Scalable Hardware Synthesis & Verification with Craig Interpolation

Jie-Hong Roland Jiang

ALCom Lab

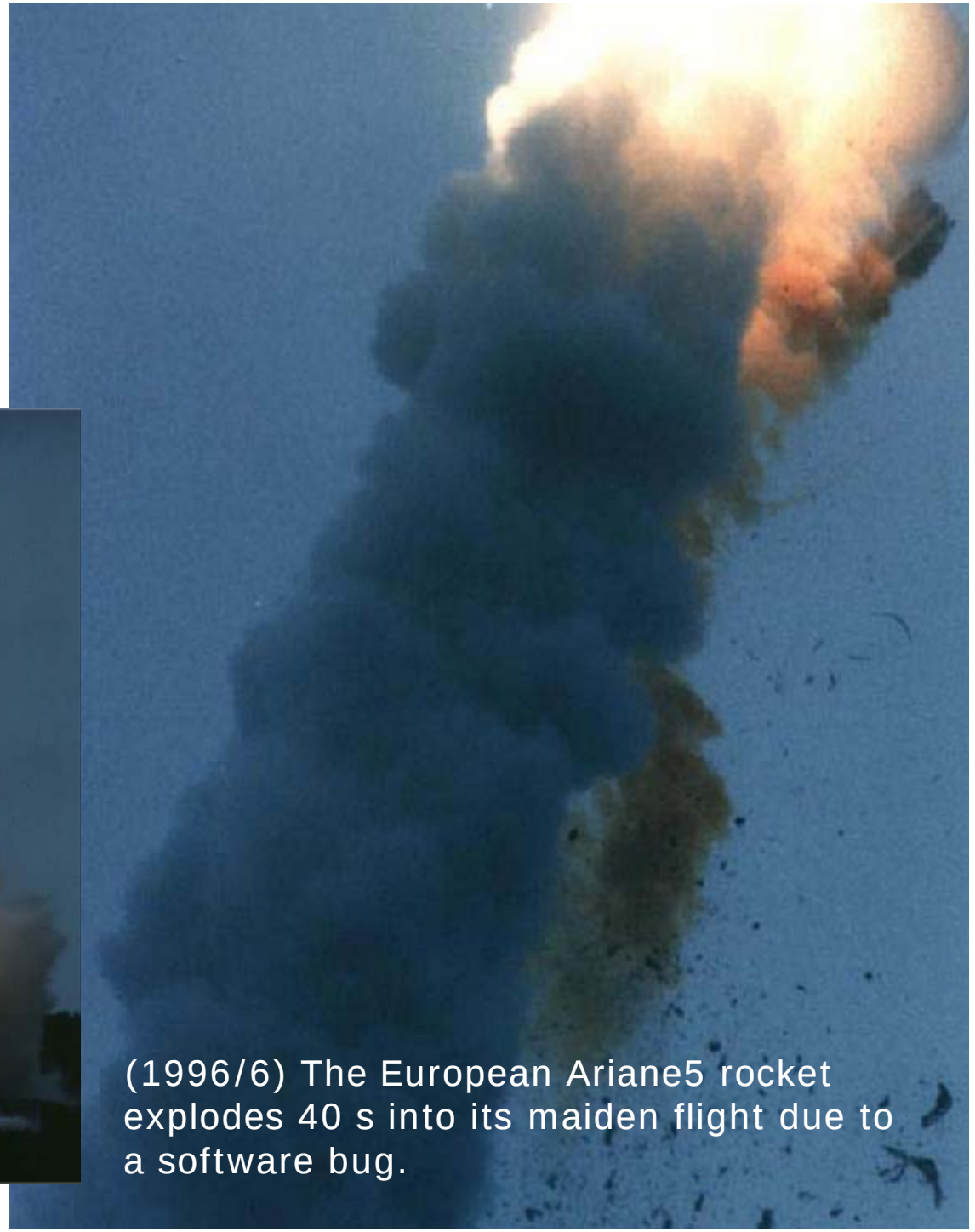
Dept. of Eelcrical Eng. / Grad. Inst. of Electronics Eng.
National Taiwan University
Taipei 10617, Taiwan

Outline

- Introduction
- Applications of Craig interpolation
  - Model checking [CAV03, McMillan]
    - Hardware / software
  - Functional dependency [ICCAD07]
    - Logic optimization [FPGA09, Mishchenko et al.]
    - ECO [FPGA09, Ling et al.]
    - Cyclic circuit synthesis [IWLS09, Backes et al.]
    - Dependent latch removal [FMCAD09, Case et al.]
  - Functional decomposition
    - Bi-decomposition [DAC08]
    - Ashenhurst decomposition [ICCAD08]
  - Quantifier elimination [CAV09]
  - Boolean relation determinization [ICCAD09]
  - Redundant linear constraint removal [TACAS09, Scholl et al.]
- Summary and outlook

Outline

- **Introduction**
- Applications of Craig interpolation
  - Model checking [CAV03, McMillan]
    - Hardware / software
  - Functional dependency [ICCAD07]
    - Logic optimization [FPGA09, Mishchenko et al.]
    - ECO [FPGA09, Ling et al.]
    - Cyclic circuit synthesis [IWLS09, Backes et al.]
    - Dependent latch removal [FMCAD09, Case et al.]
  - Functional decomposition
    - Bi-decomposition [DAC08]
    - Ashenhurst decomposition [ICCAD08]
  - Quantifier elimination [CAV09]
  - Boolean relation determinization [ICCAD09]
  - Redundant linear constraint removal [TACAS09, Scholl et al.]
- Summary and outlook

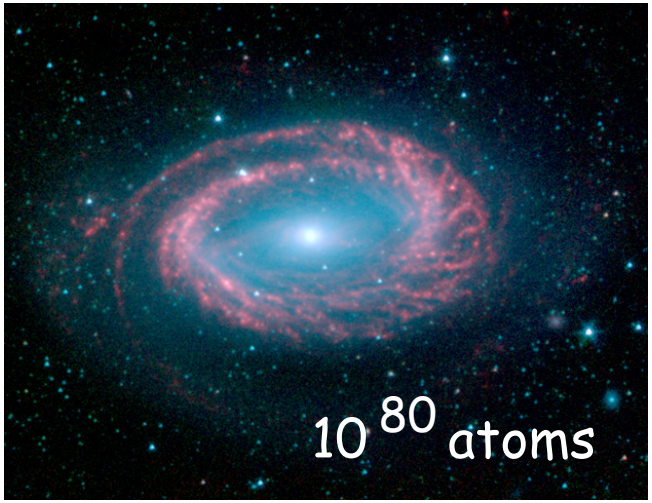


(1996/6) The European Ariane5 rocket explodes 40 s into its maiden flight due to a software bug.

The Paradigm Shift

- Data-structure revolution in formal verification

	State graph (1980s)	BDD (1990s)	SAT (2000s)
Scalability (#states)	~10^4	~10^{20}	~10^{100}



Synergy of Synthesis & Verification

- Logic synthesis techniques have been applied for verification reduction
- Formal verification techniques have been used for logic synthesis

Modern Data Structures

- Available Boolean reasoning packages
  - Satisfiability (SAT) solving
  - And-Inverter Graphs (AIGs)
  - Binary Decision Diagrams (BDDs)
  - Truth tables

- Modern logic synthesis and verification tools well combine different data structures
  - Less and less BDD-based computations
    - BDDs were used to be the pervasive computation technique

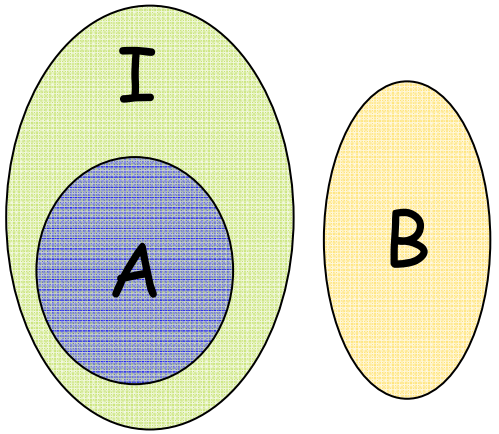
Modern SAT Solving

- Clause/circuit-based reasoning
  - Features
    - Implication
    - Non-chronological backtracking
    - Conflict-based learning
    - Two-literal watching

    E.g., SAT solvers: Chaff, miniSAT, etc.

Craig Interpolation

- SAT solving itself may not yield sufficient information for synthesis and verification
  - Useful information can be extracted from refutation proofs
- Craig Interpolation Theorem [Craig 1957]
  - $A \wedge B$ is UNSAT for clause sets A and B.
    Then there exists an interpolant I of A such that
    1. $A \Rightarrow I$
    2. $I \wedge B$ is UNSAT
    3. I refers only to the common variables of A and B
- An interpolant can be derived from a (resolution) refutation proof in linear time [Pudlak 1997]



I is an abstraction of A

Outline

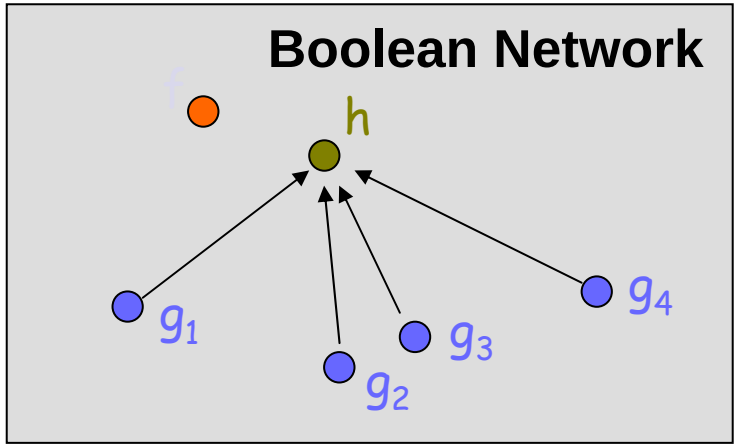
- Introduction
- **Applications of Craig interpolation**
  - Model checking [CAV03, McMillan]
    - Hardware / software
  - **Functional dependency** [ICCAD07]
    - Logic optimization [FPGA09, Mishchenko et al.]
    - ECO [FPGA09, Ling et al.]
    - Cyclic circuit synthesis [IWLS09, Backes et al.]
    - Dependent latch removal [FMCAD09, Case et al.]
  - Functional decomposition
    - Bi-decomposition [DAC08]
    - Ashenhurst decomposition [ICCAD08]
  - Quantifier elimination [CAV09]
  - Boolean relation determinization [ICCAD09]
  - Redundant linear constraint removal [TACAS09, Scholl et al.]
- Summary and outlook

Functional Dependency

- Motivations
  - Many logic synthesis problems can be formulated as computing functional dependencies
    - E.g., resynthesis / rewiring, dependent register removal, etc.
  - Prior methods can only handle circuits with ~20K gates

Functional Dependency

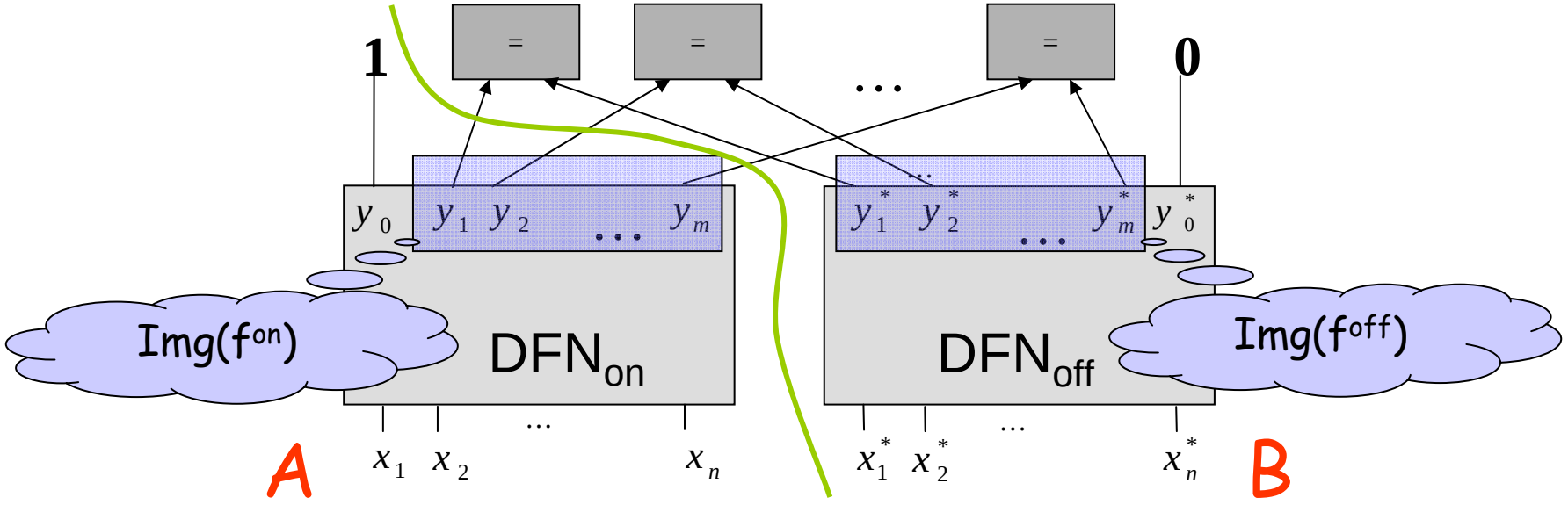
- Functional dependency
  - $f(x) = h(g_1(x), g_2(x), \ldots, g_m(x)) = h(G(x))$
  - h exists $\Leftrightarrow \exists a,b$ such that $f(a) \neq f(b)$ and $G(a)=G(b)$
    - i.e., G is more distinguishing than f



target function
base functions

Functional Dependency

- $(f(x) \equiv f(x^*)) \wedge (G(x) \equiv G(x^*))$ is UNSAT
  - Dependency function h can be obtained with Craig interpolation



Functional Dependency

		Original			Retimed			SAT (original)		BDD (original)		SAT (retimed)		BDD (retimed)	
Circuit	#Node	#FF.	#Dep-S	#Dep-B	#FF.	#Dep-S	#Dep-B	Time	Mem	Time	Mem	Time	Mem	Time	Mem
s35932	16065	1728	0	--	2026	1170	--	176.7	27	1117	164	78.1	27	--	--
s38417	22397	1636	95	--	5016	243	--	270.3	30	--	--	123.1	32	--	--
s38584	19407	1452	24	--	4350	2569	--	166.5	21	--	--	99.4	30	1117	164
b12	946	121	4	2	170	66	33	0.15	17	12.8	38	0.13	17	2.5	42
b14	9847	245	2	--	245	2	--	3.3	22	--	--	5.2	22	--	--
b15	8367	449	0	--	1134	793	--	5.8	22	--	--	5.8	22	--	--
b17	30777	1415	0	--	3967	2350	--	119.1	28	--	--	161.7	42	--	--
b18	111241	3320	5	--	9254	5723	--	1414	100	--	--	2842.6	100	--	--
b19	224624	6642	0	--	7164	337	--	8184.8	217	--	--	11040.6	234	--	--
b20	19682	490	4	--	1604	1167	--	25.7	28	--	--	36	30	--	--
b21	20027	490	4	--	1950	1434	--	24.6	29	--	--	36.3	31	--	--
b22	29162	735	6	--	3013	2217	--	73.4	36	--	--	90.6	37	--	--

Functional Dependency

- Summary
  - Functional dependency is computable with pure SAT-solving and scalable to large designs (over 200K gates)
  - Applications
    - Logic Optimization [FPGA09, Mishchenko et al.]
    - Engineering Change Orders [FPGA09, Ling et al.]
    - Cyclic circuit synthesis [IWLS09, Backes et al.]
    - Dependent latch removal [FMCAD09, Case et al.]

Outline

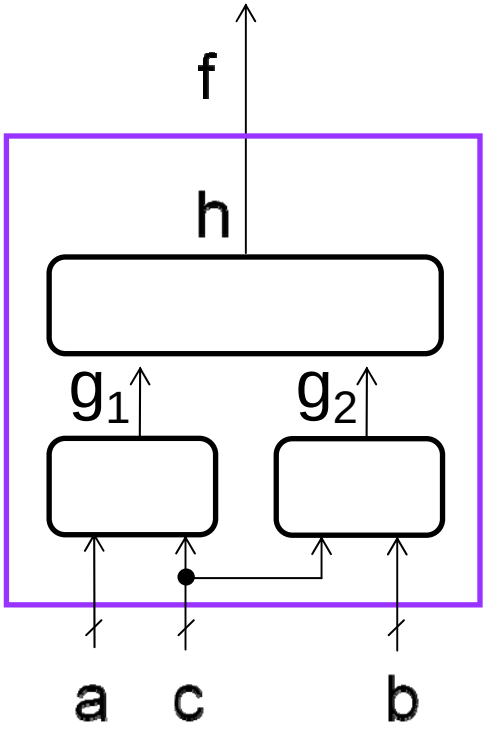
- Introduction
- **Applications of Craig interpolation**
  - Model checking [CAV03, McMillan]
    - Hardware / software
  - Functional dependency [ICCAD07]
    - Logic optimization [FPGA09, Mishchenko et al.]
    - ECO [FPGA09, Ling et al.]
    - Cyclic circuit synthesis [IWLS09, Backes et al.]
    - Dependent latch removal [FMCAD09, Case et al.]
  - **Functional decomposition**
    - Bi-decomposition [DAC08]
    - Ashenhurst decomposition [ICCAD08]
  - Quantifier elimination [CAV09]
  - Boolean relation determinization [ICCAD09]
  - Redundant linear constraint removal [TACAS09, Scholl et al.]
- Summary and outlook

Functional Decomposition

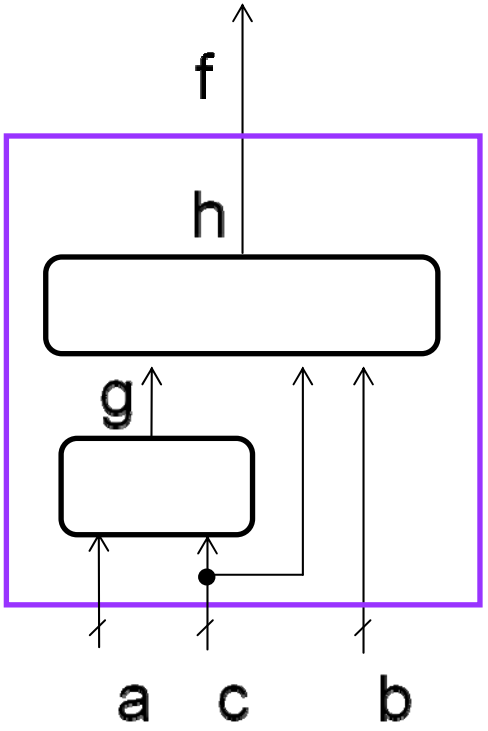
- Motivations
  - Functional decomposition plays an essential role in FPGA synthesis and in reducing circuit communication complexity
  - Prior methods can only handle functions with ~30 variables

Functional Decomposition

Bi-decomposition

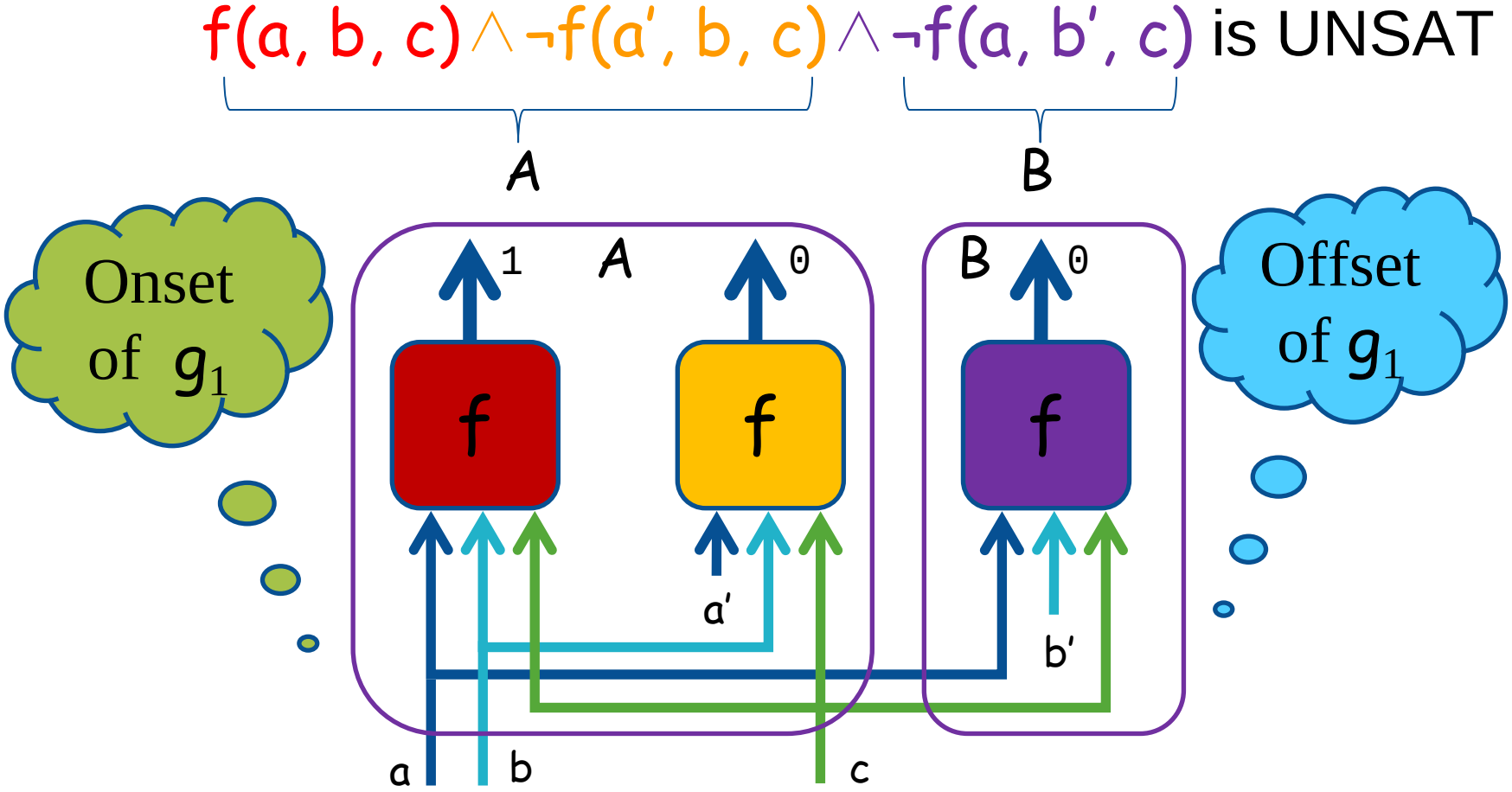


Ashenhurst decomposition



Bi-Decomposition

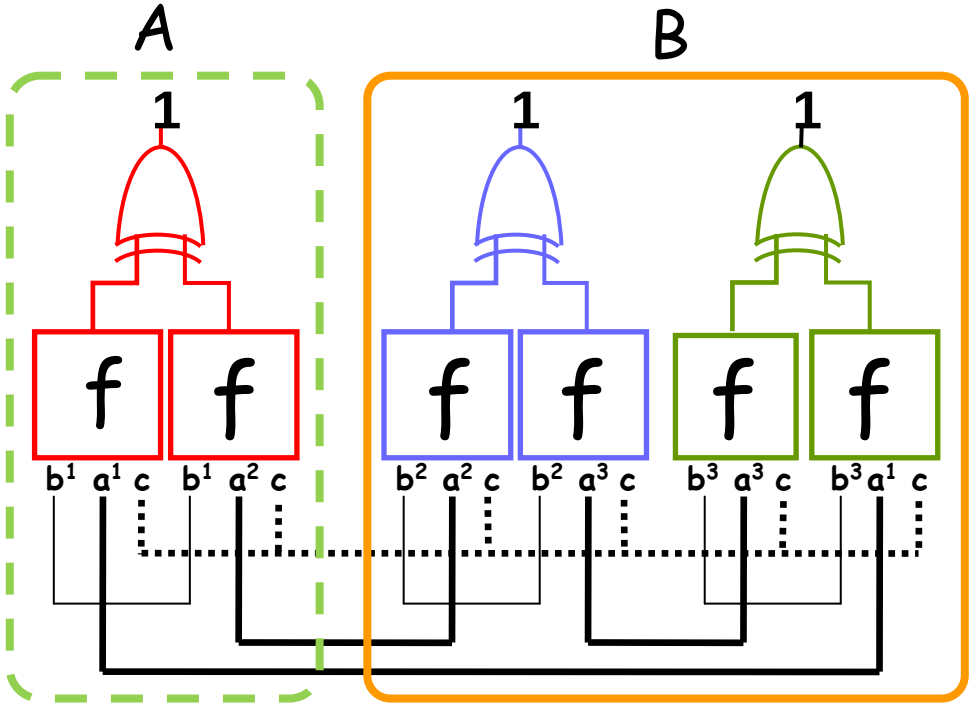
$f(a, b, c) \wedge \neg f(a', b, c) \wedge \neg f(a, b', c)$ is UNSAT



Ashenhurst Decomposition

$$f(a^1,b^1,c) \neq f(a^2,b^1,c) \wedge (f(a^2,b^2,c) \neq f(a^3,b^2,c) \wedge f(a^3,b^3,c) \neq f(a^1,b^3,c))$$

A B



Functional Decomposition

Single-output Ashenhurst decomposition

Circuit	#func	#var	#fail	#succ	#var_succ	#VP_avg	rate_ valid_VP	time_avg (sec)	Mem (MB)
b15	370	143—306	0	319	143—306	1519	0.917	96.62	107.20
b17	1009	76—308	0	861	76—308	1645	0.904	87.12	125.84
c7552	36	50—194	0	34	50—194	1350	0.455	64.38	36.65
s13207	3	212—212	0	3	212—212	569	0.908	70.26	50.62
s38417	256	53—99	6	178	53—99	1090	0.523	103.33	136.04
s38584	7	50—147	0	7	50—147	1120	0.924	47.13	51.56

Two-output Ashenhurst decomposition

Circuit	#pair	#var	#fail	#succ	#var_succ	#VP_avg	rate_ valid_VP	time_avg (sec)	Mem (MB)
b15	201	145--306	0	170	145--269	1176	0.845	113.86	224.07
b17	583	79--310	0	495	79--308	676	0.824	103.12	419.35
c7552	21	56--195	0	19	56--141	188	0.465	89.57	78.67
s13207	3	212--228	0	3	212--228	585	0.7	93.36	118.03
s38417	218	53--116	13	175	53--116	689	0.498	109.06	319.48
s38584	9	50--151	0	9	50--151	1656	0.713	46.17	207.78

Functional Decomposition

- Summary
  - Bi-decomposition and Ashenhurst decomposition are computable with pure SAT-solving and scalable to large Boolean functions (over 300 input variables)
    - Easily extendable to non-disjoint and multiple-output decompositions

Outline

- Introduction
- **Applications of Craig interpolation**
  - Model checking [CAV03, McMillan]
    - Hardware / software
  - Functional dependency [ICCAD07]
    - Logic optimization [FPGA09, Mishchenko et al.]
    - ECO [FPGA09, Ling et al.]
    - Cyclic circuit synthesis [IWLS09, Backes et al.]
    - Dependent latch removal [FMCAD09, Case et al.]
  - Functional decomposition
    - Bi-decomposition [DAC08]
    - Ashenhurst decomposition [ICCAD08]
  - **Quantifier elimination** [CAV09]
  - Boolean relation determinization [ICCAD09]
  - Redundant linear constraint removal [TACAS09, Scholl et al.]
- Summary and outlook

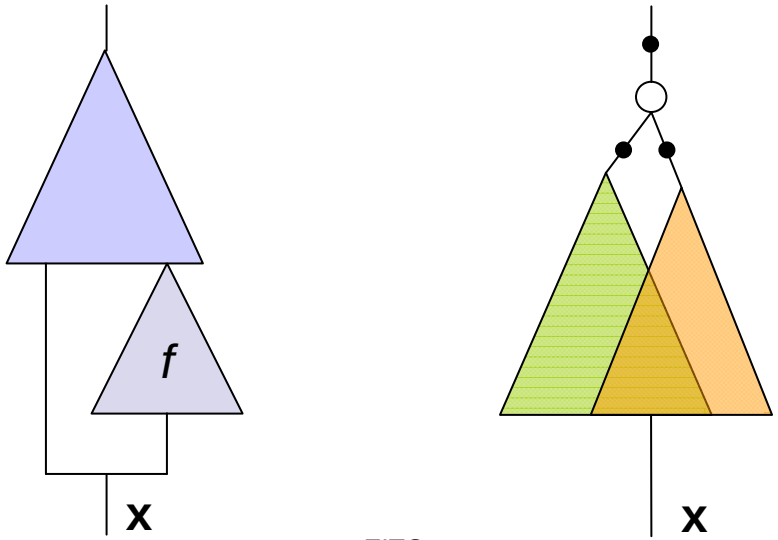
Quantifier Elimination

- Motivations
  - Many computer science problems can be cast as solving quantified Boolean formulas (QBFs)
    - PSPACE complete
  - Prior methods are not scalable
    - Formula expansion: $\exists y\ \varphi(\mathbf{x},y) = \varphi(\mathbf{x},0) \vee \varphi(\mathbf{x},1)$
    - Normal-form conversion
    - Solution enumeration

Quantifier Elimination

- Given a quantified formula $\exists y\ \varphi(\mathbf{x},y)$, what should a function f be such that $\varphi(\mathbf{x},f(\mathbf{x})) = \exists y\ \varphi(\mathbf{x},y)$?
  - Necessary and sufficient condition:

    $(\varphi(\mathbf{x},1) \wedge \neg\varphi(\mathbf{x},0)) \leq f \leq \neg(\varphi(\mathbf{x},0) \wedge \neg\varphi(\mathbf{x},1))$
  - Computable with Craig interpolation



Quantifier Elimination

circuit	(#in, #reg, #n, #l)	rel before QE		QE-Exp				QE-Cmp			
		#n	#l	#n	#l	time	mem	#n	#l	time	mem
prolog	(36, 136, 1656, 26)	1474	29	—	—	—	—	**1088**	**31**	6.27	38.0
s1196	(14, 18, 529, 24)	548	22	**3473**	**21**	5.15	37.3	21881	2532	123.15	37.3
s1269	(18, 37, 569, 35)	622	37	31005	**39**	59.24	37.5	**1694**	116	41.05	37.5
s13207.1	(62, 638, 8027, 59)	5272	45	—	—	—	—	**4741**	**44**	50.60	40.6
s1423	(17, 74, 657, 59)	757	63	17619	**59**	25.45	38.1	**3142**	452	6.19	38.1
s1488	(8, 6, 653, 17)	686	19	1269	**21**	2.90	38.1	**515**	48	3.82	38.1
s1494	(8, 6, 647, 17)	696	20	1261	**21**	2.98	38.1	**607**	42	2.54	38.1
s1512	(29, 57, 780, 30)	697	28	1187	**24**	2.64	37.7	**823**	53	3.78	37.7
s15850.1	(77, 534, 9786, 82)	5679	57	—	—	—	—	**180597**	**14247**	49409.27	427.4
s208.1	(10, 8, 104, 11)	103	14	65	**11**	0.08	37.4	**49**	12	0.06	37.4
s298	(3, 14, 119, 9)	157	15	**117**	**12**	0.08	37.4	122	**12**	0.23	37.4
s3271	(26, 116, 1573, 28)	1565	32	**1549**	**29**	3.08	38.0	1604	62	7.11	38.0
s3330	(40, 132, 1789, 29)	1434	29	—	—	—	—	**1029**	**28**	6.37	38.0
s3384	(43, 183, 1702, 60)	1801	63	1307	**58**	6.94	38.3	**1276**	**58**	17.29	38.3
s344	(9, 15, 160, 20)	164	19	**140**	**19**	0.33	37.1	155	**19**	0.81	37.1
s349	(9, 15, 161, 20)	168	19	**140**	**19**	0.26	37.5	155	**19**	0.82	37.5
s382	(3, 21, 158, 9)	220	19	**179**	**16**	0.10	37.7	189	**16**	0.27	37.7
s38417	(28, 1636, 22397, 47)	15762	44	**15705**	**40**	44.79	48.7	18865	106	149.13	46.8
s38584.1	(38, 1426, 19407, 56)	18094	48	57105	**45**	1382.97	71.4	**38089**	1362	268.94	46.0
b12	(5, 121, 952, 19)	1485	26	**1740**	**24**	0.65	38.3	1908	41	2.21	38.3
b13	(10, 53, 299, 20)	472	20	435	**16**	0.49	37.6	**423**	**16**	1.15	37.6
ratio				1.000	1.000	1.000	1.000	0.036	8.064	0.013	0.952

Quantifier Elimination

- Summary
  - Quantifier elimination with functional composition can be effective for applications satisfying the sparsity condition
    - 27x smaller in AIG size, 8x larger in AIG depth, compared with the expansion based QE

Outline

- Introduction
- **Applications of Craig interpolation**
  - Model checking [CAV03, McMillan]
    - Hardware / software
  - Functional dependency [ICCAD07]
    - Logic optimization [FPGA09, Mishchenko et al.]
    - ECO [FPGA09, Ling et al.]
    - Cyclic circuit synthesis [IWLS09, Backes et al.]
    - Dependent latch removal [FMCAD09, Case et al.]
  - Functional decomposition
    - Bi-decomposition [DAC08]
    - Ashenhurst decomposition [ICCAD08]
  - Quantifier elimination [CAV09]
  - **Boolean relation determinization** [ICCAD09]
  - Redundant linear constraint removal [TACAS09, Scholl et al.]
- Summary and outlook

Boolean Relation Determinization

- Motivations
  - Relation allows one-to-many mapping (non-determinism) and is more general than function
    - Useful in specifying systems, representing circuit flexibilities, etc.
  - Determinizing a relation (extracting functions from the relation) is needed for hardware implementation
    - Circuits are deterministic after all
  - Prior methods are not scalable

Boolean Relation Determinization

	Original				BDD				Xp			
Circuit	**(#pi, #po)**	**#n**	**#l**	**#v**	**#n**	**#l**	**#v**	**time**	**#n**	**#l**	**#v**	**time**
s5378	(214, 179)	624	12	1570	783	10	1561	286.4	1412	25	1561	49.6
s9234.1	(247, 211)	1337	25	3065	---	---	---	---	7837	59	2764	158.6
s13207	(700, 669)	1979	23	3836	---	---	---	---	5772	140	3554	769.3
s15850	(611, 597)	2648	36	15788	---	---	---	---	42622	188	13348	2700
s35932	(1763, 1728)	8820	12	7099	---	---	---	---	7280	10	6843	4178.5
s38584	(1464, 1452)	9664	26	19239	---	---	---	---	22589	277	17678	5772.8
b10	(28, 17)	167	11	159	200	10	152	0.1	197	8	152	0.9
b11	(38, 31)	482	21	416	1301	18	394	0.9	1504	57	394	5.1
b12	(126, 121)	953	16	1639	1663	14	1574	56.7	2166	25	1574	24
b13	(63, 53)	231	10	383	240	10	349	3.1	224	10	349	2.2
Ratio 1		1	1	1					3.40	4.16	0.91	
Ratio 2		1	1	1	1.70	0.89	0.97		2.24	1.79	0.97	
Ratio 3					1	1	1		1.31	2.02	1	

--- **:** BDD nodes are over 500K during computation

Boolean Relation Determinization

- Summary
  - Functions can be effectively extracted from large Boolean relations (with thousands of vars)
    - Extracted functions are typically of reasonable sizes

Outline

- Introduction
- Applications of Craig interpolation
  - Model checking [CAV03, McMillan]
    - Hardware / software
  - Functional dependency [ICCAD07]
    - Logic optimization [FPGA09, Mishchenko et al.]
    - ECO [FPGA09, Ling et al.]
    - Cyclic circuit synthesis [IWLS09, Backes et al.]
    - Dependent latch removal [FMCAD09, Case et al.]
  - Functional decomposition
    - Bi-decomposition [DAC08]
    - Ashenhurst decomposition [ICCAD08]
  - Quantifier elimination [CAV09]
  - Boolean relation determinization [ICCAD09]
  - Redundant linear constraint removal [TACAS09, Scholl et al.]
- **Summary and outlook**

Summary and Outlook

- Craig interpolation becomes an essential technique in more and more applications of system design

- Application beyond the propositional domain awaits future exploration
  - Usefulness of interpolation in Satisfiability Modulo Theories (SMT) solving

Credits

- NTU
  - Wei-Lun Hung
  - Chih-Chun Lee
  - Ruei-Rung Lee
  - Hsuan-Po Lin
- UC Berkeley
  - Robert Brayton
  - Alan Mishchenko
- Xilinx
  - Stephen Jang

Thank you for your attention!