High-Level Digital System Design with COSIDE[®] using MatchLib and Catapult[®] HLS

Petri Solanti, Senior Field Application Engineer Siemens EDA, a part of Siemens Digital Industries Software

Restricted | © Siemens 2021 | COSEDA UGM 2021-11-09 | Petri Solanti | Siemens Digital Industries Software | Where electronic innovation meets tomorro



Agenda

Challenges in Designing AI/ML Hardware AI Accelerator Development Analyzing and Optimizing System Performance Implementing Accelerator HW with High-Level Synthesis Conclusions



AI/ML Application Challenges

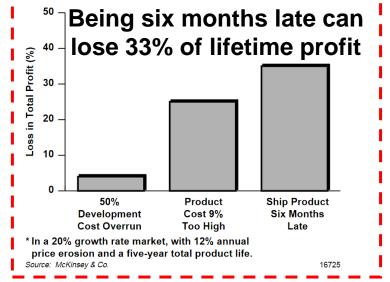
Algorithmic Complexity

- Growing faster than the ability of RTL designers to code and verify
- Memory Architecture Complexity
- Efficient data movement is key for power, performance and area
- **RTL** Verification Costs Increasing
- Increased design complexity increases bugs introduced during hand-coding of RTL
- RTL regressions involve server farms, electricity cost, licenses and time

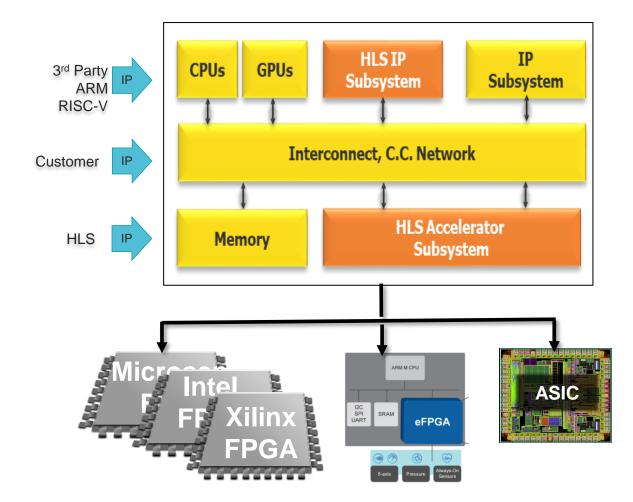
Slips in Design Schedule Kills Total Profit

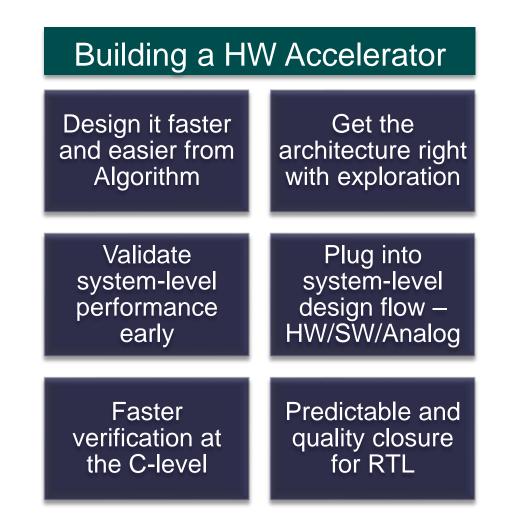
• Finding bugs during system integration is too late





Designing HW Accelerator MUCH Faster @ C-Level with HLS

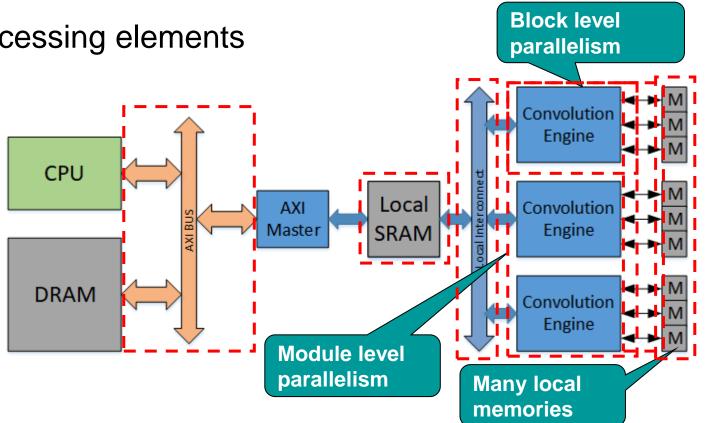






CNN Architectural Challenges

- Memory architectures need to leverage data reuse and parallelism
- May have multiple engines or processing elements
 - Block level parallelism
 - Module level parallelism
- Many local memories
- Complex interconnect
 - AXI4, local interconnect, etc.



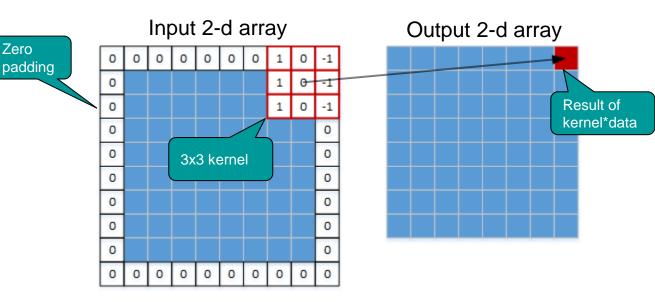
Example: Implementing 2-d 3x3 Image** Convolution Algorithm

- 2-d kernel
- 2-d input array
- Padding left, right, and top and bottom
- Stride = number of elements kernel jumps by

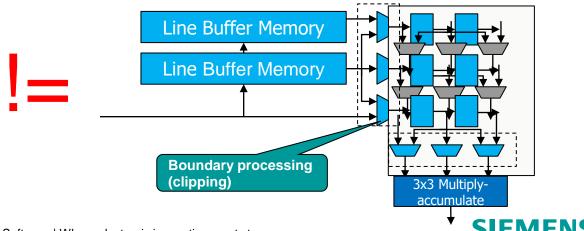
```
2-d Convolution Algorithm - stride 1, zero pad
```

```
HEIGHT:for(int r=0;r<IN_HEIGHT;r++) {
WIDTH:for(int c=0;c<IN_WIDTH;c++) {
KERNEL_R:for(int i=0;i<3;i++) {
KERNEL_C:for(int j=0;j<3;j++) {
if(r+i-1 < 0 || r+i-1 >= IN_HEIGHT)
data = 0;//zero pad
else if(c+j-1 < 0 || c+j-1 >= IN_WIDTH)
data = 0;//zero pad
else
data = data_in[r+i-1][c+j-1];
acc += data * kernel[i][j];
}
result[c] = acc;
** ML convolution usually has 2 loops more
```



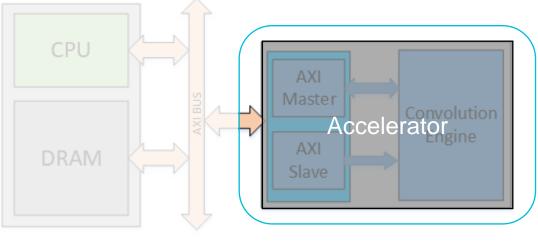


Efficient Convolution Hardware Architecture



Al Accelerator Development

- Implementing AI accelerator for software driven AI system
- Requires AXI master and slave interfaces and the accelerator algorithm
- Balancing between HW model abstraction level and SystemC complexity
- Need a tool flow that enables easy modeling, detailed analysis and seamless RTL implementation path



ML accelerator block diagram

Complex Bus Protocols are Easily Modelled with MatchLib

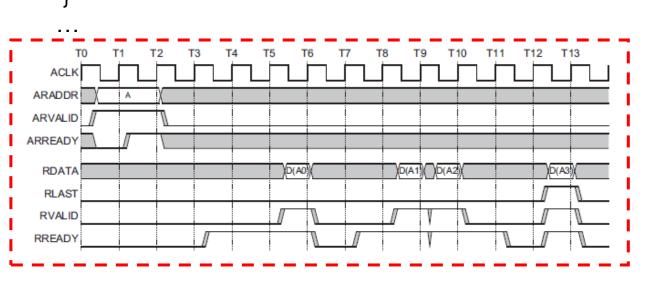
- AXI4 Master and Slaves are part of the MatchLib library
 - AXI4 segmenter transparently manages burst size, 4k boundaries, etc
- AXI4 Read Master I/F with built-in segmentation
 - r_master<>
- AXI4 Write Master I/F with built-in segmentation
- w_master<>
- AXI4 Slave I/F with built-in segmentation
- slave<>
- Interface reads/writes use connections Push/Pop methods
 - Generates AXI4 transactions

```
class bus_if: public sc_module
```

```
r master<> r master0;
```

```
...
while (1) {
```

```
r_payload r = r_master0.r.Pop();
```



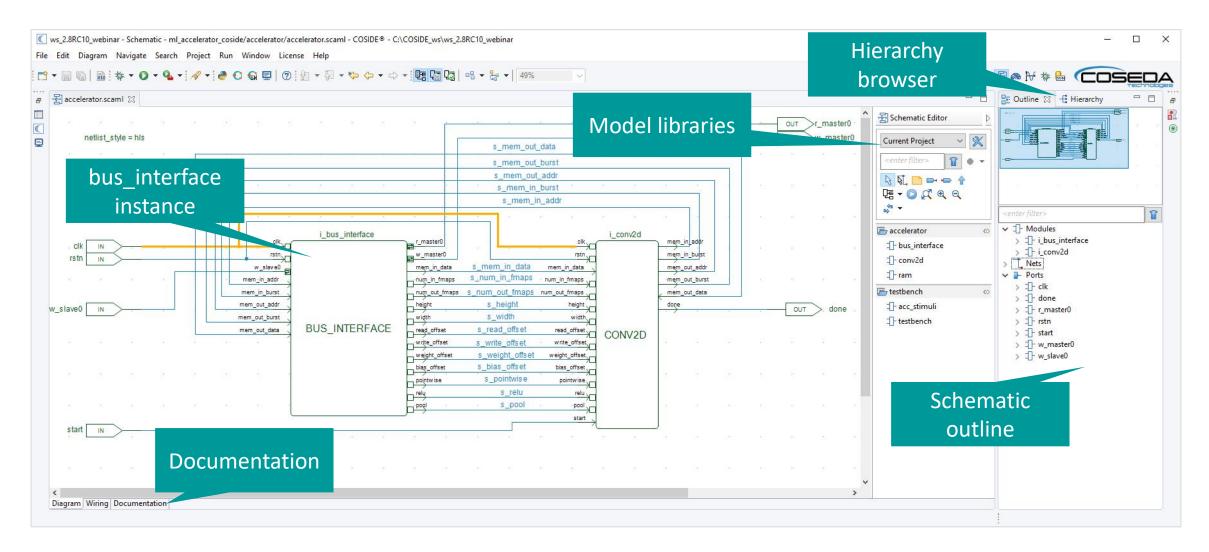
Creating Accelerator Modules with COSIDE Module Editor

- Port definition
 - SystemC
 - MatchLib Connections
- Parameter definition
- Process definition
- Enable HLS compliance
- Code generation
- Description for documentation purpose

Overview	Ports	Parameters	Processes	Includes	User Me	ethods	Registers	Document	ation Adva	anced	
▼ General	1010	1 didificters	, inclusion	mendees	oser m	Linous	negisters	bocamen		inces.	
Type so	_core::sc_modul	e 🔻		before end of		i .		start of simula			
Description bu	is interface			end of elabora	tion		(2=0)	end of simula	tion		
▼ Ports											
+ - 1 .		filter									
Name Type				Data Type			Description				
clk	sc_core::	sc_in	bool		c	lock port			default		
rstn	sc_core::	sc_core::sc_in		bool		eset port		default			
r_master0	local_axi	local_axi::read::master<>				AXI master - read connection			default		
w_master0	local_axi	local_axi::write::master<>		AXI ma		XI master -	master - write connection				
w_slave0	local_axi	local_axi4_lite::write::slave<>		AXI-lite slave - w			e - write connect	tion	default		
mem_in_addr	nnections::In <mem_addr_type></mem_addr_type>		TYPE>	memory address connect			lress connection		default		
mem_in_burst	Connect	ions::In <burst_type< td=""><td>E></td><td>BE Outline</td><td>23 -E Hierarchy</td><td>i bus</td><td>_interface?</td><td></td><td>default</td><td></td><td></td></burst_type<>	E>	BE Outline	23 -E Hierarchy	i bus	_interface?		default		
mem_in_data	Connect	ions::Out <dtype></dtype>			elk X			r_master0	rigin		
mem_out_addr	out_addrnnections::In <mem_addr_type></mem_addr_type>		TYPE>		_slave0			w_master0	default		
mem_out_burst	out_burst Connections::In <burst_type></burst_type>		E> /	1		n_addr			naps default		
mem_out_data	Connect	ions::In <dtype></dtype>			n_burst			num_out_f	^{maps} default		
num_in_fmaps	sc_core::sc_out		IN_FMAP_TY	PE H	m_out_addr			height wigth	default		
num_out_fmaps	sc_core::	sc_out	OUT_FMAP_		• >	BUS_I	NTERFACE	read_offse	default		
height	sc_core::sc_out		HEIGHT_TYP	HEIGHT_TYPE		write_offset			and a second second		
width	sc_core::sc_out		WIDTH_TYPE	WIDTH_TYPE				weight_offs	delault		
Parameters								bias_offset			
 Processes 		and the second second						relu	++-		



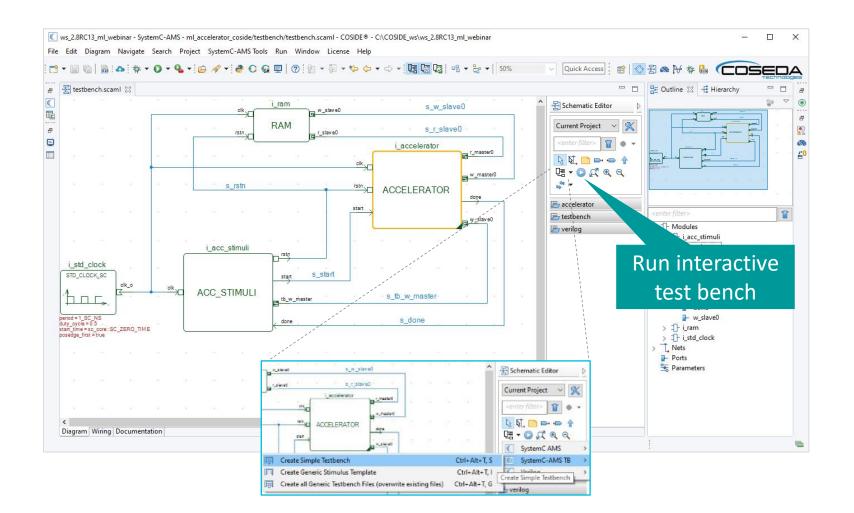
Assembling AI Accelerator with COSIDE Schematic Editor





Creating Testbench with COSIDE Schematic Editor

- Test bench code generation
- Test bench concepts:
 - Interactive/simple
 test bench
 - Stimuli sequence based
 - UVM-SystemC
- Code compilation done
 in background





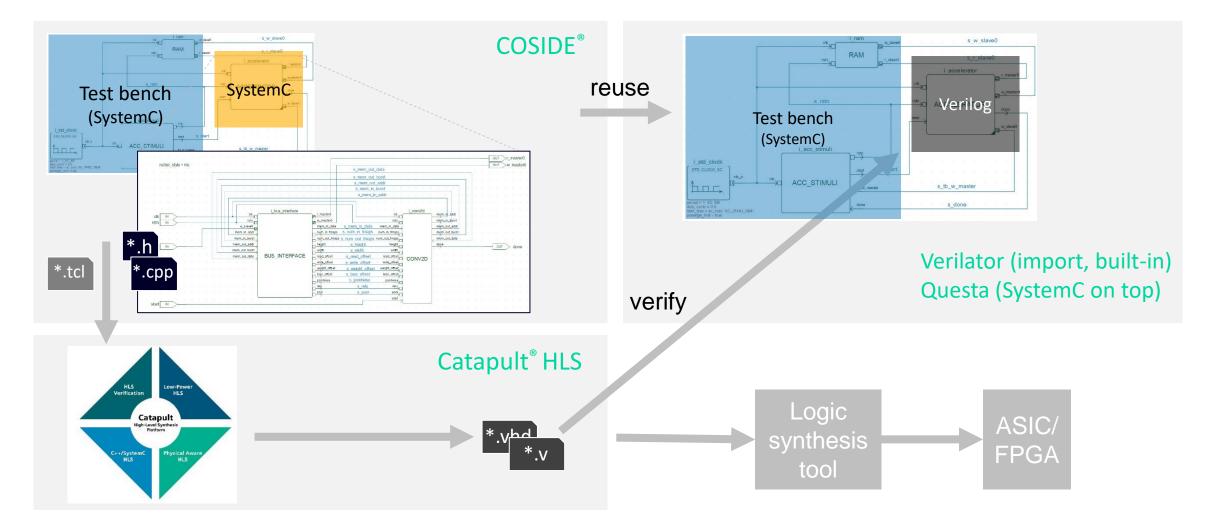
Analyzing Data Traffic with COSIDE Waveform Viewer

- Mixed signal
- Cursor handling
- Measurement utilities
- MatchLib is cycle accurate
 - **RTL-like timing results**
- Python post-processing backend
- Stores layout, cascaded execution of postprocessing

*testbench_simple_tb.vcd 🛛											- 4	📴 Outline 🔀 🕂 Hierarchy	
G* 10. respercification in the proved to				035323		20121	and all		in the second				69
(C)/24	x =1.25971e-3	500e-6		1e-3		1.5e-3	2e-3	-	2.5ę-3	r	3e-3	type filter text	Ø.,
m_out_data_comb_in_val_0	'h1	1									· · · · · · · · · · · · · · · · · · ·	 type inter text testbench_simple_tb.vcd 	
ut_burst_comb_out_msg_0	'h0A8FF	('h00000	X				'hDA8FF					 SystemC 	
out_burst_comb_out_val_0	'h1	-										🗸 達 i_testbench	
out_data_comb_in_msg_0	'h0462	('h0000		'h094A)]]('hFFEF	hFFF8	3	('h0	OCE)	'hFF14	✓ ⇒ i_accelerator	
out_data_comb_out_val_0	'h1	-										888 s_mem_in_ 888 s_mem_in_	
slave0ar_comb_in_rdy_0	'h1											1010 s mem in	
r_slave0ar_comb_in_val_0	'h0		1	T T		1	I. I.	1		1	1	and s_mem_in_	
ave0_ar_comb_out_msq_0	h080069A0900	('h08006	X 'h080	D X 'h080 X 'h0	8006 X 'h080	X 'h080 X 'h08006	X 'h080 X 'h080.	. X 'h08006	('h080) 'h0	80 X 'h08006	X 'h080 X	888 s_mem_in_	1000
slave0_ar_comb_in_msq_0		('h08006	X 'h080	D X 'h080 X 'h0	8006 X 'h080	X 'h080 X 'h08006		X 'h08006)	('h080) 'h0	80 X 'h08006	X 'h080 X	800 s_mem_in_ 800 s_mem_in_	
slave0 r comb out msg 0	'h4FF030					0X'h4FE530X 'h4FFA2		10X 'h405550		C960X 'h4272A0		lili s_mem_in_	C
	1000000000	140127	0 / 11400	//0/14005/0/ 14				10/ 11-000000		C800/ 14272A0	A marzroux	1888 s_mem_in_	
slave0r_comb_out_val_0	'h0	1					-				<u> </u>	1000 s_mem_in_	
slave0r_comb_in_msg_0	'h4FF030	h40127	0 X'h400	770X'h400570X 'h4	IFF030 Xh401EA	0)('h4FE530)('h4FFA2	0 X 'h40 X'h4087	10('h405550)	('h4D)('h4C	C960 ('h4272A0	X'h4F2F50X	888 s_mem_in_ 888 s_mem_in_	6.00
r_slave0r_comb_in_val_0	'h0										1	1010 s mem in	
ve0_aw_comb_out_msg_0	hFF001670000	(h000000	\rangle	'hFF001670000) (hFF0017C2000	hFF00191	4000	('hFFOO	A66000	hFF00	888 s_mem_in_	
lave0_aw_comb_out_rdy_0	'h1	-										100 s_mem_in_	
lave0_aw_comb_out_val_0	'h0											1000 s_mem_in_ 1000 s_mem_in_	-
slave0 b comb in msq 0	'h00	-			-	'h00						1010 s_mem_in_	
	'h0	1				100						1888 s_mem_ou	
aster_w_comb_out_val_0	1222	1							,			and s_mem_out	-
lave0_aw_comb_in_msg_0		(h000000		'hFF001670000		hFF0017C2000	*hFF00191	4000	'hFF00	IA66000)	('hFF00	1010 s_mem_ou	-
slave0_aw_comb_in_rdy_0	'h1	-				Poor	accele	rato	r			000 s_mem_ou 000 s_mem_ou	
slave0aw_comb_in_val_0	'h0	1						naiu	-			lin s_mem_ou	-
slave0b_comb_out_val_0	'h0					utiliza	ation					000 s_mem_ou	
slave0w_comb_in_msg_0	'h7094A	('h00000		'h7094A		uunza			'h7(OCE)	'h7FF14	1010 s_mem_ou	
slave0_w_comb_in_rdy_0	'h1											800 s_mem_ou	-
slave0_w_comb_in_val_0	'h0	1								ſ	1	1010 s_mem_ou	-
slave0b_comb_in_rdy_0	'h1	1			V				No			888 s_mem_ou	t_burs
	1.1994.0	1					_					alla s_mem_ou	-
slave0 b comb in val 0	'h0	1			and the second sec							lini s_mem_out	_data



Integrated High-Level Synthesis Flow



Exporting Accelerator Model to Catapult and Running HLS

Running "Export to Catapult" action – separates in user land and configuration scripts:

Constraints TCL file generation (user land)

- Constraint template e.g. set directives, technology
- e.g. <mod>_synthesis_contraints.tcl

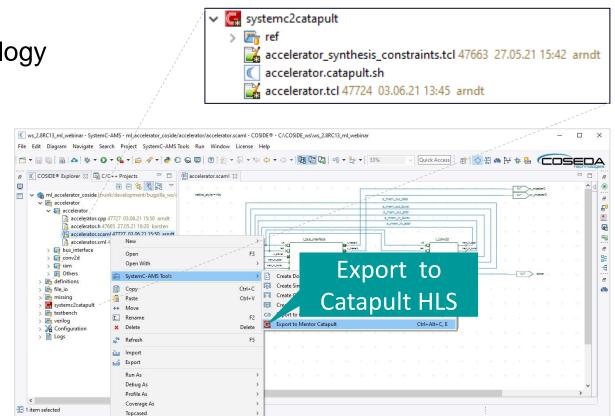
Configuration TCL file generation (auto)

- Register input files to be synthesized
- Compiler flags and invoke constraints file

• e.g. <mod>.tcl

Catapult starter (with/without GUI)

- Shell script to invoke Catapult
- Push-button to start synthesis
- e.g. <mod>.catapult.sh

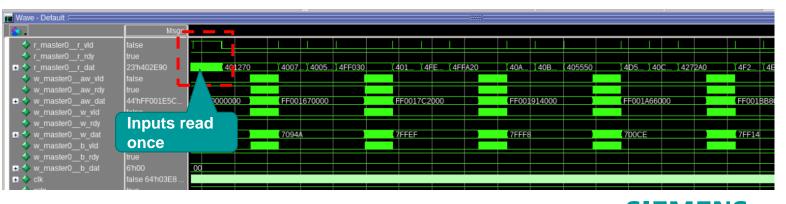




Optimizing System Performance

- Move Bias, ReLU and max pooling from SW into the accelerator
 - Cost little more in hardware area
 - Reduces data traffic between system memory and accelerator
 - Reduces CPU load
 - Memory bus is ~95% used by accelerator
- Add on-chip feature map buffering
 - Minimizes data traffic to minimum





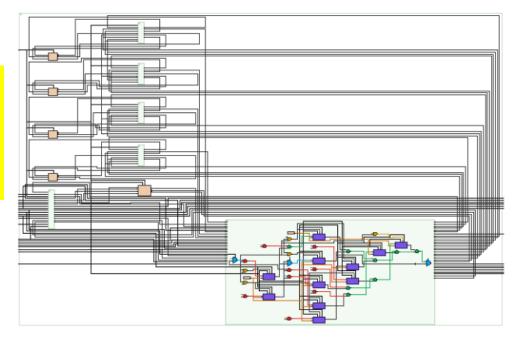
Post-HLS Synthesis RTL Simulation Results

Post-HLS simulation results were very close to the pre-HLS simulation

Post-HLS simulation runtime was over 30x longer than the pre-HLS simulation

Not practical for simulating multiple frames of video

MatchLib supports also a fast simulation mode, which is ~10x faster than accurate mode used here, but the timing accuracy is not as good.



Simulation Type		Simulation Wall-clock time(mins)
Pre-HLS	0.93	20
Post-HLS	0.97	630

Page 16 Restricted | © Siemens 2021 | COSEDA UGM 2021-11-09 | Petri Solanti | Siemens Digital Industries Software | Where electronic innovation meets tomorrow.

Conclusions

- Increasing AI/ML algorithm complexity makes design and verification at RTL-level more difficult and time consuming.
- MatchLib and SystemC allow designers to model and verify the true hardware performance, catching bugs early that would normally be exposed during system integration when it's too late.
- COSIDE SystemC IDE with integrated Catapult High-Level Synthesis provides an efficient design framework for AI/ML designs.
- Using SystemC with MatchLib and High-Level Synthesis removes the need for RTL design and keeps the abstraction level high, but still HW aware.
- MatchLib models can be directly synthesized to RTL and performance of the pre-hls and post-hls results are near identical.



Siemens EDA:

Where electronic innovation meets tomorrow.

Restricted | © Siemens 2021 | COSEDA UGM 2021-11-09 | Petri Solanti | Siemens Digital Industries Software | Where electronic innovation meets tomorrow.

