system-wide optimization rather than local optimization in a specific stage of the Chip or System design process. Yet it's not a Framework. It's a simple editor, a “Constrained Editor” albeit.

This revolutionary product enables you to create correct by construction design. There is minimal chance of making mistakes and whole classes of bugs are eliminated.

Conclusions

The economic drivers for an efficient design process are significant. Design teams must focus on attaining system-wide design efficiency maxima. This not just takes out the drudgery of engineering and let designers focus on more creative pursuits, it also helps companies deliver defect-free products.

The ROI on improving the design process is measured in days, not weeks or months, and the benefits can be reaped for subsequent generations of the design. Tools like IDesignSpec^{TM} can be used to reap a reach harvest of productivity gains.

Biography

Anupam Bakshi has more than 17 years automating digital design processes in companies such as PictureTel Corporation, Avid Technology Inc. and Blackstone EDA Inc. He earned an MS in Electronics from Delhi University, MS in Computer Engineering from Northeastern University, and a High Tech MBA from Northeastern University. He founded Agnisys, Inc. in 2007, where he is currently the President. He can be reached at ab@agnisys.us.