#### Runtime Verification Based on Executable Models: On-the-Fly Matching of Timed Traces

Mikhail Chupilko, Alexander Kamkin

# Outline

- Hardware models
- Runtime verification
- Elements of formalization
- Conformance relation
- Conclusion

#### Hardware models

- They are developed in Hardware Description Languages, like Verilog or VHDL
- The result of development is the program being executed in HDL simulator
- The common approach for verification of hardware models is testing of HDL programs
- To automatize testing is possible by means of executable models (e.g. in C++)

#### HDL programs



#### Hardware model behavior

6	- XG	- XE-	- X0+	F+	G+	G+	0+ F	+ 0+	G+	0+	(F+)	G+	0+ (	3+ <sup>(</sup>	F+ X	3+ F-	- O+	(F+)	0+ (	8+ <sup>1</sup> 6	H E	+ 0+	F+	0+	G+	0+ (	6+ G	+ XF-	+ 0+	- <del>0+</del>	<b>0</b> +	0+	G+-	F+	0000	000	0000	00+	F+	6+	0+	0000	3000	+ 0-	- 100	90000	6+ X	F+ 0	+ /F+	- \G+	G
8	- 6		- \0+	XC+	6+	8+	0+ 4	+ 0+	4+	0+	(C+)	0+	4+)(	9 <b>+</b> (	4+ X(	3+ \C+	0+	5+)	0+ )	4+ XG	+ 5	+ \0+	C+	(0+)	8+	0+ X	4+ 0	+ \C	+ 0+	8+	0+	4+)	0+)	C+	000				(C+)	64	1+	000		1.	+ X00	90	Ξx	8+ 0-	+ 0+	- XG+	14
Ē	7	Γ	7		1	Ē	Γ	7		1						Γ	7		Г		Γ	7	F	1				Γ	7		1	Ē															Ĩ			1	
		-	7						7								F	1					F									1							_		ſ			Ť							
		-		٦	, 					1	, 				Г				Г											٦										=				F							
_					-					1	, 			=	=											Г				F	1	H				Г			1				5	F							
(n)	<b>6</b>	<u> </u>	000		_γ,	. Y 1.	Vaa	0	_ Ve	. Ve	u Va	. Yı	. Va.	Yie	31	Vaaa			γ1.	Y1.	VI-	000					Ye	000		 Y	Ve	Va	Va	90	Ye	Var	Ē	000		=		Y1.	Y <sub>1</sub>			000	Ya	Veed	, y	1+ Ve	
ĕ	72	ae l	000		Ve		00		Na			e 0	- 01	Vie	10	000			Van	0.11	200	000					Vie	000	_				- 0	00	Vea	00	er.	000		=		Van	Vie	00	10	000	Ver	000	Ŧ	26 10	
H	73 80		00		2 		Vaa		- Na					Va.	Va	Veee		000	Nou	Var		00					Var	00				0, 5: Vo			Jaa Va	00	6C) (a.)	00				Jea Var	<u>, 15</u> , Vas	00	40	00	- 196 - Va	Veer		2013 2016	
÷)	<u>0+</u>	<u>ut y</u>	9991	3888	+ /0	нулон	- /00	38886	ндо	+ /0	1 1	+ / 0+	+//0+	7.04	- <u>A</u> OH-	<u>, 666</u>	3888	0004	- <u>70</u> +	<u>, 104</u>	/(0+)	000	3996	19991	33		<u>,0+</u>	.000	1999	<del>04 //0</del>	94 / U	H- / U-	F) GI	3994	-70+-	/ <u>0+</u> )	( <del>11</del> )	9999	888	3999	30	<u>, 104</u>	<u>, 0+-</u>	( <u>0+)</u>	<u>0+</u> /(	00004	- //04		0+-,(	<del>3+</del> /0	±.0
					-					J			L	-					4					1 1			4													_				╞		╘──					
					_																																			_				╇			₽				
																																												╇							
																																														┶━	_				
																								_																_		_					ᆜ				
									_								_												_	_											_										
																			_																																
									1																																										
90	66	X	<u>61 (</u>	<u>00</u> 0	4 2	0 08	3) 00	X	92 0	0	6	9 0	4 90	00	3		)(e	2 00	8	25	68	<u>00</u>	C2) (	30	G	8 34	02	66	80	<u>81)</u>	90 O	6 00	0 6	0 (OS	9 00	80	62	88	<u>)</u> (1)	3 0	8	Xae	00			<u>41)(0(</u>	3			_X1	0
																							Xe	BG	4 0	9					_)(4	0 00	0 0	4)(1)	3)20	66		43 0	8 8	3 0	4 00	)									
															XFE	01	60		02	00		80	90					67	68	80 0	90	X01	2/1	<u>9</u> (9)	3 40	66		80 0	0 0	3 2	0 41	2/01	. 04	80	00	Xe:	2 50	9 00		<u>94 )</u> 8	8
																														_)(e	7 0	8 00	8								00	C) 00	02	00							
÷	ACE	34+	885	B+) F	FFF	6EFI	-000	09100	)FFF	F6E	FF0(	300+	F)	FF-	⊧XFF	FFAB	FFGG	0054	400F	FFF	AB+	F+	F+)(	FFFF	ASF	F000	9057	3 <b>0</b> +	F+)	F+)(	FFFF	A7F	F88	0058	300+	X <sub>FFI</sub>	=F+)	FFFF	+)(F	FFF	+)(F-	+)(F+	F+	FFF	FAS	FF00	005/	100FF	FFA	5+ )(F	-+)
FF	AAF	F00	0055	501F	FFF	+ F	FFF6	FFF0	3+ )(F	FFF	÷+)(F	FFF	ASFF	00+	-XFF	FFAB	FFGG	0054	401+	XFF	FF4F	FF+	F+)	FFFF	ASF	F000	905+	X FFF	FAA	F+)(	FFFF	A7F-	+)(F	+)(0	+XFF	FFA	53F0	0005	901	FFF-	E)/EI	FFFA	15F+	FFF	F+)	F+) 0	+ XEI	FFFA	4FFG	0005	B0
FF	AAF	F00	0055	502F-	+ )(F	FFF	A9FF	90005	602	FFF	+ )(F	FFF	ASFF	00+	- XFF	FFAB	FFGG	0054	402+	) (FF	FFAT	7FF0	0+ XI	FFFF	ASF	+)(FF	FF+	FFF	FAA	F+)(	FFFF	A7F	+)(F	FFF	+XFF	FFA	7FF0	0005	802-	e Xe	FFF-	+ XFF	FFA	4FF0	000	5802	FFFF	A4FF	0000	35BG	+)
=FF	FAS	OFFO	000	5603	FFF	FA9F	F88	90560	3FFI	FFAS	9FFC	3000	H FF	FF-	•XFF	FFAB	FFGG	0054	403F	FFF	AB+	005	8+ X	FFFF	+)(F	+)FF	FFA	7FF@	9999	5803	BFFF	+) (B	+)(F	+)(FI	FFFA	7FF	3000	5803	+)(F	FFF	A63F	-00+	00	SADD	FFF	FA52-	F)(FI	FFFAS	SFF0	0005	6AG
FF	AAF	F00	0055	504FI	FFF	A+/F	FFFA	9FF00	3005	604	+)(F	FFF	ASFF	001	.)/FF	FFAB	FFGG	0054	404F	FF+	XFFF	FFA7	F+- )(	FFFF	+)(F	FFF	163F	3 <b>0</b> +	FFF	F+)(	F#XF	FFF	A7F	+ )(FI	FFFA	5FF	3000	5+ )(F	+)(F	FFF	A63+	- XFF	FFA	4FF0	000	5804	FFFF	A4FF	0000	9 <b>+</b> )(F	FF

#### Reference model-based test oracle



#### Behavior correctness checking

#### **Functional properties**

- Set of reactions is correct
- Each reaction is correct
- Reaction order is correct
- Delays between reactions are correct

#### **Time restrictions**

#### Cycle-accurate checking

#### **Reference model reactions**



## Ambiguity in reaction order

#### **Execution of reference model**



#### Arbitration of reactions

- Reaction arbiter finds a reaction corresponding to the reference model one
- Behavior checking depends on both reference model and on arbitration
- Reaction arbiters encapsulate parts of test oracle functionality aimed at reaction order checking

## Types of reaction arbiters

- Deterministic model-based arbiter arbiter:  $2^{Reaction} \rightarrow Reaction \cup \{fail\}$
- Adaptive arbiter arbiter:  $2^{Reaction} \times Reaction \rightarrow Reaction \cup \{fail\}$
- Two-level arbiter
   arbiter(reactions) = arbiter<sub>2</sub>(arbiter<sub>1</sub>(reactions), reaction)
  - Non-deterministic arbiter
  - Adaptive arbiter

#### Deterministic arbiter



#### Adaptive arbiter



#### **Two-level** arbiter



## Timed word (Alur & Dill, 1994)

 $\Sigma$  – alphabet of events T – time domain ( $\mathbf{R}^{\geq 0}$  or N)

$$w = (a_0, t_0)(a_1, t_1), \ldots \in (\Sigma \times \mathbf{T})^{\omega^{(*)}}$$

- $\forall i \cdot t_i < t_{i+1} \ (t_i \le t_{i+1}) \text{monotonicity}$
- $\forall T \exists i \cdot t_i > T \text{progress (if } |w| = \infty)$

#### Mazurkiewicz trace (1977)

 $\Sigma-$  alphabet of events  $I \subset \Sigma {\times} \Sigma - \text{relation of independence}$ 

*Equivalent*:  $u \equiv v \Leftrightarrow u$  is derived from v by means of reordering of closest independence events

*Trace* is a class of equivalence of event chains in respect to equivalent relation  $\equiv$ 

#### Mazurkiewicz trace (1977) - Example

$$[ab]_{\equiv} = \{ab, ba\}$$

$$[bc]_{=} = \{ bc \}$$

 $[abcd]_{=} = \{ abcd, bacd, abdc, badc \}$ 

#### Partially ordered set – Pratt (1982)

- $\Sigma$  alphabet of events Pomset is tuple  $\langle V, \leq, \lambda \rangle$
- V set of vertexes
- $\leq \subset V \times V partial set$
- $\lambda: V \to \Sigma$  labeling function

#### Partially ordered set – Pratt (1982) Examples



## Timed trace – Chieu & Hung (2012)

- $\Sigma$  alphabet of events, **T** time domain Timed trace –  $\langle V, \leq, \lambda, \theta [, \delta] \rangle$
- V set of vertexes
- $\leq \subset V \times V partial order$
- $\lambda: V \rightarrow \Sigma$  labeling function
- $\theta: V \rightarrow T time of event$
- $\delta: V \rightarrow \Delta T allowed interval$

#### Timed trace – Chieu & Hung (2012) Examples



- { abcd, bacd, abdc, badc }
- { abcd, bacd } time restrictions

# Behavior of specification and implementation

Implementation behavior  $\langle \mathbf{V}_{I}, \mathcal{O}, \lambda_{I}, \mathbf{\theta}_{I} \rangle$ 

Specification behavior  $\langle \mathbf{V}_{S}, \leq, \lambda_{S}, \boldsymbol{\theta}_{S}, \boldsymbol{\delta}_{S} \rangle$ 

Allowed time interval  $\delta_{s}(x) = [\theta_{s}(x) - \Delta t(x), \theta_{s}(x) + \Delta t(x)]$ 

Correspondence of events **match**(x, y) =  $(\lambda_I(y) = \lambda_S(x)) \& (\theta_I(y) \in \delta_S(x))$ 

#### **Conformance relation**

 $I \sim S \Leftrightarrow \forall t \in \mathbf{T}$ .

 $\exists \mathbf{M} \subseteq \{ (x, y) \in \mathbf{past}_{S}(t) \times \mathbf{past}_{I}(t) \mid \mathbf{match}(x, y) \}$ 

- **M** one-to-one relation
- $\forall x \in \mathbf{past}_{S}(t \Delta t) \exists y \in \mathbf{past}_{I}(t) . (x, y) \in \mathbf{M}$
- $\forall y \in \mathbf{past}_I(t \Delta t) \exists x \in \mathbf{past}_S(t) . (x, y) \in \mathbf{M}$
- $\forall (x, y), (x', y') \in \mathbf{M} . x \leq x' \Longrightarrow \theta(y) \leq \theta(y')$

#### **Reaction arbiters**

# $arbiter_1 = \min_{\leq}(X)$ $X \subseteq \mathbf{V}_S$

**arbiter**<sub>2</sub>(y, X) = 
$$\begin{cases} x, \exists x \in X \text{ match}(x, y) \\ \epsilon, \exists x \in X \text{ match}(x, y) \end{cases}$$
  
y  $\in \mathbf{V}_I, X \subseteq \mathbf{V}_S$ 

#### **Conformance relation checking**



#### C++TESK Testing ToolKit

#### Web: <u>http://forge.ispras.ru/projects/cpptesk-toolkit</u> E-mail: <u>cpptesk-support@ispras.ru</u>

Home My page Projects Help		
ISPRAS C++TESK Testing ToolKit		
Overview Activity Roadmap Issues New issue Gantt Calend	lar News Documents Wiki <mark>Files</mark> Repository Hudson Se	ettings
Files		
File	Date Size	D/L
<b>9 1.0</b>		
cpptesk-toolkit-1.0.1-beta-110415.tar.gz	04/15/2011 04:17 pm 2.1 MB	12
cpptesk-toolkit-1.0.2-beta-110504.tar.gz	05/04/2011 03:14 pm 2.6 MB	6
cpptesk-toolkit-1.0.3-beta-110510.tar.gz	05/10/2011 10:32 pm 4 MB	8
cpptesk-toolkit-1.0.4-beta-110520.tar.gz	05/20/2011 07:31 pm 5.9 MB	5
cpptesk-toolkit-1.0.5-beta-110528.tar.gz	05/28/2011 07:22 pm 6.5 MB	3
cpptesk-toolkit-1.0.6-beta-110621.tar.gz	06/21/2011 09:10 pm 6.8 MB	3
cpptesk-toolkit-1.0.7-beta-110625.tar.gz	06/25/2011 07:10 pm 7.5 MB	1
cpptesk-toolkit-src-1.0.1-beta-110415.tar.gz	04/15/2011 04:17 pm 916.3 kB	23
cpptesk-toolkit-src-1.0.2-beta-110504.tar.gz	05/04/2011 03:14 pm 3.8 MB	12
cpptesk-toolkit-src-1.0.3-beta-110510.tar.gz	05/10/2011 10:32 pm 5.4 MB	19
cpptesk-toolkit-src-1.0.4-beta-110520.tar.gz	05/20/2011 07:31 pm 7.6 MB	9
cpptesk-toolkit-src-1.0.5-beta-110528.tar.gz	05/28/2011 07:22 pm 8.7 MB	13
cpptesk-toolkit-src-1.0.6-beta-110621.tar.gz	06/21/2011 09:10 pm 9.2 MB	4
cpptesk-toolkit-src-1.0.7-beta-110625.tar.gz	06/25/2011 07:10 pm 10 MB	7

## Conclusion

- Based on the theory of traces and partially ordered multisets method of on-the-fly analysis of hardware systems has been developed
- The method has been implemented in C++TESK Testing ToolKit and has been successfully used in a number of projects
- Future research is connected with failure diagnostics: giving hints to localization of bugs

#### THANK YOU

• Any questions?