Deeper Bound in BMC by Combining Constant Propagation and Abstraction

Tamir Heyman

R. Armoni, R. Fraer, L. Fix, M. Vardi, Y. Visel and Y. Zbar

ASP-DAC 2007

Hardware verification

- There is a growing use of model checking in hardware verification
- SAT-based Bounded Model Checking (BMC) can handle thousands of state elements
- BMC can find errors up to a small search bound (~100) which limits confidence in correctness

Deeper Bound in BMC is Required

- Communication devices includes long transactions
- Large circuits have larger diameter
- Occasionally, BMC can not reach even a small bound for small circuits

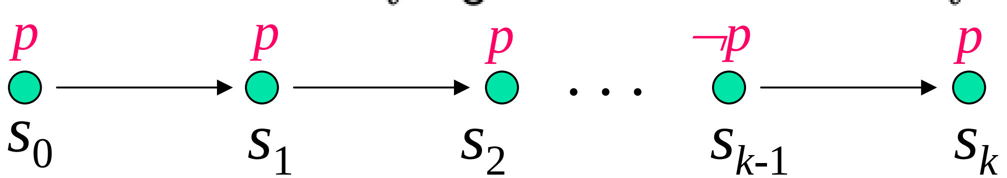
Our Contributions

- Improve BMC using assumptions substitution
- Improve BMC using abstraction refinement
- Combine assumptions substitution and abstraction refinement to reach deeper bounds

Bounded Model Checking (BMC)

- Bounded Model Checking takes a model M, a specification p and a bound k
- The safety property p is valid up to step k iff φ = Ω(k) is unsatisfiable:

$$\Omega(k) = I(S_0) \wedge \bigwedge_{i=0}^{k-1} R(S_i, S_{i+1}) \wedge \bigvee_{i=0}^{k} \neg P(S_i)$$



Assumptions Substitution

- It is common that a hardware specification includes assumptions on the inputs
  - For example, “The input clk toggles every cycle”
- We use these assumptions to reduce φ
- Smaller φ implies smaller SAT instance
- We substitute the value of the input into φ according to the assumption

Example

- Variables x,y
- x'=(!clk/¥clk')? !y' : x
- y'=(clk/¥!clk')? x' : y
- An input clock clk
- The property to check: p(x,y)=(x=y)
- Assumption: Clock initialized to 0 and toggles every cycle

After unfolding

- x_0 , y_0 - free
- x_1 =!clk $_0$/¥clk$_1$? !y_1 : x_0
- y_1=!clk$_0$/¥clk$_1$? x_1 : y_0
- x_2=!clk$_1$/¥clk$_2$? !y_2 : x_1
- y_2=!clk$_1$/¥clk$_2$? x_2 : y_1
- The assumption on the clock
  - clk$_0$=0,clk$_1$=1,clk$_2$=0

After injecting the clock values

- x_0, y_0 - free
- x_1=!0/¥ 1 ? !y_1 : x_0
- y_1= 0/¥!1 ? x_1 : y_0
- x_2=!1/¥ 0 ? !y_2 : x_1
- y_2= 1/¥!0 ? x_2 : y_1

After Constant Propagation clk

- x_0, y_0 - free
- $x_1 = !y_1$
- $y_1 = y_0$
- $x_2 = x_1$
- $y_2 = x_2$

Improved BMC with Abstraction Refinement

- We construct from the hardware circuit a sequence of abstract models
- The model is a conservative abstraction
  - The hardware satisfies the property up to bound k if the abstract model satisfies P up to k
  - Usually, the abstract model is smaller than the hardware

ABMC algorithm

function ABMC(M, P, t, δ)

initialize k

While $k < t$

V = concrete model variables

φ = Build-BMC-formula (V, k)

if SAT-solver(φ)$= SAT$ return CEX

V' = variables in unSAT core

if $\frac{|V'|}{|V|} \leq \delta$

φ = Build-BMC-formula(V', t)

if SAT-solver(φ) $= unsat$

return Valid up to t

$k = k + 1$

return Valid up to t

ABMC algorithm

function ABMC(M, P, t, δ)

initialize k

While $k < t$

  V = concrete model variables

  φ = Build-BMC-formula (V, k)

  if SAT-solver(φ)= SAT return CEX

  V' = variables in unSAT core

  if $\frac{|V'|}{|V|} \leq \delta$

    φ = Build-BMC-formula(V', t)

    if SAT-solver(φ) = $unsat$

      return Valid up to t

  $k = k + 1$

return Valid up to t

ABMC-CCP: BMC with Abstraction and Assumptions

- The abstract model needs to satisfy p up to bound k
- An input that gets values by an assumption may be required for the abstract model
- We use a new conservative constant propagation (CCP)
  - Reduces parts that are don’t care
- The SAT solver complement CCP

CCP: Conservative CP

- We evaluate each node e1 in φ considering all the assumptions
- We create new expression φ' by replacing the expression e=e1 /¥ e2 with e=e1, if e1 evaluated to false
- The expression e2 is not included in φ'

ABMC-CCP is Sound

- The expression φ' translated to CNF and is sent to the SAT solver
- If the solver returns unsat, identify all the parts of the circuit that relate to the unsat proof
- These parts forms the abstract model
- It is a conservative abstract model
  - The circuit satisfies the property if the abstract model does

ABMC-CCP is complete

- If a circuit satisfies a property p up to bound k, then the appropriate φ is unsat
- If φ is unsat φ' is unsat
  - Any satisfying assignment to φ satisfies φ'
- The abstract model build from the unsat proof for φ' does not have any counterexample of lengh k
  - We includes all the parts for repeat the unsat proof

Circuit	#var	ABMC		ABMC-CCP		Ratio
		bound	time	bound	time	
P8	27,201	20	Mout	48	Mout	240%
P15	5,946	281	Tout	512	Tout	182%
P19	6,907	153	Tout	296	Tout	193%
P24	5,954	271	Tout	312	Tout	115%
P38	6,028	187	Tout	300	Tout	160%
P54	6,028	245	Tout	296	Tout	121%
P69	5,938	199	Tout	276	Tout	139%
P45	6,219	369	Tout	1,000	7,869	271%
P37	7,180	355	Tout	694	Tout	195%
Pf	1,585	715	Tout	686	Tout	96%
Pbb	1,458	30	Tout	61	Tout	203%
Pc	1,648	71	Mout	498	Tout	701%
Ave						218%

The End

- Thanks you for listening