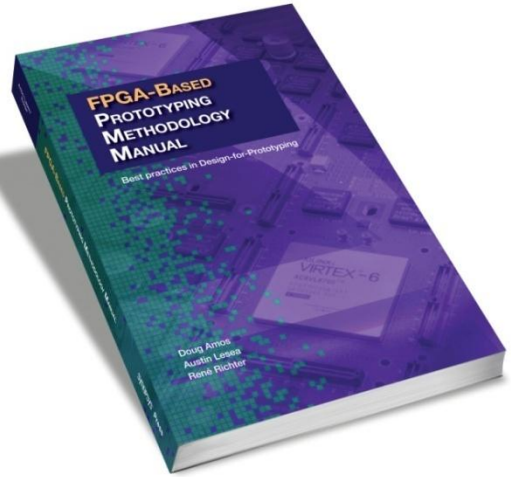
XILINX

SYNOPSYS®

Introducing the FPGA-Based Prototyping Methodology Manual (FPMM)



Best Practices in Design-for-Prototyping

What's the News?

- Introducing the FPMM:
  FPGA-Based Prototyping Methodology Manual
- Launch of new online community for prototyping
- Concept
  - Introducing Design-for-Prototyping
  - Collaboration between Xilinx & Synopsys
  - Capturing more than a decade of best practices
  - Additional reviews and content contributions from prototyping experts at 20 companies

About the Authors

Authors

Doug Amos & René Richter – Synopsys

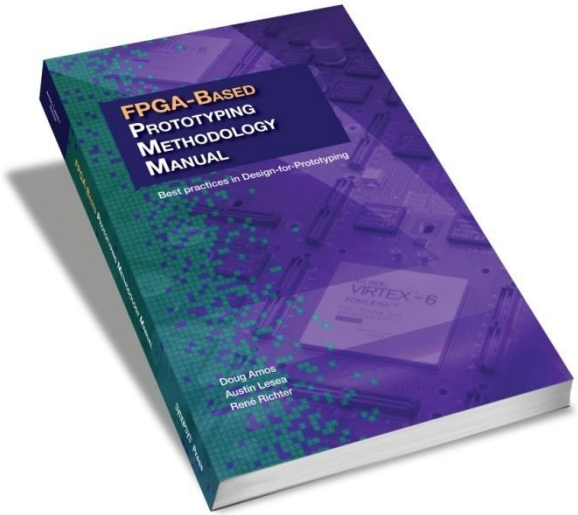
Austin Lesea – Xilinx

Contributors Include:

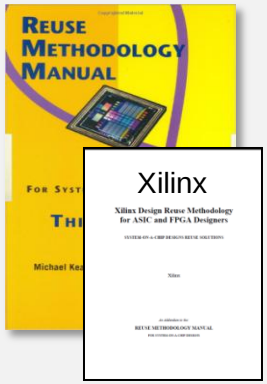
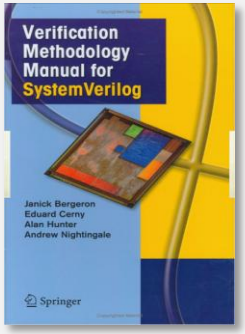
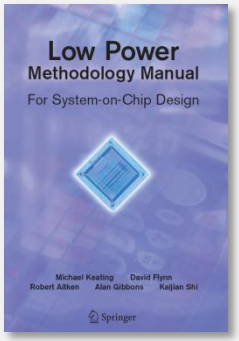
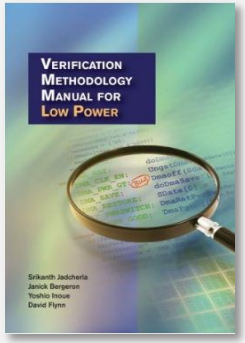
Freescale Semiconductors, LSI,
STMicroelectronics, Synopsys,
Texas Instruments, Xilinx & others

40 engineers in the review council

(Full list in acknowledgements pages)



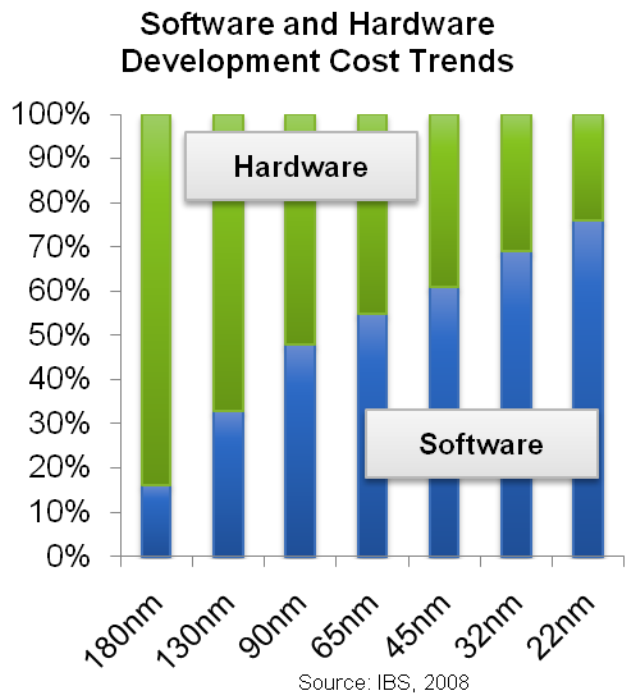
Synopsys' History of Methodology Development

	RMM	VMM	LPMM	VMM-LP	FPMM
Subject	Design Reuse	Functional Verification with SystemVerilog	Low Power SoC Design	Functional Verification of Low Power Designs	FPGA-Based Prototyping of SoC Design & Embedded Software
					
Partners	Mentor Graphics (Xilinx)	ARM	ARM	ARM Renesas	Xilinx
Release	1999-2002	2005	2007	2009	2011

Why Prototype?

System complexity & software

Increasing SW Development Effort

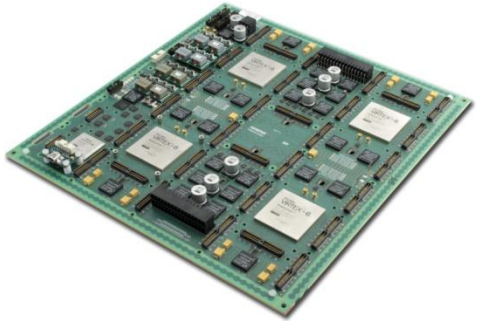


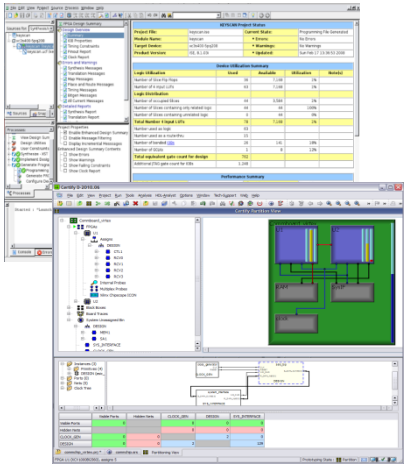
Software Content Dominates

The need is critical!

Great Prototyping Technology is Available

- Large, high-performance FPGAs
  - Xilinx® Virtex®-6 LX760 is excellent for prototyping
  - Synopsys has designed LX760 into all new platforms
- Reliable commercial prototyping hardware platforms
- Powerful software tools
  - Partitioning with high-speed serial TDM interconnect
  - Incremental and parallel implementation
  - High debug visibility





FPGA-based Prototyping Until Now

"FPGAs provide a platform for SoC development and verification unlike any other, and their greatest value is in their unique ability to provide a fast and accurate model of the SoC in order to allow pre-silicon validation of the embedded software."

"It is the hardware team's solution to the software team's problems"

– Helena Krupnova, STMicroelectronics

Advantages	Disadvantages *until now*
Find software bugs	FPGA expertise needed
Debug multicore designs	Partition across multiple chips
Real Time interfaces	Hard to achieve silicon speeds
Cycle-accurate	Complicated debug
Real World Speed	RTL must be available
Portable	RTL not optimized for FPGA
Low-cost replication	Considerable set-up time
Uses same RTL as SoC	

How Can Things Be Improved?

Design-for-Prototyping

Procedural Guidelines

Recommendation	Comment
Integrate RTL team and prototypers.	Have prototyping team be part of RTL design team, trained on same tools and if possible co-located.
Prototypers work with software team.	The software engineers are most common end-users for working prototypes.
Have prototype progress included in verification plan.	Branch RTL for prototype use only at pre-agreed milestones avoiding many incremental RTL dumps.
Keep prototype-compatible simulation environment.	At various points in the prototyping project it helps to compare results with original SoC simulations.
Clear documentation and combined revision control.	Track software and RTL changes for prototyping together using same source control.
Adopt company-wide standard for hardware and add-ons.	Common hardware approach avoids waste and encourages reuse.
Include Design for Prototyping in company RTL coding standards.	Most RTL style changes which make prototyping easier are also good for SoC design in general.

Technical Guidelines

Recommendation	***Comment/Detail***
Avoid latches	*Latch-based designs allow lower power SoC but are hard to time when mapped into FPGA.*
Avoid combinatorial loops	*Sometimes not seen in SoC RTL because of bottom-up design flow.*
Pre-empt RTL changes with `define	*`define and `ifdef included in source style guide to include/remove prototyping changes. Use single define for all RTL changes. Used to isolate BIST, memory instantiations, etc.*
Low-impact source changes	*Always use wrappers and make changes inside those. Replace files, rather than edit them. Back-annotate changes to real source.*
Write pure RTL	*Allow SoC tool flow to infer clock gating, insert test, apply low-power mitigation etc. avoid instantiating such measures directly into RTL source.*
Isolate RTL changes	*Make changes inside library elements (RAM, IO library etc.) rather than outside of them in the RTL structure. This improves portability, and places the prototyping code close to the original code it is replacing.*
Reuse file-lists/scripts	*Common project make-files for SoC and FPGAs with macro-driven branching for different targets.*

Design-for-Prototyping guidelines also help SoC design reuse

Procedural Improvements - Example

Table 22: Procedural recommendations in Design for Prototyping

Recommendation	Comment
Integrate RTL team and prototypers.	Have prototyping team be part of RTL design team, trained on same tools and if possible co-located.
Prototypers work with software team.	The software engineers are most common end-users for working prototypes.
Have prototype progress included in verification plan.	Branch RTL for prototype use only at pre-agreed milestones avoiding many incremental RTL dumps.
Keep prototype-compatible simulation environment.	At various points in the prototyping project it helps to compare results with original SoC simulations.
Clear documentation and combined revision control.	Track software and RTL changes for prototyping together using same source control.
Adopt company-wide standard for hardware and add-ons.	Common hardware approach avoids waste and encourages reuse.
Include Design for Prototyping in company RTL coding standards.	Most RTL style changes which make prototyping easier are also good for SoC design in general.

* Excerpt from FPMM, Chapter 9

Technical Improvements - Examples

Recommendation	Comment/Detail
Avoid latches	Latch-based designs allow lower power SoC but are hard to time when mapped into FPGA.
Avoid combinatorial loops	Sometimes not seen in SoC RTL because of bottom-up design flow.
Pre-empt RTL changes with `define	`define and `ifdef included in source style guide to include/remove prototyping changes. Use single define for all RTL changes. Used to isolate BIST, memory instantiations, etc.
Low-impact source changes	Always use wrappers and make changes inside those. Replace files, rather than edit them. Back-annotate changes to real source.
Write pure RTL	Allow SoC tool flow to infer clock gating, insert test, apply low-power mitigation etc. avoid instantiating such measures directly into RTL source.
Isolate RTL changes	Make changes inside library elements (RAM, IO library etc.) rather than outside of them in the RTL structure. This improves portability, and places the prototyping code close to the original code it is replacing.
Reuse file-lists/scripts	Common project make-files for SoC and FPGAs with macro-driven branching for different targets.
Memory compatibility	For each new memory generated for SoC, generate FPGA-compatible version. This could be alongside and options controlled with `define.
PHY compatibility	PHY blocks in SoC will need modeling in FPGA, or in off-chip test chip, if available. Keep this in mind when choosing PHY components for the SoC.
Design synchronously	Avoid asynchronous loops, double-edge clocking and other structures which do not map easily to FPGA.

PHY compatibility | PHY blocks in SoC will need modeling in FPGA, or in off-chip test chip, if available. Keep this in mind when SoC.

-edge clocking and easily to FPGA.

very long chains of owly in FPGA.

own block, preferably lete replacement by

boundary. Helps avoid d reset after

lex. If possible, cking options for o this.

ld latency but raints and increase

chronizers.

nd curious design for late design fix.

avior can be checked evel testbench useful

at different rates. n slower running ed.

nchronous, globally rototyping.

* Excerpt from FPMM, Chapter 9

Technical Improvements

Recommendation	Comment/Detail
Avoid latches	Latch-based designs allow lower power SoC but are hard to time when mapped into FPGA.
Avoid combinatorial loops	Sometimes not seen in SoC RTL because of
Pre-empt RTL changes with `define	`define and `ifdef included in so include/remove prototyping ch for all RTL changes. Used to is instantiations, etc.
Low-impact source changes	Always use wrappers and make Replace files, rather than edit changes to real source.
Write pure RTL	Allow SoC tool flow to infer cl apply low-power mitigation et measures directly into RTL sou
Isolate RTL changes	Make changes inside library el etc.) rather than outside of the This improves portability, and code close to the original code
Reuse file-lists/scripts	Common project make-files for macro-driven branching for di
Memory compatibility	For each new memory generat FPGA-compatible version. Thi options controlled with `define
PHY compatibility	PHY blocks in SoC will need mo off-chip test chip, if available. Ke choosing PHY components for the So
Design synchronously	Avoid asynchronous loops, double-edge clocking and other structures which do not map easily to FPGA.

PHY compatibility	PHY blocks in SoC will need modeling in FPGA, or in off-chip test chip, if available. Keep this in mind when SoC.
	-edge clocking and easily to FPGA.
	very long chains of owly in FPGA.

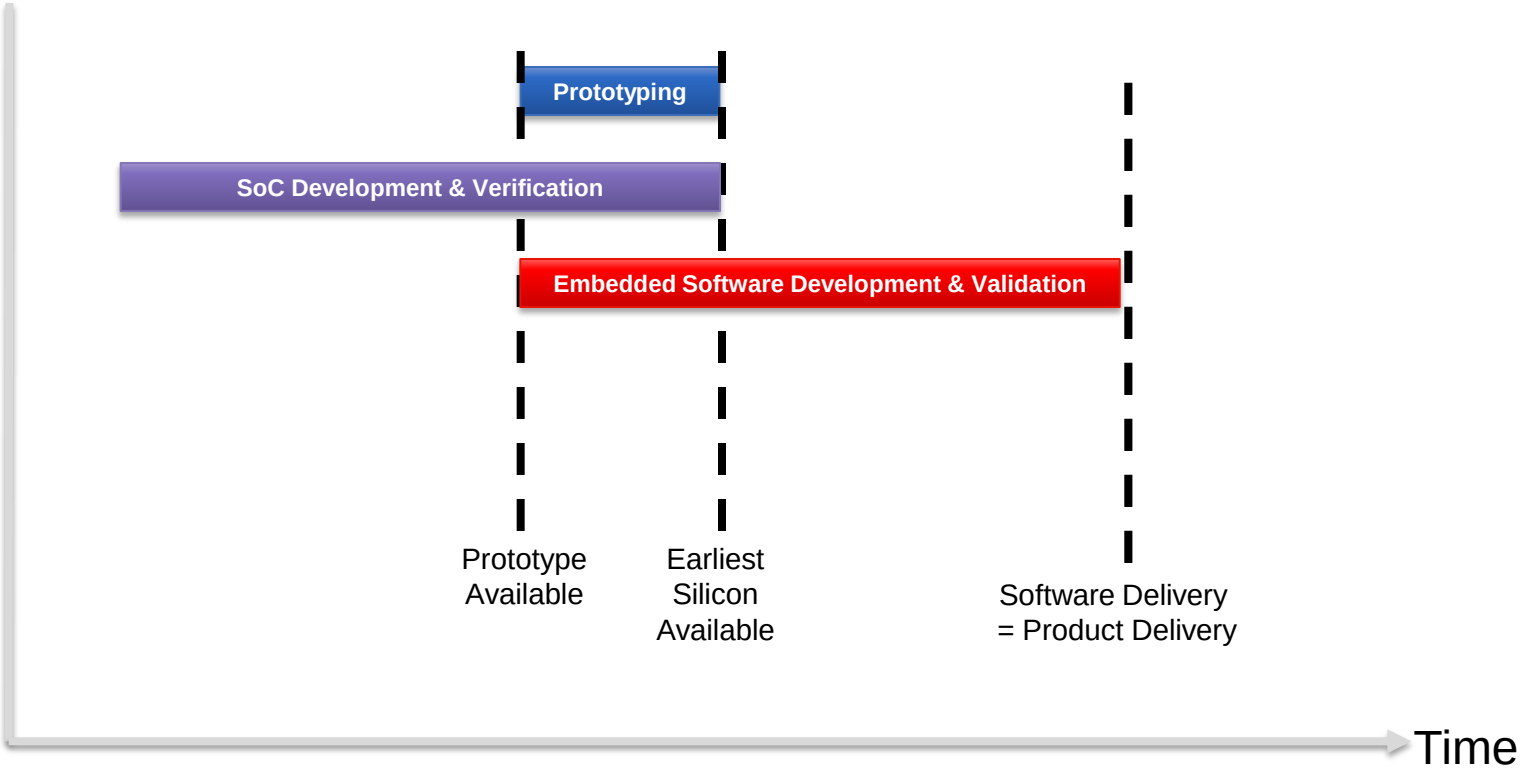
Synopsys Reuse Methodology (RMM) has become foundation of IP design and reuse.

Design-for-Prototyping aims to do the same for FPGA-based Prototyping

* Excerpt from FPMM, Chapter 9

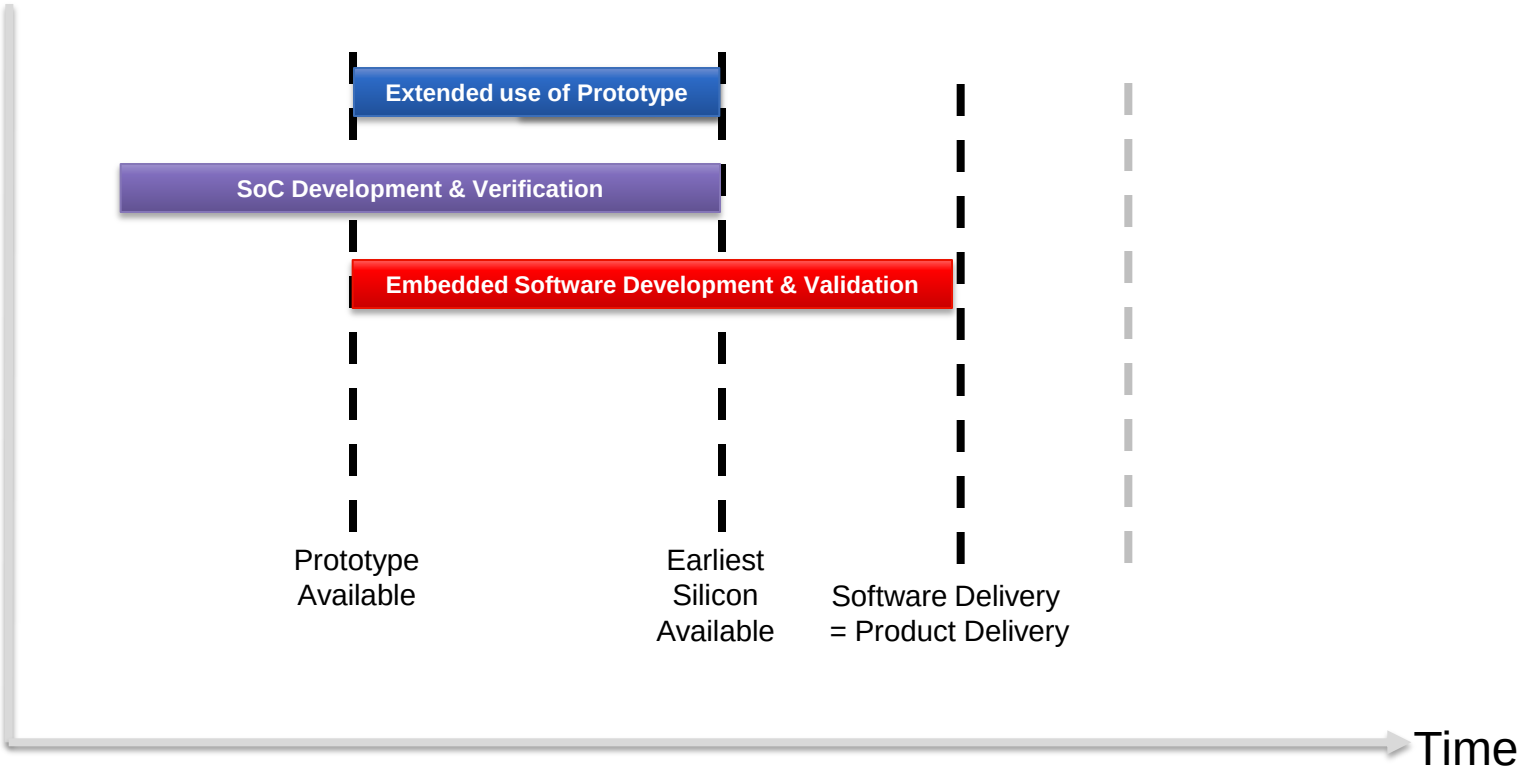
XILINX SYNOPSYS

The Effect of Design-for-Prototyping



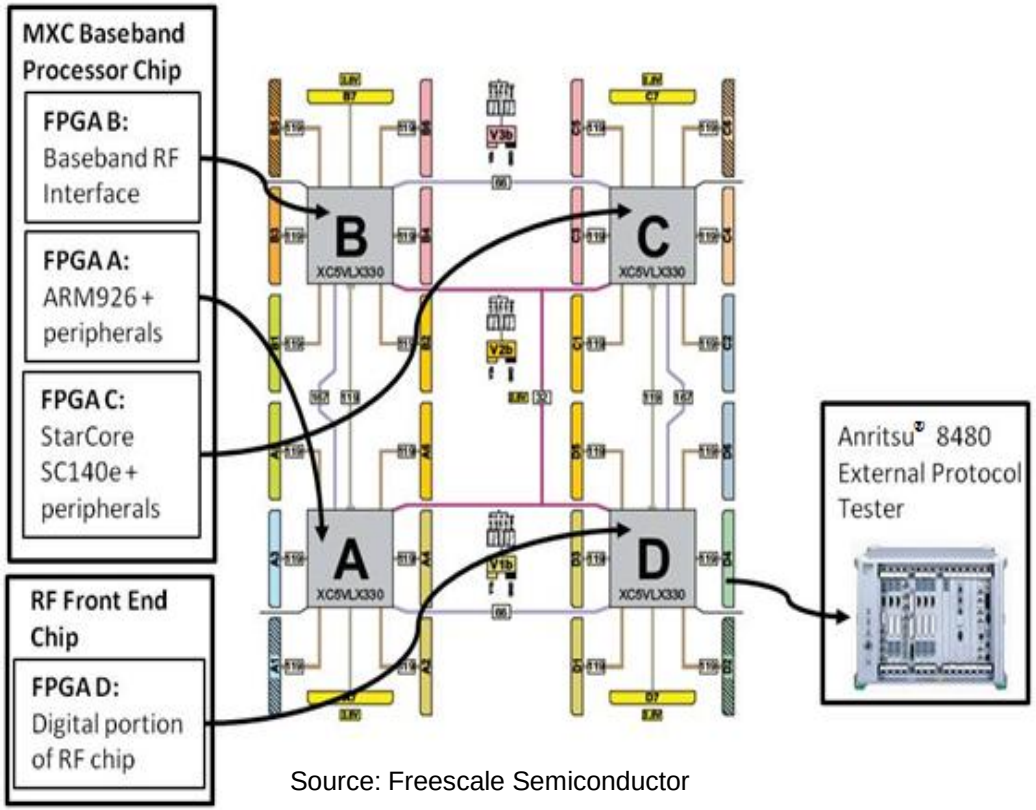
Prototyping is most useful pre-silicon

The Effect of Design-for-Prototyping



Earlier prototype saves time & improves quality

A Typical Example from the FPMM



Source: Freescale Semiconductor

Design Project

- Freescale Semiconductor
- Cellular communications chip
- RF, ARM® CPU, StarCore® DSP, Digital logic

Prototype was used for

- 3G Protocol testing
- System integration testing
- Driver development
- Early application software porting

Challenges

- Short Product life-cycles
- Long test runtimes (weeks on an emulator)

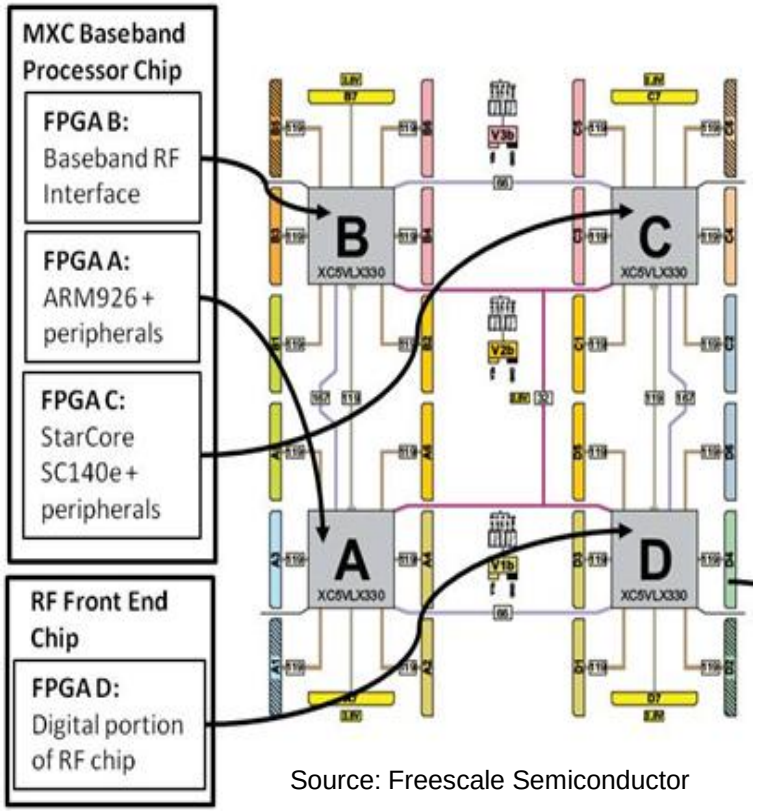
Techniques used

- Used techniques described in FPMM
- Partitioned by Certify onto HAPS-54 board
- Four Xilinx Virtex-5 devices

Results

- Protocol tests runtime reduced to 1 day

Project Results



Source: Freescale Semiconductor

- Completed protocol testing
- Accelerated project schedule
- Got HW, SW and systems engineers working together earlier

"the prototype proved its worth many times over"

"most important was the immeasurable human benefit of getting engineers involved earlier in the project schedule, and having all teams from design to software to validation to applications very familiar with the product six months before silicon even arrived"

- Scott Constable
Freescale Semiconductor Corp.
Austin, TX

Endorsements

"The FPMM is a great place to start, for management and engineers alike.
I wish we'd had this when we started."

– Brian Nowak, LSI, Inc.

"I recommend the FPMM to anybody considering prototyping as a validation vehicle for developing silicon products."

– Fernando Martinez, NVIDIA Corp.

The FPGA-Based Prototyping Methodology Manual

"Best Practices in Design-for-Prototyping"

500 pages in 15 Chapters

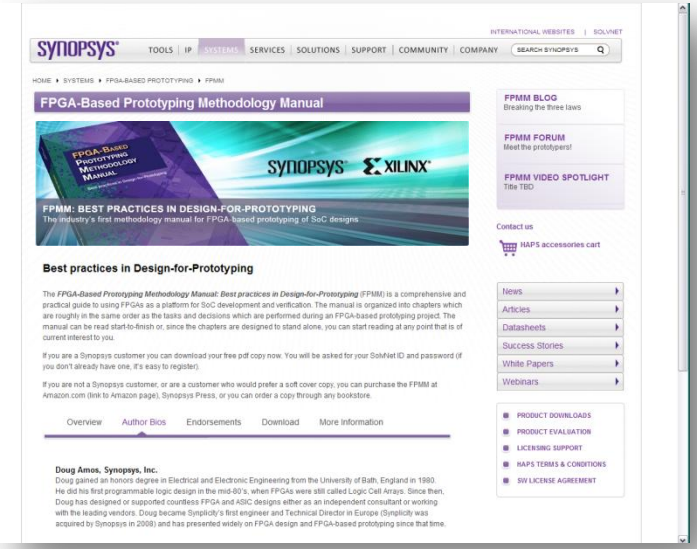
1	Introduction: the challenge of system-level verification
2	What can FPGA-based p rototyping do for us?
3	FPGA technology today: Chips and tools
4	Getting started
5	Which Platform? (1): build-your own
6	Which Platform? (2): ready-made
7	Getting the Design ready for the Prototype
8	Partitioning and reconnecting
9	Design-for-Prototyping
10	IP and high-speed interfaces
11	Bring-up and debug: the prototype in the lab
12	Breaking out of the lab: the prototype in the field
13	Prototyping + verification = the best of both worlds
14	The future of prototyping
15	Conclusions
Appx A	Worked example: Texas Insruments
Appx B	Economics of making prototyping boards

New Online Community for FPGA Prototyping

- The FPMM online web community
  - Starting the conversation between prototypers worldwide
  - The first go-to place for exchange of information and best practices
  - Driven by FPMM Review Council

- www.synopsys.com/fpmm

  (Live on March 2, 2011)



Summary
FPGA-Based Prototyping Methodology Manual

"Best Practices in Design-for-Prototyping"

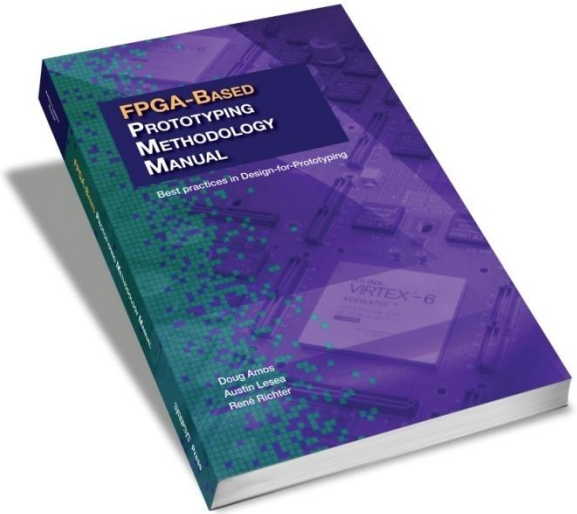
Authors:	Synopsys and Xilinx
Other Contributors	STMicroelectronics, Texas Instruments, Freescale, LSI and others
Review Council	Over 40 practitioners worldwide

Release date: March 2, 2011

Free eBook download (www.synopsys.com/fpmm)

Printed book from retailers including Amazon.com

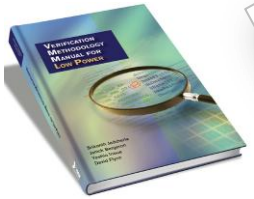
Online community launch date March 2, 2011



Thank You for Your Interest

To learn more about other **Synopsys Press** methodology manuals and educational publications, visit www.synopsys.com/synopsys_press

Technical Series



VMM For Low Power

Synopsys Journal

FPGA-Based Prototyping Methodology Manual

LPMM Chinese Edition
LPMM,VMM-LP Japanese Editions

Business Series

10 Commandments For Effective Standards