EPR-Based Bounded Model Checking at Word Level

Christoph Sticksel

The University of Iowa

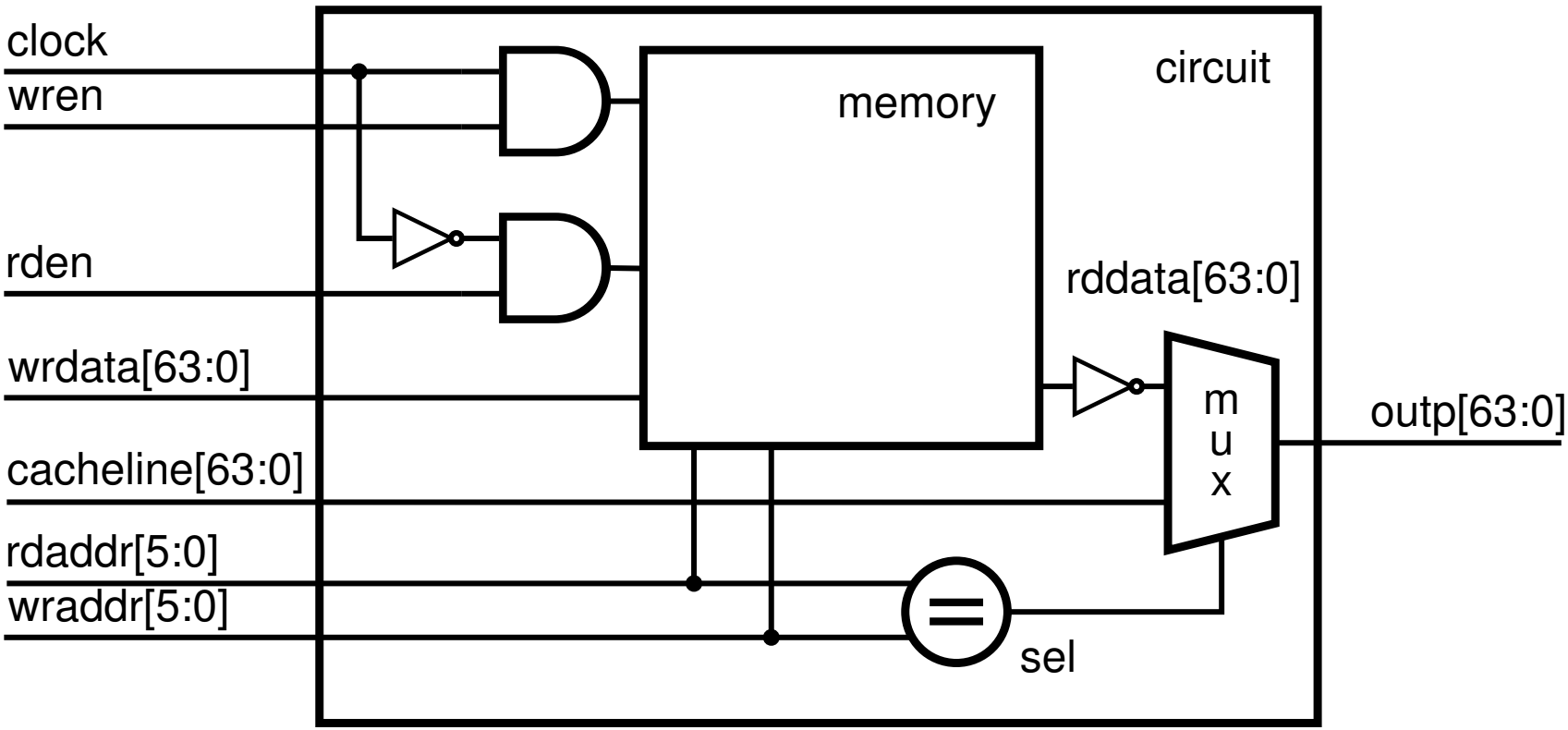
September 21, 2012

Joint work with

Moshe Emmer, Zurab Khasidashvili
Intel Development Center, Haifa

Konstantin Korovin, Andrei Voronkov
The University of Manchester

A Word Level Design



Typical word level components:
bit-vectors, memories, and addresses

Bounded Model Checking (BMC)

- System model with state variables in $\bar{x}$
- Initial state I, transition relation T, and (safety) property P
- Verify safety property by stepwise unrolling the transition

$$I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x}_2) \wedge \cdots \wedge T(\bar{x}_{n-1}, \bar{x}_n) \models P(\bar{x}_n)$$

- Solve one SAT problem on each unrolling
- Counterexample if property is violated

Bottlenecks in SAT-based BMC

- Unrolling creates **copies of the transition relation**.
- Word level components have to be **encoded bit-wise** for SAT.

Our contribution: move to higher level of abstraction.

1. **Avoid unrolling** the transition relation, and
2. **succinctly encode** word level components with into EPR.

Bounded Model Checking (BMC)

- System model with state variables in $\bar{x}$
- Initial state I, transition relation T, and (safety) property P
- Verify safety property by stepwise unrolling the transition

$$I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x}_2) \wedge \cdots \wedge T(\bar{x}_{n-1}, \bar{x}_n) \models P(\bar{x}_n)$$

- Solve one SAT problem on each unrolling
- Counterexample if property is violated

Bottlenecks in SAT-based BMC

- Unrolling creates **copies of the transition relation**.
- Word level components have to be **encoded bit-wise** for SAT.

Our contribution: move to higher level of abstraction.

1. **Avoid unrolling** the transition relation, and
2. **succinctly encode** word level components with into EPR.

Bounded Model Checking (BMC)

- System model with state variables in $\bar{x}$
- Initial state I, transition relation T, and (safety) property P
- Verify safety property by stepwise unrolling the transition

$$I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x}_2) \wedge \cdots \wedge T(\bar{x}_{n-1}, \bar{x}_n) \models P(\bar{x}_n)$$

- Solve one SAT problem on each unrolling
- Counterexample if property is violated

Bottlenecks in SAT-based BMC

- Unrolling creates **copies of the transition relation**.
- Word level components have to be **encoded bit-wise** for SAT.

Our contribution: move to higher level of abstraction.

1. **Avoid unrolling** the transition relation, and
2. **succinctly encode** word level components with into EPR.

Effectively Propositional Reasoning (EPR)

- First-order logic $\exists^*\forall^*$ formulas without function symbols
- No Skolem functions in CNF
- Aka. Bernays-Schönfinkel or Function-free Clause Logic

- Decidable by instantiation-based methods
- Efficient solvers, e.g. **iProver**

- Important for applications
- SAT encoding: many isomorphic or nearly isomorphic subsets
- EPR resolution proofs can be exponentially shorter than propositional resolution proofs.

Effectively Propositional Reasoning (EPR)

- First-order logic $\exists^*\forall^*$ formulas without function symbols
- No Skolem functions in CNF
- Aka. Bernays-Schönfinkel or Function-free Clause Logic

- Decidable by instantiation-based methods
- Efficient solvers, e.g. **iProver**

- Important for applications
- SAT encoding: many isomorphic or nearly isomorphic subsets
- EPR resolution proofs can be exponentially shorter than propositional resolution proofs.

Effectively Propositional Reasoning (EPR)

- First-order logic $\exists^*\forall^*$ formulas without function symbols
- No Skolem functions in CNF
- Aka. Bernays-Schönfinkel or Function-free Clause Logic

- Decidable by instantiation-based methods
- Efficient solvers, e.g. **iProver**

- Important for applications
- SAT encoding: many isomorphic or nearly isomorphic subsets
- EPR resolution proofs can be exponentially shorter than propositional resolution proofs.

Encoding BMC in EPR

The BMC problem

$$I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x}_2) \wedge \cdots \wedge T(\bar{x}_{n-1}, \bar{x}_n) \models P(\bar{x}_n)$$

is dominated by $T(\bar{x}_i, \bar{x}_{i+1})$ terms for large n.

Idea [Navarro-Perez, Voronkov 2007]

- Introduce a **symbolic constant** s_i for each bound,
- instead of a state variable p use a **predicate** $p(s_i)$, and
- **quantify** over predicates $p(x)$ instead of unrolling.

Encoding BMC in EPR

The BMC problem

$$I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x}_2) \wedge \cdots \wedge T(\bar{x}_{n-1}, \bar{x}_n) \models P(\bar{x}_n)$$

is dominated by $T(\bar{x}_i, \bar{x}_{i+1})$ terms for large n.

Idea [Navarro-Perez, Voronkov 2007]

- Introduce a **symbolic constant** s_i for each bound,
- instead of a state variable p use a **predicate** $p(s_i)$, and
- **quantify** over predicates $p(x)$ instead of unrolling.

Translating BMC into EPR

Preliminaries:

- Transition relation $T(\bar{x}, \bar{x}')$ contains **current state symbols** in $\bar{x}$ and **next state symbols** in $\bar{x}'$.
- Initial state constraint $I(x)$ and property $P(x)$ **only** contain **current state symbols**.

Translating BMC into EPR

Let S and S' be fresh variables

$$\mathcal{B}(p(\bar{r})) \stackrel{def}{=} \begin{cases} p_t(S, \mathcal{B}(\bar{r})) & \text{if p is a current state symbol} \\ p_t(S', \mathcal{B}(\bar{r})) & \text{if p is a next state symbol} \\ p(\mathcal{B}(\bar{r})) & \text{otherwise} \end{cases}$$

Let $s_1, \ldots, s_n$ be new constants and next be a new binary predicate, then the n-step unrolling is

$$\mathcal{B}(I)[S \mapsto s_0]$$
$$\forall S \forall S' \, (\text{next}(S, S') \rightarrow \mathcal{B}(T))$$
$$\mathcal{B}(P)[S \mapsto s_n]$$
$$\text{next}(s_0, s_1) \wedge \cdots \wedge \text{next}(s_{n-1}, s_n)$$

Encoding Word Level Components

Relational encoding [Khasidashvili, Kinanah, Voronkov 2009]

Predicate representation

Bit-vector $\texttt{wrdata}(S, B)$: Boolean value of bit B in state S

Memory $\texttt{mem}(S, A, B)$: Boolean value of bit at row A, column B in state S

Functional representation

Bit-vector $\texttt{wraddrFunc}(S)$: bit-vector value in state S

Bit-index bitindex_i: i-th bit in a bit-vector

$$\texttt{mem}(S, \texttt{rdaddrFunc}(S), \text{bitindex}_5)$$

Dimension of bit-vectors and memories is abstract

Encoding Word Level Components

Relational encoding [Khasidashvili, Kinanah, Voronkov 2009]

Predicate representation

Bit-vector $\texttt{wrdata}(S, B)$: Boolean value of bit B in state S

Memory $\texttt{mem}(S, A, B)$: Boolean value of bit at row A, column B in state S

Functional representation

Bit-vector $\texttt{wraddrFunc}(S)$: bit-vector value in state S

Bit-index bitindex_i: i-th bit in a bit-vector

$$\texttt{mem}(S, \texttt{rdaddrFunc}(S), \text{bitindex}_5)$$

Dimension of bit-vectors and memories is abstract

Axiomatizing the Relational Encoding

$$\texttt{wraddrFunc}(S) = \texttt{rdaddrFunc}(S) \leftrightarrow$$
$$\forall B\ (\mathsf{less}_6(B) \rightarrow (\texttt{wraddr}(S, B) \leftrightarrow \texttt{rdaddr}(S, B)))$$

$$\mathsf{less}_k(x) \leftrightarrow (x = \mathsf{bitindex}_0 \vee \cdots \vee x = \mathsf{bitindex}_{k-1})$$

$$\begin{array}{rl} \mathsf{less}_k(\mathsf{bitindex}_j) & \text{if } j < k \\ \neg\, \mathsf{less}_k(\mathsf{bitindex}_j) & \text{otherwise} \end{array}$$

$$\texttt{wraddrFunc}(S) = \texttt{rdaddrFunc}(S) \leftrightarrow$$
$$\forall B\ (\mathsf{range}_{[0,6]}(B) \rightarrow (\texttt{wraddr}(S, B) \leftrightarrow \texttt{rdaddr}(S, B)))$$

$$\mathsf{range}_{[m,k]}(x) \leftrightarrow (x = \mathsf{bitindex}_m \vee \cdots \vee x = \mathsf{bitindex}_k)$$

Axiomatizing the Relational Encoding

$$\texttt{wraddrFunc}(S) = \texttt{rdaddrFunc}(S) \leftrightarrow$$
$$\forall B\ (\mathsf{less}_6(B) \rightarrow (\texttt{wraddr}(S, B) \leftrightarrow \texttt{rdaddr}(S, B)))$$

$$\mathsf{less}_k(x) \leftrightarrow (x = \mathsf{bitindex}_0 \vee \cdots \vee x = \mathsf{bitindex}_{k-1})$$

$$\begin{array}{rl} \mathsf{less}_k(\mathsf{bitindex}_j) & \text{if } j < k \\ \neg\, \mathsf{less}_k(\mathsf{bitindex}_j) & \text{otherwise} \end{array}$$

$$\texttt{wraddrFunc}(S) = \texttt{rdaddrFunc}(S) \leftrightarrow$$
$$\forall B\ (\mathsf{range}_{[0,6]}(B) \rightarrow (\texttt{wraddr}(S, B) \leftrightarrow \texttt{rdaddr}(S, B)))$$

$$\mathsf{range}_{[m,k]}(x) \leftrightarrow (x = \mathsf{bitindex}_m \vee \cdots \vee x = \mathsf{bitindex}_k)$$

Back to EPR: Address Unrolling (1)

(akin to EPR-based finite model finding [Baumgartner, Fuchs, de Nivelle, Tinelli 2007])

Add clauses

$$\mathrm{assoc}_{\mathtt{rdaddr}}(s_0, \mathtt{rdaddr}_0) \wedge \cdots \wedge \mathrm{assoc}_{\mathtt{rdaddr}}(s_n, \mathtt{rdaddr}_n)$$

and turn

$$\Phi[\mathtt{rdaddrFunc}(x)]$$

into

$$\forall y\, \mathrm{assoc}_{\mathtt{rdaddr}}(x, y) \rightarrow \Phi[y].$$

Back to EPR: Address Unrolling (2)

$\mathrm{val}(b, i)$ represents the value of bit-vector b at index i

$\mathrm{addr}_k(x)$ is true iff x is a bit-vector of length k

Equality between bit-vectors:

$$\forall x \forall y \, (\mathrm{addr}_k(x) \wedge \mathrm{addr}_k(y) \rightarrow (x = y \leftrightarrow \forall B \, (\mathrm{range}_{[0,k-1]}(B) \rightarrow (\mathrm{val}(x, B) \leftrightarrow \mathrm{val}(y, B)))))$$

Not yet in EPR: $\leftrightarrow$ results in $\forall^2\exists$ prefix
Again: turn Skolem function into Skolem predicate

Implementation and Incrementality

- **iProver** approximates a ground model and delegates propositional solving to **MiniSat**.
- Incremental solving with **activation literals** b_k

$$\text{reachable}(s_0) \wedge \cdots \wedge \text{reachable}(s_k)$$
$$b_k \rightarrow \forall x \, (\text{reachable}(x) \rightarrow x = s_0 \vee \cdots \vee x = s_k)$$

- Unsatisfiable cores to lift information about relevant clauses from one bound to the next

Evaluation: EPR-Based BMC vs. SAT-Based BMC

Intel's SAT-based BMC tool vs. iProver incremental mode, Intel benchmarks

Problem	Memories #	Memories bits	Trans. #	BVs bits	Const. #	Const. bits	BMC max.	BMC1 bound
PMS1	8	46080	1486	6109	3	47	2	**10**
SCD1	2	16384	556	1923	5	45	4	**12**
SCD2	2	16384	80	756	3	10	4	**14**
BPB2	4	10240	550	4955	6	42	**50**	11
DCI1	32	9216	3625	6496	3	9	**6**	4
DCC2	4	8960	426	1844	2	2	8	**11**
DCC1	4	8960	1827	5294	5	106	7	**8**
ROB2	2	4704	255	3479	26	129	**50**	8

Evaluation: Encoding Bit-Ranges

SMT solver Z3 vs. iProver, not incremental

Problem (bound)	Z3 less_k	Z3 $\text{range}_{[m,k]}$	Z3 arith.	iProver less_k	iProver $\text{range}_{[m,k]}$
BPB (2)	—	—	—	42s	**41s**
BPB (4)	—	—	—	**634s**	669s
DCC (2)	78s	56s	**29s**	55s	79s
DCC (4)	1204s	636s	**157s**	266s	238s
DCC (6)	8540s	3396s	3512s	—	**1407s**
PMS (2)	44s	1266s	**9s**	161s	163s
PMS (4)	638s	**149s**	188s	1295s	1298s
PMS (6)	2898s	5730s	**4564s**	—	—
PMS (8)	12303s	**3062s**	—	—	—
ROB (2)	—	—	—	**250s**	282s
SCD (2)	167s	119s	178s	**15s**	**15s**
SCD (4)	434s	316s	346s	**276s**	277s
SCD (6)	886s	548s	699s	**635s**	**635s**
SCD (8)	2037s	**1017s**	1497s	—	—

Future Work

[Emmer, Khasidashvili, Korovin, Sticksel, Voronkov 2012]

- Build in arithmetic reasoning into EPR solving
- Boost incremental solving by propagating information from lower bounds
- Abstraction refinement to add bit-vector information lazily