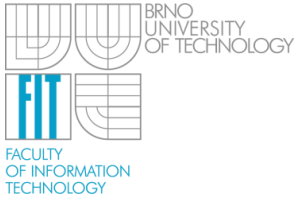
Verifikace číslicových obvodů: využití metod a nástrojů formální verifikace

Marcela Šimková, Michal Kajan

Fakulta informačních technologií
Vysoké učení technické v Brně



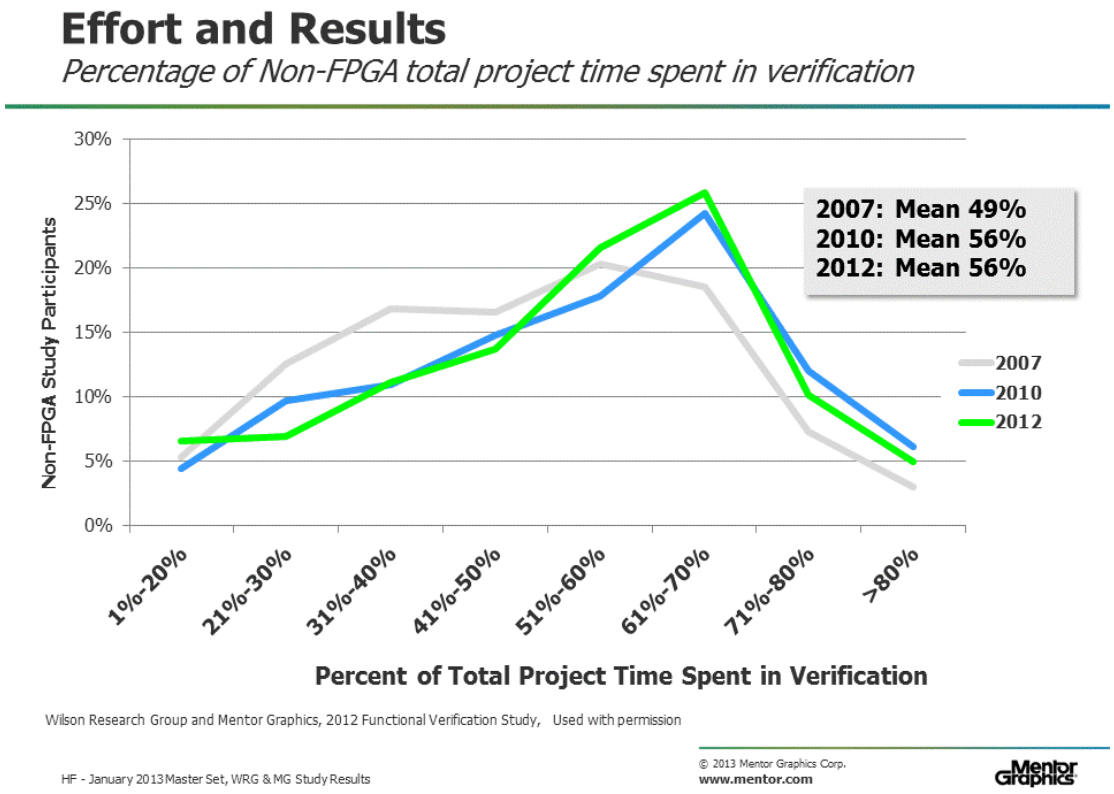
Tento materiál vznikl za podpory Fondu rozvoje vysokých škol (projekt FR1086/2013) a statutárního města Brna (program Brno Ph.D. talent).

Osnova

- Motivace
- Základní pojmy
- Jazyky pro definici formálních tvrzení
- Verifikace založená na formálních tvrzeních (ABV – *Assertion Based Verification*)
- Nástroje ABV
- Užitečné zdroje a odkazy

Motivace

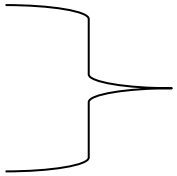
Se složitostí obvodu roste až exponenciálně složitost verifikace !!!
→ více logiky na čipu, složitější funkce a chování systému, časové závislosti, atd.



Motivace

Přístupy pro verifikaci číslicových obvodů:

- RTL simulace (testbench)
- funkční verifikace



nedokazují korektnost obvodu !

- formální verifikace

Výhodná strategie: KOMBINACE těchto přístupů.

Základní pojmy – formální verifikace

Využívá matematických metod k formálnímu popisu systému nebo jeho vlastností a ověřuje, zda funkčnost systému je v souladu se specifikací.

Výsledkem je:

Důkaz správnosti (angl. *proof*) – neexistuje žádný vstup, který by způsobil porušení sledované podmínky.

nebo

Protipříklad (angl. *fire, counter-example*) – určitý běh systému, nebo vstup, který vede k porušení sledované podmínky.

Základní pojmy – formální verifikace

Nejznámější techniky formální verifikace:

- **Model Checking** – ověřuje vlastnosti systému (*properties*) úplným prozkoumáním jeho stavového prostoru.
- **Statická analýza** – automatická analýza zdrojového kódu, nevyžaduje model systému, používá se i pro optimalizaci a generování kódu.
- **Theorem Proving** – deduktivní metoda, podobná matematickému dokazování.
- **Equivalence Checking** – formálně dokazuje, že dvě reprezentace obvodu mají stejné chování (např. na různých úrovních abstrakce).
- **SAT Solving** – SAT (*satisfiability*) representuje NP-úplný problém splnitelnosti Booleovských formulí.
- **Verifikace založená na formálních tvrzeních (ABV)**

Základní pojmy – ABV

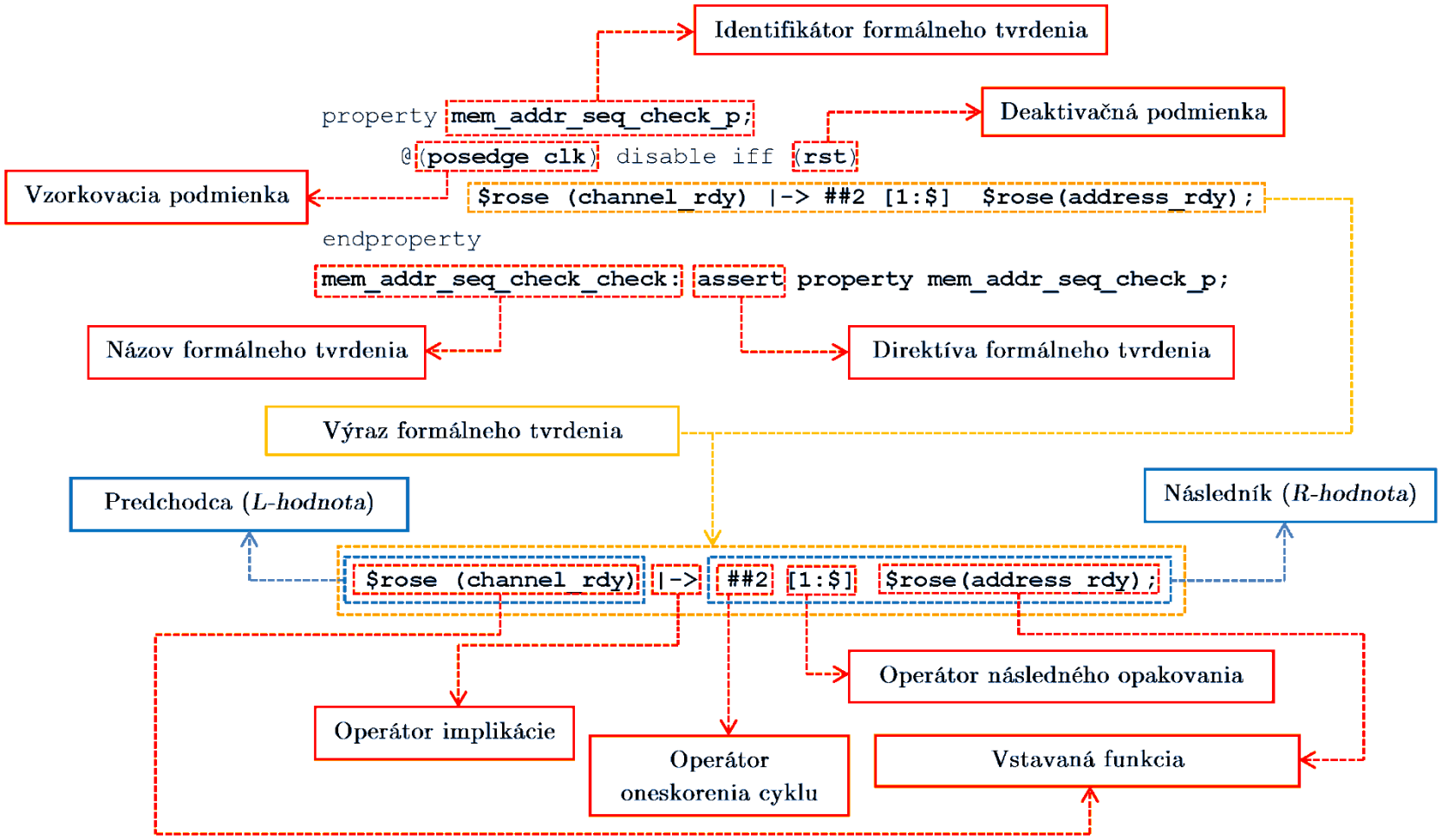
Verifikace založená na formálních tvrzeních (ABV, *Assertion-Based Verification*)

Vlastnost (*property*) – vlastnost systému.

Formální tvrzení (*assertion*) – tvrzení (výraz v temporální logice) pokrývající určitou vlastnost systému, které musí v daném systému vždy platit.

- Zachycení časových vlastností obvodu.
- Vyjádření očekávaných operací, vnitřní synchronizace, postupnosti instrukcí, atd.
- Kontrola dodržení protokolů vstupních a výstupních rozhraní, křížení hodinových domén, stavových automatů.
- Při porušení formálního tvrzení je verifikace přerušena a je reportována chyba → rychlá lokalizace zdroje problému.

Základní pojmy – formální tvrzení



Jazyky pro tvorbu formálních tvrzení

- **PSL** – *Properties Specification Langua*ge:
  - původně jazyk Sugar od IBM,
  - dnes standard IEEE 1850-2005,
  - použitelný pro různé samostatné HDL jazyky.

- **SVA** – *SystemVerilog Assertions:*
  - součást jazyka SystemVerilog,
  - základem se stal jazyk PSL a OpenVera Assertions,
  - dnes standard IEEE 1800-2005.

- **Knihovny** předpřipravených formálních tvrzení:
  - OVL – *Open Verification Library* (30 kategorí),
  - *SystemVerilog Assertion Library* (50 kategorí),
  - *CheckerWare Library* (70 kategorí).

- Podpora za strany dostupných nástrojů pro vývoj číslicových systémů.

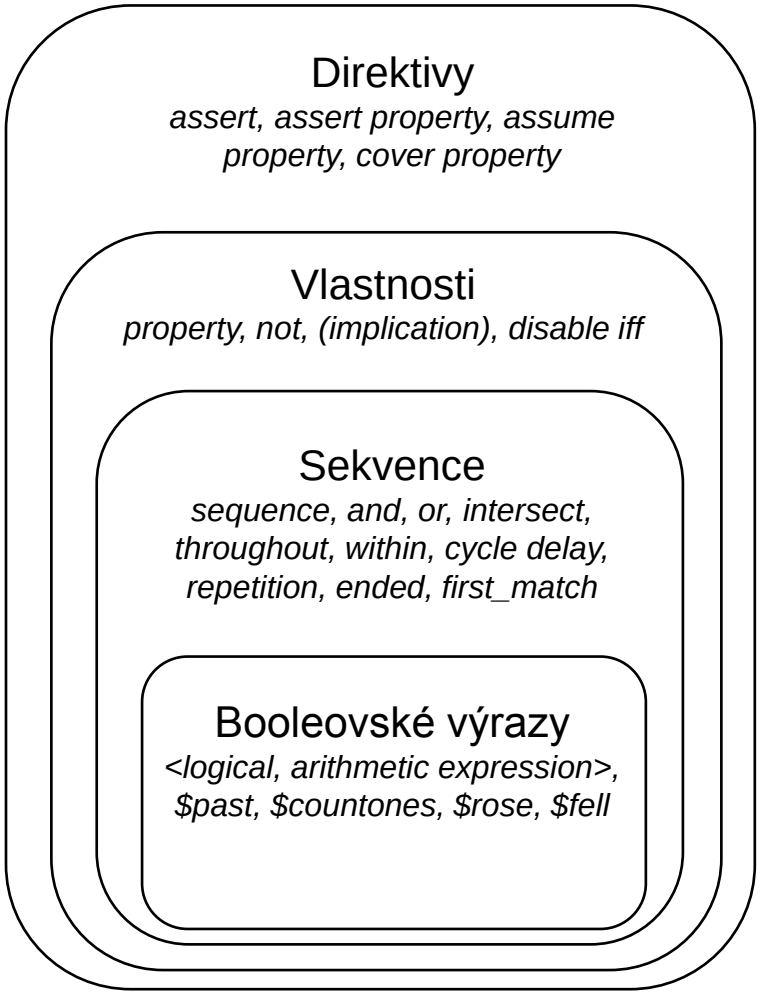
SysemVerilog Assertions

Podpora dvou typů formálních tvrzení:

1. *Immediate:*
   - založeny na událostech (*event-driven*),
   - procedurální charakter,
   - slouží jen k vyhodnocování Booleovských výrazů (**ne** temporálních vlastností),
   - vyhodnocují se **okamžitě** (ne s hranou hodin).

1. *Concurrent:*
   - umožňují specifikaci chování v simulačním čase založené na sekvenci událostí,
   - tento typ je vytvářen 4 konstrukty:
     - Booleovské výrazy,
     - sekvence (*sequences*),
     - vlastnosti (*properties*),
     - direktivy (*assertions directives*).

Hierarchie SVA konstruktů



Booleovské výrazy

- Specifikují události v jednom konkrétním taktu hodin.
- Hodnoty proměnných v Booleovském výrazu se vyhodnocují s nástupnou anebo sestupnou hranou hodin.
- Výsledek True/False.
- Příklady:

```
// reset není aktivní
!reset

// neaktivní reset, aktivní povolovací zápisový signál a plná fronta
!reset && we && fifo_full

// hodnota data_in je rovna 3E hexadecimálně
data_in[7:0] == 8'h3E
```

Sekvence

- Definují vztahy mezi Booleovskými výrazy v simulačním čase.
- Jednotlivé Booleovské výrazy jsou odděleny jednou nebo více hranami hodin → vyhodnocení výrazů v specifických taktech hodin.
- Sekvence je možné skládat.
- Příklady:

```
// data adresy jsou následována potvrzením jejich platnosti za 3 takty

sequence AddrAck;
  addr ##3 ack;
endsequence

// potvrzení může přijít v rozmezí 2 až 4 taktů

sequence AddrAck2;
  addr ##[2:4] ack;
endsequence
```

Vlastnosti

- Umožňují lepší práci s Booleovskými výrazy a sekvencemi: povolování, rušení sekvencí, přidávání podmínek pro sekvence a vlastnosti (pomocí implikace).
- Příklad:

```
// sekvence Sack je platná až jeden takt po sekvenci Sframe

property Ppci_burst
   @(posedge clock) Sframe |=> Sack;
endproperty

// Když je žádost validní (req_valid = true) tak musí platit, že v tom stejném taktu hodin
// má req_id validní rozsah (<10) a potvrzení žádosti (ack_valid) není aktivní. Potvrzení
// žádosti musí přijít až v následujících 1-10 taktech.

property req_imp_ack;
 integer id;
 @(posedge CLK)
 (req_valid, id=req_id) |->
   ((req_id <= 9) && !ack_valid)
      ##[1:10] (ack_valid && (ack_id == id));
endproperty
```

```
property req_ack;
  integer id;
  @(posedge CLK)
  if (req_valid)
    ((req_id <= 9) && !ack_valid, id=req_id)
      ##[1:10] (ack_valid && (ack_id == id));
endproperty
```

Direktivy

- Určují, jak budou vlastnosti, sekvence, a Booleovské výrazy interpretovány v kontextu daného systému.
- **Formální tvrzení** (*assertions*) jsou tvořeny vlastnostmi a direktivami.
- Typy direktiv pro tvorbu formálního tvrzení:

1. ***Assert*** *property* – určuje vlastnost, která by nikdy neměla být v systému porušena.
2. ***Cover*** *property* – definuje požadavek na sledování výskytu dané vlastnosti nebo sekvence pro účely statistického vyhodnocení.
3. ***Assume*** *property* - používá se pro definování omezujících podmínek pro vstupní hodnoty verifikované jednotky.

Direktivy

- Příklady:

```
property Ppci_burst
   @(posedge clock) Sframe |=> Sack;
endproperty

// příklad assert property
assert property (Ppci_burst)
   else $error("Sequence Sack is not valid after sequence Sframe!");

// příklad cover property
cover property (Ppci_burst);
```

Jazyky pro tvorbu formálních tvrzení

- Formální tvrzení v jazyce SVA:

```
property dma buffer overflow;
  @(posedge dma clk)
  not (fifo full && fifo write);
endproperty
assert property (dma buffer overflow);
```

- Fromální tvrzení v jazyce PSL:

```
property dma buffer overflow =
  never (fifo full and fifo write) @ (posedge dma clk);
assert dma buffer overflow;
```

- Formální tvrzení z knihovny OVL:

```
dma buffer overflow assert fifo index
  #(4, 16) // severity level, depth
  (dma clk, dma rst, push, pop);
```

- Formální tvrzení z knihovny CheckerWare:

```
/ 0in fifo -enq push -deq pop depth 16 -severity 4
-name dma buffer overflow /
```

ABV- aplikace

Důraz na **znovu-použitelnost**:

1. Automatická kontrola formálních tvrzení
2. Statická formální verifikace
3. Funkční verifikace, simulace s využitím formálních tvrzení
4. Dynamická formální verifikace

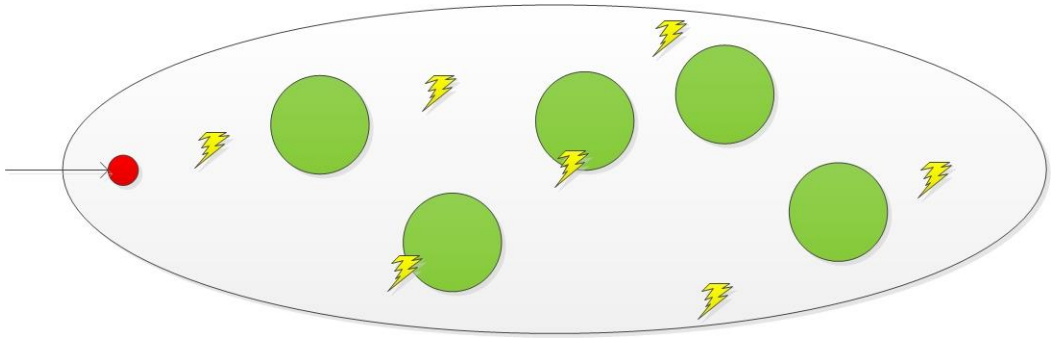
Návrh

Výroba

5. Ladění na čipu, online monitoring

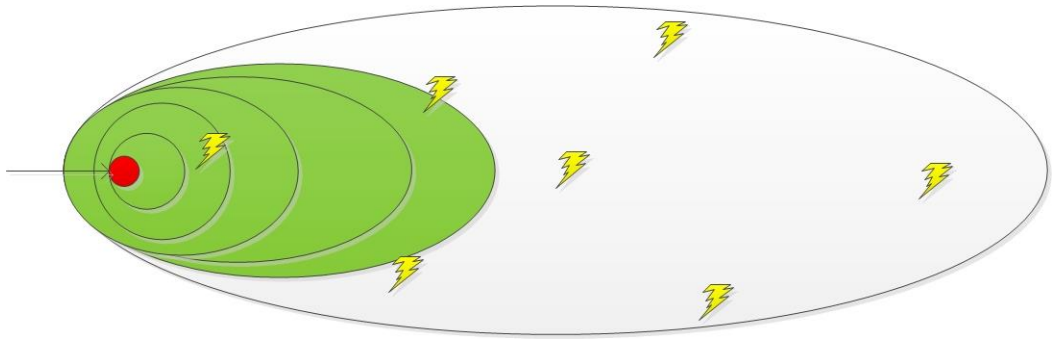
Automatická kontrola formálních tvrzení

- Předpřipravené formální tvrzení v podobě knihoven (i vlastní).
- Odhalení běžně se vyskytujících chyb v návrhu obvodu:
  - aritmetické chyby,
  - chyby v konečných automatech,
  - připojení registrů.



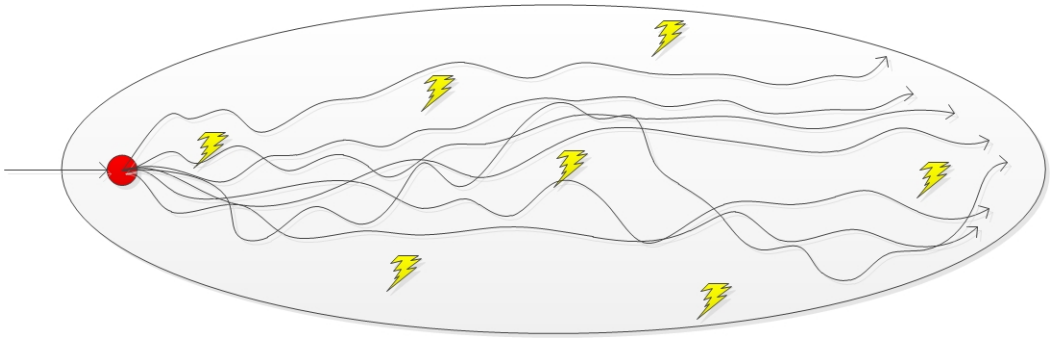
Statická formální verifikace

- Typicky *model-checking* anebo *bounded model-checking*.
- Dokazuje se, že vlastnosti obvodu, které jsou definovány pomocí formálních tvrzení, platí (**důkaz**) anebo neplatí (**protipříklad**).
- Ověřují se klíčové vlastnosti obvodu – formální tvrzení vytvořené dle specifikace.



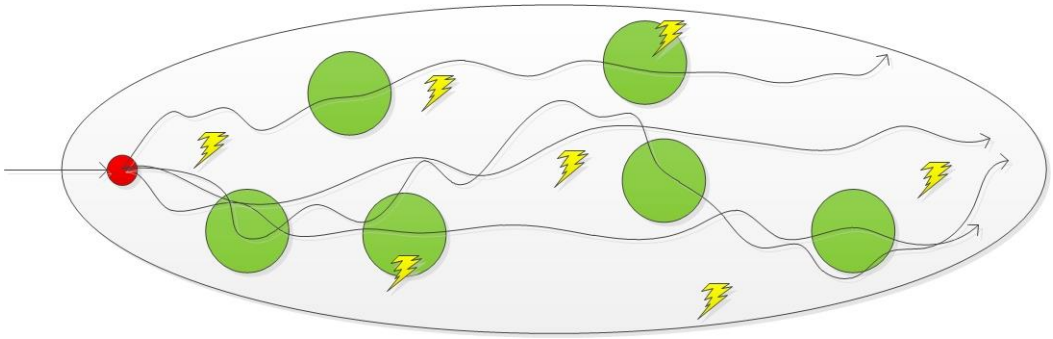
Funkční verifikace

- Splnění formálních tvrzení (ze statické verifikace anebo z knihoven) je možné sledovat v simulaci.
- V případě porušení sledované podmínky se ihned reportuje chyba.
- Informace o pokrytí sledovaných podmínek testovacími vektory.



Dynamická formální verifikace

- Kombinace funkční verifikace a statické formální verifikace.
- Lokální formální analýza v okolí zajímavých stavů → typicky místa, kde došlo k porušení formálního tvrzení.



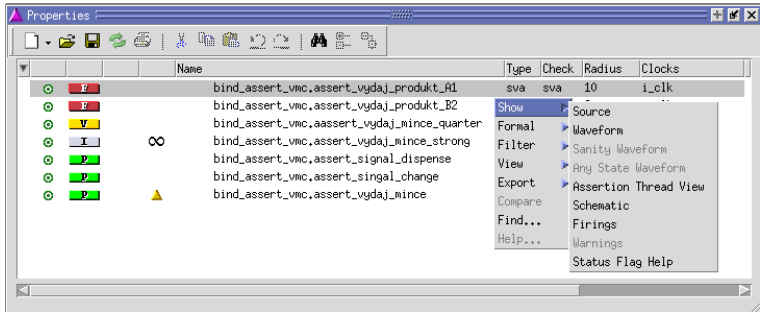
Monitorovací obvody

- Vznikají syntézou formálních tvrzení do podoby samostatných hardwarových jednotek.
- Umožňují monitorovat chování obvodu i **po výrobě**:
  - ladění prototypů,
  - online monitoring výsledného produktu.

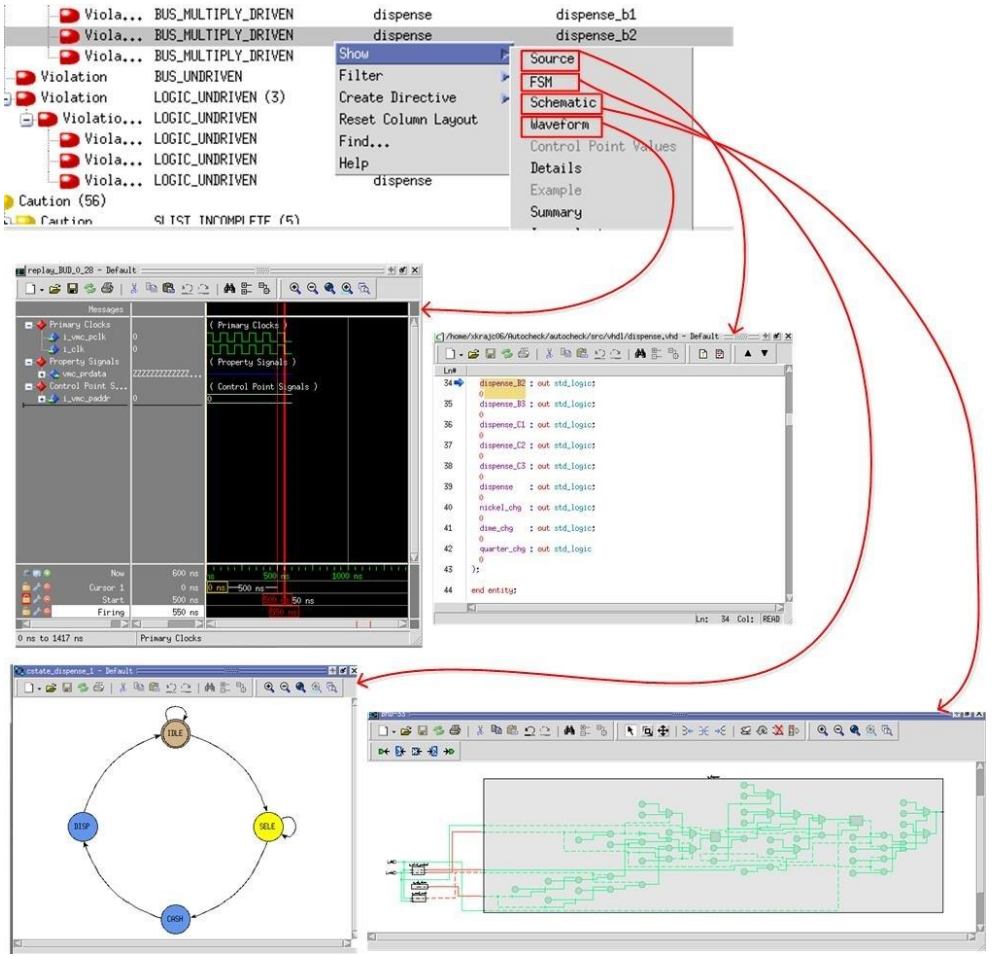
Nástroje ABV

Sada nástrojů Questa od společnosti Mentor Graphics:

- **Questa Static**: automatická kontrola formálních tvrzení
- **Questa Formal**: statická formální verifikace



Úroveň závažnosti	Popis
porušenie	Závažné porušenie formálneho tvrdenia.
varovanie	Varovanie pri porušení formálneho tvrdenia.
informácia	Informácia o porušení formálneho tvrdenia.



Užitečné zdroje a odkazy

- Foster, H., Krolnik, A.: Creating Assertion-Based IP, Springer, 2007, ISBN 978-0387366418
- Haque, F., Michelson, J., Khan, K. A.:The Art of Verification with SystemVerilog Assertions, Verification Central, USA, 2006, ISBN: 978-0-9711994-1-5
- Boulé, M., Zilic, Z.: Generating Hardware Assertion Checkers, Springer, 2008, ISBN: 978-1-4020-8585-7
- Eisner, C., Fisman, D.: A Practical Introduction to PSL, Springer, 2006, ISBN 978-038735313
- Cerny, E., Dudani, S.: The Power of Assertions in SystemVerilog, Springer, 2010, ISBN 978-1441965998
- IEEE Computer Society. IEEE Std 1800-2009: *IEEE Standard for SystemVerilog - Unified Hardware Design, Specification, and Verification Language*, 2009, ISBN: 978-0-7381-6129-7.
- Spear, Chris: *SystemVerilog for Verification*, 2nd Edition, Springer, New York, 2008, ISBN 978-0-387-76529-7.
- Verification Academy [online]. <http://verification-academy.mentor.com/>