Flow Engineering for Physical Implementation: Theory and Practice

Steve Golson Trilobyte Systems Pete Churchill Magma Design Automation

June 10, 2008 45th Design Automation Conference

Define: flow

flow n.

- 1. the act or manner of flowing
- 2. the smooth motion characteristic of fluids
- 3. anything that flows; stream or current
- 4. a continuous production
- 5. the sequence in which operations are performed

Define: audience

- design engineer
- EDA vendor

manager



Define: flow

Management perspective

flow n.

a four-letter word often associated with work that can't be scheduled and seems to never end

CAD perspective

flow n.

a four-letter word representing an endless project whose result is underappreciated and underused by the design community

Design engineer perspective

flow n.

a four-letter word representing a necessary tool which you are not given time to work on, even though what you have is not adequate or complete

EDA vendor perspective

flow n.

a four-letter word representing all the stuff that customers do with other vendor's tools, often making us look bad

Flow (our definition)

flow n.

 all the files and infrastructure required to implement a methodology

"The CAD team has created a new flow for us to use on our next design."

a sequence of tool invocations

"You need to run the flow for clock synthesis."

Define: physical implementation

- "IC design" or "chip design" or "IC implementation"
- Everything from RTL → GDSII
- Everything other than functionality
- Not writing RTL, not functional verification
- Hold function constant, change the representation

Define: engineering

engineering n.

The application of scientific and mathematical *principles* to practical ends such as the design, manufacture, and operation of efficient and economical structures, machines, processes, and systems.

The Principles of Flow Engineering

#1 Flows are hard

#2 Flows are iterative

#3 Flows are automated

#4 Flows fail more than they succeed

Flows are expensive

 We spend as much on flow as we do on EDA tools themselves

> "The IDMs and large fabless companies are spending between \$1 and \$2 on CAD support activities for every dollar they spend on the tools themselves. In fact, there are nearly as many CAD engineers inside the semiconductor industry as there are in all of the EDA vendors... combined!"

> > Thomas Harms Infineon Technologies SCDsource 12 Dec 2007

We've been designing chips for a long time Why hasn't this been solved already?

EDA perspective:

- There's no money in it
- Vendors can only work with their own tools

CAD perspective:

- Too many tools to support them all
- Too many process nodes/libraries to support them all
- Customization capabilities are difficult to include

"We build chips, not flows"

- This is only true if you reuse a flow
- Flows are only reusable when they are *designed* to be reused
- Making changes and improvements to your flow for reuse will *not* make your current chip project finish sooner...

however your next chip will finish sooner...

but only if you actually reuse your flow and don't start over from scratch!

Flow must change/adapt as the design matures

- Prototype flows needed in the beginning
- Production flows needed once implementation matures
- Hopefully we are building the process while the RTL is still being written

- EDA tools are complex and expensive
- Combining them into a flow is *also* complex and expensive
 - Always lots of time pressure
 - it seems there is never any time to develop what we need
 - instead, we have to make do



"Can't we use the flow from the last project?"

"Nope, we just hacked something together to get the chip out."

"You should just use our recommended flow."

1. Management

2. CAD

3. EDA

4. Design

Management
 CAD
 EDA

4. Design



Management
 CAD
 EDA

4. Design

Principle #2: Iteration

Iteration is an overloaded term

We have two competing definitions:

iteration n.

 hand-fixing a single netlist or placement or routed block in order to meet your design goals (e.g., timing, DRC)
 These are *micro iterations* Goal: single iteration convergence using automated flow

iteration n.

 running your automated flow when changes occur to RTL, floorplan, constraints, etc.
 These are *macro iterations* Reality: many many runs

Principle #2: Iteration

Physical implementation...

 is *not* like building a bridge because you only build a bridge once

Physical implementation...

is like building a *machine* that builds a bridge

Imagine building a new bridge *every day!*

Our goal is to build a *flow* that builds a chip (RTL \rightarrow GDSII)

Principle #2: Iteration

- Iterate early, iterate often
- Iterations will overlap
- How fast can you iterate?
- Each iteration must be
 - repeatable
 - reliable
 - robust

How can we accomplish this?



"The RTL freeze was three weeks ago."

"What about the five changes since then?"

"Those weren't releases, they were ECOs." Management
 CAD

3. EDA

4. Design

Management
 CAD
 EDA

4. Design

- Management
 CAD
 EDA
- 4. Design

Principle #3: Automation

Automation means your computer does the work

- Scripted, not GUI
 - No GUI implies no hand edits
- Batch, not interactive
- Self-checking, not user judgment
- Repeatable, reliable, robust
- Predictable results

Can I sleep while this is running?

Principle #3: Automation

"Success in physical design will come from relentless automation."

Paul Rodman DAC 2002

Quote Quiz

"We've got scripts to automate your ECO flow. The documentation is on our website."

"146 pages of documentation. How long will it take us to learn this?"

"I don't need an ECO flow. Get me a six-pack of Red Bull and I'll have it ready in the morning." Management
 CAD
 EDA
 Design

Management
 CAD
 EDA



1. Management

- 2. CAD
- 3. EDA

4. Design

Principle #4: Failure

Failure is an overloaded term

We have two competing definitions:

failure n.

when a particular flow (sequence of tool invocations) does not generate tapeout-quality results
 Reality: Most iterations *fail* for one reason or another
 Reality: Many many more failures than tapeouts

failure n.

 when a particular flow (all the scripts and infrastructure) is painfully awkward and difficult to use
 Reality: Many flows *fail* although we rarely admit it

Principle #4: Failure (of scripts and infrastructure)

Fallacy: Flows never fail

- You always get the chip out the door, so by definition your flow "works"
- The only flows that fail are the ones that are developed *without* a chip

Who develops flows without a chip project?

Principle #4: Failure (of scripts and infrastructure)

Dangerous fallacies:

- Our flow is a success
- The CAD department's flow is a failure
- The EDA vendor's flow is a failure

Outrageous truths:

- Failure of your flow infrastructure is *bad* failure
- Acknowledge what is broken
- Consider what else might work

Principle #4: Failure (of sequence of tool invocations)

- This is good failure
- Failure is inherent in the process

Failure is your friend

Principle #4: Failure

Success

- How to define success?
 - Must have something to measure
 - Zero errors (DRC, timing, etc.)
 - Success metrics will change as the design matures
- We should not expect *first* pass success
 - rather, we want *last* pass success
 - eventually, every pass success

Quote Quiz

"Look at the cool wrapper we built around the tool. Now it's much easier for you to use."

"I got a segmentation fault. What do I do now?"

"Can you provide a test case? Without your wrapper?" Management
 CAD
 EDA
 Design
 Management

- 2. CAD
- 3. EDA
- 4. Design

- Management
 CAD
- 2. CAD 3. EDA
- 4. Design

Principle #4: Failure

All tools suck

Goals

Building from the 4 principles:

- What are the goals for our flow?
- How should it look and work?
- Goals are used to drive decisions during flow development and ongoing maintenance



"Our goal is to show you the best way to use our tools."

"Our goal is to build flows, not chips."

"Our goal is to meet the original schedule."

"Our goal is to build chips, not flows."

30

Management

CAD

Design

FDA

The Goals of Flow Development

#1 Design independent flow

#2 Tool independent flow

#3 Technology independent flow

#4 User independent flow

Principle: iteration Goal: user independence

- implies isolated sandbox (aka workspace)
- all inputs, libs, etc.
- no external sources
- no common files (except: EDA tools, OS)
- no rug to get pulled out
- isolation for each iteration (multiple sandboxes)

Iteration

can you exactly repeat what you did before?

- if you can't repeat it, then you are not iterating, you are flailing
- if your result is a failure, you want to make sure you don't do it that way again
- if your result is a success you want to repeat it
- what files went into your successful run? this requires...

Configuration management

- Configuration management means which...
 - RTL?
 - scripts?
 - libraries?
 - EDA tool versions?
- ...means any user should be able to repeat
 - same user rerun
 - different user rerun
 - rerun next year
 - EDA vendor rerun (e.g., test case)
- ...implies revision control

Revision control

all sorts of these have been used for IC design

- copy directory structure
- SCCS
- RCS
- CVS
- Subversion
- Synchronicity, Clearcase, AccuRev, ICManage, ClioSoft, etc.

Revision control

- What files need to be under revision control?
- Some that you might not immediately consider:
 - libraries
 - derived files: netlist, layout
 - anything that the next step in the flow might need
 - anything that you might modify for an ECO

Revision control needs to handle:

- large binary files
- directories as primary objects
- symbolic links as primary objects
- CVS doesn't cut it...

Revision Tags

- Configuration management uses revision tags or equivalent
- Must uniquely identify each iteration
- Bad names:
 - Latest
 - Bronze, silver, gold
- Good names:
 - Anything that will never run out
 - Integers
 - Dates
 - but be sure and include the year
 - and allow for multiple names per day

Iteration + user independence

Consider the user's *shell environment*.

- It should *not* affect your flow, because it isn't under revision control!
- Tool version selection should *not* be determined by your shell search path

Automation + tool independence

- The flow should work with different tools and any number of copies of the tools you have
- There is one precious resource that you *must* manage in an automated fashion...

Licenses!

License management

- Job should never fail due to lack of license
- Implies job queueing system (LSF, SGE)
 - even if you have only one license!
 - even if you have only one execution host!
 - even if you have only one user!
- Your job control system *must* be license-aware
- Each command within a tool may require a different set of licenses
- Far easier to be efficient with licenses if the tool invocations are *short* and *focused*



target *n.* An invocation of a tool

modular target n.

An invocation of a tool to perform a single focused task

Why modular targets are a good idea

- simple to specify/understand/write/reuse
- more efficient license use (check in/out)
- more efficient host use
 - different steps have different resource requirements (e.g., memory, threading)
- enables parallelism: execute targets in parallel
- faster results (critical target first)
- ease of modification
- self-checking targets can detect a problem earlier in the flow
- noncritical target can fail and rest can complete

Example modular targets

- dc_read_sdc
 - 1. read ddc database
 - 2. read SDC
 - 3. write ddc database
- dc_compile
 - 1. read ddc database
 - 2. compile
 - 3. write ddc database
- dc_write_verilog
 - 1. read ddc database
 - 2. write out netlist

- pt_analysis_using_sdc
 - 1. read netlist
 - 2. read parasitics (optional)
 - 3. read SDC
 - 4. update timing
 - 5. save session
- talus_fix_time
 - 1. read constrained volcano
 - 2. fix time
 - 3. write optimized volcano
- talus_fix_cell
 - 1. read floorplan volcano
 - 2. fix cell
 - 3. write placed volcano

Sequence management

GNU make

- Understands and manages dependencies
- Supports parallel execution
- Free, and available on all the tool platforms

flowfile: a make-based target sequence

dc load design i1: @\$(start-target) @\$(MAKE) -- no-print-directorv \ MAKEFILES=../../flow/design compiler/design compiler master.mk \ TARGET DIR=\$ (TARGET DIR) \ TARGET INSTANCE=\$ (TARGET INSTANCE) \ DC TOPO="1" \ dc load design @\$(end-target) dc load constraints i1: dc load design i1 @\$(start-target) @\$(MAKE) -- no-print-directorv \ MAKEFILES=../../flow/design compiler/design compiler master.mk \ TARGET DIR=\$ (TARGET DIR) \ TARGET INSTANCE=\$ (TARGET INSTANCE) \ input ddc="\$(DERIVED DIR)/dc load design i1/\$(BLOCK NAME).ddc" \ constraints file="flowfiles/cpu constraints.tcl" \ DC TOPO="0" \ dc load constraints @\$(end-target) dc compile i1: dc load constraints i1 @\$(start-target) @\$(MAKE) -- no-print-directorv \ MAKEFILES=../../flow/design_compiler/design_compiler_master.mk \ TARGET DIR=\$ (TARGET DIR) \ TARGET INSTANCE=\$ (TARGET INSTANCE) \ input ddc="\$(DERIVED DIR)/dc load constraints i1/\$(BLOCK NAME).ddc" \ DC TOPO="0" \ dc compile @\$(end-target) dc_syn_reports_i2: dc_compile_i1 @\$(start-target) @\$(MAKE) -- no-print-directorv \ MAKEFILES=../../flow/design compiler/design compiler master.mk \ TARGET DIR=\$ (TARGET DIR) \ TARGET INSTANCE=\$ (TARGET INSTANCE) \ input ddc="" \ DC TOPO="0" \ dc svn reports @\$(end-target)

dc compile incremental i1: dc svn reports i2 @\$(start-target) @\$(MAKE) -- no-print-directorv \ MAKEFILES=../../flow/design compiler/design compiler master.mk \ TARGET DIR=\$ (TARGET DIR) \ TARGET INSTANCE=\$ (TARGET INSTANCE) \ input ddc="\$(DERIVED DIR)/dc compile i1/\$(BLOCK NAME).ddc" \ DC TOPO="0" \ dc compile incremental @\$(end-target) dc_verilog_out_i1: dc_compile_incremental_i1 @\$(start-target) @\$(MAKE) -- no-print-directorv \ MAKEFILES=../../flow/design compiler/design compiler master.mk \ TARGET DIR=\$ (TARGET DIR) \ TARGET INSTANCE=\$ (TARGET INSTANCE) \ input ddc="\$(DERIVED DIR)/dc compile incremental i1/\$(BLOCK NAME).ddc" \ TOOL VERSION="B-2008.06-SP2" \ DC TOPO="0" \ dc verilog out @\$(end-target) post compile reports: dc compile incremental i1 @\$(start-target) @\$(MAKE) -- no-print-directorv \ MAKEFILES=../../flow/design compiler/design compiler master.mk \ TARGET DIR=\$ (TARGET DIR) \ TARGET INSTANCE=\$ (TARGET INSTANCE) \ input ddc="\$(DERIVED DIR)/dc compile incremental i1/\$(BLOCK NAME).ddc" \ DC TOPO="0" \ dc svn reports @\$(end-target)

```
complete: parallel_2
```

How would you create and maintain these sequences?

 A *flowfile* is a makefile that describes your flow (a sequence of tool invocations, i.e., targets)

flowMaker

- Tcl/Tk script written to build and maintain *flowfiles*
- Only thing we couldn't get off the shelf
- Runs on all tool platforms (e.g., RHEL)
- Download a copy at <u>www.flowMaker.info</u>

flowMaker window

💙 flow Ma	ıker							_ C >
Quit	1 1	Load Now	Save					lleip
Row	Option Selec	tor						
		Active Tar	gets				Available Targets	
↑ ·	Move Up	dc_load dc_comp	(Jesiyn_i Constrai ile_1 log_out_i	nts_il		← Activate →	<pre>Misster_target_fles:</pre>	*
↓ -	Move Down	4			5	Disable	<pre>wile sol . //flov/primetime/sdc_naster.mk . //flov/syncps⊽s/starro_master.mk</pre>	V
	larget Master:	:				Master Source:		

Select a master target

💙 flow Ma	ıker						
Quit	Clear	Load Now	Save				ileip
Row	Option Sele	ctor					
		Active Tag	gets			Available Targets	
↑	Move Up Move Down	dc_load dc_comp	Lderiyn_i: _constrain ile_i log_out_i	nts_il	Activate	<pre>Master_target_files:</pre>	
	larget Mastei	r:			Master Source:		
		1					

Activate the master target

Catit Clear Load flow Save How Option Selector Available Torgets Image: Interpret								
How Option Salector Active Targets Image: I had bering if	💙 flow Ma	ıker						
Active Targets dc_load_design_ii dc_load_corstrainte_ii dc_load_corstrainte_ii dc_somplie_if dc_weillog_out_iii dc_somplie_if dc_weillog_out_iii dc_somplie_if dc_weillog_out_iii dc_weillog_out_iii dc_weillog_out_iii dc_weillog_out_iii dc_weillog_out_iiii dc_weillog_out_iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii	Quit	Clear	Load Now	Save				ileip
Move Up Move Up Move Up Activate Move Up Activate Move Up Move Up Move Up Activate Activate Comparison Move Down Activate Move Down	Row	Option Selec	ctor					
Image: Move Down Image: Disable Image: First Structure Structus	↑.		Active Tary dc_load dc_load dc_comp	_design_i _cor.strai]e i	nte_il	Activate ► Add	<pre>Mixibility Larget_files: // flow/design_compiler/design_compiler_master_mk co-compile co-compile incremental co-compile_ultre_incremental co-compile_ultre_incremental co-compile_ultre_incremental co-compile_ultre_incremental co-compile_ultre_incremental co-compile_ultre_incremental co-compile_ultre_incremental co-compile_ultre_incremental co-compile_ultre_incremental co-compile_ultre_incremental co-compile_ultre_incremental co-source_tol_file</pre>	
Iarget Master: Master Source:						 Disable	<pre>frinctimc_reports frinctimc_reports write odd write odf frinctime/sdi_naster.nk frinctime/sdi_naster.nk frinctime/svs/starro_master.nk frinctime/svs/starro_master.nk</pre>	Y

Added new target instance

💙 flow Ma	ıker							,
Quit	Clear	Load Now	Save					lleip
Row	Option Sele	ector						
		Active Tar	gets			۸	vailable Targets	
1	Move Up	dc_load dc_comy dc_veri dc_syn_	l_design_i1 l_comstrain ille_i log_out_i1 _cepcrts_i1	ds_il	Actival		<pre>isster_target_files: / /flow/tesign_compiler/tesign_compiler_master_mk do_compile do_compile_ultro_incremental do_compile_ultro_incremental do_load_constraints do_load_constraints do_load_ests do_load_ests do_report_math_slack do_set_wire_load do_set_wire_load do_set_comprete do_unipuity do_worilec_out do_worilec_out do_worilec_out do_worilec_out talus_fix_rel talus_fix_rel talus_fix_rel talus_fix_rel talus_fix_rel talus_fix_rel talus_fix_rel talus_fix_rel talus_fix_rel_met_metetmk primetime_analyse_using_sdo primetime_mastyse_using_sdo primetime_report_path_slack primetime_report_path_slack primetime_report_slack primetime_report_master.mk</pre>	
	larget Maste) () () () () () () () () () (Master So		. //flov/syncpsvs/starre master.nk	
	i ai get maste	•			waster 30			

New target instance

💙 flow M	laker						_ C >
Quit	Clear L	oad Now Save					ileip
Row	Option Select	זי					
		Active Targets		A		Available Targets Master_target_lites: ///finw/design_pompiler/design_pompiler_master_mk	A
		dc_load_corstrai dc_compile_i dc_verilog_out_i dc_syn_cepcrts_i bcsic pos :	1	eports]	de_compile de_compile_inerenental de_compile_ultre_ineremental de_compile_ultre_ineremental de_load_design de_load_sde de_uilkyway_uut	
ſ	Move Up				Activate	dc_captrt_math_slack dc_set_wire_load dc_set_wire_load dc_syn_reports dc_ungroup dc_ungroup dc_ungroup dc_wrile_ad	
						 .//flow/magma/talua_master.mk talua_ctt_scan_insect talua_fix_netList talua_fix_netList talua_fix_time talua_impost constraints .//flow/prime_time/master.mk 	
1	Maus Dour					primetime_analysis_usinc_sdc primetime_analyse_ddc primetime_report_path_slack primetime_reports write_ccc write_sdf	
↓ ·	Move Down			¥.	Disable	<pre>+ ././flow/primelime/sdo_mester.mk + ././flow/synopsys/starro mester.mk</pre>	
	larget Master:	de	_syn_reparts	М	laster Source:	//llow/design_compiler/design_compiler_master.mk	

Target menu

🐓 flow M								
 	laker	<u>aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa</u>			aaaa	aanaaaaaaaa		
Quit	Clear	Load Now	Save					llelp
Row	Option Selec							
1	Move Up	Active Ta	d Jesign_ii d_corstrain pile_i ilog_out_i centrts_i Rename i: Orea Collap Collap Ure Colla	nts_il		Activate	<pre>Avoitable Targets Master_target_files:</pre>	
, _ [Move Down Target Master:					→ Disable Master Source:	<pre> prinetime_report_path_sLack prinetime_reports write_sdf</pre>	

Rename the target instance

💙 flow Ma	aker		$- \subset X$
Quit	Clear L	oad Now Save	ileip
Row	Option Select	or	
		Active Targets Available Targets	
Ŷ	Моте Цр	dc_load_design_i1 // flow/design_compiler_master_nk dc_load_constraints_i1 // flow/design_compiler/design_compiler_master_nk dc_compile_i // flow/design_compiler/design_compiler_master_nk dc_werilog_out_ii // flow/design_compiler/design_compiler_master_nk dc_syn_ceperts_iii // flow/design_compiler/design_compiler_master_nk dc_syn_ceperts_iii // flow/design_compiler/design_compiler_master_nk dc_sopile_ultrop_incremental // flow/design_compiler_master_nk dc_sopile_ultrop_incremental // flow/design_compiler_master_nk dc_load_cestints // flow/design_compiler_master_nk dc_sopile_ultrop_incremental // flow/design_compiler_master_nk dc_load_cestints // flow/design_compiler_master_nk dc_sopile_ultrop_incremental // flow/design_compiler_master_nk dc_load_cestints // flow/design_compiler/design_compiler_master_nk dc_load_cestints // flow/design_compiler_master_nk dc_load_losite // flow/design_compiler/design_compiler_master_nk dc_load_losite // flow/design_compiler_master_nk dc_load_losite // flow/design_compiler_master_nk dc_load_losite // flow/design_compiler_master_nk dc_load_losite // flow/design_compiler_m	
.	Move Down	July of the set o	
·	larget Master:	Master Source:	

Target instance renamed

💙 flow Ma	aker							- 31	R
Quit	Liear L	.aad How	Save					нөгр	
Flow	Option Select	lor							
		Active Targ	ots				Available Targets		
Ŷ	Move Up	cc_load co_compi cc_veril	design_i constrain le_l .og_put_1 wile_rep	nt∈_il l			<pre>Naster_target_file:: //flew/design_compiler/decign_compiler_moster.nl dc_compile_incremental dc_compile_incremental dc_compile_incremental dc_compile_ultre_incremental dc_load_constraints dc_load_decign. dc_load_ode dc_milkyway_out dc_respert_path_slack dc_set_wire_idec dc_source_til_tile dc_source_til_tile dc_source_til_tile dc_wijnify dc_verilog_out dc_wijnify dc_verilog_out dc_wijnify dc_verilog_out dc_wijnify dc_verilog_out dc_wijnify dc_verilog_out dc_wijnify talus_tile_son_insert talus_tile_tilet talus_filet_time talus_</pre>	k	
$\downarrow \downarrow$					4 -		//flcw/synopsys/starrc_maater.nk		ł
-	Target Master:				Mas	iter 3ource:			Ĩ

Option Selector

😪 flow Maker	_ C X
Quit Clear Load Row Save	i leip
Row Option Selector	
Select the desired target switches below:	
📕 File Inpuls 📕 Target Variables 🔄 Job Submission Settings 📕 File Outpuls	
dc_load_design_i1	
DDC_FILE_Design database in DDC format. <ddc_file></ddc_file>	
dc_load_constraints_fi	
input dde odd das gn 11:000 FILE dde_tile	
constraints_file flowFilec/opu_constraints.tol de_tel_file	
DDC_HLE_Constrained design database in DDC_ <ddc_tlb> format</ddc_tlb>	
a	
mput_ddc >dc_lcad_constraints_i1:DDC_FILE ddc_file	
DDC_FILE Compiled design database in DDC format <ddc_file></ddc_file>	
dc_verilog_out_ii	
input_ddc site_c impile_ifiCDC_FILE ddc_file	
NETLIST_FILE Colput design netlist in Verilog format. <verilog></verilog>	
DDC_HLE_Output design database in DDC format_ <ddc_tile></ddc_tile>	
post_compile_reports	
inpul_dd:	

Derived file forwarding

Y flow Maker		
Quit Clear Load Flow Save		Help
Flow Option Selector		
	Select the desired target switches below:	
📕 File Inputs 📕 Target Variables 🔄 Job Submission Se	ettings 📕 File Outputs	
dc_load_design_i1		
DDC_FILE Design database in DDC	format adda files	1
dc_load_constraints_i1		
i nput_ddc >dc_load_design_i1:DDC_	_FILE ddc_file	
constraints file flowfiles/cpu constraints.to	dc_tcl_file	
DDC_FILE Constrained design data format	base in DDC <ddc_file></ddc_file>	
dc_compile_i1		
input_ddc >dc_load_constraints_i1:D		
DDC_FILE Compiled design databas	se in DDC format <ddc_file></ddc_file>	
de verilog out il		
input_ddc >dc_compile_i1:DDC_FILE	ddc file —	
NETLIST_FILE Output design netlist in '		
DDC_FILE Output design neutration	-	
post_compile_reports		
input_ddc	◆ ddc_file	
,		
	◇ suc_load_design_i1.000C_FILE ◇ sdc_load_constraints_i1:DDC_FILE	
	♦ Store The Strain	
	<pre>>dc_verilog_out_i1:DDC_FILE</pre>	
M		

Derived file selected

Y flowMaker	- D X
Quit Clear Load Flow Save	Heip
How Option Selector	
Select the desired target switches below:	
📕 File Inputs 📕 Target Variables 🔄 Job Submission Settings 📕 File Outputs	
dc load design i1	
DDC_FILE Design database in DDC format <ddc_file></ddc_file>	
dc_load_constraints_i1	
input_ddc >dc_load_design_i1:DDC_FILE ddc_file	
DDC_FILE Constrained design database in DDC <ddc_file> format</ddc_file>	
dc_compile_i1	
input_ddc >dc_load_constraints_I1:DDC_FILE ddc file	
DDC_FILE Compiled design database in DDC format <ddc_file></ddc_file>	
dc_verilog_out_i1 input_ddc >dc_compile_i1:DDC_FILE ddc_file	
NETLIST_FILE Output design netlist in Verilog format «verilog»	
DDC_FILE Output design database in DDC format <ddc_file></ddc_file>	
post_compile_reports	
input_ddc >dc_compile_11:DDC_FILE ddc_file	

Job Submission Settings

Y flow Maker	
Quit Clear Load Flow Save	Help
Flow Option Selector	
Select the desired target switches below:	
📕 File Inputs 📕 Target Variables 📕 Job Submission Settings 📕 File Outputs	
DC_TOPO	
DDC_FILE Constrained design database in DDC <ddc_file> format</ddc_file>	
dc_compile_i1	
input_ddc >dc_load_constraints_i1:DDC_FILE ddc_file	
TOOL_VERSION	
TOOL_RESOURCES	
DC_TOPO	
DDC_FILE Compiled design database in DDC format <ddc_file></ddc_file>	
dc_verilog_out_i1	
input_ddc >dc_compile_11:DDC_FILE ddc_file	
TOOL_VERSION	
TOOL_RESOURCES	
DC_TOPO 🔄	
NETLIST_FILE Output design netlist in Verilog format. <verilog></verilog>	
DDC_FILE Output design database in DDC format <ddc_file></ddc_file>	
post_compile_reports	
input_ddc >dc_compile_i1:DDC_FILE ddc_file	
TOOL_VERSION	
TOOL_RESOURCES	
DC_ТОРО 🔄	

Change tool version for a target

▼ flow Maker	
Quit Clear Load Flow Save	Heip
Flow Option Selector	
Select the desired target switches below:	
👅 File Inputs 🔎 Target Variables 🔎 Job Submission Settings 🔎 File Outputs	
DC_TOPO	Δ
DDC_FILE Constrained design database in DDC <ddc_file> format</ddc_file>	
dc compile i1	
input_ddc >dc_load_constraints_i1:DDC_FILE ddc_file	
TOOL_VERSION	
TOOL_RESOURCES	
DC_TOPO	
DDC_FILE Compiled design database in DDC format <ddc_file></ddc_file>	
de verilog out i1	
input_ddc >dc_compile_i1:DDC_FILE ddc_file	
TOOL_VERSION B-2008.08-SP2	
TOOL_RESOURCES	
DC_TOPO	
NETLIST_FILE Output design netlist in Verilog format <verilog></verilog>	
DDC_FILE Output design database in DDC format <ddc_file></ddc_file>	
post_compile_reports	
input_ddc >dc_compile_i1:DDC_FILE ddc_file	
TOOL_VERSION	
TOOL_RESOURCES	
DC_TOPO	

Target menu

Y flow Ma	lker						×
Quit	Clear	Load Flow	Save				Help
	1 1	Active Targ dc_load dc_load dc_comp dc_veril	ets design_i1 constrain le_i1 og_out_i1 Renan (C C C C C C	.ts_i1	- Activate	Available Targets Master_target_files: /./flow/design_compiler/design_compiler_master.ml dc_compile_incremental dc_compile_ultra dc_compile_ultra dc_compile_ultra dc_load_constraints dc_load_design dc_load_sdc dc_milkyway_out dc_report_path_slack dc_set_wire_load dc_source_tol_file dc_wire_tol_file dc_wire_tol_file dc_wire_sdc /./flow/magma/talus_master.mk talus_fit_scan_insert talus_fit_ntl talus_fit_thme talus_inport_constraints /./flow/primetime/master.mk primetime_analysis_using_sdc primetime_reports write_sdc /./flow/primetime/sdc_master.mk /./flow/primetime/sdc_master.mk	Help
	Move Down Farget Maste				Disable Master Source:		7

Parallel targets

Y flow Maker								- 0 3
Quit	Clear Lo	ad Flow	Save					Help
▲	ption Selecto	Active Tarç dc load	_design_i _constrai ile_i1 rilog_out	ints_il t_il		← Activate	Available Targets Master_target_files: /flow/design_compiler/design_compiler_master_nk dc_compile_incremental dc_compile_ultra dc_compile_ultra dc_load_constraints dc_load_design dc_load_sdc dc_ast_wire_load dc_set_wire_load dc_source_tol_file dc_syn_reports dc_ungroup dc_uniquify dc_verilog_out dc_write_sdc //flow/magma/talus_master.nk talus_fix_relist talus_fix_time talus_fix_time talus_fix_time talus_fix_time talus_fix_time talus_fix_topath_slack primetime_constraints //flow/primetime/master_nk primetime_reports write_sdc */flow/primetime/master.nk #/flow/primetime/master.nk #/flow/primetime/master.nk #/flow/primetime/master.nk */flow/primetime/master.nk */flow/primetime/master.nk */flow/primetime/master.nk */flow/primetime/master.nk	
	ve Down				2	Disable		$\overline{\nabla}$
Targ	jet Master:					Master Source:	:	

Add another target

Y flow Maker		- C X
Quit Clear	r Load Flow Save	Help
Quit Clear	Selector Active Targets Acti	<pre>compiler/design_compiler_master.mk ntal noremental kk alus_master.mk ert maints</pre>
Move Do	bown	using_sdc Mdc wth_slack
Target Ma	laster: Master Source:	

Rearrange target order

✓ flow Maker	-	0 ×
Quit Clear Load Flow Save	н	leip
		· · · · · ·
	Disable	$\overline{\nabla}$
Target Master:	Master Source:	

Update target options

✓ flow Maker		*
Quit Clear Load Flow Save	Help	Ĩ
Flow Option Selector		
👅 File Inputs 👅 Target Variables 👅 Job S	Select the desired target switches below: Submission Settings 🛛 📕 File Outputs	
	aunnasion actungs – The outputs	
		5
	design database in DDC format <ddc_file></ddc_file>	
dc_compile_incremental_i1		
input_ddc >dc_compil	ile_i1:DDC_FILE	
TOOL_VERSION		
TOOL_RESOURCES		
DC_ТОРО 🔲		
DDC_FILE Compiled a	design database in DDC format <ddc_file></ddc_file>	
dc_verilog_out_i1		
input_ddc >dc_compil	lle_incremental_i1:DDC_FILE ddc_file	1
TOOL_VERSION B-2008.06-	-SP2	
TOOL_RESOURCES		
DC_ТОРО 🔲		
NETLIST_FILE Output de:	esign netlist in Verilog format. <verilog></verilog>	
DDC_FILE Output de:	esign database in DDC format <ddc_file></ddc_file>	
post compile reports		
	ile_incremental_i1:DDC_FILE ddc_file	
TOOL VERSION		
TOOL_RESOURCES		
M		Ĩ

Add duplicate target

Y flow Maker					×
Quit C	ear Load Flow Save				Help
Guit Ci	ear Load Flow Save on Selector Active Targets dc load design i1 dc load constraint dc compile i1 dc compile increme dc compile increme (post_compile_re) ve Up	ntal_i1	- Activate	Available Targets Master_target_files: . /. /flow/design_compiler/design_compiler_master_nk dc_compile_ultra dc_compile_ultra_incremental dc_load_constraints dc_load_design dc_load_design dc_load_sdc dc_report_path_slack dc_set_wire_load dc_source_tol_file dc_syn_reports dc_ungroup dc_uniquify dc_verilog_out dc_write_sdc 	Help
	e Down	2		//flow/synopsys/starrc_master.mk	V
Target	Master:		Master Source:		

Save flowfile

💙 flow Ma	aker	- 1	30
Quit	Cear La	I Flow Save Help	
	,		Ē
Flow	Option Selecto		
		tive Targets Available Targets	
ſ	Move Up	<pre>de load design il de load constraints_il de compile incremental de compile incremental de compile incremental de compile_incremental de compile_incremental de compile_incremental de compile_incremental de compile_incremental de compile_incremental de load constraints de load design de load design design de load design de load design de load design de load des</pre>	
	Move Down		
-	Target Master:	Master Source:	

How to run a flow

- % make FLOW=dc complete
- % make FLOW=dc complete -j -k >& make.log &

Anatomy of a flowMaker master target

```
dc_syn_reports:
## Basic post-synthesis Design Compiler reports
#
# in input_ddc
#
    input_ddc(type) = ddc_file
#
    input_ddc(comment) = Input design database in DDC format
#
  job DC TOPO
#
#
    DC_TOPO(type) = boolean
#
   DC TOPO(value) = 0
#
   DC TOPO(comment) = Enable DC topographical mode
#
  $(call dispatch, dc_shell \
       $(if $(findstring $(DC_TOPO), 1), -topographical) \
       -f $(FLOW_ROOT)/design_compiler/dc_syn_reports.tcl \
        , $ (TARGET DIR) / $ (TARGET INSTANCE).log)
```

Anatomy of a flowMaker master target

```
talus_fix_time:
## Reads in volcano, runs fix time
#
# in input_volcano
     <u>input_volcano(type) = volcano</u>
#
     input volcano(comment) = Input design database in volcano format
#
# var target_slack
    target_slack(type) = string
#
    target_slack(value) = 0p
#
     target_slack(comment) = Target slack with time unit
#
# var timing_effort
    timing_effort(type) = menu
#
    timing_effort(value) = medium
#
    timing_effort(allowedvalues) = low medium high
#
     timing_effort(comment) = timing effort option to fix time
#
# out VOLCANO
    VOLCANO(type) = volcano
#
    VOLCANO(filename) = $ (BLOCK) % fix-time.volcano
#
#
    VOLCANO (comment) = Output design database in volcano format
#
   $(call dispatch, \
          talus -f $(FLOW_ROOT)/magma/$@.tcl -logdir $(TARGET_DIR) \
          , $(TARGET DIR) /$ (TARGET INSTANCE).log )
```

Master target options

Quit Clear Load Flow Save Hell Flow Option Selector Select the desired target switches below: Imput_voicanu Select the desired target switches below: Imput_voicanu Imput_voicanu<	а Т
Select the desired target switches below: File Inputs Target Variables Job Submission Settings File Outputs Input_voicano >talus_fix_fit_fit:VOLCANO volcano	
Select the desired target switches below: File Inputs Target Variables Job Submission Settings File Outputs Input_voicano >talus_ttx_tti_tt:VOLCANO Volcano	X
■ File Inputs ■ Target Variables □ Job Submission Settings ■ File Outputs Imput_voicano >talus_TIX_ITI_IT: YOLCANO volcano	
	Δ
	Δ
VOLCANO Fix netlist volcano <volcano></volcano>	
talus_dft_scan_insert_i1	
input_volcano >talus_fix_netlist_i1:VOLCANO volcano	
VOLCANO Output design database in volcano format «volcano»	
talus_import_constraints_i1	
input_volcano >talus_dft_scan_insert_i1:VOLCANO volcano	
SDC >write_sdc_i1:SDC sdc file	
VOLCANO Constrained design database in volcano <volcano> format</volcano>	
talus_fix_time_i1	
input_volcano >talus_import_constraints_i1:VOLCANO volcano	
target_slack 200p (string)	
timing_effort medium 😫 (menu)	
VOLCANO Output design database in volcano format «volcano»	

Different flowMaker users

- Just runs an existing flowfile. Invokes "make" from the command line.
- Uses flowMaker to edit the target options, then runs the modified flowfile.
- Uses flowMaker to edit the target options, edit the target sequence (add, delete, rearrange targets), then runs the modified flowfile.
- Writes new master targets to support new EDA tools and new features.

Things to consider

- What sort of directory structure do you have?
- What sort of revision control?
- What sort of configuration management and tags?
- How do you manage job control?
- How do you manage licenses?
- How do you manage tool versions?
- How do you manage your project?

Notes on project management

What you do not need: MS Project

- IC design is an iterative process, particularly if your flow is not stable
 - Flows will never be stable when used on an immature design
 - PERT charts are linear tools
 - OK to have major milestones but minor milestones are too detailed
- Don't be afraid of another iteration
 - with automation, rerun is easy

A suggested management method

- Make a list of everything you haven't done yet for your design (past successes don't count)
- Make a second list of everything not automated in your flow yet
- Schedule each item from both lists to occur concurrent with maturation of the RTL
- How long will the implementation take?
 - You mean to run through the tools the final time, or to get all the items in the list achieved?
 - You won't know the run time until both lists are empty
 - Hopefully both lists will be empty before the RTL stops changing
- Increase frequency of RTL releases as project matures
 - You need time between releases early on to work on the two lists
 - Towards the end, you need to iterate to make sure you are capable of running it all through

Managing iterations

- Iterate early, iterate often
- The last change in the RTL will be small
- A big change in the RTL won't be the last change
- How do you tell if you are done?
- How do you measure progress?
- Consider your process from RTL→GDSII:
 - Do you know the theoretical minimum time?
 - Do you know exactly how many steps are required?

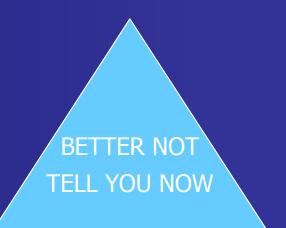
Simple progress report

min_delay

	_		31Mar1019	30Mar0609	28Mar2229	27Mar1956	27Mar0000	25Mar1953
	post_route		2328	2471	1703	2146	1318	1376
		TYP					5412	5788
		FAST	4933	4923	81732	71500	62504	60014
	DFT.scan_setup		212	209	209	1573	1602	1646
		TYP					1652	1665
		FAST	538	535	535	1592	1666	1671
	DFT.scan_shift		14	14	27	15	15	15
		TYP					23	25
		FAST	54	44	49	21	42	43
DFT.	stuck-at_scan_capture		203	152	171	433	125	178
		TYP					166	241
		FAST	751	517		933	292	349
	DFT.AC_scan_capture	SLOW	405	354		1730	1423	1476
		TYP					1458	1531
		FAST	1214	992		2219	1552	1609
	DFT.boundary_scan	SLOW	44	51		1607	1651	1696
		TYP					1856	1881
		FAST	370	431		1843	1921	1926
	DFT.dft_p1500	SLOW	200	3868		7711	7711	7711
		TYP					7711	7711
		FAST	526	3868		7711	7711	7711
	DFT.sms_p1500	SLOW	200	659		6785	7067	7184
		TYP					7253	7271
		FAST	526	904		7224	7280	7281
	t	total	12518	19992	84426	115043	129411	127999

The Magic 8-Ball as a management tool

"Will there be problems with my new IP?" "Is two weeks enough time to complete a floorplan?" "Will we really see new timing problems if we turn on SI?" "Will this chip go out on time?"





Let's make up a checklist for physical implementation!

 Something like the Reuse Methodology Manual and Verification Methodology Manual

All the things you need:

- Software
- Hardware
- Other stuff

What you need: Other stuff

- Books: Tcl, perl, make, shell, UNIX, revision control
- Training
- Tool documentation
- Design documentation

What you need: Hardware and Infrastructure

- CPU -- compute farm (how many?)
- memory (how much?)
- disk (1Mbyte per placeable instance?)
- backups, printers, power, maintenance...
- desktop: usually PC so you can have office apps (email, Word)
- VNC or other connection to CPU how easily can you transfer files to/from desktop to CPU? how easily can you cut-and-paste?
- remote access: VPN
- bug tracking
- team communication (email, conferences, video links)

What you need: Software

- configuration management: revision control
- license management: job control
- flow management: flowMaker
- dependence management: make
- tool version management: job control
- compute farm management: job control
- error management: logscan + self-checking
- physical design tools
- libraries (standard cell, IO, memory, analog, 3rd party IP)



"I still don't see why we can't use the flow from the last chip."

"How about a flow from the EDA vendor? Can't we start with what they offer?"

"Didn't the guys in Bangalore just tape out? We can take their flow, right?" Management
 CAD
 EDA
 Design
 Management



- Management
 CAD
- 3. EDA

2. CAD

3. EDA

4. Design

4. Design

Thanks

- These companies loaned software keys to build up the example flow:
 - Magma
 - Synopsys
 - Cadence
 - CLK Design Automation
- Designs and libraries provided by:
 - Sun Microsystems
 - Artisan/ARM
 - Nangate
- Special thanks to Narendra Shenoy, Leon Stok and David Reda

Flow engineering: Things to remember

Principles

#1 Flows are hard#2 Iterate#3 Automate#4 Failure is your friend

Goals

#1 Design independent flow#2 Tool independent flow#3 Technology independent flow#4 User independent flow