



A formal study of two physical countermeasures against side channel attacks

PROOFS'2012

Sébastien Briaïs Sylvain Guilley Jean-Luc Danger

2012, September 13th

- 1. Introduction**
- 2. Combinational Circuits**
- 3. Formalisation of WDDL and BCDL**
- 4. Discussion**
- 5. Conclusion**

- 1. Introduction**
- 2. Combinational Circuits**
 1. Language
 2. Formal semantics, equivalences
- 3. Formalisation of WDDL and BCDL**
 1. Preliminaries
 2. WDDL
 3. BCDL
- 4. Discussion**
- 5. Conclusion**

Physical Attacks

Cryptographic devices need to be protected.

Physical Attacks

Cryptographic devices need to be protected.

Side-Channel Attacks

- **Passive** attacks.
- Power consumption, electromagnetic radiation, computation time... may leak sensitive data.

Physical Attacks

Cryptographic devices need to be protected.

Side-Channel Attacks

- **Passive** attacks.
- Power consumption, electromagnetic radiation, computation time... may leak sensitive data.

Extra logic is required in order to mask the sensitive data or to balance the leakage.

Dual-rail Precharge Logic

- Aims at making the device activity independent on the data being processed.

Dual-rail Precharge Logic

- Aims at making the device activity independent on the data being processed.
- A signal is represented by a pair of wires: $T = 10$, $F = 01$.

Dual-rail Precharge Logic

- Aims at making the device activity independent on the data being processed.
- A signal is represented by a pair of wires: $T = 10$, $F = 01$.
- A cycle of computation alternates two phases:
 - **precharge** phase: propagation of $\text{NULL} = \{(0, 0)\}$ through the combinational part of the circuit.
 - **evaluation** phase: the data is processed by the combinational part of the circuit.

Dual-rail Precharge Logic

- Aims at making the device activity independent on the data being processed.
- A signal is represented by a pair of wires: $T = 10$, $F = 01$.
- A cycle of computation alternates two phases:
 - **precharge** phase: propagation of $\text{NULL} = \{(0, 0)\}$ through the combinational part of the circuit.
 - **evaluation** phase: the data is processed by the combinational part of the circuit.

Many proposals: WDDL, STTL, DRSL, BCDL, ...

Dual-rail Precharge Logic

- Aims at making the device activity independent on the data being processed.
- A signal is represented by a pair of wires: $T = 10$, $F = 01$.
- A cycle of computation alternates two phases:
 - **precharge** phase: propagation of $\text{NULL} = \{(0,0)\}$ through the combinational part of the circuit.
 - **evaluation** phase: the data is processed by the combinational part of the circuit.

Many proposals: WDDL, STTL, DRSL, BCDL, ...

Possible vulnerabilities:

- Glitches
- Early evaluation

Dual-rail Precharge Logic

- Aims at making the device activity independent on the data being processed.
- A signal is represented by a pair of wires: $T = 10$, $F = 01$.
- A cycle of computation alternates two phases:
 - **precharge** phase: propagation of $\text{NULL} = \{(0,0)\}$ through the combinational part of the circuit.
 - **evaluation** phase: the data is processed by the combinational part of the circuit.

Many proposals: **WDDL**, STTL, **DRSL**, BCDL, ...

Possible vulnerabilities:

- **Glitches**
- **Early evaluation**

1. Introduction
2. **Combinational Circuits**
 1. Language
 2. Formal semantics, equivalences
3. Formalisation of WDDL and BCDL
 1. Preliminaries
 2. WDDL
 3. BCDL
4. Discussion
5. Conclusion

1. Introduction
2. **Combinational Circuits**
 1. Language
 2. Formal semantics, equivalences
3. Formalisation of WDDL and BCDL
 1. Preliminaries
 2. WDDL
 3. BCDL
4. Discussion
5. Conclusion

Syntax

A combinational circuit is a directed acyclic graph of logical gates.

Syntax

A combinational circuit is a directed **acyclic** graph of logical gates.

Syntax

A combinational circuit is a directed acyclic graph of logical gates.

Combinational circuits

Let \mathcal{G} be a set of logical gates.

Syntax

A combinational circuit is a directed acyclic graph of logical gates.

Combinational circuits

Let \mathcal{G} be a set of logical gates.

We define by induction the set of combinational circuits over \mathcal{G} :

$$P, Q ::= 0$$

empty circuit

Syntax

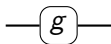
A combinational circuit is a directed acyclic graph of logical gates.

Combinational circuits

Let \mathcal{G} be a set of logical gates.

We define by induction the set of combinational circuits over \mathcal{G} :

$$P, Q ::= 0 \mid g$$



logical gate $g \in \mathcal{G}$

Syntax

A combinational circuit is a directed acyclic graph of logical gates.

Combinational circuits

Let \mathcal{G} be a set of logical gates.

We define by induction the set of combinational circuits over \mathcal{G} :

$$P, Q ::= 0 \mid g \mid I$$

a single wire



Syntax

A combinational circuit is a directed acyclic graph of logical gates.

Combinational circuits

Let \mathcal{G} be a set of logical gates.

We define by induction the set of combinational circuits over \mathcal{G} :

$$P, Q ::= 0 \mid g \mid I \mid Y$$



a fork

Syntax

A combinational circuit is a directed acyclic graph of logical gates.

Combinational circuits

Let \mathcal{G} be a set of logical gates.

We define by induction the set of combinational circuits over \mathcal{G} :

$$P, Q ::= 0 \mid g \mid I \mid Y \mid X$$



a swap

Syntax

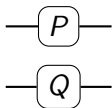
A combinational circuit is a directed acyclic graph of logical gates.

Combinational circuits

Let \mathcal{G} be a set of logical gates.

We define by induction the set of combinational circuits over \mathcal{G} :

$$P, Q ::= 0 \mid g \mid I \mid Y \mid X \mid P \mid Q$$



parallel composition

Syntax

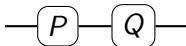
A combinational circuit is a directed acyclic graph of logical gates.

Combinational circuits

Let \mathcal{G} be a set of logical gates.

We define by induction the set of combinational circuits over \mathcal{G} :

$$P, Q ::= 0 \mid g \mid I \mid Y \mid X \mid P \mid Q \mid P ; Q$$



sequential composition

Well-formedness

Circuit with n inputs and m outputs

$$\frac{\mathcal{T}(g) = (n, m)}{g : n \otimes m} \quad g \in \mathcal{G}$$

$$\overline{\mathbf{0} : 0 \otimes 0}$$

$$\overline{\mathbf{1} : 1 \otimes 1}$$

$$\overline{\mathbf{Y} : 1 \otimes 2}$$

$$\overline{\mathbf{X} : 2 \otimes 2}$$

$$\frac{P_1 : n_1 \otimes m_1 \quad P_2 : n_2 \otimes m_2}{P_1 | P_2 : n_1 + n_2 \otimes m_1 + m_2}$$

$$\frac{P_1 : n \otimes m \quad P_2 : m \otimes p}{P_1 ; P_2 : n \otimes p}$$

Well-formedness

Circuit with n inputs and m outputs

$$\frac{\mathcal{T}(g) = (n, m)}{g : n \otimes m} \quad g \in \mathcal{G}$$

$$\overline{\mathbf{0} : 0 \otimes 0}$$

$$\overline{\mathbf{1} : 1 \otimes 1}$$

$$\overline{\mathbf{Y} : 1 \otimes 2}$$

$$\overline{\mathbf{X} : 2 \otimes 2}$$

$$\frac{P_1 : n_1 \otimes m_1 \quad P_2 : n_2 \otimes m_2}{P_1 | P_2 : n_1 + n_2 \otimes m_1 + m_2}$$

$$\frac{P_1 : n \otimes m \quad P_2 : m \otimes p}{P_1 ; P_2 : n \otimes p}$$

Well-formedness

Circuit with n inputs and m outputs

$$\frac{\mathcal{T}(g) = (n, m)}{g : n \otimes m} \quad g \in \mathcal{G}$$

$$\overline{\mathbf{0} : 0 \otimes 0}$$

$$\overline{\mathbf{1} : 1 \otimes 1}$$

$$\overline{\mathbf{Y} : 1 \otimes 2}$$

$$\overline{\mathbf{X} : 2 \otimes 2}$$

$$P_1 : n_1 \otimes m_1 \quad P_2 : n_2 \otimes m_2$$

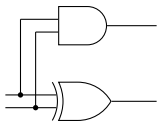
$$\frac{P_1 \mid P_2 : n_1 + n_2 \otimes m_1 + m_2}{}$$

$$P_1 : n \otimes m \quad P_2 : m \otimes p$$

$$\frac{P_1 ; P_2 : n \otimes p}{}$$

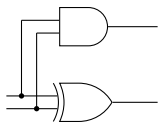
Example: a half-adder

$$\mathcal{G} = \{\text{AND}, \text{XOR}\}$$



Example: a half-adder

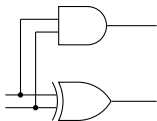
$$\mathcal{G} = \{\text{AND}, \text{XOR}\}$$



$$\text{Half} := (\mathbf{Y} \mid \mathbf{Y}); (\mathbf{I} \mid \mathbf{X} \mid \mathbf{I}); (\text{AND} \mid \text{XOR})$$

Example: a half-adder

$$\mathcal{G} = \{\text{AND}, \text{XOR}\}$$



$$\text{Half} := (\mathbf{Y} \mid \mathbf{Y}); (\mathbf{I} \mid \mathbf{X} \mid \mathbf{I}); (\text{AND} \mid \text{XOR})$$

Half is a combinational circuit with 2 inputs and 2 outputs, i.e.
Half : $2 \otimes 2$.

1. Introduction
2. **Combinational Circuits**
 1. Language
 2. Formal semantics, equivalences
3. Formalisation of WDDL and BCDL
 1. Preliminaries
 2. WDDL
 3. BCDL
4. Discussion
5. Conclusion

Preliminary definitions

Alphabet, words, concatenation.

An **alphabet** Σ is a finite set of letters.

A **word** u over Σ is a finite sequence of letters $u = u_1 \cdots u_n$ where $u_i \in \Sigma$.

The set of words over Σ is noted Σ^* .

The integer n is the **length** of u and noted $|u|$.

The **empty word** is noted ϵ and is the unique word of length 0.

The set of words of length n is noted Σ^n .

The **concatenation** of $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_m$ is defined $u \bullet v := u_1 \cdots u_n v_1 \cdots v_m$.

Formal semantics

C computes y on input x

The semantics of circuits is given by a relation of $\Sigma^* \times \Sigma^*$.

$$\frac{x \in \Sigma^* \quad \mathcal{E}(g)(x) = y \in \Sigma^*}{g \Vdash x \rightsquigarrow y} \quad g \in \mathcal{G}$$

$$\frac{}{\mathbf{0} \Vdash \epsilon \rightsquigarrow \epsilon}$$

$$\frac{}{\mathbf{I} \Vdash a \rightsquigarrow a} \quad a \in \Sigma$$

$$\frac{}{\mathbf{Y} \Vdash a \rightsquigarrow aa} \quad a \in \Sigma$$

$$\frac{}{\mathbf{X} \Vdash ab \rightsquigarrow ba} \quad a, b \in \Sigma$$

$$\frac{P_1 \Vdash x_1 \rightsquigarrow y_1 \quad P_2 \Vdash x_2 \rightsquigarrow y_2}{P_1 | P_2 \Vdash x_1 \bullet x_2 \rightsquigarrow y_1 \bullet y_2}$$

$$\frac{P_1 \Vdash x \rightsquigarrow y \quad P_2 \Vdash y \rightsquigarrow z}{P_1 ; P_2 \Vdash x \rightsquigarrow z}$$

Formal semantics

C computes y on input x

The semantics of circuits is given by a relation of $\Sigma^* \times \Sigma^*$.

$$\frac{x \in \Sigma^* \quad \mathcal{E}(g)(x) = y \in \Sigma^*}{g \Vdash x \rightsquigarrow y} \quad g \in \mathcal{G}$$

$$\frac{}{\mathbf{0} \Vdash \epsilon \rightsquigarrow \epsilon}$$

$$\frac{}{\mathbf{I} \Vdash a \rightsquigarrow a} \quad a \in \Sigma$$

$$\frac{}{\mathbf{Y} \Vdash a \rightsquigarrow aa} \quad a \in \Sigma$$

$$\frac{}{\mathbf{X} \Vdash ab \rightsquigarrow ba} \quad a, b \in \Sigma$$

$$\frac{P_1 \Vdash x_1 \rightsquigarrow y_1 \quad P_2 \Vdash x_2 \rightsquigarrow y_2}{P_1 | P_2 \Vdash x_1 \bullet x_2 \rightsquigarrow y_1 \bullet y_2}$$

$$\frac{P_1 \Vdash x \rightsquigarrow y \quad P_2 \Vdash y \rightsquigarrow z}{P_1 ; P_2 \Vdash x \rightsquigarrow z}$$

$\mathcal{E}(g)$ is a partial function $\Sigma^* \rightarrow \Sigma^*$, defined consistently w.r.t. the typing function.

Formal semantics

C computes y on input x

The semantics of circuits is given by a relation of $\Sigma^* \times \Sigma^*$.

$$\frac{x \in \Sigma^* \quad \mathcal{E}(g)(x) = y \in \Sigma^*}{g \Vdash x \rightsquigarrow y} \quad g \in \mathcal{G}$$

$$\overline{\mathbf{0} \Vdash \epsilon \rightsquigarrow \epsilon}$$

$$\overline{\mathbf{I} \Vdash a \rightsquigarrow a} \quad a \in \Sigma$$

$$\overline{\mathbf{Y} \Vdash a \rightsquigarrow aa} \quad a \in \Sigma$$

$$\overline{\mathbf{X} \Vdash ab \rightsquigarrow ba} \quad a, b \in \Sigma$$

$$\frac{P_1 \Vdash x_1 \rightsquigarrow y_1 \quad P_2 \Vdash x_2 \rightsquigarrow y_2}{P_1 | P_2 \Vdash x_1 \bullet x_2 \rightsquigarrow y_1 \bullet y_2}$$

$$\frac{P_1 \Vdash x \rightsquigarrow y \quad P_2 \Vdash y \rightsquigarrow z}{P_1 ; P_2 \Vdash x \rightsquigarrow z}$$

$$P \simeq Q \iff \forall x, y : P \Vdash x \rightsquigarrow y \iff Q \Vdash x \rightsquigarrow y$$

Structural congruence

\equiv identifies circuits that only differ in some minor wiring details.

It is the smallest congruence that satisfies the following equations:

- $(P_1 | P_2) | P_3 \equiv P_1 | (P_2 | P_3)$
- $P | \mathbf{0} \equiv \mathbf{0} | P \equiv P$

Structural congruence

\equiv identifies circuits that only differ in some minor wiring details.

It is the smallest congruence that satisfies the following equations:

- $(P_1 | P_2) | P_3 \equiv P_1 | (P_2 | P_3)$
- $P | \mathbf{0} \equiv \mathbf{0} | P \equiv P$
- $(P_1 ; P_2) ; P_3 \equiv P_1 ; (P_2 ; P_3)$
- If $P : n \otimes m$ then $\mathbf{I}^n ; P \equiv P ; \mathbf{I}^m \equiv P$

Structural congruence

\equiv identifies circuits that only differ in some minor wiring details.

It is the smallest congruence that satisfies the following equations:

- $(P_1 | P_2) | P_3 \equiv P_1 | (P_2 | P_3)$
- $P | \mathbf{0} \equiv \mathbf{0} | P \equiv P$
- $(P_1 ; P_2) ; P_3 \equiv P_1 ; (P_2 ; P_3)$
- If $P : n \otimes m$ then $\mathbf{I}^n ; P \equiv P ; \mathbf{I}^m \equiv P$
- If $P_1 : n \otimes m$ and $P_2 : m \otimes p$ then
 $(P_1 ; P_2) | (P_3 ; P_4) \equiv (P_1 | P_3) ; (P_2 | P_4)$

Structural congruence

\equiv identifies circuits that only differ in some minor wiring details.
It is the smallest congruence that satisfies the following equations:

- $(P_1 | P_2) | P_3 \equiv P_1 | (P_2 | P_3)$
- $P | \mathbf{0} \equiv \mathbf{0} | P \equiv P$
- $(P_1 ; P_2) ; P_3 \equiv P_1 ; (P_2 ; P_3)$
- If $P : n \otimes m$ then $\mathbf{I}^n ; P \equiv P ; \mathbf{I}^m \equiv P$
- If $P_1 : n \otimes m$ and $P_2 : m \otimes p$ then
 $(P_1 ; P_2) | (P_3 ; P_4) \equiv (P_1 | P_3) ; (P_2 | P_4)$
- $\mathbf{Y} ; (\mathbf{I} | \mathbf{Y}) \equiv \mathbf{Y} ; (\mathbf{Y} | \mathbf{I})$
- $\mathbf{Y} ; \mathbf{X} \equiv \mathbf{Y}$
- $\mathbf{X} ; \mathbf{X} \equiv \mathbf{I} | \mathbf{I}$
- $\mathbf{X} ; (\mathbf{Y} | \mathbf{Y}) \equiv (\mathbf{Y} | \mathbf{Y}) ; (\mathbf{I} | \mathbf{X} | \mathbf{I}) ; (\mathbf{X} | \mathbf{X}) ; (\mathbf{I} | \mathbf{X} | \mathbf{I})$

Some results

- If $P : n \otimes m$ and $P : n' \otimes m'$ then $n = n'$ and $m = m'$.
- If $P \equiv Q$ then $P : n \otimes m \iff Q : n \otimes m$.
- If $P \Vdash x \rightsquigarrow y$ then $P : |x| \otimes |y|$.
- If $P : n \otimes m$ and $P \Vdash x \rightsquigarrow y$ then $|x| = n$ and $|y| = m$.
- If $P : n \otimes m$ then for any x such that $|x| = n$ there exists y such that $P \Vdash x \rightsquigarrow y$.
- If $P \Vdash x \rightsquigarrow y$ and $P \Vdash x \rightsquigarrow z$ then $y = z$.
- \simeq is a congruence.
- If P and Q are ill-formed then $P \simeq Q$.
- $\equiv \subseteq \simeq$.

1. Introduction
2. Combinational Circuits
 1. Language
 2. Formal semantics, equivalences
- 3. Formalisation of WDDL and BCDL**
 1. Preliminaries
 2. WDDL
 3. BCDL
4. Discussion
5. Conclusion

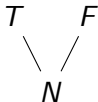
1. Introduction
2. Combinational Circuits
 1. Language
 2. Formal semantics, equivalences
- 3. Formalisation of WDDL and BCDL**
 1. Preliminaries
 2. WDDL
 3. BCDL
4. Discussion
5. Conclusion

Definitions

- In the following, let $\Sigma = \{0, 1\}$.
- We pose $T = 10$, $F = 01$, $N = 00$ et $E = 11$.
NULL = $\{N\}$, VALID = $\{T, F\}$, FAULT = $\{E\}$.

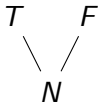
Definitions

- In the following, let $\Sigma = \{0, 1\}$.
- We pose $T = 10$, $F = 01$, $N = 00$ et $E = 11$.
NULL = $\{N\}$, VALID = $\{T, F\}$, FAULT = $\{E\}$.
- Let \preceq be the partial order defined by:



Definitions

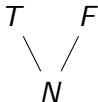
- In the following, let $\Sigma = \{0, 1\}$.
- We pose $T = 10$, $F = 01$, $N = 00$ et $E = 11$.
 $\text{NULL} = \{N\}$, $\text{VALID} = \{T, F\}$, $\text{FAULT} = \{E\}$.
- Let \preceq be the partial order defined by:



- Let \sim be the equivalence relation on Σ^2 whose equivalence classes are NULL, VALID and FAULT.
- We extend these definitions to words of even length.

Definitions

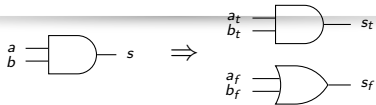
- In the following, let $\Sigma = \{0, 1\}$.
- We pose $T = 10$, $F = 01$, $N = 00$ et $E = 11$.
 $\text{NULL} = \{N\}$, $\text{VALID} = \{T, F\}$, $\text{FAULT} = \{E\}$.
- Let \preceq be the partial order defined by:



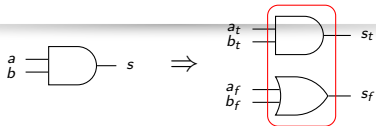
- Let \sim be the equivalence relation on Σ^2 whose equivalence classes are NULL, VALID and FAULT.
- We extend these definitions to words of even length.
- For $u \in \Sigma^*$, we let $[u] \in \text{VALID}^*$ be the **corresponding word in dual-rail logic**.
example: $[0110] = FTTF = 01101001$

1. Introduction
2. Combinational Circuits
 1. Language
 2. Formal semantics, equivalences
- 3. Formalisation of WDDL and BCDL**
 1. Preliminaries
 2. WDDL
 3. BCDL
4. Discussion
5. Conclusion

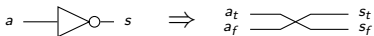
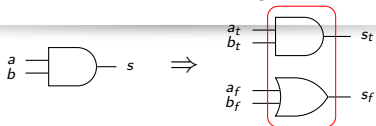
Transformation process



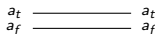
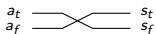
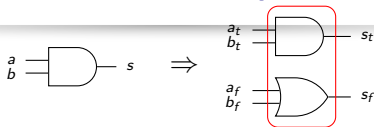
Transformation process



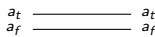
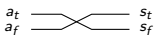
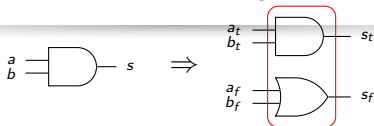
Transformation process



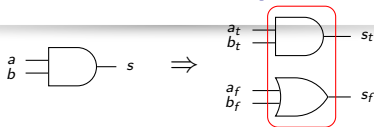
Transformation process



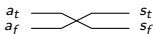
Transformation process



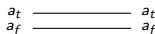
Transformation process



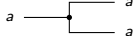
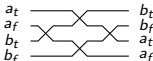
\Rightarrow



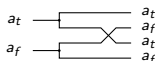
\Rightarrow



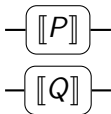
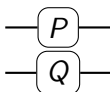
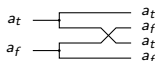
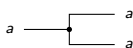
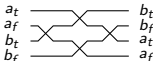
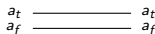
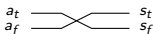
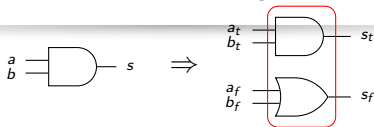
\Rightarrow



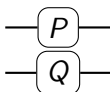
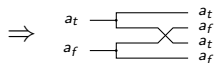
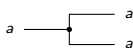
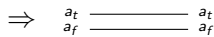
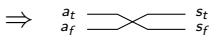
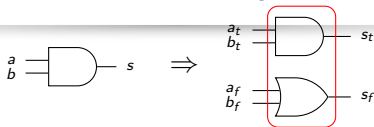
\Rightarrow



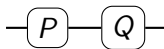
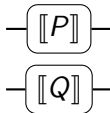
Transformation process



Transformation process



\Rightarrow



\Rightarrow



Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

The transformation is sound.

If $P \Vdash x \rightsquigarrow y$ then $\llbracket P \rrbracket \Vdash [x] \rightsquigarrow [y]$.

Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

The transformation is sound.

If $P \Vdash x \rightsquigarrow y$ then $\llbracket P \rrbracket \Vdash [x] \rightsquigarrow [y]$.

No glitches are possible.

$$\llbracket P \rrbracket \Vdash x \rightsquigarrow y$$

Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

The transformation is sound.

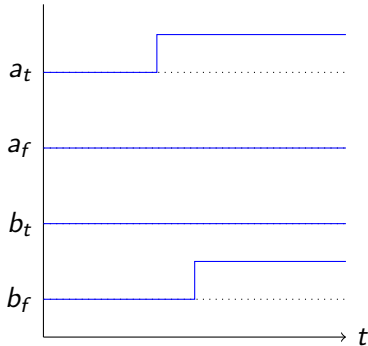
If $P \Vdash x \rightsquigarrow y$ then $\llbracket P \rrbracket \Vdash [x] \rightsquigarrow [y]$.

No glitches are possible.

$$\begin{array}{c} \llbracket P \rrbracket \Vdash x \rightsquigarrow y \\ \rightsquigarrow \\ x' \end{array}$$

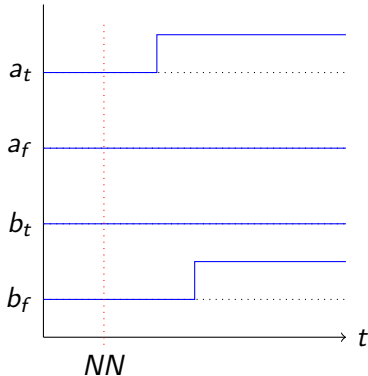
Interpretation of \preceq

\preceq models the transition of signals from precharge state (NULL) to evaluation state (VALID).



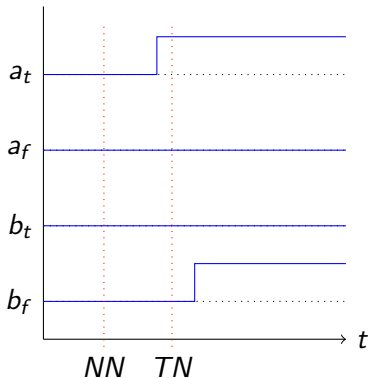
Interpretation of \preceq

\preceq models the transition of signals from precharge state (NULL) to evaluation state (VALID).



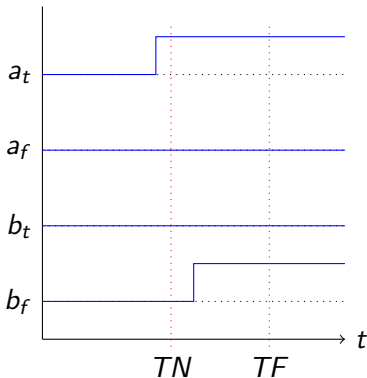
Interpretation of \preceq

\preceq models the transition of signals from precharge state (NULL) to evaluation state (VALID).



Interpretation of \preceq

\preceq models the transition of signals from precharge state (NULL) to evaluation state (VALID).



Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

The transformation is sound.

If $P \Vdash x \rightsquigarrow y$ then $\llbracket P \rrbracket \Vdash [x] \rightsquigarrow [y]$.

No glitches are possible.

$$\begin{array}{c} \llbracket P \rrbracket \Vdash x \rightsquigarrow y \\ \Downarrow \\ x' \end{array}$$

Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

The transformation is sound.

If $P \Vdash x \rightsquigarrow y$ then $\llbracket P \rrbracket \Vdash [x] \rightsquigarrow [y]$.

No glitches are possible.

$$\begin{array}{c} \llbracket P \rrbracket \Vdash x \rightsquigarrow y \\ \Downarrow \\ \llbracket P \rrbracket \Vdash x' \rightsquigarrow y' \end{array}$$

Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

The transformation is sound.

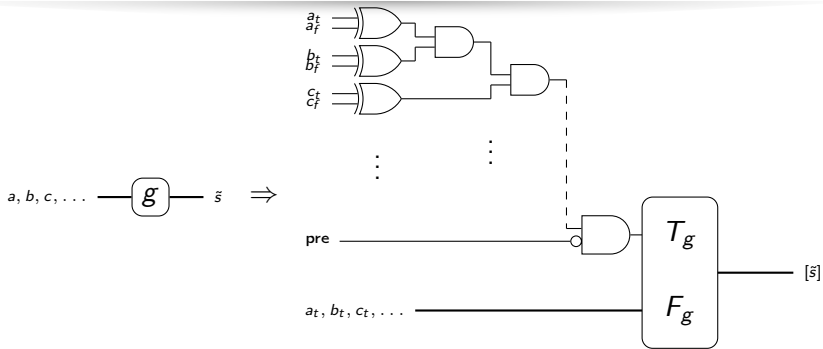
If $P \Vdash x \rightsquigarrow y$ then $\llbracket P \rrbracket \Vdash [x] \rightsquigarrow [y]$.

No glitches are possible.

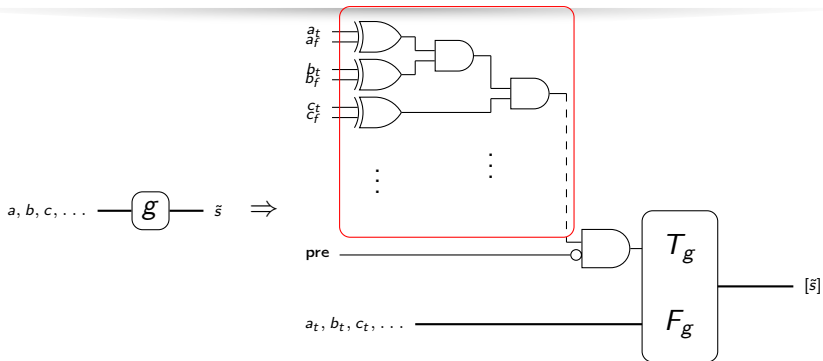
$$\begin{array}{ccc} \llbracket P \rrbracket \Vdash x \rightsquigarrow y & & \\ \Downarrow & & \Downarrow \\ \llbracket P \rrbracket \Vdash x' \rightsquigarrow y' & & \end{array}$$

1. Introduction
2. Combinational Circuits
 1. Language
 2. Formal semantics, equivalences
- 3. Formalisation of WDDL and BCDL**
 1. Preliminaries
 2. WDDL
 3. BCDL
4. Discussion
5. Conclusion

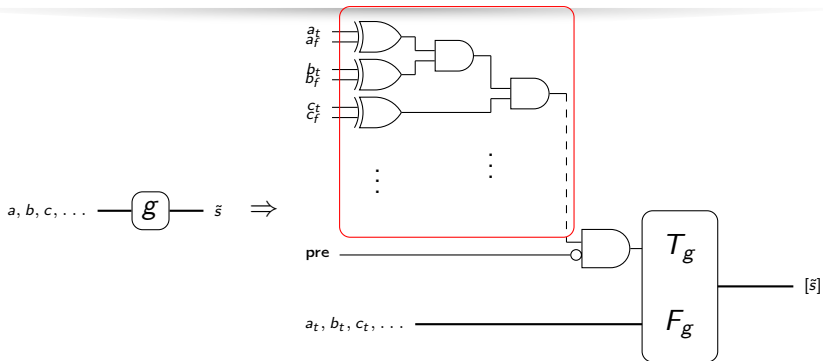
Transformation process



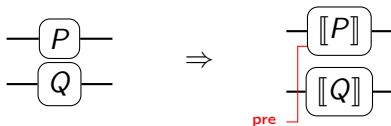
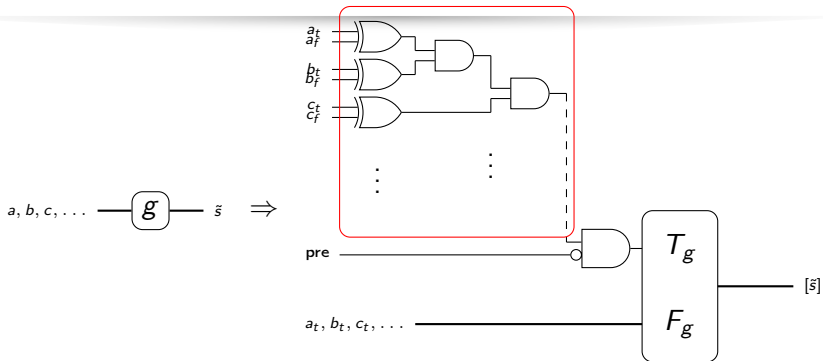
Transformation process



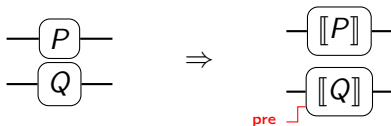
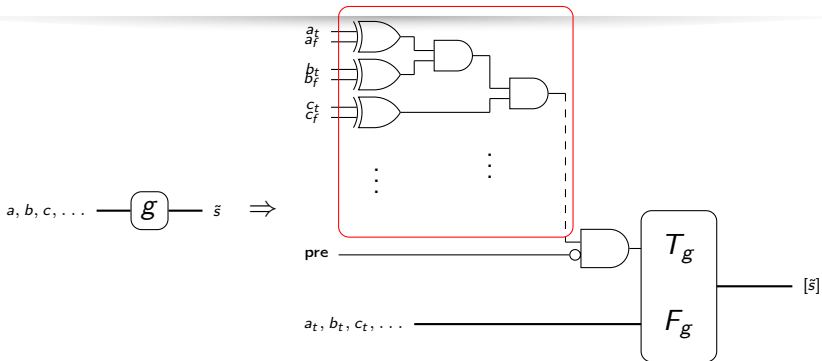
Transformation process



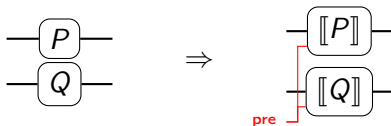
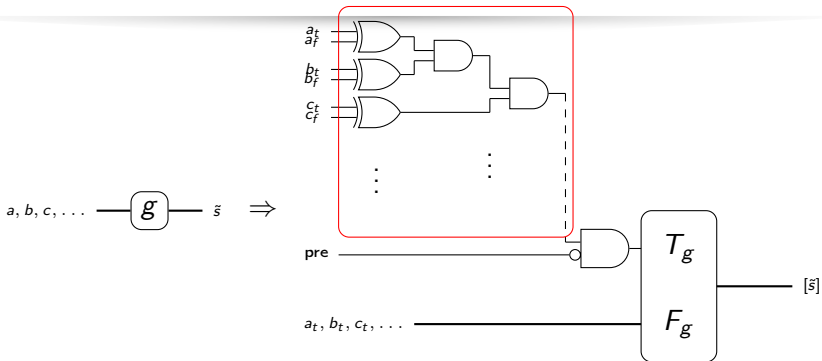
Transformation process



Transformation process



Transformation process



Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket g \rrbracket \Vdash 1x \rightsquigarrow y$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash \hat{1}x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash \hat{0}x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket g \rrbracket \Vdash 1x \rightsquigarrow y$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash \hat{1}x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash \hat{0}x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

The transformation is sound.

If $P \Vdash x \rightsquigarrow y$ then $\llbracket P \rrbracket \Vdash \hat{0}[x] \rightsquigarrow [y]$.

Properties

$\llbracket P \rrbracket$ fulfils the DPL invariants.

- If $\llbracket g \rrbracket \Vdash 1x \rightsquigarrow y$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash \hat{1}x \rightsquigarrow y$ and $x \in \text{NULL}^*$ then $y \in \text{NULL}^*$.
- If $\llbracket P \rrbracket \Vdash \hat{0}x \rightsquigarrow y$ and $x \in \text{VALID}^*$ then $y \in \text{VALID}^*$.

The transformation is sound.

If $P \Vdash x \rightsquigarrow y$ then $\llbracket P \rrbracket \Vdash \hat{0}[x] \rightsquigarrow [y]$.

No glitches are possible.

If $\llbracket P \rrbracket \Vdash \hat{p}x \rightsquigarrow \hat{p}y$, $x \preceq x'$ and $\llbracket P \rrbracket \Vdash \hat{p}x' \rightsquigarrow \hat{p}y'$ then $y \preceq y'$.

Properties

There is no early-evaluation.

If $\llbracket P \rrbracket \Vdash \hat{0}x \rightsquigarrow y$ and $y \in \text{VALID}^*$ then $x \in \text{VALID}^*$.
(provided that P does not contain gates with 0 outputs)

Properties

There is no early-evaluation.

If $\llbracket P \rrbracket \Vdash \hat{0}x \rightsquigarrow y$ and $y \in \text{VALID}^*$ then $x \in \text{VALID}^*$.
(provided that P does not contain gates with 0 outputs)

The transformation is complete.

If $\llbracket P \rrbracket \Vdash \hat{0}x' \rightsquigarrow y'$ and $y' \in \text{VALID}^*$ then
there exists x, y such that $x' = [x]$, $y' = [y]$ and $P \Vdash x \rightsquigarrow y$.
(provided that P does not contain gates with 0 outputs)

Properties

There is no early-evaluation.

If $\llbracket P \rrbracket \Vdash \hat{O}x \rightsquigarrow y$ and $y \in \text{VALID}^*$ then $x \in \text{VALID}^*$.
(provided that P does not contain gates with 0 outputs)

The transformation is complete.

If $\llbracket P \rrbracket \Vdash \hat{O}x' \rightsquigarrow y'$ and $y' \in \text{VALID}^*$ then
there exists x, y such that $x' = [x]$, $y' = [y]$ and $P \Vdash x \rightsquigarrow y$.
(provided that P does not contain gates with 0 outputs)

The secured circuit behaves the same on equivalent inputs.

$$\llbracket P \rrbracket \Vdash \hat{p}x \rightsquigarrow y$$

Properties

There is no early-evaluation.

If $\llbracket P \rrbracket \Vdash \hat{0}x \rightsquigarrow y$ and $y \in \text{VALID}^*$ then $x \in \text{VALID}^*$.
(provided that P does not contain gates with 0 outputs)

The transformation is complete.

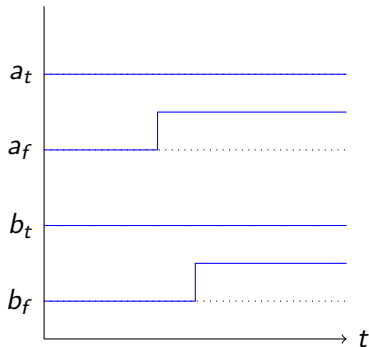
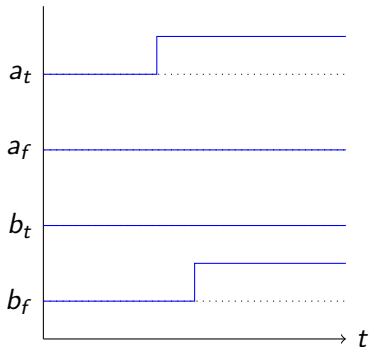
If $\llbracket P \rrbracket \Vdash \hat{0}x' \rightsquigarrow y'$ and $y' \in \text{VALID}^*$ then
there exists x, y such that $x' = [x]$, $y' = [y]$ and $P \Vdash x \rightsquigarrow y$.
(provided that P does not contain gates with 0 outputs)

The secured circuit behaves the same on equivalent inputs.

$$\begin{array}{c} \llbracket P \rrbracket \Vdash \hat{p}x \rightsquigarrow y \\ \sim \\ \hat{p}x' \end{array}$$

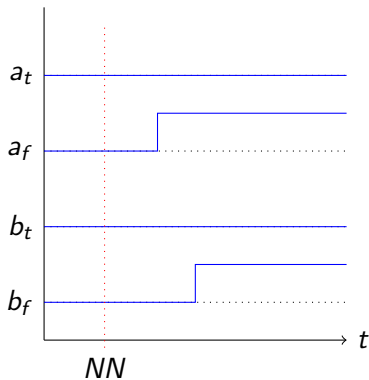
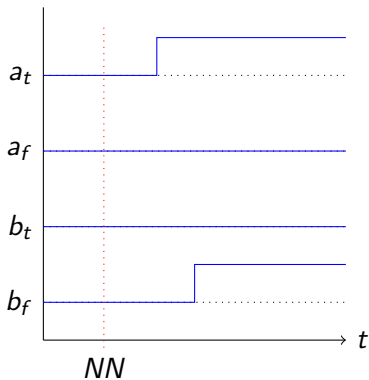
Interpretation of \sim

\sim equates words which have the same amount of information, i.e. in which corresponding dual-rail signals have the same nature.



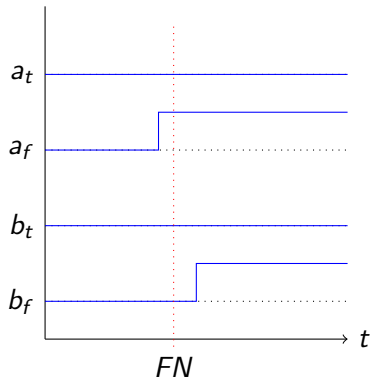
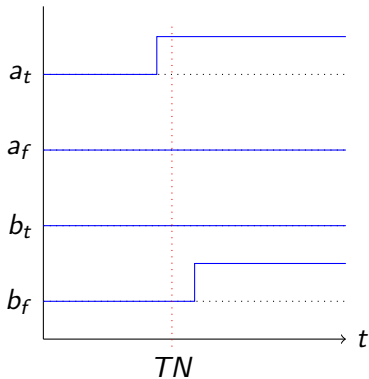
Interpretation of \sim

\sim equates words which have the same amount of information, i.e. in which corresponding dual-rail signals have the same nature.



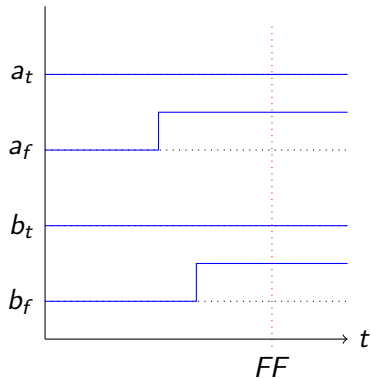
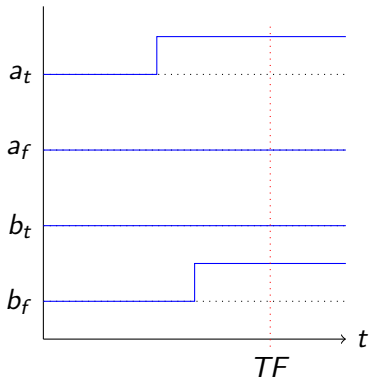
Interpretation of \sim

\sim equates words which have the same amount of information, i.e. in which corresponding dual-rail signals have the same nature.



Interpretation of \sim

\sim equates words which have the same amount of information, i.e. in which corresponding dual-rail signals have the same nature.



Properties

There is no early-evaluation.

If $\llbracket P \rrbracket \Vdash \hat{0}x \rightsquigarrow y$ and $y \in \text{VALID}^*$ then $x \in \text{VALID}^*$.
(provided that P does not contain gates with 0 outputs)

The transformation is complete.

If $\llbracket P \rrbracket \Vdash \hat{0}x' \rightsquigarrow y'$ and $y' \in \text{VALID}^*$ then
there exists x, y such that $x' = [x]$, $y' = [y]$ and $P \Vdash x \rightsquigarrow y$.
(provided that P does not contain gates with 0 outputs)

The secured circuit behaves the same on equivalent inputs.

$$\begin{array}{c} \llbracket P \rrbracket \Vdash \hat{p}x \rightsquigarrow y \\ \sim \\ \hat{p}x' \end{array}$$

Properties

There is no early-evaluation.

If $\llbracket P \rrbracket \Vdash \hat{0}x \rightsquigarrow y$ and $y \in \text{VALID}^*$ then $x \in \text{VALID}^*$.
(provided that P does not contain gates with 0 outputs)

The transformation is complete.

If $\llbracket P \rrbracket \Vdash \hat{0}x' \rightsquigarrow y'$ and $y' \in \text{VALID}^*$ then
there exists x, y such that $x' = [x]$, $y' = [y]$ and $P \Vdash x \rightsquigarrow y$.
(provided that P does not contain gates with 0 outputs)

The secured circuit behaves the same on equivalent inputs.

$$\begin{array}{c} \llbracket P \rrbracket \Vdash \hat{p}x \rightsquigarrow y \\ \sim \\ \llbracket P \rrbracket \Vdash \hat{p}x' \rightsquigarrow y' \end{array}$$

Properties

There is no early-evaluation.

If $\llbracket P \rrbracket \Vdash \hat{0}x \rightsquigarrow y$ and $y \in \text{VALID}^*$ then $x \in \text{VALID}^*$.
(provided that P does not contain gates with 0 outputs)

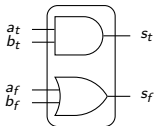
The transformation is complete.

If $\llbracket P \rrbracket \Vdash \hat{0}x' \rightsquigarrow y'$ and $y' \in \text{VALID}^*$ then
there exists x, y such that $x' = [x]$, $y' = [y]$ and $P \Vdash x \rightsquigarrow y$.
(provided that P does not contain gates with 0 outputs)

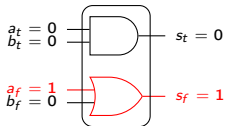
The secured circuit behaves the same on equivalent inputs.

$$\begin{array}{c} \llbracket P \rrbracket \Vdash \hat{p}x \rightsquigarrow y \\ \sim \quad \sim \\ \llbracket P \rrbracket \Vdash \hat{p}x' \rightsquigarrow y' \end{array}$$

Back to WDDL



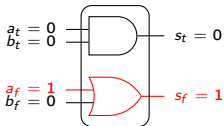
Back to WDDL



The AND_{WDDL} gate suffers from early-evaluation.

$x = 0100 \notin \text{VALID}^*$ and $y = 01 \in \text{VALID}^*$

Back to WDDL



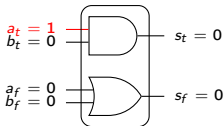
The AND_{WDDL} gate suffers from early-evaluation.

$x = 0100 \notin \text{VALID}^*$ and $y = 01 \in \text{VALID}^*$

The AND_{WDDL} gate behaves differently on equivalent inputs.

$x = 0100$ and $y = 01$

Back to WDDL



The AND_{WDDL} gate suffers from early-evaluation.

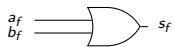
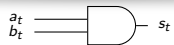
$x = 0100 \notin \text{VALID}^*$ and $y = 01 \in \text{VALID}^*$

The AND_{WDDL} gate behaves differently on equivalent inputs.

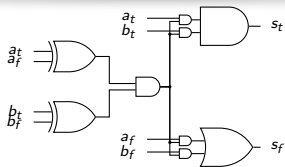
$x = 0100 \sim x' = 1000$ and $y = 01 \approx y' = 00$

1. Introduction
2. Combinational Circuits
 1. Language
 2. Formal semantics, equivalences
3. Formalisation of WDDL and BCDL
 1. Preliminaries
 2. WDDL
 3. BCDL
4. Discussion
5. Conclusion

Discussion

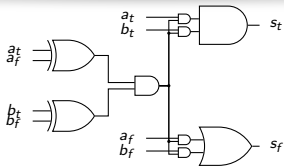


Discussion



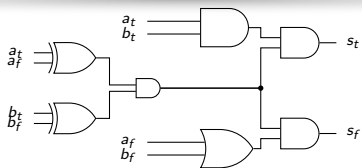
- BCDL fixes WDDL by adding a synchronisation barrier.

Discussion



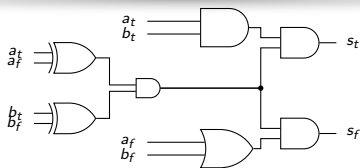
- BCDL fixes WDDL by adding a synchronisation barrier.
- How to address the race between the synchronisation signal and the data signals? (DRSL vulnerability)

Discussion



- BCDL fixes WDDL by adding a synchronisation barrier.
- How to address the race between the synchronisation signal and the data signals? (DRSL vulnerability)
- How to discriminate this circuit?

Discussion



- BCDL fixes WDDL by adding a synchronisation barrier.
- How to address the race between the synchronisation signal and the data signals? (DRSL vulnerability)
- How to discriminate this circuit?
Measure the activity of circuits and show that the activity of a circuit is constant on equivalent inputs, i.e.

$$x \sim x' \Rightarrow \mu_C(x) = \mu_C(x')$$

1. Introduction
2. Combinational Circuits
 1. Language
 2. Formal semantics, equivalences
3. Formalisation of WDDL and BCDL
 1. Preliminaries
 2. WDDL
 3. BCDL
4. Discussion
5. Conclusion

Summary

- We defined a calculus to describe combinational circuits.

Summary

- We defined a calculus to describe combinational circuits.
- We defined formally WDDL and BCDL securisation process.

Summary

- We defined a calculus to describe combinational circuits.
- We defined formally WDDL and BCDL securisation process.
- We proved the correctness of these two transformations.

Summary

- We defined a calculus to describe combinational circuits.
- We defined formally WDDL and BCDL securisation process.
- We proved the correctness of these two transformations.
- Regarding security properties, we identified some necessary conditions to fulfil.

Perspectives

- Apply the model to other dual-rail styles.

Perspectives

- Apply the model to other dual-rail styles.
- Refine the model.

The End

Thank You

Rotations

On words

- $\vec{\epsilon} = \epsilon$ and, for $a \in \Sigma, u \in \Sigma^*, \vec{ua} = au$
- $\overleftarrow{\epsilon} = \epsilon$ and, for $a \in \Sigma, u \in \Sigma^*, \overleftarrow{au} = ua$

Rotations

On words

- $\vec{\epsilon} = \epsilon$ and, for $a \in \Sigma, u \in \Sigma^*, \vec{ua} = au$
- $\overleftarrow{\epsilon} = \epsilon$ and, for $a \in \Sigma, u \in \Sigma^*, \overleftarrow{au} = ua$

We define by induction on $n \in \mathbb{N}$ the circuit ror_n :

$$\text{ror}_0 := \mathbf{0}$$

$$\text{ror}_1 := \mathbf{1}$$

$$\text{ror}_{n+2} := (\mathbf{1}^n \mid \mathbf{X}); (\text{ror}_{n+1} \mid \mathbf{1})$$

We have

$$\text{ror}_n \Vdash x \rightsquigarrow y \iff |x| = n \wedge y = \vec{x}$$

Rotations

On words

- $\vec{\epsilon} = \epsilon$ and, for $a \in \Sigma, u \in \Sigma^*, \vec{ua} = au$
- $\overleftarrow{\epsilon} = \epsilon$ and, for $a \in \Sigma, u \in \Sigma^*, \overleftarrow{au} = ua$

We define by induction on $n \in \mathbb{N}$ the circuit rol_n :

$$\text{rol}_0 := \mathbf{0}$$

$$\text{rol}_1 := \mathbf{1}$$

$$\text{rol}_{n+2} := (\text{rol}_{n+1} \mid \mathbf{1}); (\mathbf{1}^n \mid \mathbf{X})$$

We have

$$\text{rol}_n \Vdash x \rightsquigarrow y \iff |x| = n \wedge y = \overleftarrow{x}$$

Rotations

On words

- $\vec{\epsilon} = \epsilon$ and, for $a \in \Sigma, u \in \Sigma^*, \vec{ua} = au$
- $\overleftarrow{\epsilon} = \epsilon$ and, for $a \in \Sigma, u \in \Sigma^*, \overleftarrow{au} = ua$

We also have that:

- $\text{ror}_n; \text{rol}_n \equiv \mathbf{I}^n$
- $\text{rol}_n; \text{ror}_n \equiv \mathbf{I}^n$

Interleaving

On words

$\epsilon \parallel \epsilon := \epsilon$ and for $a, b \in \Sigma$, $u, v \in \Sigma^*$, $(au) \parallel (bv) := ab(u \parallel v)$

Interleaving

On words

$\epsilon \parallel \epsilon := \epsilon$ and for $a, b \in \Sigma$, $u, v \in \Sigma^*$, $(au) \parallel (bv) := ab(u \parallel v)$

We define by induction on $n \in \mathbb{N}$ the circuit int_n :

$$\text{int}_0 := \mathbf{0}$$

$$\text{int}_{n+1} := (\mathbf{1} \parallel \text{ror}_{n+1} \parallel \mathbf{1}^n); (\mathbf{1} \parallel \mathbf{1} \parallel \text{int}_n)$$

We have

$$\text{int}_n \Vdash x \rightsquigarrow y \iff x = u \bullet v \wedge |u| = |v| = n \wedge y = u \parallel v$$

Interleaving

On words

$\epsilon \parallel \epsilon := \epsilon$ and for $a, b \in \Sigma$, $u, v \in \Sigma^*$, $(au) \parallel (bv) := ab(u \parallel v)$

We define by induction on $n \in \mathbb{N}$ the circuit uint_n :

$$\text{uint}_0 := \mathbf{0}$$

$$\text{uint}_{n+1} := (\mathbf{1} \parallel \mathbf{1} \parallel \text{uint}_n); (\mathbf{1} \parallel \text{rol}_{n+1} \parallel \mathbf{1}^n)$$

We have

$$\text{uint}_n \Vdash x \rightsquigarrow y \iff y = u \bullet v \wedge |u| = |v| = n \wedge x = u \parallel v$$

Interleaving

On words

$\epsilon \parallel \epsilon := \epsilon$ and for $a, b \in \Sigma$, $u, v \in \Sigma^*$, $(au) \parallel (bv) := ab(u \parallel v)$

We also have that:

- $\text{int}_n ; \text{unint}_n \equiv \mathbb{I}^{2n}$
- $\text{unint}_n ; \text{int}_n \equiv \mathbb{I}^{2n}$