Note written by 尤存翰 2022/12/15

A Hardware Design Language for Timing-Sensitive Information-Flow Security

Danfeng Zhang Yao Wang G. Edward Suh Andrew C. Myers Cornell University

ACM SIGPLAN NoticesVolume 50Issue 4 April 2015 pp 503–516 https://doi.org/10.1145/2775054.2694372

信息安全可能会因底层硬件实现的泄漏而受到损害。最近一个突出的例子是 cache probing attacks，它依赖于缓存创建的timing通道。本文介绍了一种硬件设计语言SecVerilog，它可以在硬件级别静态分析信息流。有了SecVerilog，可以通过对定时信道和其他信息信道的可验证控制来构建系统。SecVerilog是基于Verilog，扩展了表达型注释，可以对信息流进行精确推理的语言。

它还提供了严格的形式保证（证明SecVerilog强制执行对时间的不敏感，从而确保了信息流的安全。）

本文通过构建一个安全的MIPS处理器及其缓存，证明SecVerilog使构建复杂的硬件设计成为可能。

本文主要贡献

- 设计了SecVerilog，一种新的硬件描述语言，它通过对硬件内信息流的跟踪来扩展Verilog。
- 包含相关安全类型的表达式静态注释，实现flexible, fine-grained的重用和跨安全级别的硬件共享，
- HDL型系统能soundly控制信息流的正式证明（？文章里的证明很片面，看不明白，打算等学完了Program Verification再来看看）
- 使用SecVerilog设计了一个安全的微处理器，证明了该方法的power。delay, area, power, performance and designer effort都很低。

SecChisel框架

The SecVerilog approach

参考下图的cache实现，灰字部分是从Verilog到SecVerilog的label扩展，这点和SecChisel很像，在设计硬件安全框架时也许是个公认的好方法（？）

```
reg[18:0]{L} tag0[256],tag1[256];
reg[18:0]{H} tag2[256],tag3[256];
wire[7:0]{L} index;
//Par(0)=Par(1)=L  Par(2)=Par(3)=H
wire[1:0]{Par(way)} way;
wire[18:0]{Par(way)} tag_in;
wire{Par(way)} write_enable;

always @(posedge clock) begin
  if (write_enable) begin
    case (way)
    0: begin tag0[index]=tag_in; end
    1: begin tag1[index]=tag_in; end
    2: begin tag2[index]=tag_in; end
    3: begin tag3[index]=tag_in; end
    endcase
  end
end
```

(a) SecVerilog code for cache tags

```
wire{L} isLoad,isStore;
wire{L} hit0,hit1; // hitX: 1 iff way X gets a cache hit
wire{H} hit2,hit3;
//LH(0)=L  LH(1)=H
wire{LH(timingLabel)} stall, hit, timingLabel;
reg[2:0]{LH(timingLabel)} dFsmState;

assign stall = ((isLoad | isStore) &
  (~hit | (dFsmState != DFSM_IDLE)));
assign hit = (timingLabel == 0) ?
  ((hit0|hit1)?1:0) : ((hit0|hit1|hit2|hit3)?1:0);
...
case (dFsmState)
  DFSM_IDLE: begin
    // load hit
    if (isLoad && hit) begin
      dFsmState <= DFSM_IDLE; // nonblocking assignment
    ...
endcase
```

(b) SecVerilog code for a cache controller

Figure 2. SecVerilog extends Verilog with security label annotations (shaded in gray).

这样设计有什么好处?

首先，验证是在编译时完成的，避免了运行时的开销，并在早期设计阶段检测到错误。这在GLIFT[1]和Sapper[2]中是不可能的。其次，变量和逻辑可以在多个安全级别中共享（例如，方式和命中与各种时序标签共享），这在Caisson [3]中是不可能的。此外，SecVerilog增加的编程工作量很小。Verilog代码几乎可以按原样进行验证，仅在变量声明中需要注释（安全标签）。

语法和语义

除了添加的注释外，SecVerilog基本上具有与Verilog相同的语法和语义。

$$
\begin{array}{llll}
\text{Program} & \texttt{Prog} & ::= & B_1 \ldots B_n \\
\text{Thread} & B & ::= & \texttt{always}\ @(\gamma)\ c \\
\text{Trigger} & \gamma & ::= & \texttt{posedge clock} \mid \texttt{negedge clock} \mid \vec{v} \\
\text{Cmds} & c & ::= & \texttt{skip}_\eta \mid \texttt{begin}\ c_1; \ldots; c_n;\ \texttt{end} \\
 & & & \mid v =_\eta e \mid v \Leftarrow_\eta e \mid \texttt{if}_\eta\ (e)\ c_1\ \texttt{else}\ c_2 \\
\text{Expr} & e & ::= & v \mid n \mid \textbf{uop}\ e \mid e\ \textbf{bop}\ e \\
\text{Vars} & x, y, v & \in & \textbf{Vars}
\end{array}
$$

Figure 3. Syntax of SecVerilog.

$$
\begin{array}{ll}
\text{Level} & \ell \in \mathcal{L} \\
\text{Family} & f \in \mathbb{Z}_n \to \mathcal{L} \\
\text{Label} & \tau ::= \ell \mid f(v) \mid \tau_1 \sqcup \tau_2 \mid \tau_1 \sqcap \tau_2
\end{array}
$$

Figure 4. Syntax of security labels.

SecVerilog的类型系统

SecVerilog中的类型只是用安全标签表达式扩展的Verilog类型，见下图。

$$\frac{}{\Gamma, pc, \mathcal{M} \vdash \texttt{skip}_\eta}\ \textsc{T-Skip} \qquad \frac{\Gamma, pc, \mathcal{M} \vdash c_1 \quad \Gamma, pc, \mathcal{M} \vdash c_2}{\Gamma, pc, \mathcal{M} \vdash c_1; c_2}\ \textsc{T-Seq} \qquad \frac{\Gamma \vdash e : \tau \quad v \notin \texttt{FV}(\Gamma(v)) \quad \models P(\bullet\eta) \Rightarrow \tau \sqcup pc \sqsubseteq \Gamma(v)}{\Gamma, pc, \mathcal{M} \vdash v =_\eta e \quad \Gamma, pc, \mathcal{M} \vdash v \Leftarrow_\eta e}\ \textsc{T-Assign}$$

$$\frac{\Gamma \vdash e : \tau \quad \begin{array}{c} v \in \texttt{FV}(\Gamma(v)) \\ v' \notin \Gamma \end{array} \quad \begin{array}{r} \models P(\bullet\eta) \Rightarrow pc \sqsubseteq \Gamma(v) \ ^{\text{if } v \notin \mathcal{M}} \\ \models P(\bullet\eta), v' = \lfloor e \rfloor_a \Rightarrow \tau \sqcup pc \sqsubseteq \Gamma(v)\,\{v'/v\} \end{array}}{\Gamma, pc, \mathcal{M} \vdash v =_\eta e \qquad \Gamma, pc, \mathcal{M} \vdash v \Leftarrow_\eta e}\ \textsc{T-Assign-Rec} \qquad \frac{\Gamma \vdash e : \tau \quad \begin{array}{c} \Gamma, pc \sqcup \tau, \mathcal{M} \cap \texttt{DA}(\eta) \vdash c_1 \\ \Gamma, pc \sqcup \tau, \mathcal{M} \cap \texttt{DA}(\eta) \vdash c_2 \end{array}}{\Gamma, pc, \mathcal{M} \vdash \texttt{if}_\eta\ (e)\ c_1\ \texttt{else}\ c_2}\ \textsc{T-If}$$

Figure 5. Typing rules: commands.

（原文摘抄）SecVerilog类型系统以严格和可验证的方式静态控制信息流。该类型系统最新颖的特点包括：

- 可变的、依赖的安全标签（类似于SecChisel）
- 控制标签通道的permissive但是sound的方式
- 模块化设计，将精度所需的程序分析与类型系统分开控制

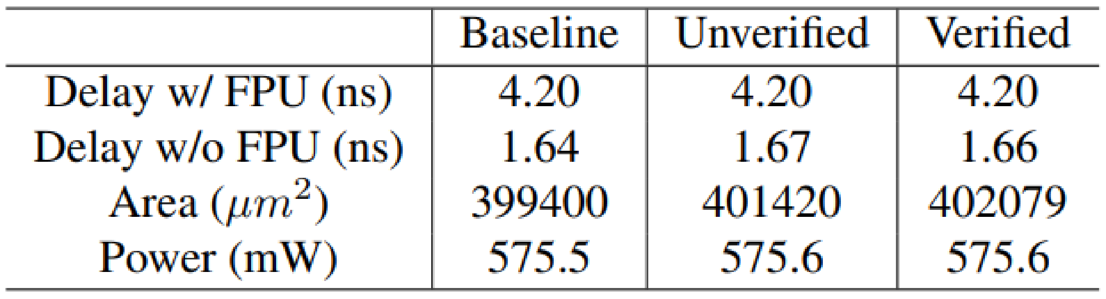
与以前大多数基于语言的安全性工作不同，SecVerilog支持动态标签，也也就是可以在运行时更改的标签。使用应用于变量v的类型值函数f来构造动态标签$f(v)$。

Soundness证明

（待填写）

实验结果

下图是对验证过的，没验证过的做区分的evaluation，包含Delay, area and power.



	Baseline	Unverified	Verified
Delay w/ FPU (ns)	4.20	4.20	4.20
Delay w/o FPU (ns)	1.64	1.67	1.66
Area (μm^2)	399400	401420	402079
Power (mW)	575.5	575.6	575.6

Table 3. Comparing processor designs.

下图是OpenSSL的例子，其中评估了两种安全策略：“nomix”，整个程序标记为H，对应于以前安全硬件设计方法所针对的安全策略；“mixed”，允许混合H和L指令，是SecVerilog的新特性。

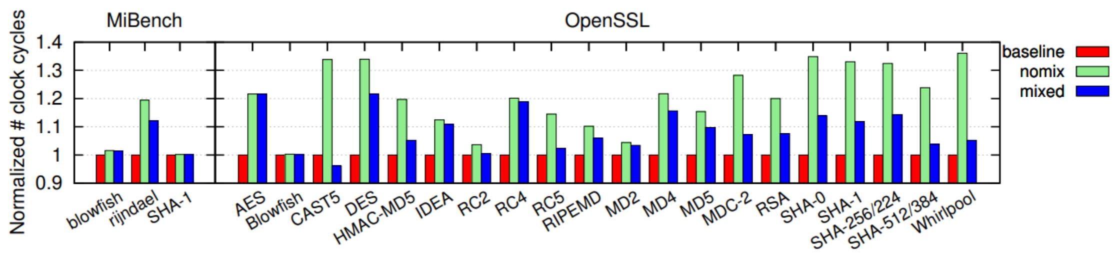


Figure 9. Performance overhead of timing channel protection.

Ref

[1] M. Tiwari, H. M. Wassel, B. Mazloom, S. Mysore, F. T. Chong, and T. Sherwood. Complete information flow tracking from the gates up. In ASPLOS XIV, pages 109–120, 2009.

[2] X. Li, V. Kashyap, J. K. Oberg, M. Tiwari, V. R. Rajarathinam, R. Kastner, T. Sherwood, B. Hardekopf, and F. T. Chong. Sapper: A language for hardware-level security policy enforcement. In Proc. 19th Int'l Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pages 97–112, 2014.

[3] X. Li, M. Tiwari, J. Oberg, V. Kashyap, F. Chong, T. Sherwood, and B. Hardekopf. Caisson: a hardware description language for secure information flow. In ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI), pages 109–120, 2011.

Caisson: a hardware description language for secure information flow.（目前正在看，预计12.21左右完成）

ACM SIGPLAN NoticesVolume 46Issue 6 June 2011 pp 109–120 https://doi.org/10.1145/1993316.1993512

Xun Li, Mohit Tiwari, Jason K. Oberg, Vineeth Kashyap, Frederic T. Chong, Timothy Sherwood, Ben Hardekopf

Department of Computer Science University of California, Santa Barbara, CA

信息流是一个重要的安全属性，必须从头开始纳入，包括在硬件设计时，为系统的信任根基提供一个正式的基础。本文结合了设计信息流安全编程语言的见解和技术，为设计安全硬件提供了一个新的视角。作者描述了一种新的硬件描述语言Caisson，它将硬件设计中常见的特定领域的抽象与安全编程语言中使用的基于类型的技术的见解相结合。这些元素的适当组合允许一个有表达力的、可证明是安全的HDL，在该语言的目标受众--硬件设计师--熟悉的抽象水平上运行。

作者为Caisson实现了一个编译器，将设计翻译成Verilog，然后使用现有的工具对设计进行综合。作为Caisson有用性的一个例子，作者已经解决了安全硬件中的一个开放性问题，创建了有史以来第一个具有微架构特征（包括流水线和高速缓存）的可证明的信息流安全处理器。本文合成了安全处理器，并在芯片面积、功耗和时钟频率方面与标准（不安全）的商业处理器和在门级增强的处理器进行了经验比较，以动态跟踪信息流。本文合成的处理器与不安全的处理器相比是有竞争力的，而且明显优于动态跟踪。

本文主要贡献

- 设计了Caisson，一种针对可静态验证的信息流安全硬件设计的硬件描述语言。
- 正式证明了Caisson可以执行对时间敏感的非干涉。
- 设计并实现了一个可验证的信息流安全处理器，它具有复杂的微结构特征，包括流水线和缓存。
- 将Caisson与不安全的商业CPU设计以及动态跟踪信息流的GLIFT CPU设计进行了经验性比较。在芯片面积（1.35倍对3.34倍）、时钟频率（1.46倍对2.63倍）和功率（1.09倍对2.82倍）方面，Caisson比GLIFT的基线处理器引入的开销少得多。

语法和语义

$$r \in \mathit{Register} \qquad v \in \mathit{Variable} \qquad x \in \mathit{Register} \cup \mathit{Variable}$$

$$n \in \mathbb{Z} \qquad \oplus \in \mathit{Operator} \qquad l \in \mathit{Program\ Label}$$

$$\begin{aligned}
\mathit{prog} \in \mathit{Prog} &::= \textsf{prog}\ l = \vec{r_\ell}\ \textsf{in}\ d \\
d \in \mathit{Def} &::= \textsf{let}\ \vec{s}\ \textsf{in}\ c \ \mid\ c \\
s \in \mathit{State} &::= \textsf{state}\ l_\tau\ (\vec{v_\alpha})\, \kappa = d \\
e \in \mathit{Exp} &::= n \ \mid\ x \ \mid\ e \oplus e \\
c \in \mathit{Cmd} &::= \textsf{skip} \ \mid\ x := e \ \mid\ c\ \textsf{;}\ c \ \mid\ \textsf{fall}_l \\
&\quad\ \mid\ \textsf{goto}\ l(\vec{x}) \ \mid\ \textsf{if}\ e\ \textsf{then}\ c\ \textsf{else}\ c \\
p \in \mathit{Phrase} &::= \mathit{prog} \ \mid\ d \ \mid\ s \ \mid\ e \ \mid\ c
\end{aligned}$$

Figure 3. Abstract syntax. Type annotations ℓ, α, τ, and κ are described in Figure 5

$$\gamma \in \mathit{Env} : (v \mapsto r) \cup (l \mapsto l) \qquad \sigma \in \mathit{Store} : r \mapsto n$$
$$\delta \in \mathit{Time} : \mathbb{N} \qquad \mathbb{C} \in \mathit{Config} : \langle p, \gamma, \sigma, \delta \rangle$$

$$E ::= \square \mid E \oplus c \mid n \oplus E \mid x := E \mid E \;\mathbf{;}\; c$$
$$\mid \textbf{prog } l = \vec{r_\ell} \textbf{ in } E \mid \textbf{let } \vec{s} \textbf{ in } E$$

$$\langle E[r], \gamma, \sigma, \delta \rangle \rightsquigarrow \langle E[\sigma(r)], \gamma, \sigma, \delta \rangle \qquad \text{(REG)}$$

$$\langle E[v], \gamma, \sigma, \delta \rangle \rightsquigarrow \langle E[\sigma(\gamma(v))], \gamma, \sigma, \delta \rangle \qquad \text{(VAR)}$$

$$\langle E[n_1 \oplus n_2], \gamma, \sigma, \delta \rangle \rightsquigarrow \langle E[n_1 \llbracket \oplus \rrbracket n_2], \gamma, \sigma, \delta \rangle \qquad \text{(OP)}$$

$$\frac{\langle e, \gamma, \sigma, \delta \rangle \rightsquigarrow^* \langle n, \gamma, \sigma, \delta \rangle}{\langle E[r := e], \gamma, \sigma, \delta \rangle \rightsquigarrow \langle E[\textbf{skip}], \gamma, \sigma[r \mapsto n], \delta \rangle} \qquad \text{(ASSIGN-R)}$$

$$\frac{\langle e, \gamma, \sigma, \delta \rangle \rightsquigarrow^* \langle n, \gamma, \sigma, \delta \rangle}{\langle E[v := e], \gamma, \sigma, \delta \rangle \rightsquigarrow \langle E[\textbf{skip}], \gamma, \sigma[\gamma(v) \mapsto n], \delta \rangle} \qquad \text{(ASSIGN-V)}$$

$$\frac{\langle e, \gamma, \sigma, \delta \rangle \rightsquigarrow^* \langle n, \gamma, \sigma, \delta \rangle \qquad c' = \begin{cases} c_1 & : n = 0 \\ c_2 & : n \neq 0 \end{cases}}{\langle E[\textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2], \gamma, \sigma, \delta \rangle \rightsquigarrow \langle E[c'], \gamma, \sigma, \delta \rangle} \qquad \text{(IF)}$$

$$\langle E[\textbf{skip} \;\mathbf{;}\; c], \gamma, \sigma, \delta \rangle \rightsquigarrow \langle E[c], \gamma, \sigma, \delta \rangle \qquad \text{(SEQ)}$$

$$\langle E[\textbf{fall}_l], \gamma, \sigma, \delta \rangle \rightsquigarrow \langle \mathcal{F}_{cmd}(\gamma(l)), \gamma, \sigma, \delta \rangle \qquad \text{(FALL)}$$

$$\frac{\begin{array}{c} \gamma_1 = \mathit{Reset}(\gamma, l) \\ \gamma_2 = \gamma_1[\mathcal{F}_{pnt}(l) \mapsto l] \\ \gamma_3 = \gamma_2[\mathcal{F}_{prm}(l) \mapsto \gamma(\vec{x})] \end{array}}{\langle E[\textbf{goto } l(\vec{x})], \gamma, \sigma, \delta \rangle \rightsquigarrow \langle \mathcal{F}_{root}, \gamma_3, \sigma, \delta{+}1 \rangle} \qquad \text{(GOTO)}$$

Figure 4. Small-step semantic rules for Caisson ($\rightsquigarrow^*$is the reflexive transitive closure of $\rightsquigarrow$).

Challenges and Opportunities of Security-Aware EDA （计划中，待填写）

JAKOB FELDTKELLER, Ruhr University Bochum, Germany

PASCAL SASDRICH, Ruhr University Bochum, Germany

TIM GÜNEYSU, Ruhr University Bochum, Germany

DFKI, Cyber-Physical Systems, Germany

TODO: 把各种HDL安全框架放一起做个比对，然后看一下有没有什么可以提升的空间