EDP-2002

Scaling Methodology

Dan Smith
Director HW Engineering
dsmith@nvidia.com

Introduction

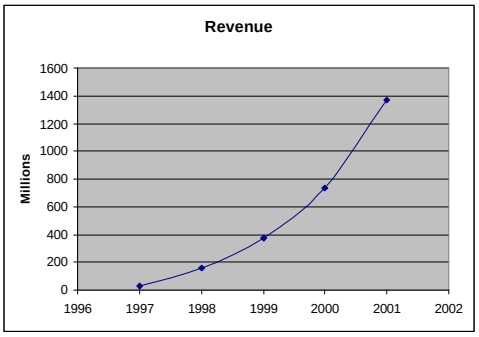
- NVIDIA
- NVIDIA's growth
- Methodology Group
- NVIDIA's methodology beginnings
- How the methodology has changed
  - Philosophy
  - Practicality
- Examples
- What's the future like

NVIDIA Characteristics

- NVIDIA is the global leader in advanced graphics processing technology for mainstream platforms
  - GPU's, Platform Chipsets, XBOX chipset
- Frequent design refreshes
- Large, complex designs
- Verilog RTL and C++ behavioral
- COT back-end
- Design team primarily in Santa Clara, but reasonable number of people in other areas

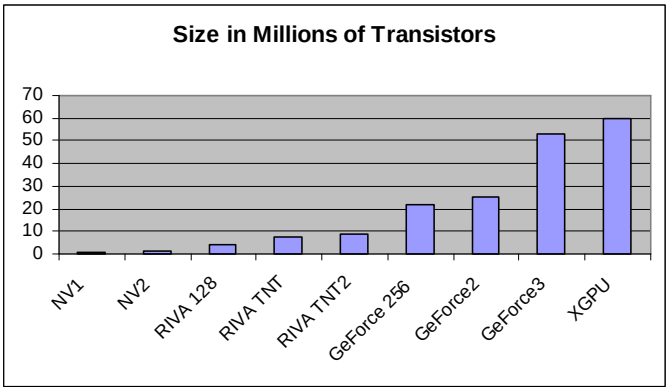
NVIDIA Growth

- NVIDIA has grown significantly in many dimensions over the last 5 or so years



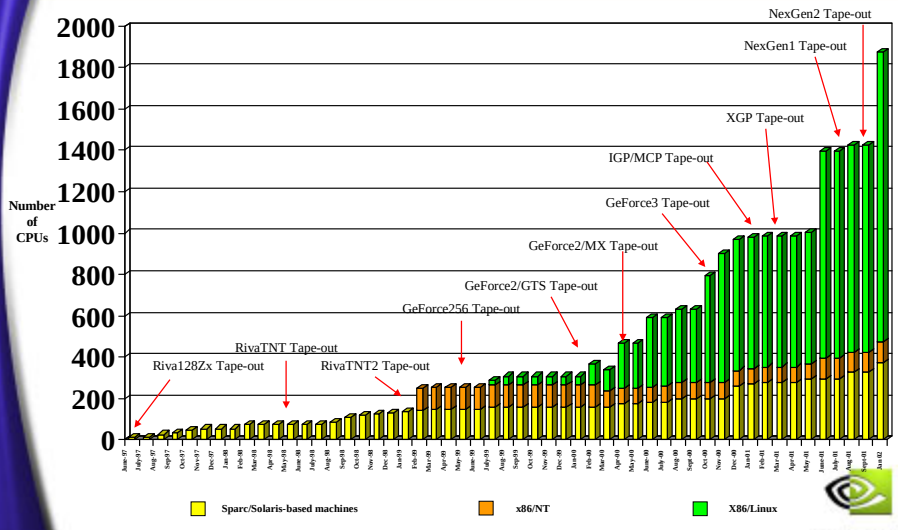
Design Growth

- Many simultaneous design projects
- Design complexity is growing



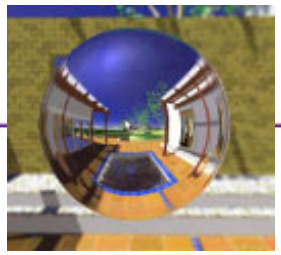
EDP-2002

Compute Server Growth



EDP-2002

Methodology Group



- Central group responsible for standardizing, enhancement, and automation of NVIDIA's methodology
- Common methodology for platform and GPU products
- Practical approach
- Manage globally used EDA vendor tools

NVIDIA.

Methodology Beginnings

- Several generations of GPU's done by same team
  - One designer per unit
  - No specialization
  - Little turn-over
  - Design code base and tools code base copied from project to project
- Consistent high level methodology
- Lots of variability at the implementation level
- Many "standard" methodologies in use simultaneously on the same chip
- Custom tools crafted for a specific chip or unit

NVIDIA.

Forces Driving Methodology Change

- Volume
- Time to Market
- New Markets
- Hitting all market segments
- Lowering Cost
- Competition
- Engineering staff growth
- Increased design complexity

NVIDIA.

How to scale design methodology

- Continually find better ways to do design
- Standardization
  - Reduces the learning curve
  - Provides for better reuse
  - Helps ensure consistent quality
  - Combines the best practices from all engineers
  - Allows global upgrades/enhancements
- Documentation and training
- Automation
- Communication
- Continue to hire great engineers
- Specialize

NVIDIA.

What to Standardize

- Anything that takes engineering time is a candidate for standardization
  - Verification environments
  - Synthesis/Timing/Layout flows
  - Emulation
  - Manufacturing test methodology
  - Formal verification

- Custom scripts, process flows, and software tools should be written only once
- Selection and use of commercial EDA tools
- All design rule checks

Drawbacks to Standardization

- Can slow adoption of new techniques if they have to work globally before being used at all
- Requires dedicated staff to support
- Different chips may be best designed with different techniques
- Difficult for chip designers to customize tools quickly to get chip's out.

How to Standardize

- Programmers develop robust tools and flows, scalable across projects
- Tool experts working with chip designers and programmers
- Hire great chip designers, encourage them to innovate, and leverage the work they do across other projects
- Plan ahead, develop and acquire technology, and be opportunistic is deployment

Standardization Hurtles

- Picking the right methodology and getting everyone to agree on the selection
- Disruption to projects during transition period
- Projects sometimes diverge from common flows as they get close to tape-out
- Common flows/tools usually more complex than those written to work on a particular project
- “Owner” of flows/tools doesn’t usually have the same sense of urgency as those on a project

Structured Tool & Flow Development

- Unlike EDA companies, it does make sense for chip companies to develop custom code for particular chips or design types.
- All tools and flows should use common set of libraries for accessing information
- Clear designation between chip specific and chip independent code. Subclass chip independent classes to extend or specialize functionality
- Tool independent configuration files for all chip specific information
- Consistent set of tools and libraries “frozen” for each design

Synthesis - Legacy

- Originally
  - >20 different ways to go about automating synthesis
  - Synthesis responsibility with designer of unit
  - Some were attempts to standardize and used by many modules within a unit
  - Impossible to change things globally, like cycle time
  - Many copies of design quality scripts
- Chip designers had problems moving from block to block as they had to relearn these relatively complex environments
- No documentation meant that new engineers had a steep learning curve

Synthesis - Common

- Merged good characteristics from all environments into one automated and documented flow, used across all projects
  - Automatic dependency generation
  - Standard script structure
  - Centralized default constraints
  - Centralized library definitions
  - Automatic invocation of design quality scripts
  - Common code base, freezable and tweak-able for individual projects if necessary
  - Simple setup but completely customizable without breaking automation
  - Automatic checking of failure conditions

Resources

- NVIDIA has always had a shared pool of resources
  - Compute servers: linux and solaris
  - Network and file servers
  - EDA tool licenses
- Sharing requires additional priority schemes as the number of projects grows
- Simple scaling of resources breaks all sorts of things
- Need to get more sophisticated about our use of resources

Manufacturing Test - Traditional

- 100% scan
- ATPG patterns for all stuck-at faults
- At-speed Functional patterns for detecting all speed faults and doing speed characterization
- Test cost expensive per chip
  - Large chips with huge patterns sets
  - No characterization or optimization of shift speed
  - Functional patterns require tight timing tollerances
- Minimal FA

Manufacturing Test - Today

- Automatic structural tests for transition faults
- Automatic tests for chip delay characterization
- Better process for isolating faults based on structural tests – both transition and static faults
- Techniques for compressing vectors
- New techniques for improving coverage
- Logic optimized to run structural tests fast
- Characterization of various pattern sets to optimize test programs
- Requires much larger investment in time, engineers, and capital

Manufacturing Test - Future

- Continual drive towards lower cost test
  - Timing critical testing moved on-chip
    - Critical timing relationships for functional testing
    - Analog interfaces
    - PLL
  - Structural vector compression
- Higher quality vectors
  - Bridging fault models
  - Higher transition fault coverage
- Better fault isolation techniques

NVIDIA.

Tracking the Design Process

- Design process is a complex and involved process taking many people many months to accomplish
- Problems:
  - Engineers and managers need to understand the flows and where their chip is in the flow.
  - Need to better manage revisions of design files to ensure that the proper set was used for building the chip, as well as for all verifications steps along the way.

NVIDIA.

Design File Revisions - Traditionally

- Files are kept in a source code control system
- Latest revision number of a file is considered the revision to be used.
- Individual engineers “sync” to a set of files and perform some activities which might span many days, and then check in the resulting files.
- Leads to confusion on what tools were run with what revision of the files. Relies on very careful engineers with an in-depth knowledge of the process to manage successfully
- Big waste of time and hit to schedules

Design File Revisions - Future

- Mechanism for formally specifying a process
- Database for tracking:
  - execution of process activities
  - revisions of design files involved in the process flow executions
  - checkins to source code system

Tracking Tools

- Tools will read tracking database and process description files. These can be used to:
  - Verify before running a process that all the input files are available and up to date.
  - Automatically launch process flows when input files have been updated
  - Ensure all verification steps have been run on the file revisions used to tape out a chip

Conclusion

- Nothing is a substitute for intelligent, diligent engineers working well together
- Well thought out, standard, automated processes will enable growth without compromising quality or efficiency