



Mixed Mode Simulation – First Steps

P. Fischer

Lehrstuhl für Schaltungstechnik und Simulation ZITI, Uni Heidelberg

(Original slides from Florian Erdinger)

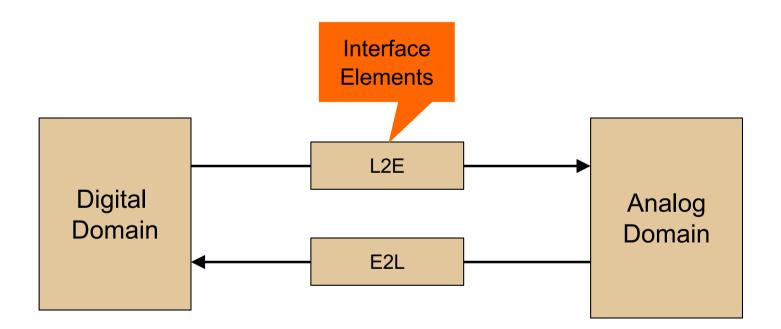
VLSI Design - Mixed Mode Simulation

© P. Fischer, ZITI, Uni Heidelberg Page 1

Why Simulate in Mixed Mode?

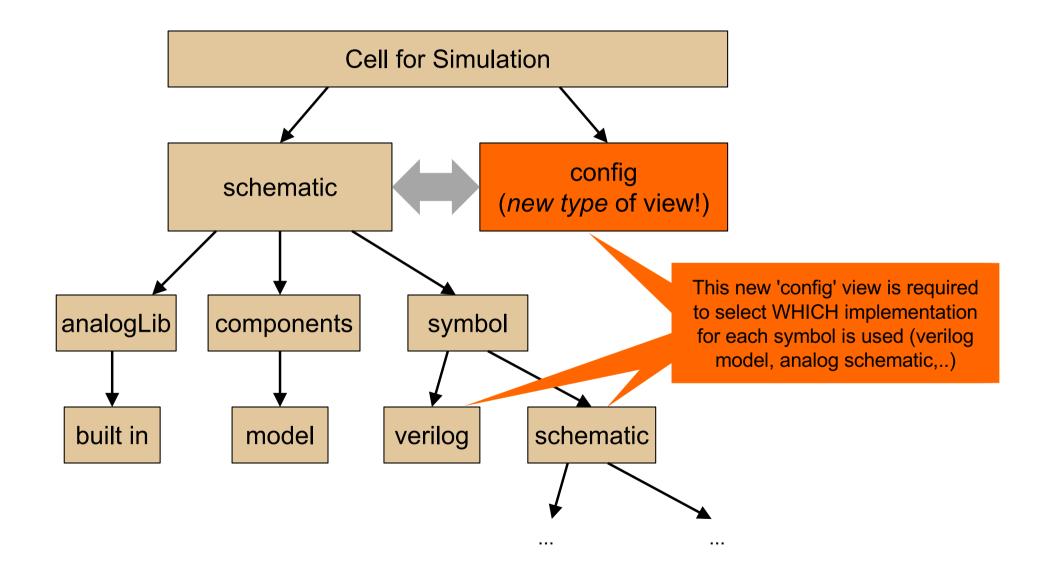
- Most analog circuits need interaction with digital circuits
 - control logic
 - processing / verification of results
- Simple digital functionality can be obtained by Spice sources (vpulse, vpwl,...), but this is tedious, inflexible,...
- More flexibility by using Verilog-A. Good for simple extensions (DAC..), but not suited for large digital parts
- \rightarrow Mixed Mode Simulation:
- Describe the digital parts by a hardware description language (HDL).
- Analog part: schematics & analog simulator
- Digital part: HDL & digital simulator
 - HDL (Verilog, VHDL, ...) much more flexible
 - Simulation much faster (but some simulator setup overhead..)

- Two simulators run in parallel
 - Digital Simulator for digital part
 - Analogue simulator for analogue part (we use Spectre)
- Interface Elements translate between both domains





What do we Need?

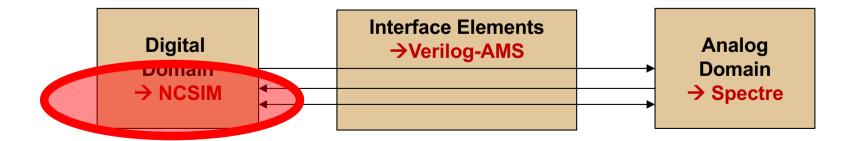




A SIMPLE EXAMPLE

A Simple Example

- The following slides show how to set up a simple mixed mode simulation in the Virtuoso ADE environment with the following steps:
 - 1. Creating a Verilog module with a matching symbol
 - 2. Creating a *top level simulation schematic* instantiating the Verilog symbol and some analog circuit connected to it
 - 3. Creating a 'config' view of the top level simulation schematic, which describes the hierarchy
 - 4. Specifying 'Interface elements' which connect the digital and analog domains.



Before You Start

- Check that your start script 'start.sh' includes version 181 of spectre with ./opt/eda/environment/spectre181_path.bash
 - The 'later' versions are not working properly with UMC180..
- Search in .cdsinit the a line editor="..." and insert the text editor of your choice.

In cds.lib, add the line DEFINE connectLib /opt/eda/XCELIUM2009/tools.lnx86/affirma_ams/etc/connect_ lib/connectLib

1. Creating a New Verilog Module

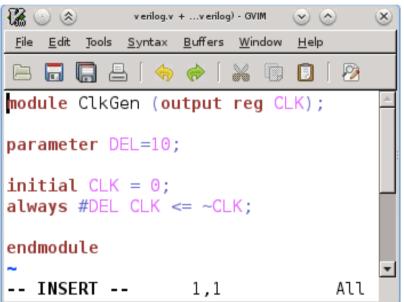
☐ Show Categories ☐ Show Fil Library	es Cell	X ⊙ ⊗ File	New File 🕜 📀 📀	8
playground		Library	playground	
D0_DUMMY_WAFER_V4_ D0_SENSOR DSSC_ONTRL1 DSSC_ONTRL1 DSSC_ONTRL1_PIxe1 DSSC_F1 DSSC_F1 DSSC_F1_ADAPTER DSSC_F1_CNTRL DSSC_F1_CNTRL DSSC_F1_SENSOR DSSC_F1_SENSOR DSSC_F1_ADAPTER DSSC_F1_ADAPTER DSSC_L1 DSSC_L1 DSSC_L1_Submitted DSSC_IMM2 DSSC_IMM2 DSSC_IMM3 	A8FCELL8_D A8FCELL6_D A8FCELL0_D BinGen_4SIM BumpPad Cnt_9bit_Lfsr1 CrossSection CurrentSource D0_DSSC_Sensor DRAM1 FamilyLogo Fill_E1_3 Fill_M1 Fill_M1 Fill_M4_3 ICG	Application Open with Always use Library path	VerilogExample Verilog Verilog Read HDL this application for thi file er/IBM013.0A/cds.lib	s type
Messages			OK Cancel H	lelp

- In 'Library Manager:
 - File \rightarrow New \rightarrow Cell View
 - 'Cell': name of verilog module
 - 'View': 'verilog' (Non-Capital!)
 - 'Type': Verilog
- The Cadence text editor opens with a 'naked' Verilog module
- The editor of your choice has been specified in .cdsinit.
 - You can also use the shell: export EDITOR=gvim
 - or the CIW: editor="gvim"

- Fill the Verilog module with some code.
 - The code need not be synthesizable
- For instance

initial out = 1'b0; always #10 out <= ~out;</pre>

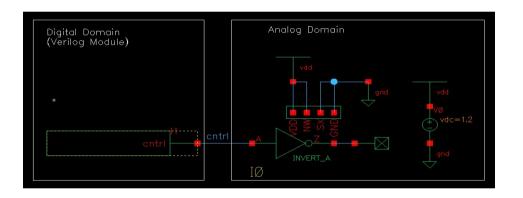
 When you close the text file, it is automatically parsed. Correct it until there are no errors left.



- When the Verilog file is closed, Virtuoso offers to create a symbol if there is none (or modify it if it does not fit to the declared interface). Create the symbol.
 - (If the Verilog contains *parameters*, the symbol inherits them.
 - In the instantiated symbol, select CDFParameter -> Verilog, not 'Use Tool Filter')

2. Creating A Top Level Simulation Module

- In 'Library Manager'
 - File \rightarrow New \rightarrow Cell View
 - Create a schematic



The Fare Tree Fordul Hanader Herb		- 🕅 🔿 🔿	New File		(X)
<pre>ile Edit View Design Manager Help Show Categories Show Files Library playground MM7_subwission_to_Florian Pads SUSIRE_BANDING SUSULTE_BANDING SUSDAC Temp_ADC_F1 US_8ths analogLib avTech basic cdsDefTechLib connestf conrEdf basic condectLib esample sbalib short_io</pre>	Cell SIM_MixedModeExample Fill_M1 Fill_MA_3 IGG LATCH_SYNC LFSR_3 LFSR_8_GAL LFSR_8_GAL LVS_mUltiplcity NMASP PielJesper SIM_Cht_9_LfsrTog SIM_DFFRE_Reset SIM_CC SIM_LFSR_3 SIM_SIM_SIM_SIM_SIM_SIM_SIM_SIM_SIM_SIM_	Library path	playground SIM_MixedM Schematic Schematics e this applica	ation for this	×
Messages Deleting view "symbol" trom cell "Verilog Deleted cellview 'playground/VerilogExamp Deleting view "VerilogExample" from cell Deleted cellview 'playground/SIM_MixedMod	le/symbol'.Deletion of 1 vi "SIM_MixedModeExample" in 1	ew done. ibrary "playground"	ne.		Hr.

- Put an instance of your Verilog module
- Add some analog circuit (symbols, primitives, sources, ...)
- 'Digital' and analog circuits can directly be connected

3. Creating the Simulation Configuration View

🕺 🔾 🛞		New File	$\odot \odot \odot$	\otimes	
File	(C)				
Library	ccs	2013			
Cell	Mix	edSim			
View	con	fig			
Туре	conf	ig			
Application	n			-	
Open with	Hiera	archy Ec	litor 🔽		
🔲 Always	use this app	lication	for this type of f	île	
$\mathbb{P} \odot \otimes$		Ne	w Configuration		$\odot \odot \odot$
⊂ Top Cell —					
Library:	CCS2013				
Cell:	MixedSim				
View:	schematic				
- Global Bir	`∲ © ⊗				
			Use Template		$\odot \odot \odot$
Library Li:	Template				
View List:					
VIGW LISE	Name:	AMS			
Stop List:		C	scher/hierEditor/ter	nplates/A	NS I
		C	scher/hierEditor/ter	nplates/A	AMS
Stop List:		C	scher/hierEditor/ter		AMS elp
Stop List: Constraint		/home/fis			
Stop List: Constraint		/home/fis			
Stop List: Constraint		/home/fis			
Stop List: Constraint		/home/fis			
Stop List: Constraint		/home/fis	Cancel App	ly H	elp
Stop List: Constraint	From File:	/home/fis	Cancel App	ly H	elp

- The AMS simulator needs a 'config' view (specifics on next slide) for the simulation schematic
- In 'Library Manager:
 - Select your simulation schematic
 - File \rightarrow New \rightarrow Cell View
 - 'Type': config (name changes to 'config')
- Note that 'Application' switches automatically to 'Hierarchy Editor'
- In the next window: change 'View' to 'schematic'
- Click 'Use Template' (bottom)
 - Select 'AMS' (this will be our simulator)
 - OK

VLSI Design - Mixed Mode Simulation



3. Changing 'config' view with the Hierarchy Editor

- The config view is edited in the 'Hierarchy Editor' and configures the netlisting procedure for simulation.
- Cells can have multiple representations, for instance a 'verilog' view and a 'schematic' view at the same time.
- The config view specifies the view to use for netlisting for each cell (or even instance)

eral
nd
,
s

4. Adding the Interface Elements

- There are built-in Interface Elements (IE) supplied with the simulator (which can also be customized if necessary)
- They are located in the 'connectLib'
- To add the connectLib to your library path
 - In the Library Manager: Edit \rightarrow Add Library Path...
 - In the table add a row with Library = connectLib
 - Path =

/opt/eda/XCELIUM2009/tools.lnx86/affirma_ams/etc/connect_lib/ connectLib

- (or edit cds.lib)
- (this must only be done once, library definition is saved in .cdslib)
- The IEs to be used are selected in the ADE when setting up the simulation (specifics see later)
- They are inserted automatically (do not have to be placed in the schematic manually)

Setting Up the Simulation and Outputs

🐒 💿 🛞 🛛 Virtuoso® Analog Design Environment (1) - playground SIM_MixedModeExample config 💿 🔿 👘	×
Launch Session Setyp Analyses Variables Qutputs Simulation Results Iools Help $c\bar{a}dence$	2
الط 🔊 ا 🕫 27 🛛 👌 🎾 🖄 🗹 🗁	
Design Variables ? @ X	c
Name Value [Enable] Arguments C	_
🕅 🕢 🛞 Choosing Simulator/Directory/Host Virtuoso® An 🕐 😒 🔗 🔛	
Simulator ans	
Project Directory //tmp/ADE-Sim-endinger	-
Host Mode local remote distributed	
Host	-
Remote Directory	6
OK Cancel Defaults Apply Help	
> Results in /tmp/ADE-Sim-erdinge Plot after simula(Auto 🎽 Plotting mocReplace 🍸	
mouse L: M: R	:
4(8) Simulator/ Status: Ready T=27 C Simulator: ams(Spectre)Mode: batch State: tmpstate	

1.0	Setup Analyses Variables Qutputs Simulation Results Iools Help	
Design Variable	Analyses	? 🗗 🗙 💽
Name 🕺 🖸) 🛞 Choosing Design Virtuoso® Analog Desi 🥐 📎 🚫 🛞	Su S
Libra	rry Name playground	1
Cell	Name SIM_GCC SIM_LATCH_SYNC SIM_LFSR_3 SIM_LFSR_3 SIM_LVDS_LOBoard SIM_NixedVodeExample SIM_TOGGLE_FF SIM_leakage SIM_ringosc	? # ×
View	Name config	
Open	Mode 💿 edit 🔾 read	
> Results in /t	Concel Help noceplace	e 🔽

- Open the top level simulation schematic
- From the menu: Launch $\rightarrow ADE$
- Setup \rightarrow Design
 - Change 'View Name' to 'config' (which we have created before)
- Setup → Simulator/Directory/...
 - Change 'Simulator' to 'ams'
- Add a transient simulation
- AMS saves nothing by default, to save everything:
 - go to 'Outputs \rightarrow Save All'
 - in the category NETS, select 'all' to save all node voltages
 - In the category CURRENTS, select 'all' if you also want to save all currents

Specifying the Interface Elements

🕺 🔾 🛞	ams	0: Select Connect Rules	
List of Connec	t Rules Used in Simulat:	ion	
Type Modified bui	Rule Name ConnRules_18V_full_	Details Fast1 Lib:conne	ctLib View;connect
Enable 1	Disable Delete Re	name) Copy	Jp Down ustomize
Rules Name	Customized rules	BV_full_fast1	View
User-defined Library Cell View	f rules for novlog,noela	b,ncsim Browse Add	Choose
			from the built-in rules here
	User- defined rules can be used		

- In the ADE window: select
 'Setup → Connect Rules ...'
- The standard connect rules use 1.8V supply and work fine for UMC018 so nothing has to be really done here...
- There are several 'built-in' interface elements (fast, medium, slow, 1.2V, 3V, ...), which can be customized
- Parameters are: vsup, trise, tfall, rlo, …;
 → logic levels, driving strength, …
- Own module can also be specified
- These modules are automatically inserted in every digital to analog connection in the entire design

(Viewing and Customizing the Interface Elements)

(This is for information only..)

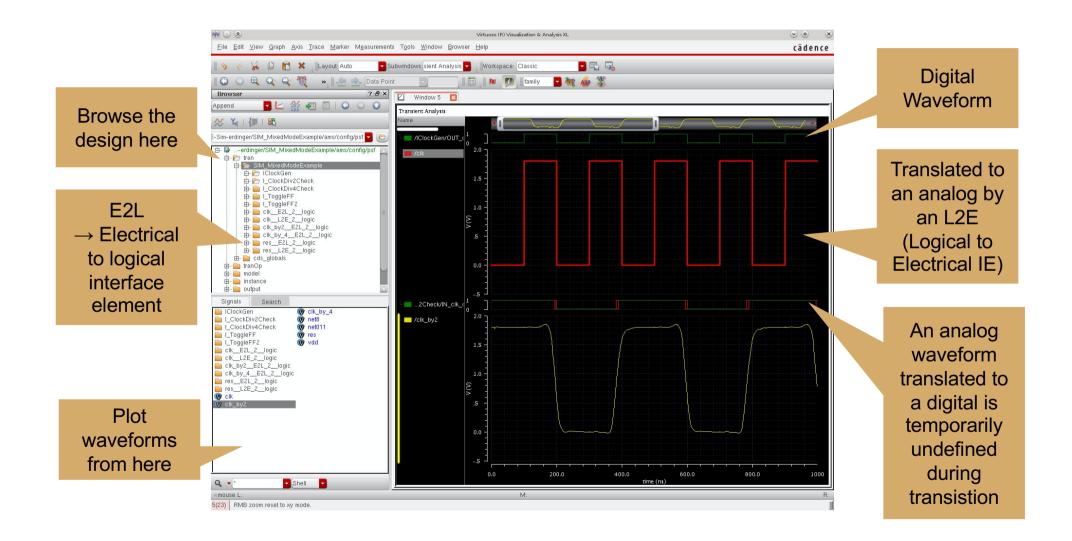
🔾 🛞 ar	is0: Select Connect Rules 🕐 📀 🛞	
List of Connect Rules Used in Simula Type Rule Name Modified bui ConnRules_18V_ful C Enable Disable Delete	Details	To customize or view the rules click here
Built-in and Customized rules Rules Name connectLib.ConnRules Description This is the descrip User-defined rules for novlog.nos Library Cell View	Image: Second	All interface elements are listed here The important ones for us are: L2E → Logical to Electrical E2L → Electrical to Logical
	Mode Parameters Parameter Parameter Value Change M Cancel Apply Disciplines Help	Select one and click here if you are interested in the code (Verilog-AMS)

Running and Viewing the Simulation

- Run the simulation ('play button')
- In the log file you can see that there are several steps:
 - Compilation
 - Elaboration
 - Simulation
- Verilog \$display task prints to the log file
- Open the results browser to look at the results: in the ADE menu: Tools → Results Browser …

• Select Outputs \rightarrow to be plotted \rightarrow all,...

The Results Browser





EXERCISE: MIXED MODE SIMULATION

Exercise: Clock Generation

- Step 1: Create a 'ClockGenerator' cell
 - Generate a Verilog view
 - Use a parameter
 - parameter del=10;
 - to set the clock frequency. (Parameters can be overwritten in the properties of the schematic instance. You may have to change the 'CDF Parameter of view' combo box to 'verilog')
 - Follow all steps until you have the symbol
- Step 2: Create a new schematic (for simulation)
 - Instantiate the ClockGenerator
 - Add an inverter or at lease a RC element to do something with the clock
- Step 3: Mixed mode simulation
 - Follow all described steps to setup and run a mixed mode simulation
 - Browse through the results

- Step 4: Divide by 2:
 - Create an edge triggered flipflop from two latches (or take if from a SUSLIB..)
 - Use it to divide the clock by 2.
- Step 5: Checking via Verilog: 'ClockChecker' cell
 - Make a Verilog module which has a clock output and an *input* for the divided clock
 - Use Verilog code to verify that the clock is divided correctly
- NOTE: When re-running the simulation, the results in the lower hierarchy might be missing despite for 'save all'.
 → Closing and re-opening the results browser should fix this.