

# A Fully Compositional Theory of Digital Circuits

---

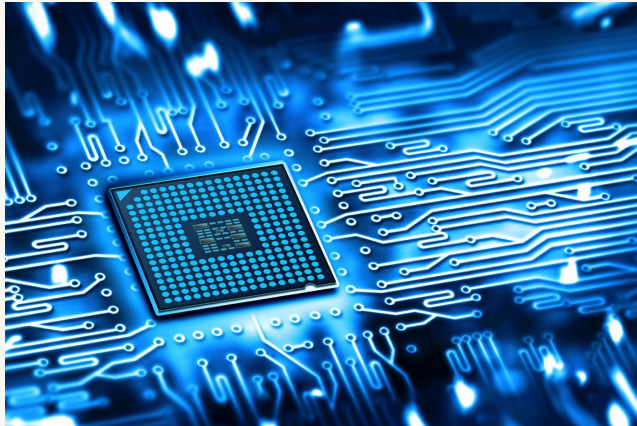
**George Kaye**

University of Birmingham

27 November 2023 – SYNCHRON 2023

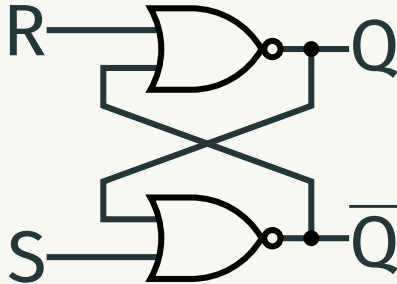
What are we going to be talking about?

## Digital circuits!



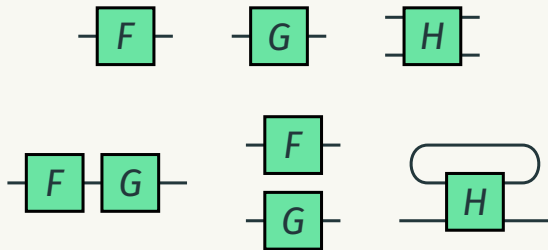
# What are we going to be talking about?

Digital circuits!



## What are we going to be talking about?

We want a **compositional** theory of digital circuits.



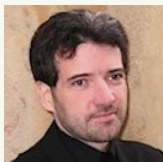
## Why all the pictures?

We want to reason **equationally** about circuits.

Using **string diagrams** removes much of the  
bureacracy

# What came before

Lafont (2003) *'Towards an algebraic theory of Boolean circuits'*



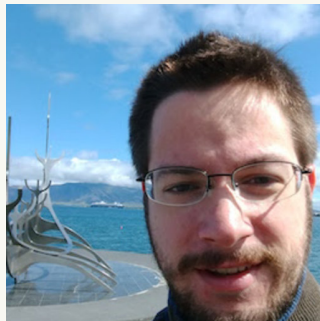
Ghica, Jung, Lopez (2017) *'Diagrammatic semantics for digital circuits'*



## Joint work with...



Dan Ghica  
University of Birmingham



David Sprunger  
Indiana State University

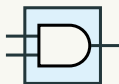
# Syntax

---

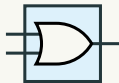


# Combinational circuit components

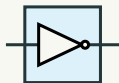
gates



AND gate



OR gate



NOT gate

(co)monoid structure



disconnected



fork



join



stub

categorical structure



identity



symmetry




Light circuits



only contain gates and structure.

# Sequential circuit components

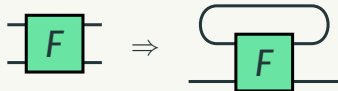
## Values


	false
	true
	short circuit

## Delay

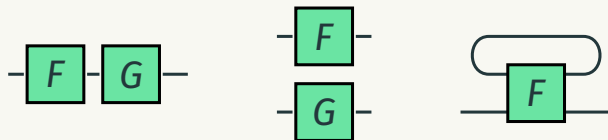


## Feedback

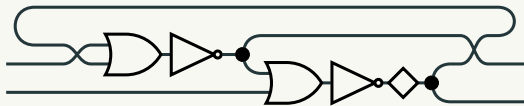
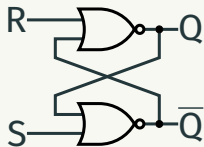


Dark circuits  may contain delay or feedback.

Circuits are morphisms in a **freely generated symmetric traced monoidal category** (STMC).



Need an example?

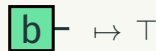
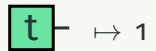
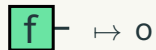
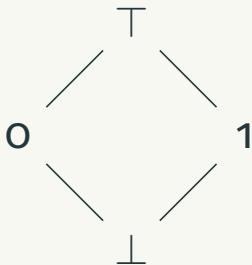


# Semantics

---

# We need some meaning

Values are interpreted in a **lattice**:



# Let's make everything a function



monotone functions

$$\bar{g}: \mathbf{V}^m \rightarrow \mathbf{V}$$



initialise

$$() \mapsto (\perp)$$



copy

$$x \mapsto (x, x)$$



join in the lattice

$$(x, y) \mapsto x \sqcup y$$



discard

$$x \mapsto ()$$

Feedback is interpreted as the **least fixed point**.

How do we model **delay**?

**Streams!**



A **stream**  $\mathbf{V}^\omega$  is an infinite sequence of values.

$$V_0 :: V_1 :: V_2 :: V_3 :: V_4 :: V_5 :: V_6 :: V_7 :: \dots$$

A **stream function**  $\mathbf{V}^\omega \rightarrow \mathbf{V}^\omega$  consumes and produces streams.

$$f(V_0 :: V_1 :: V_2 :: V_3 :: V_4 :: \dots) = W_0 :: W_1 :: W_2 :: W_3 :: W_4 :: \dots$$

## Interpreting the sequential components

$$\boxed{v} - () := v :: \perp :: \perp :: \perp :: \dots$$

$$\boxed{\diamond} - (v_0 :: v_1 :: v_2 :: \dots) := \perp :: v_0 :: v_1 :: v_2 :: \dots$$

Does every circuit correspond to a stream function

$$(\mathbf{V}^m)^\omega \rightarrow (\mathbf{V}^n)^\omega?$$

**No.**

(but this is to be expected!)

## Restricting the stream functions

Circuits are **causal**.

They can only depend **what they've seen so far**.

Circuits are **monotone**.

They are constructed from **monotone functions**.

Is that all? **Not quite...** (but we'll get there)

## Some operations on stream functions

Given a causal stream function  $f: (\mathbf{V}^m)^\omega \rightarrow (\mathbf{V}^n)^\omega$  and an element  $a \in \mathbf{V}^m$ ...

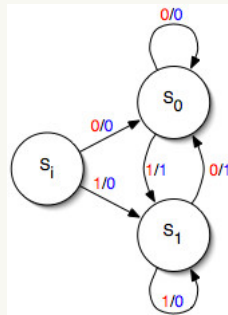
**initial output**  $f[a] \in \mathbf{V}^n$

‘the first thing  $f$  produces given  $a$ ’

**stream derivative**  $f_a \in (\mathbf{V}^m)^\omega \rightarrow (\mathbf{V}^n)^\omega$

‘how  $f$  behaves after seeing  $a$  first’

Hold on, these look familiar...



## Mealy machines!

Stream functions are the *states* in a Mealy machine.

## Circuits have finitely many behaviours

Circuits have a finite number of components.

So there are finite number of states in the Mealy machine.

So the outputs of streams given some input must be **periodic**.

(There are finitely many **stream derivatives**).

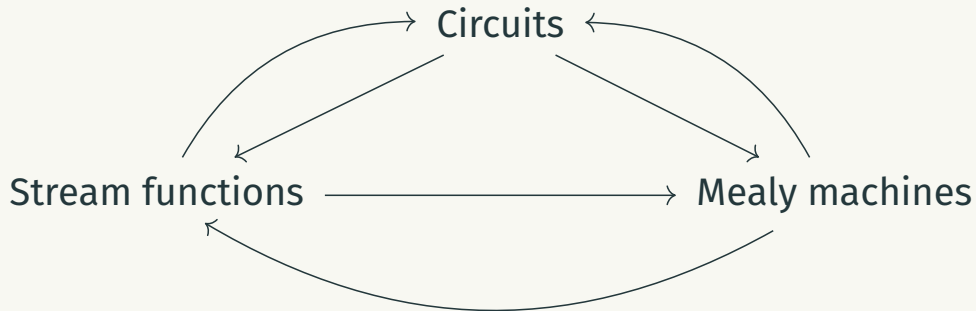
## These are the streams we're looking for

### Theorem

*A stream function is the interpretation of a sequential circuit if and only if it is **causal**, **monotone** and has **finitely many stream derivatives**.*



## The correspondence

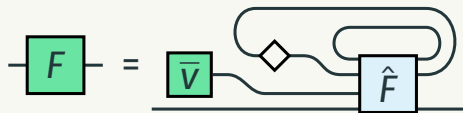


# Operational semantics

---

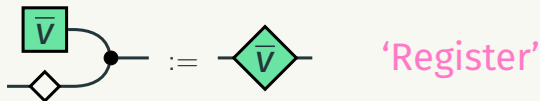
Equational reasoning      **Unstructured pen and paper proofs**






Operational semantics      **Mechanical step-by-step reduction**

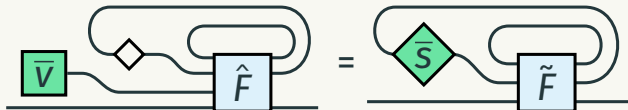


by moving boxes and wires around

Some notation:



By adding equations  =  =  and  = 

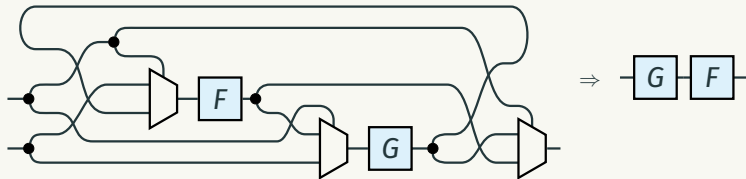


What are we going to do with the non-delay-guarded trace?

## Do we even need it?

In industry, normally circuits must be **delay-guarded**.

But this rules out some **clever** circuits!



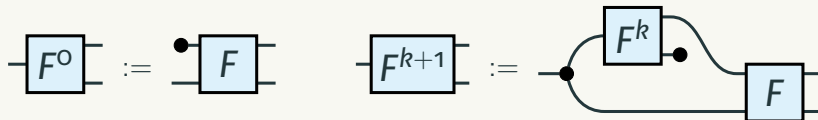
(And also it would be cheating)

# Getting rid of non-delay-guarded feedback

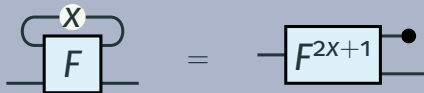
$V$  is a **finite** lattice...

The functions are monotone...

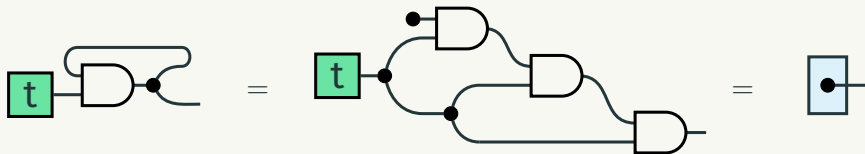
We can compute the **least fixed point** in finite iterations!



Axiom(s)

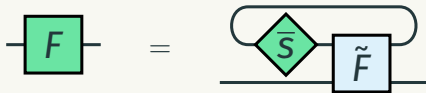


## Getting rid of non-delay-guarded feedback





For **any** circuit



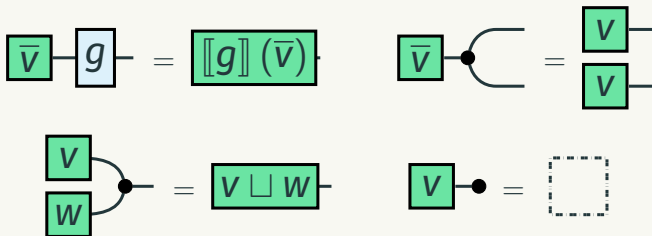
## What is the goal

We want to compute the **outputs** of circuits given some **inputs**



How does a circuit **process** a value?

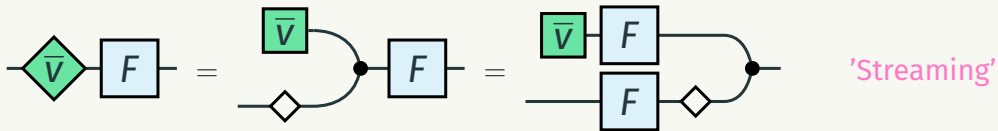
## Reducing values



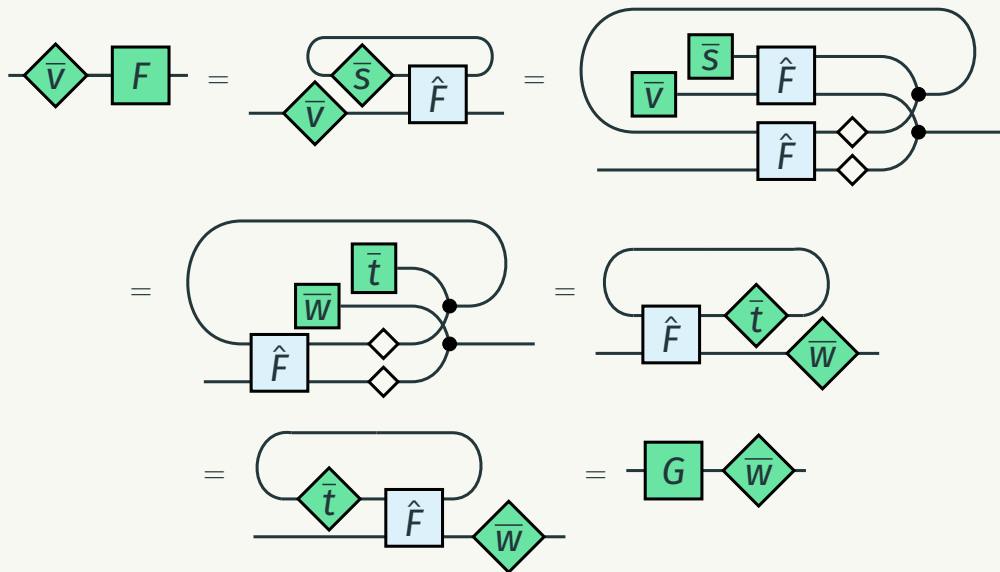
### Lemma

For every  $\text{---} F \text{---}$  there exists  $\bar{W} \text{---}$  s.t.  $\bar{V} \text{---} F \text{---} = \bar{W} \text{---}$ .

What about **delays**?



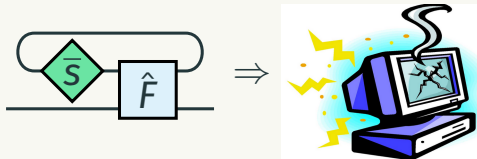
# Catching the jet stream



## From diagrams to graphs

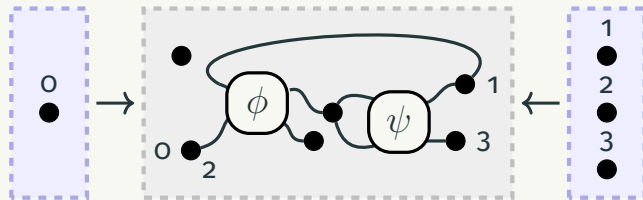
---

## Making it combinatorial

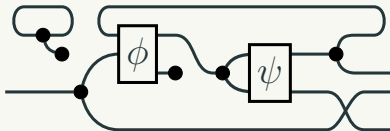


It is **hard** for computers to work with string diagrams...  
...but computers **love** graphs!

## A hyper kind of graph

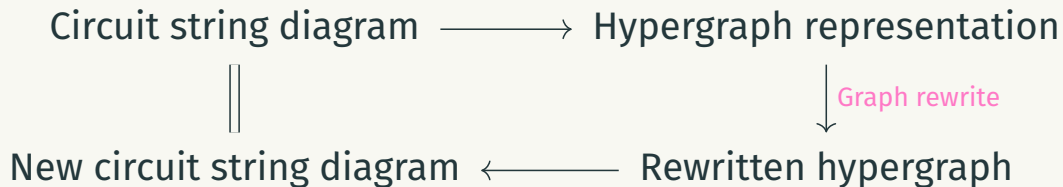


There are correspondences between **certain classes of hypergraphs** and **circuit string diagrams**.





## Using the correspondence



The rewriting framework has been *implemented* in a  
hardware description language.

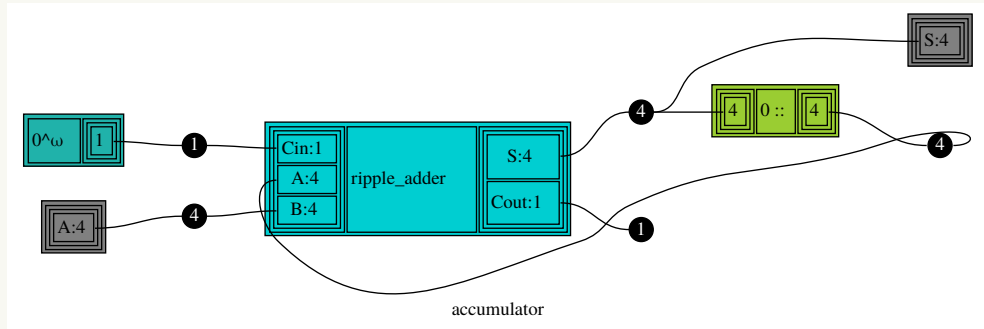


(the language is still in development)

(so I've been warned not to accidentally announce anything)

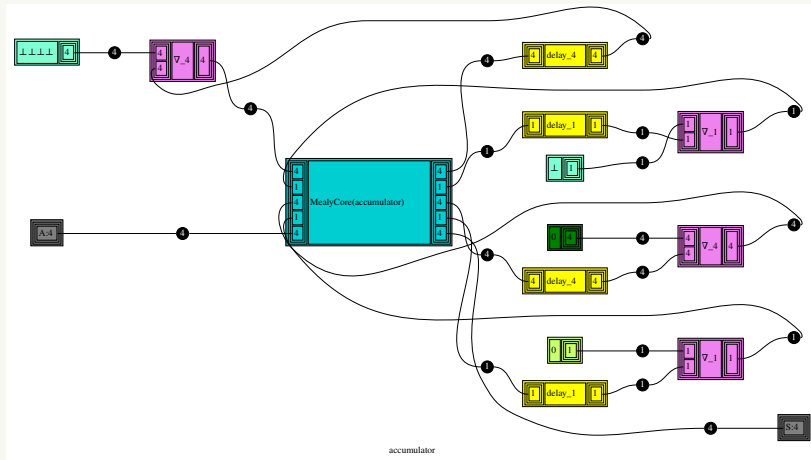
# I can still show you something

Create a circuit...



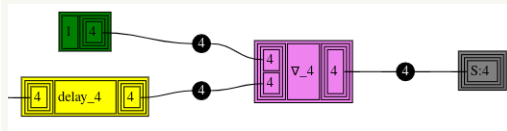
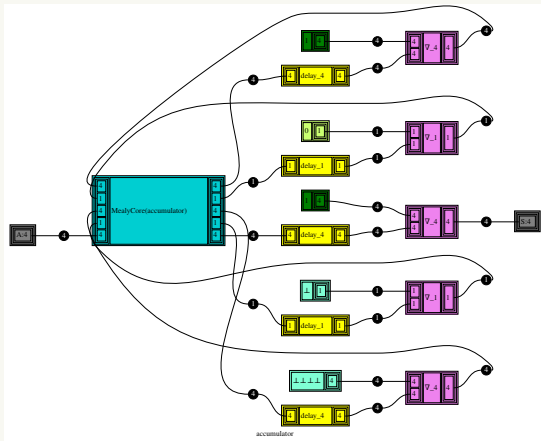
# I can still show you something

The evaluator converts it into Mealy form...



# I can still show you something

...and then evaluates an input.



We have developed a **compositional framework** for digital circuits

We have an **operational semantics** for digital circuits

We can model this using **graph rewrites** on **hypergraphs**

This has been **implemented** in a hardware description language