HDL-Designer & Modelsim & ISE Cheatsheet

Hes·so VALAIS WALLIS
School of Engineering

HDL Shortcuts

File

Save	Ctrl+s
Print	Ctrl+p
Undo / Redo	Ctrl+z/y

Edit

Copy Element	Ctrl+c
Paste Element	Ctrl+v
Cut Element	Ctrl+x
Delete Element	delete
Select All	Ctrl+A

View

View All	Home
Zoom In/Out	Shift+Up/Down
Pan Left/Right	Shift+Wheel
Pan Up/Down	Wheel

HDL

View Generated HDL	Ctrl+g
Object Properties	Alt+Enter
Connect	Alt+c
Disconnect	Shift+Alt+c

Add

Add Signal	s
Add Bus	u
Add Bundle	n
Add In /Out Port	i
Add Out Port	o
Add Inout Port	t
Add Blue Bloc	b
Add Green Component	c
Add Yellow Embedded Code	e

Modelsim Shortcuts

Standard

Load	
Save	Ctrl+s
Print	
Copy Element	Ctrl+c
Paste Element	Ctrl+v
Cut Element	Ctrl+x
Undo / Redo	Ctrl+z/y

Mode

Select Mode	
Zoom Mode	

Zoom

Zoom In	i
Zoom Out	o
Zoom Full	f

Wave Cursor

Add Cursor	a
Delete Cursor	
Jump to Cursor	c
Jump to next/prev edge	
Jump to next/prev falling edge	
Jump to next/prev rising edge	

Simulation

Restart Simulation	
Run	
Run All	
Break Simulation	
Stop Simulation	

Simulation

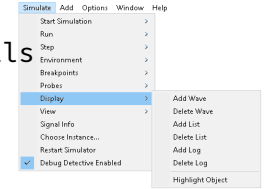
Start Simulation

1. Select Testbench to simulate
2. Compile
   - Compile through components
   - Compile toplevel only
   - Compile through blocs only
   - Compile design root (ifdef)
3. Select .do file

Waveform Signals

Add Signal

Select Signals
Simulate ⇒
Display ⇒
Add Wave



Run (commands)

run <time>	run for given time
run 100 us	
run -all	run until wait in TB
restart -f	force restart

Run (graphic)

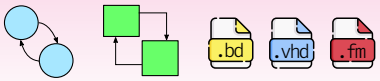
run <time> | 100 us
run -all
restart

Time Units

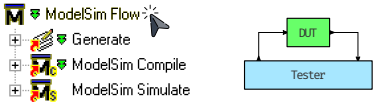
sec|ms(10^{-3})|us(10^{-6})|ns(10^{-9})|ps(10^{-12})

How to Program an FPGA

1. Generate Design using HDL-Designer

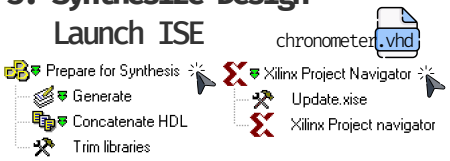


2. Test Design in Simulation



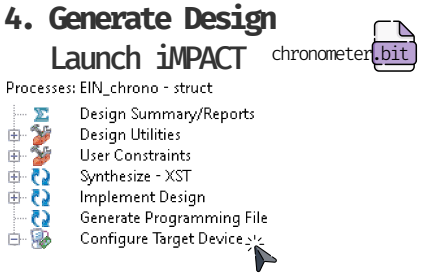
3. Synthesize Design

Launch ISE

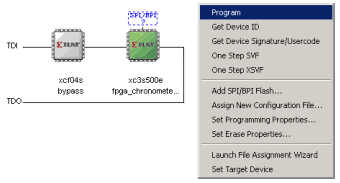


4. Generate Design

Launch iMPACT



5. Programming FPGA



Don't Panic

Designed by AmA & ZaS - 2020 - Icons made by Freepik from Flaticon

HDL-Designer & Modelsim & ISE Cheatsheet

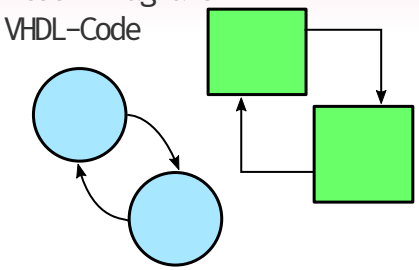
Hes·so VALAIS WALLIS
School of Engineering

How to Program an FPGA

Click

1. Generate Design using HDL-Designer

- Finite State Machines
- Block Diagrams
- VHDL-Code

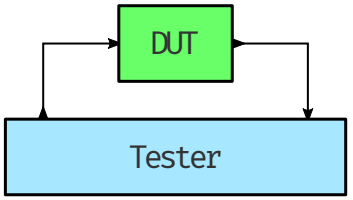


.vhd

.bd .vhd .fm

2. Test Design in Simulation

ModelSim Flow
- Generate
- ModelSim Compile
- ModelSim Simulate



.bd .vhd .fm

chronometer.vhd

3. Prepare for Synthesis

Launch Xilinx Project Navigator (ISE)

- Generate VHDL Files
- Concatenate VHDL Files
- Trim Libraries
- Create ISE Project File
- Start ISE Program

Board
- EIN_chrono [root]

Prepare for Synthesis
- Generate
- Concatenate HDL
- Trim libraries

Xilinx Project Navigator
- Update.xise
- Xilinx Project navigator

4. Synthesize Design

Launch iMPACT

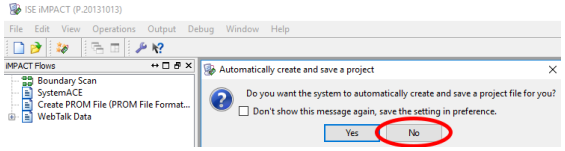
Synthesize → .edif → Place & Route → Generate Programming File

Processes: EIN_chrono - struct
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
- Implement Design
- Generate Programming File
- Configure Target Device

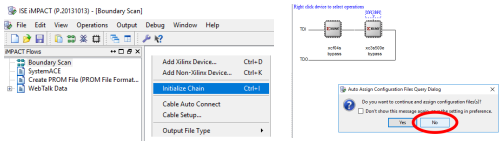
chronometer.bit

5. Programming FPGA

Launch Impact

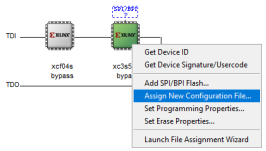


Initialize Chain

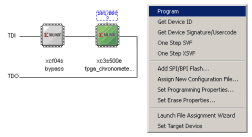


Assign Configuration File

.msc (ROM / left) → keeps code when FPGA is reset
.bit (FPGA RAM / right) → loses code when FPGA is reset



Program



Congratulations!

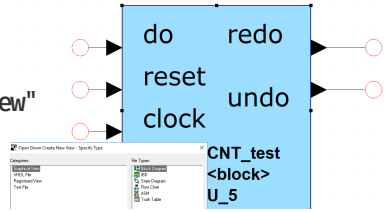
Designed by AmA & ZaS - 2020 - Icons made by Freepik from Flaticon

HDL-Designer & Modelsim & ISE Cheatsheet

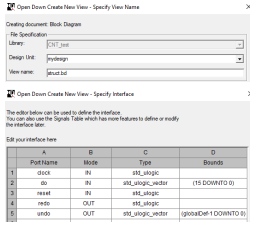
Hes·so VALAIS WALLIS
School of Engineering

Create component

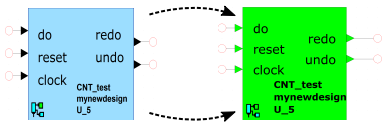
Add bloc
Wire needed I/O
Double-click or "Open As" → "New View"
⇒ "Graphics View"
⇒ "Block Diagram" for blocks
⇒ "State Diagram" for state machine



Give your unit a name (block name)
Give the view a name
(per bloc different view are possible)
Ensure the correct I/O type
Set the bounds for multi-bits types



Right-click on bloc
⇒ Convert Bloc to Component



Component interface

Right click on component
⇒ "Open As" ⇒ "Interface"

	A	B	C	D	E	F	G
	Group	Name	Mode	Type	Bounds	Initial	Comment
1		clock	IN	std_ulogic			
2		do	IN	std_ulogic_vector	(15:0)		
3		reset	IN	std_ulogic			
4		redo	OUT	std_ulogic			
5		undo	OUT	std_ulogic_vector	(globalDef-1:0)		
6							

Interface tab allow to modify I/O

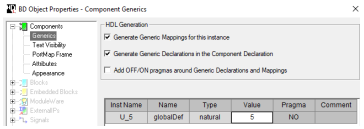
Symbol tab allow to modify visual representation :
- position of I/O Ports
- size of the component

	A	B	C	D	E	F
	Group	Name	Type	Value	Pragma	Comment
1		globalDef	natural	16	NO	
2						

Generics tab allow to set default generic values

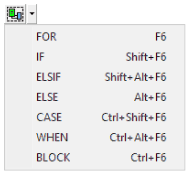
Generics mapping

Generics given in last tab are default values.
They can be overriden :
right-click on component
⇒ "Generics / Parameters"
⇒ "Edit mappings"



FOR / IF Generate

They are both available while creating the circuit under :

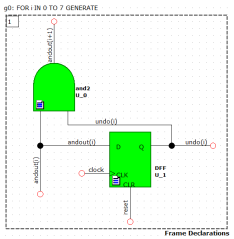


VHDL Types

Types tell us "how the data is represented" and "what value it can take"

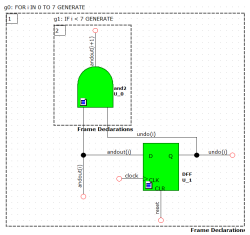
Std_logic	1 bit resolved signal
Std_ulogic	1 bit unresolved signal The difference is while wiring two outputs together, the ulogic would produce an error during compilation, while the logic would allow the simulation to run, and output an 'X' value.
Std_(u)logic_vector	Bus of multiple bits
Unsigned	Bus of multiple bits read as unsigned value
Signed	Bus of multiple bits read as 2cl value
Natural	Integer type ranging from 0 to 2'147'483'647.
Time	"physical" type representing the time primary base of 1 [fs] - Ex: TIME_DELTA: time := 100ns;

FOR Generate frames allow to create n iterations of the same structure



Watch out !
Here, we will do this block with i from (0 to 7)
Signal **undo** must be at least (7 DOWNTO 0)
But signal **andout** must be at least (8 DOWNTO 0)
(as we have andout(i+1),so when i = 7 ⇒ i+1 = 8)
Also, ensure that the first bit **andout(0)** is connected.

IF Generate frames allow to create structure if a given condition is fulfilled



Here, the **IF Generate** tells HDL to create the **AND** gate only if **i < 7**.
So, when **i** will be 7 (or more if it could), the gate would not be generated, and so **andout** stays in the bounds (7 DOWNTO 0).

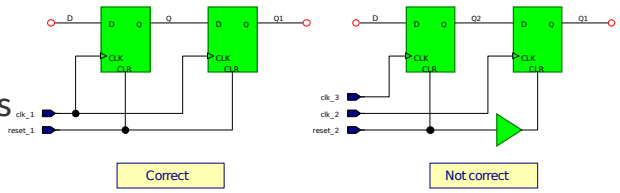
HDL-Designer & Modelsim & ISE Cheatsheet

Hes·so VALAIS WALLIS
School of Engineering

1st rule

Same clock and reset

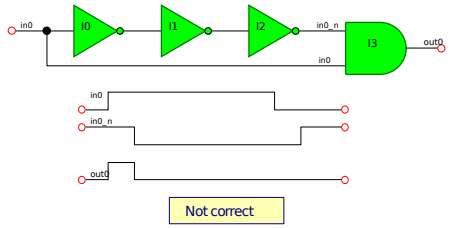
Each sequential logic uses the same clock and reset



2nd rule

No use of gate delay's

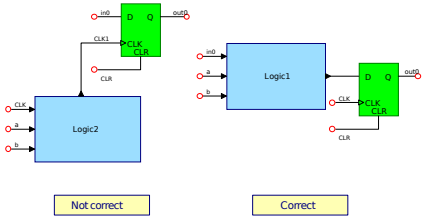
Never use logical elements to create a delay



3rd rule

Clock connection

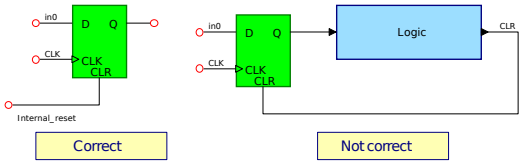
No control of the clock with a logical element



4th rule

Reset connections

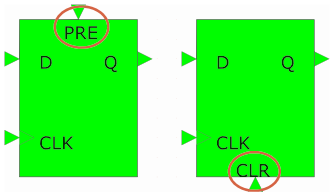
Asynchronous inputs cannot be used to create a functionality of the circuit



5th rule

Initial state

Every sequential logic needs to be initialized at the beginning

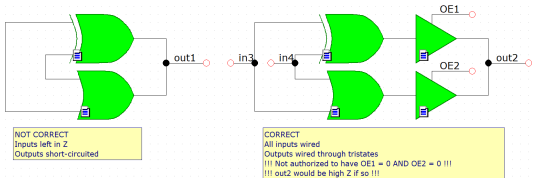


6th rule

I/O connections

Input must not be left in high impedance

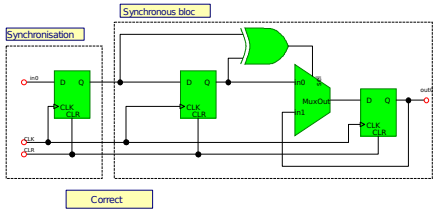
Outputs must not be short-circuited (unless tristates are used)



7th rule

Synchronization

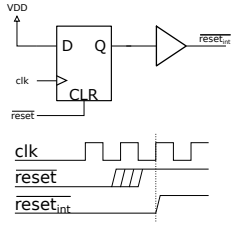
Inputs needs to be synchonized with D-FlipFlops



8th rule

Asynchronous reset

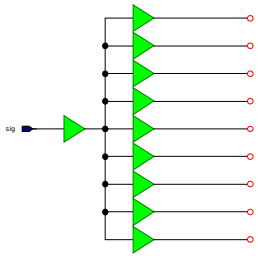
The internal reset disappearance must be synchronous to the clock, but its appearance is asynchronous



9th rule

Gate fan-out

Evaluate the fan-in and fan-out of the gates



Maximal clock frequency

$$T_{min} \leq T_{ClkQ_{max}} + TQD_{max} + T_{skew} - T_{setup_{max}}$$

- TClkQ - delay time between the clock edge and the flip-flop output Q
- TQDmax - delay of the longest chain of gates between an output Q of a sequential logic and an input D of a sequential logic responsive to the same edge of the same clock
- Tskew - is the clock shift with respect to the clock inputs of the sequential logic
- Tsetup - is the setup time of the sequential logic