Building Design Process in a Startup Company

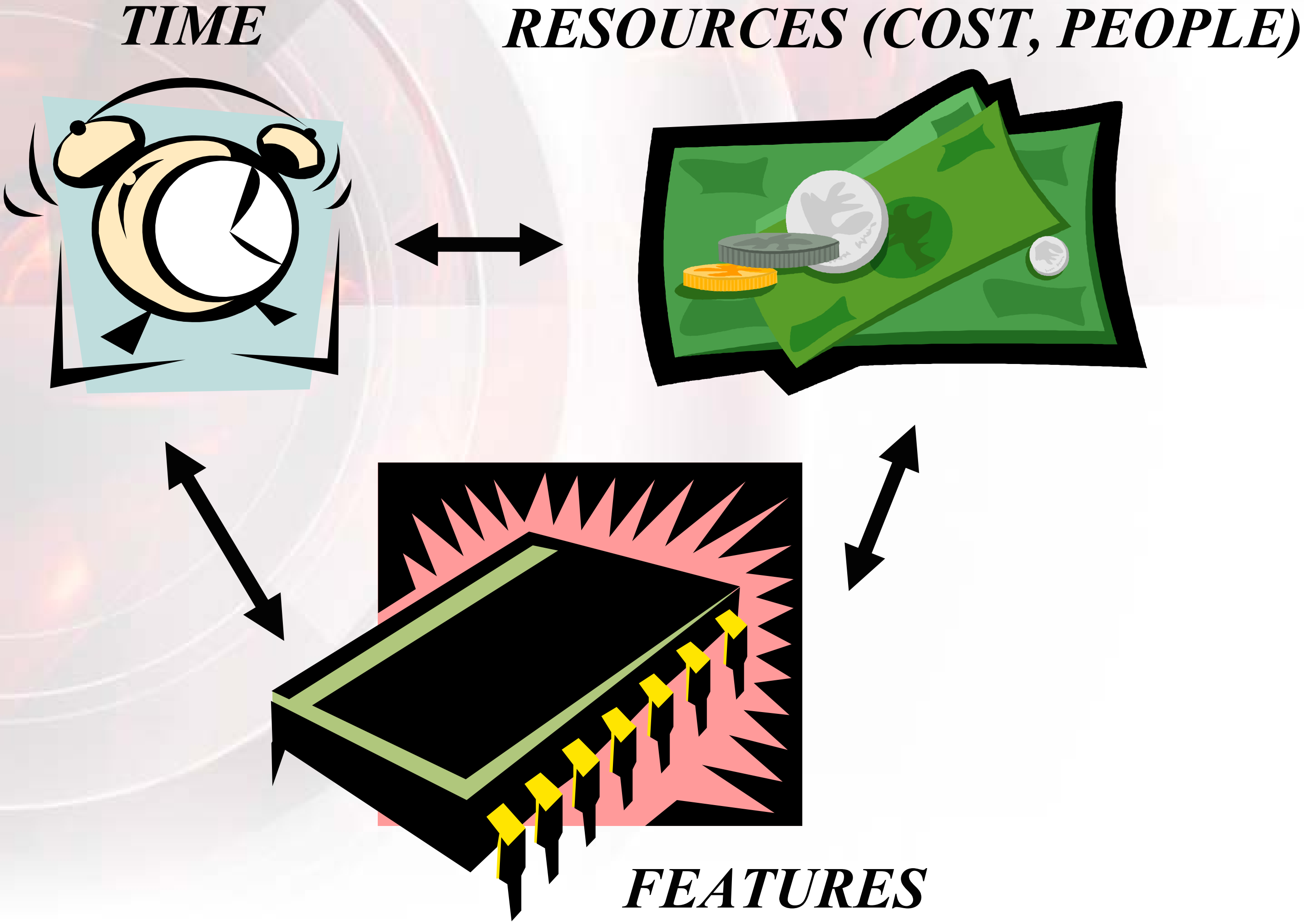
David A. Gates
ATI Research Silicon Valley

EDP-2003

Outline

- Motivation
- Startup Challenges
- Functional Verification
- Design Process Evolution
- DV Architecture
- DV Implementation and Experience
- Conclusions

Project Tradeoffs



TIME

RESOURCES (COST, PEOPLE)

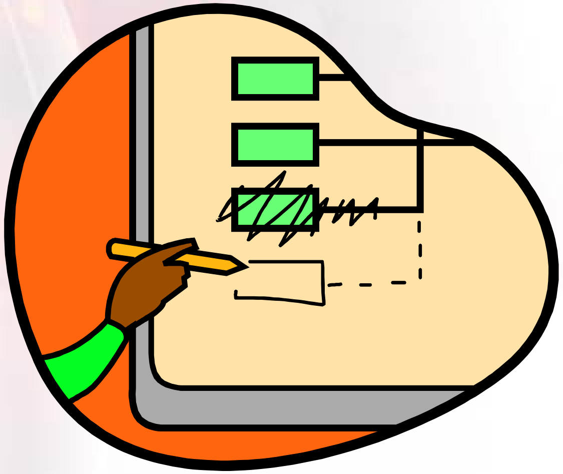
FEATURES

Why Build a Design Process?

- Design Process brings Chaos under Control.
- Design Automation makes Tradeoffs Easier!
  - More exploration ➨ more, better **FEATURES**.
  - Lower **TIME** of iterations / design changes.
  - Lower **COST** of finding / fixing human errors.
  - Enables more work by fewer **PEOPLE**.

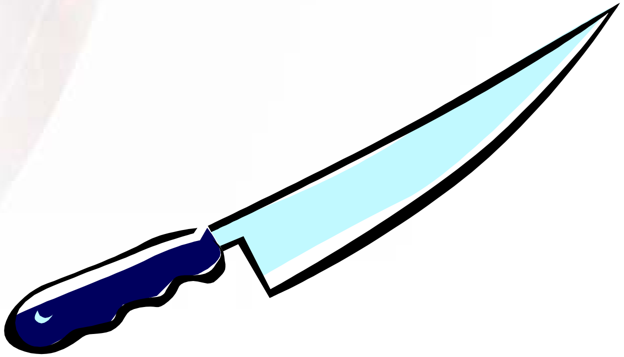
Startup Challenges

- Change is Inevitable
  - Small company inherently more agile.
  - Redirect when goals can't be met.
  - Design process must be MODIFIABLE.
- Time is of the Essence
  - Win the race or die.
  - Process quality suffers.
  - BUT 2nd product can't take as long as 1st.
  - Design process must be CAPTURED.

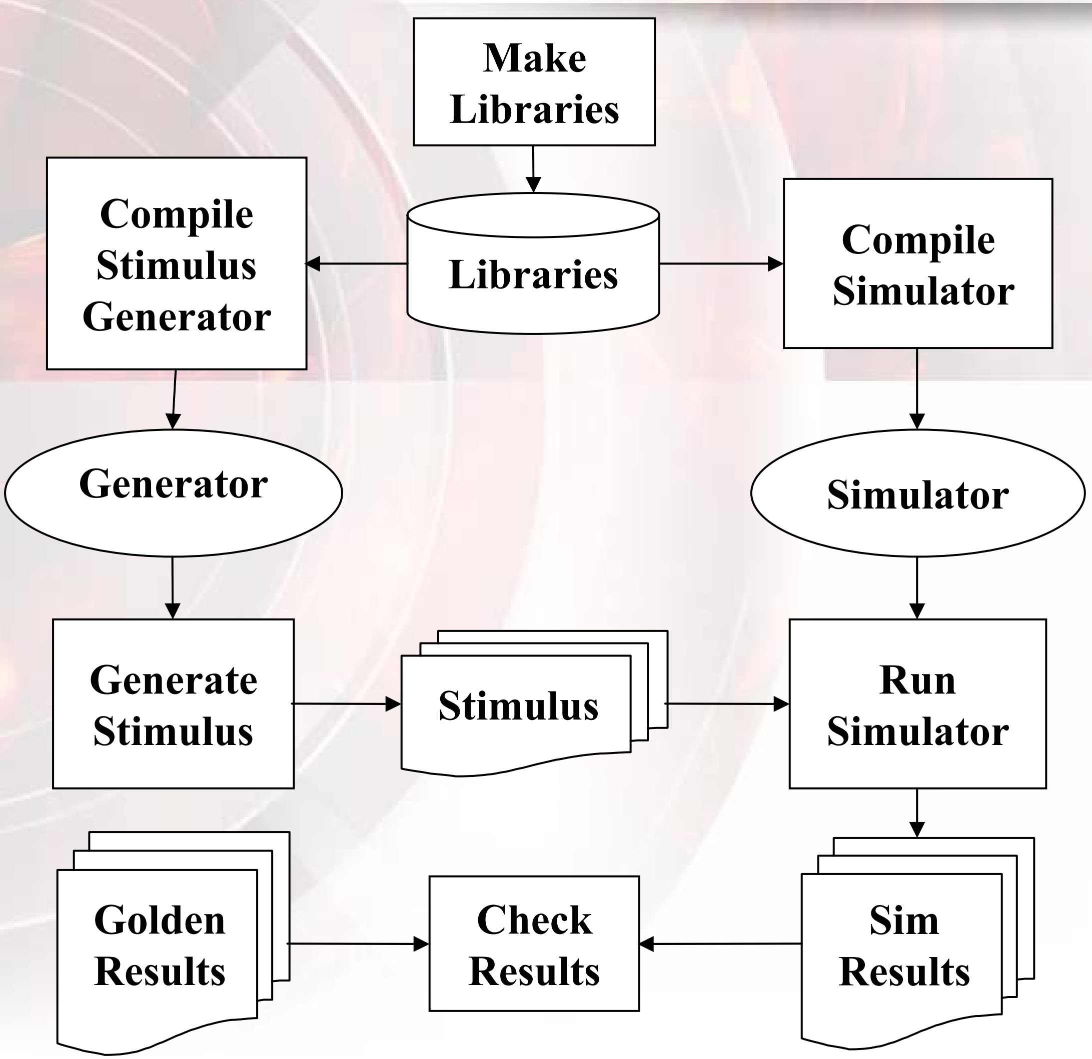


Startup Challenges (Cont.)

- Few Hands make Heavy Work ☺
  - Deep Experience balances Limited Workforce.
  - “Best Practices” differ from one to the next.
  - Design process must be ABSTRACTED.
- Cutting Edge Design
  - Must discover right process via experiment.
  - Design process must be EXTENSIBLE.



Functional Verification – Example Flow



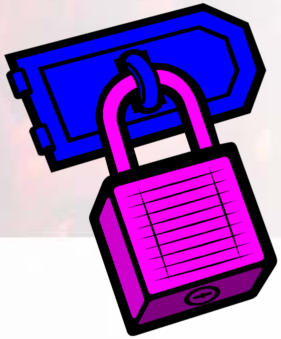
Design Process Evolution

- Piecemeal Automation
  - Developed in Isolation
  - Diverse Implementation
  - One Task per Piece
  - No One Knows the Flow

- Manual Flows / Checklists
  - Stitch Together Pieces
  - Trial-and-Error Creation
  - Mystery Dependencies
  - Clobber and Rebuild for Safety
  - Observable, Executed One Step at a Time

Design Process Evolution (Cont.)

- Hard-coded Scripts
  - Automate Manual Flow via Direct Translation
  - Difficult to Spot Bugs
  - Locked to User, Project, Block, Activity
  - Difficult to Select Alternate Flows or Entry Points

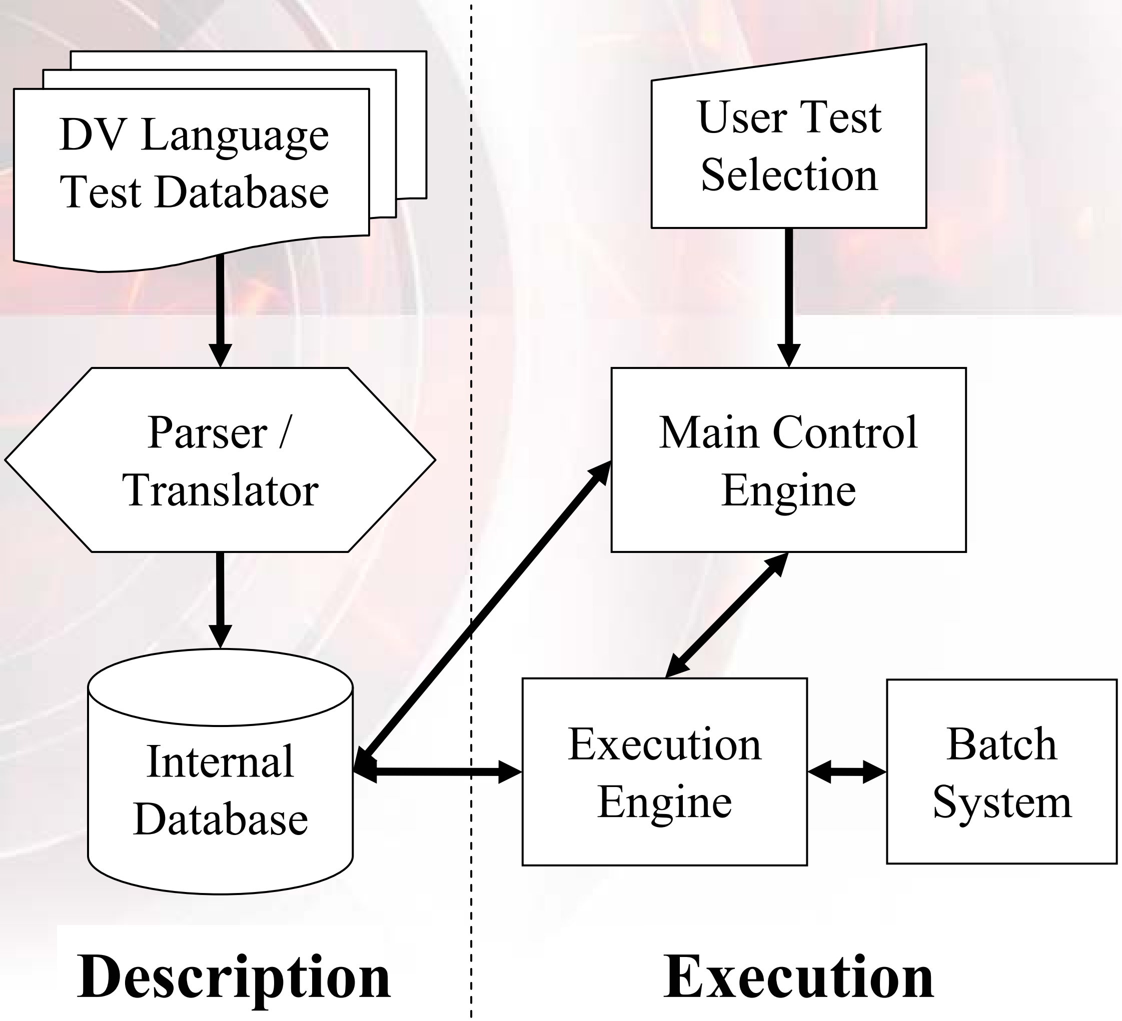


- Generalized Scripts
  - Work Across Users, Projects and Blocks

DV Requirements

- Flexible
  - Handle multiple flows and levels of detail.
- Extensible
  - Add or modify flows specific to one area.
- Formatted
  - Maintained by design and verification engineers.
- Automated
  - Run tests or sets of tests interactively / as batch.
- Observable
  - Easily follow flow of control to spot problems.
- Modifiable
  - Tool can be updated as project progresses.

DV Architecture



DV Language: Elements

- Configurations : Design Structure
  - Configuration = Components + Views
  - Components include Blocks, Interfaces, Monitors
  - Views include Behavioral, RTL, Netlist
- Tools : Verification Actions
  - Tool = Directory + Command + Arguments
  - Parameterized with Block, Test, and Configuration names
- Flows : Verification Process
  - Flow = Order / Dependencies
  - Can use Sequential, Parallel and Selective execution
- Tests : Tie Together Structure and Action
  - Test = Configuration + Flow
  - Tests can pass Tools unique Parameters

DV Language: Example

```
// Describe test.
test t1
  conf = c1
  flow = default
endtest
// Describe configuration.
conf c1
  block1 = rtl
  block1_mon = live
endconf
// Describe tools.
tool make_lib
  dir = $TOP/lib
  cmd = make
endtool
```

```
...
// Describe flow.
flow default
  make_lib
  par
    seq
      make_gen
      run_gen
    endseq
    bld_sim
  endpar
  run_sim
  chk_sim
endflow
```

DV Execution

```
Use ARGs to set INTERACTIVE/BATCH mode.
Use ARGs to get list of BLOCKs.
foreach BLOCK
  Use ARGs to get list of TESTs.
  foreach TEST
    Query Database to get CONF/FLOW.
    Dispatch FLOW to Execution Engine.
  end
end
```

DV Implementation

- User-Interface & Execution
  - 4000 lines of PERL
  - rapid development and evolution
- Parser / Translator
  - 1000 lines of lex / yacc / C
  - better translation speed
  - set of objects (conf, tool, flow, test) is **fixed**
  - set of properties on objects is **open**
- Batch Execution uses Platform LSF
- Verified Nintendo GAMECUBE 3D Pipeline

DV Experience

- Phased Introduction
  - Initial Rollout Follows Architecture ②
  - Batch Execution Added to Manage Machines and Licenses
  - Active Use suggested Optimizations & Extensions
  - Changes made to both Description and Execution
- Process Capture
  - Avoid Proliferation of Arguments and Environment Variables
  - Encapsulate Detailed Argument Settings
  - Don’t Use Environment Variables to Make Choices!
- Pattern Finding
  - Search Database looking for Commonality
  - Optimize Language to Simplify Patterns
  - Document and Communicate to Improve Process

Conclusions

- Startups are Flexible, Fast and Fearless.
- Automated Process Essential for Startup Success.
- Verification Process Must Handle:
  - Design Structure (Data)
  - Verification Flow (Methods)
- DV Evolved from Ad Hoc Verification Solutions
- DV Captures Verification Process
- DV Still Used for Verification at ATI