Timely Computation

Conal Elliott

ICFP, September 2023

What is a digital circuit?

Computers vs calculations

What computers are vs what we use them for:

- *Computer:* electronic circuit, transforming analog signals.
- *Calculation:* mathematical function, transforming discrete data.

How do we use one to accomplish the other?

In other words, what is a *digital circuit*?

What makes for a good definition?

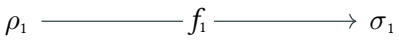
- Clear
- Simple
- Useful
- Formal
- Constructive
- Compositional

My definition

A “digital circuit” is an analog circuit that respects discrete meanings.

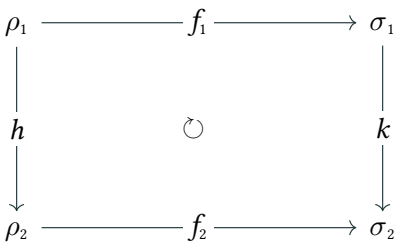
Compositionally correct computing

computing



operation

correct computing

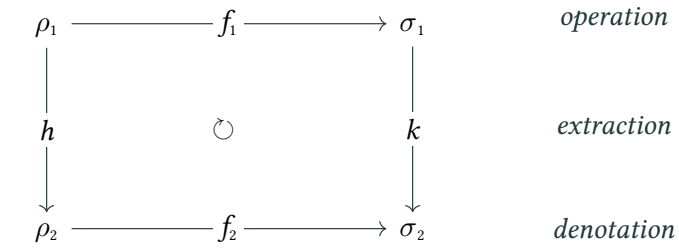


operation

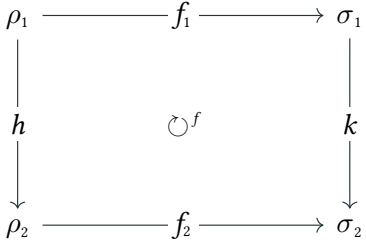
extraction

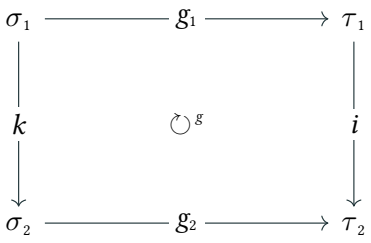
denotation

Compositionally correct computing

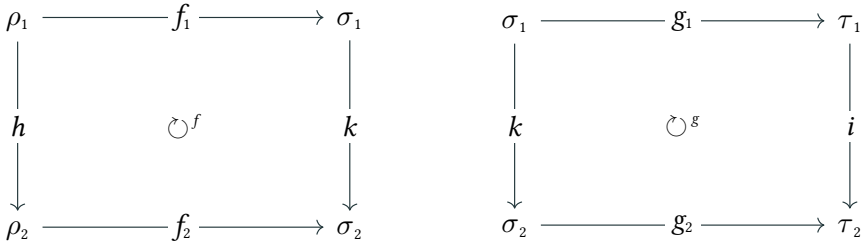


Compositional correctness: *sequential*

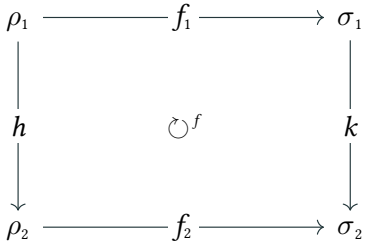


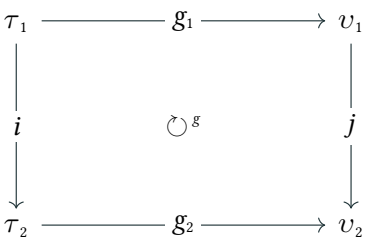


Compositional correctness: *sequential*

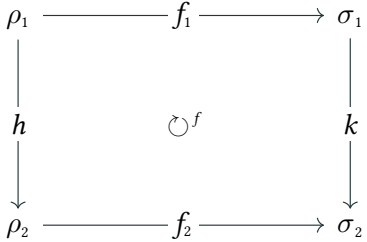


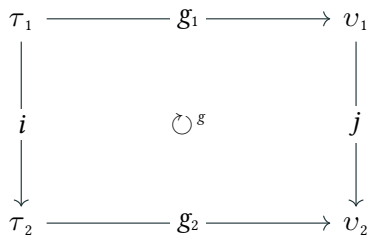
Compositional correctness: *parallel*

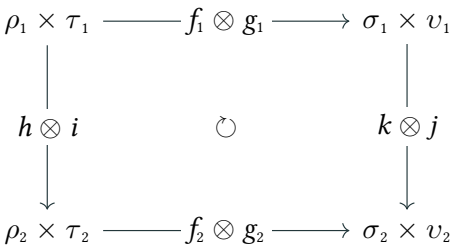




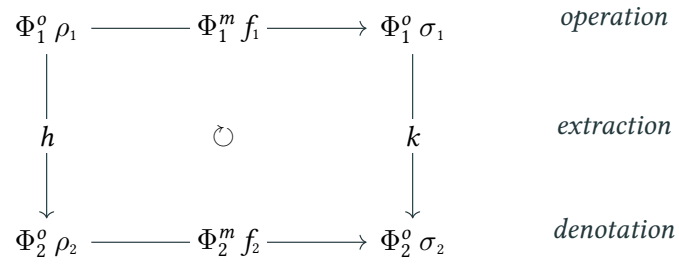
Compositional correctness: *parallel*



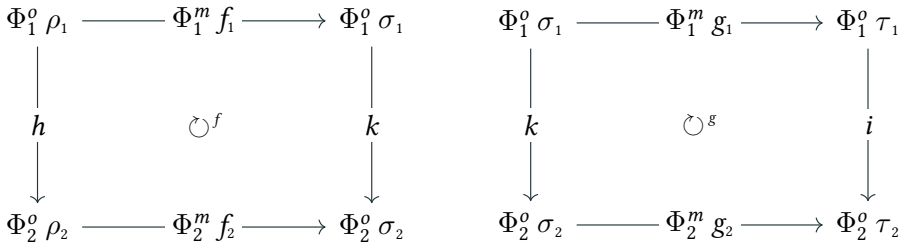




Compositionally correct computing with representations



Compositional correctness with representations: *sequential*



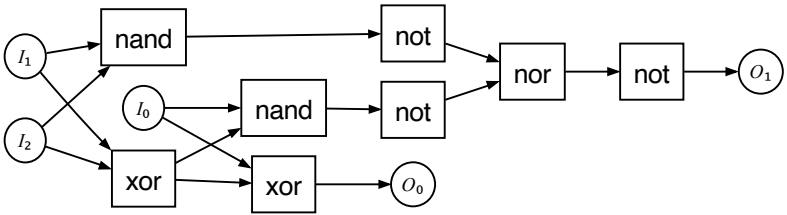
Compositional correctness with representations: *parallel*

$$\begin{array}{ccc}
\Phi_1^o\,\rho_1 & \longrightarrow & \Phi_1^m\,f_1 \longrightarrow \Phi_1^o\,\sigma_1 \\
\downarrow h & \circlearrowright^{f} & \downarrow k \\
\Phi_2^o\,\rho_2 & \longrightarrow & \Phi_2^m\,f_2 \longrightarrow \Phi_2^o\,\sigma_2
\end{array}
\qquad
\begin{array}{ccc}
\Phi_1^o\,\tau_1 & \longrightarrow & \Phi_1^m\,g_1 \longrightarrow \Phi_1^o\,\upsilon_1 \\
\downarrow i & \circlearrowright^{g} & \downarrow j \\
\Phi_2^o\,\tau_2 & \longrightarrow & \Phi_2^m\,g_2 \longrightarrow \Phi_2^o\,\upsilon_2
\end{array}$$

$$\begin{array}{ccc}
\Phi_1^o\,(\rho_1 \times \tau_1) & \longrightarrow & \Phi_1^m\,(f_1 \otimes g_1) \longrightarrow \Phi_1^o\,(\sigma_1 \times \upsilon_1) \\
\downarrow h \otimes i & \circlearrowright & \downarrow k \otimes j \\
\Phi_2^o\,(\rho_2 \times \tau_2) & \longrightarrow & \Phi_2^m\,(f_2 \otimes g_2) \longrightarrow \Phi_2^o\,(\sigma_2 \times \upsilon_2)
\end{array}$$

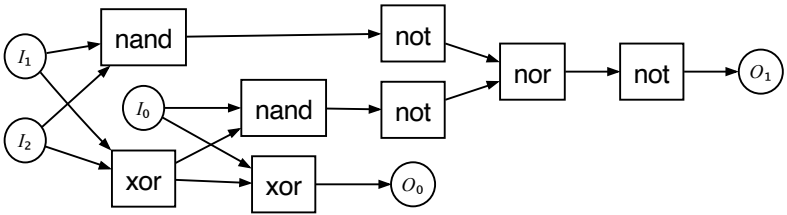
Back to circuits

Example: full adder



Given when input is (digitally) valid, determine when output is valid. *What is when?*

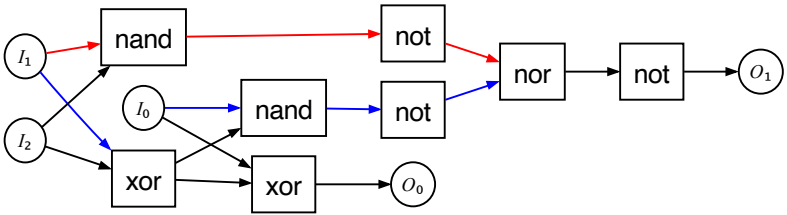
Example: full adder



Given when input is (digitally) valid, determine when output is valid. *What is when?*

- Per-bit timing.
-

Example: full adder



Given when input is (digitally) valid, determine when output is valid. *What is when?*

- Per-bit timing.
- Differing path lengths.

Analog computation

Time & signals:

$\mathbb{T} : \mathsf{Set}$

$\mathbb{T} = \mathbb{Q}$

$\mathbb{S} : \mathsf{Set}$

$\mathbb{S} = \mathbb{T} \to \mathbb{B}$

Analog gates:

$\mathsf{analog}_0 : \mathbb{T}^{\,0} \to (\mathbb{B}^{\,0} \to \mathbb{B}) \to (\mathbb{S}^{\,0} \to \mathbb{S})$

$\mathsf{analog}_0\ \mathsf{tt}\ h\ \mathsf{tt} = \lambda\ t \to h\ \mathsf{tt}$

$\mathsf{analog}_1 : \mathbb{T} \to (\mathbb{B} \to \mathbb{B}) \to (\mathbb{S} \to \mathbb{S})$

$\mathsf{analog}_1\ \delta\ h\ \tilde{x} = \lambda\ t \to h\ (\tilde{x}\ (t - \delta))$

$\mathsf{nand}^{\,\mathrm{A}} : \mathbb{S}^{\,2} \to \mathbb{S}$

$\mathsf{nand}^{\,\mathrm{A}} = \mathsf{analog}_2\ \delta\text{-}\mathsf{nand}\ \mathsf{nand}$

$\mathsf{analog}_2 : \mathbb{T}^{\,2} \to (\mathbb{B}^{\,2} \to \mathbb{B}) \to (\mathbb{S}^{\,2} \to \mathbb{S})$

$\mathsf{analog}_2\ (\delta_1\ ,\ \delta_2)\ h\ (\tilde{x}_1\ ,\ \tilde{x}_2) = \lambda\ t \to h\ (\tilde{x}_1\ (t - \delta_1)\ ,\ \tilde{x}_2\ (t - \delta_2))$

Stable signals

Constrain to stable signals & stability-preserving transformations:

$$\mathsf{stable} : \mathsf{Pred}\ \mathbb{S}\ 0\ell$$
$$\mathsf{stable}\ \tilde{x} = \exists_2\ \lambda\ (\hat{x} : \mathbb{I})\ (x : \mathbb{B}) \to \forall\ \{t : \mathbb{T}\} \to t \in \hat{x} \to \tilde{x}\ t \equiv x$$

Stable signals

Constrain to stable signals & stability-preserving transformations:

$$\mathsf{stable} : \mathsf{Pred}\ \mathbb{S}\ 0_\ell$$
$$\mathsf{stable}\ \tilde{x} = \exists_2\ \lambda\ (\hat{x} : \mathbb{I})\ (x : \mathbb{B}) \to \forall\ \{t : \mathbb{T}\} \to t \in \hat{x} \to \tilde{x}\ t \equiv x$$

Constructive logic, so timely digital circuits *compute* $\hat{x}$ and x.

Form nested pairings: higher-dimensional time intervals.

Uses general construction for constraints and their preservation.

Stable gates

$$\mathsf{stable}{\Rightarrow_0} : \mathbb{T}^{\,0} \to (\mathbb{B}^{\,0} \to \mathbb{B}) \to (\mathbb{B}^{\,0} \Rightarrow \mathbb{B})$$
$$\mathsf{stable}{\Rightarrow_0}\ \delta\ h = \mathsf{mk}{\Rightarrow}\ \{f = \mathsf{analog}_0\ \delta\ h\}\ \lambda\,\{\ \mathsf{tt} \to \delta\ \hat{\S}^0\ \mathsf{tt}\ ,\ h\ \mathsf{tt}\ ,\ \lambda\ \_ \to \mathsf{refl}\ \}$$

$$\mathsf{stable}{\Rightarrow_1} : \mathbb{T} \to (\mathbb{B} \to \mathbb{B}) \to (\mathbb{B} \Rightarrow \mathbb{B})$$
$$\mathsf{stable}{\Rightarrow_1}\ \delta\ h = \mathsf{mk}{\Rightarrow}\ \{f = \mathsf{analog}_1\ \delta\ h\}\ \lambda\ (\hat{x}\ ,\ x\ ,\ P) \to \delta\ \hat{\S}\ \hat{x}\ ,\ h\ x\ ,\ \mathsf{cong}\ h \circ P \circ \in\!\star$$

$$\mathsf{stable}{\Rightarrow_2} : \mathbb{T}^{\,2} \to (\mathbb{B}^{\,2} \to \mathbb{B}) \to (\mathbb{B}^{\,2} \Rightarrow \mathbb{B})$$
$$\mathsf{stable}{\Rightarrow_2}\ \delta\ h = \mathsf{mk}{\Rightarrow}\ \{f = \mathsf{analog}_2\ \delta\ h\}\ \lambda\ ((\hat{x}_1\ ,\ x_1\ ,\ P_1)\ ,\ (\hat{x}_2\ ,\ x_2\ ,\ P_2)) \to$$
$$\delta\ \hat{\S}^2\ (\hat{x}_1\ ,\ \hat{x}_2)\ ,\ h\ (x_1\ ,\ x_2)\ ,\ \mathsf{cong}_2{}'\ h \circ (P_1 \circ \in\!\star \otimes P_2 \circ \in\!\star) \circ \in\!\cap^{\,e}$$

Gate timings and interval lemmas:

$$\_\hat{\S}^0\_ : \mathbb{T}^{\,0} \to \mathbb{I}^{\,0} \to \mathbb{I} \qquad \mathsf{tt}\ \hat{\S}^0\ \mathsf{tt} = \mathsf{U}$$

$$\_\hat{\S}\_ : \mathbb{T} \to \mathbb{I} \to \mathbb{I} \qquad \delta\ \hat{\S}\ \hat{x} = \{\ \delta\ \} \star \hat{x}$$

$$\_\hat{\S}^2\_ : \mathbb{T}^{\,2} \to \mathbb{I}^{\,2} \to \mathbb{I} \qquad (\delta_1\ ,\ \delta_2)\ \hat{\S}^2\ (\hat{x}_1\ ,\ \hat{x}_2) = \delta_1\ \hat{\S}\ \hat{x}_1 \cap \delta_2\ \hat{\S}\ \hat{x}_2$$

$$\in\!\star : t \in \{\ \delta\ \} \star p \to t - \delta \in p \qquad \in\!\cap^{\,e} : \forall\ \{t\} \to t \in p \cap q \to t \in p \times t \in q$$

Stable logic

```
logic : Logic _⇒_
logic = record { false = stable⇒₀ δ-false false
               ; true  = stable⇒₀ δ-false true
               ; not   = stable⇒₁ δ-not   not
               ; nand  = stable⇒₂ δ-nand  nand
               ; nor   = stable⇒₂ δ-nor   nor
               ; xor   = stable⇒₂ δ-xor   xor }
```

The interval semiring and linearity

Note timing operations:

$$\_\hat{\S}^0\_ : \mathbb{T}^{\,0} \to \mathbb{I}^{\,0} \to \mathbb{I} \qquad \_\hat{\S}\_ : \mathbb{T} \to \mathbb{I} \to \mathbb{I} \qquad \_\hat{\S}^2\_ : \mathbb{T}^{\,2} \to \mathbb{I}^{\,2} \to \mathbb{I}$$

$$\mathsf{tt}\ \hat{\S}^0\ \mathsf{tt} = \mathsf{U} \qquad \delta\ \hat{\S}\ \hat{x} = \{\ \delta\ \} \star \hat{x} \qquad (\delta_1\ ,\ \delta_2)\ \hat{\S}^2\ (\hat{x}_1\ ,\ \hat{x}_2) = \delta_1\ \hat{\S}\ \hat{x}_1 \cap \delta_2\ \hat{\S}\ \hat{x}_2$$

U, $\star$, and $\cap$ are semiring operations, and *timing is linear*.

The interval semiring and linearity

Note timing operations:

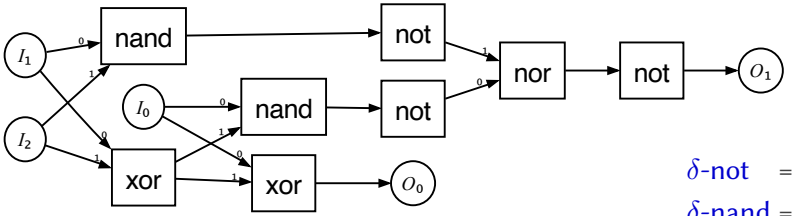
$$\_\hat{\S}^0\_ : \mathbb{T}^{\,0} \to \mathbb{I}^{\,0} \to \mathbb{I} \qquad \_\hat{\S}\_ : \mathbb{T} \to \mathbb{I} \to \mathbb{I} \qquad \_\hat{\S}^2\_ : \mathbb{T}^{\,2} \to \mathbb{I}^{\,2} \to \mathbb{I}$$

$$\mathsf{tt}\ \hat{\S}^0\ \mathsf{tt} = \mathsf{U} \qquad \delta\ \hat{\S}\ \hat{x} = \{\ \delta\ \} \star \hat{x} \qquad (\delta_1\ ,\ \delta_2)\ \hat{\S}^2\ (\hat{x}_1\ ,\ \hat{x}_2) = \delta_1\ \hat{\S}\ \hat{x}_1 \cap \delta_2\ \hat{\S}\ \hat{x}_2$$

U, $\star$, and $\cap$ are semiring operations, and *timing is linear*. Some consequences:

- Represent timing as interval matrices (data).
- Timing as automatic differentiation.
- Statically timed hardware is affine.

Example: full adder



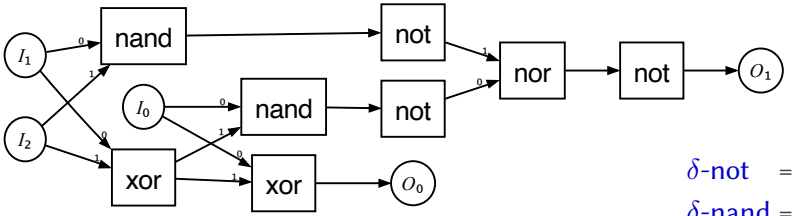
δ-not = 1 / 10

δ-nand = dup (1 / 5)

δ-nor = dup (1 / 5)

δ-xor = dup (1 / 4)

Example: full adder



δ-not = 1 / 10

δ-nand = dup (1 / 5)

δ-nor = dup (1 / 5)

δ-xor = dup (1 / 4)

Timing analysis (nanoseconds):

```
_ : matrix (2 + 1) 2 (timing fa) ≡ ( (((1 // 4 , 1 // 2) , 1 // 2)
                                    , ((3 // 5 , ((17 / 20) ↑ , (3 / 5) ↓)) , ((17 / 20) ↑ , (3 / 5) ↓)) )
_ = refl
```

Improvements

- Multi-stable signals and cycles/recursion/trace.
- Richer examples.
- Nondeterministic gate timing.
- Full analog (voltages).
- Layout and wires.
- Data-dependent timing.
- Transistors.
- Relationship to automatic differentiation.