

Diamondization of HP's memristive flip-flop circuit

Strategies of implementing memristors as second-order devices

Rudolf Kaehr Dr.phil.

Copyright © ThinkArt Lab ISSN 2041-4358

Abstract

The aim of this exercise or first draft is to train the strategy of transforming and diamondizing buffer-like constructions between agents into interactional procedures to avoid unnecessary wastage of resources. This is not supporting the new global megatrend of parsimonious thriftiness but tries to open up a more liberated play of interactionality for nano-electronic devices. Hence, memristive systems are conceived as second-order constructions. A crucial conceptual challenge for such an approach is given by the chance to diamondize HP's memristive flip-flop circuit towards a more interactional and diamondized implementation.

1. Memristive Flip-Flop Device

1.1. Memristor as a first-order device in a flip-flop circuit

1.1.1. Interactionality and memristive systems

Interactivity, interactionality and reflexionality is not conceivable without the involvement of time- and memory-related concepts and constructs. Therefore, a great chance to develop interaction-based computing systems arrives with the discovery/invention of the memristor and memristive systems opened up by these new possibilities of nano-electronic design.

This new approach shall be experienced with the transformation of a channel-based to an interaction-based design of a flip-flop circuit.

How can a memristor construction save the states of a flip-flop device after a power crash and enabling the flip-flop to continue to work at the state it crashed after the power is re-established?

In other words, how can a volatile flip-flop be modified to a *nonvolatile* flip-flop device as the basic unit of a nonvolatile processor design?

This is an attempt to connect CMOS circuits with memristors. It is therefore not

attempted to create processors on the base of memristors alone.

Recall the definition of a master-slave flip-flop:

"A master-slave D flip-flop is created by connecting two gated D latches in series, and inverting the enable input to one of them. It is called master-slave because the second latch in the series only changes in response to a change in the first (master) latch." (Wiki)

A master-slave flip-flop is a *serial* connection of two latches.

The whole approach, to add memristors to CMOS, might be turned into the possibility to program memristive systems connected with CMOS devices to change the behavior of the CMOS systems and not only to save their functioning after a power collapse.

It would be another challenge to construct a flip-flop on the base of memristors only. A first step would be to consider a translation from the material implication plus negativ constant to NAND gates. If the whole construction is based on memristors, a special memristive save-system for the case of power interruptions seems then to become obsolete.

1.1.2. A memristor-based nonvolatile latch circuit

It has to be recognized as a nice 'bargain' in the world of academic papers that IOP SCIENCE is offering HP's text "A memristor-based nonvolatile latch circuit" for free to the interested audience. Finally, there are now more original papers accessible. This will certainly improve my own conceptual work on memristive systems.

Warren Robinett et al, 2010 Nanotechnology 21 235203
<http://iopscience.iop.org/0957-4484/21/23/235203/>

"We demonstrated that a memristor can be used as a memory element in a nonvolatile latch circuit. This circuit could be used to make a 'nonvolatile processor' —a processor for which the state could be saved in one machine cycle—thus enabling computing systems powered by highly erratic intermittent power sources, such as scavenged environmental energy from vibrations or heat differentials.

The TiO₂ memristors used in this experiment, which were fabricated in our laboratory, exhibit endurance of 10³ - 10⁴ write operations, which we believe can be improved by modifications in device geometry, materials, and processing. Other workers, using other material systems, report memristor endurances up to 10⁸ write/erase cycles." (HP)

HP's interpretation of Figure 3.

"A schematic of the nonvolatile latch test circuit that was designed and constructed for this experiment is shown as figure 3, and the circuit itself is shown in figure 4.

"The components of the circuit include a master flip-flop (FF1), a slave flip-flop (FF2), write circuit switches (S1, S2) a read circuit switch (S3), a memristor (M1), a resistor (R1), and a read comparator (C0).

"When the power-down signal (called write-enable WE in figure 3) is received, the state of the master flip-flop Q1 is copied into the memristor (M1) by closing the write-enable switch (S2): this applies a write-voltage pulse to the memristor (figure 4(c)), with the pulse duration controlled by the WE signal, and the pulse voltage selected as one of the two voltages VW0 and VW1, depending on the state Q1 to be saved; this sets the conductance state of the memristor: high conductance (ON) for a saved one, or low conductance (OFF) for a saved zero.

"Later, when the power-up signal (called readable RE in figure 3) is received, the saved state is copied from the memristor to the slave flip-flop by closing the switch S3 and capturing the saved state at input D2 of the slave FF2; then, computation resumes.

"When switch S3 closes, this forms a voltage divider using resistor R1 and memristor M1; the saved state is embodied physically in the conductance state of the memristor, and the voltage divider's output voltage VOUT signals this saved state, which is measured by the voltage comparator C0.

"After power-up is complete, the circuit is in the same state as it was when power-down occurred.

Thus, we created a nonvolatile latch test circuit that demonstrates the saving of a flip-flop state to a memristor, and the restoration of this state from the memristor to a second flip-flop."

Warren Robinett et al, 2010 Nanotechnology 21 235203; for short: (HP)

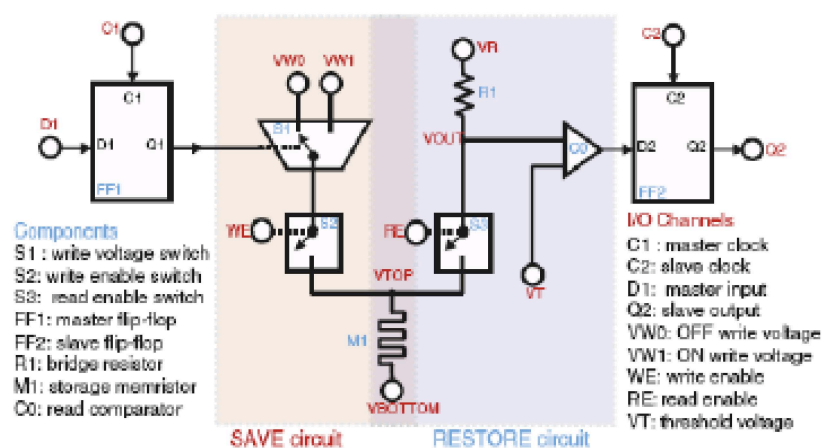


Figure 3.

"Schematic of the circuit used to demonstrate a nonvolatile flip-flop (FF). The circuit stores the state of the master FF into the memristor within one clock cycle of the

power-down (WE) signal and copies that saved state from the memristor to the slave FF upon power-up (RE signal)." (HP)

The memristor as a reliable charity

"The circuit stores the state of the master FF into the memristor within one clock cycle of the power-down (WE) signal and copies that saved state from the memristor to the slave FF upon power-up (RE signal)."

FF1 → (SAVE circuit → Memristor → Restore circuit) → FF2

Memristor as a reliable buffer: Master FF-state → memristor/state → slave FF-state

The aim of this approach is to demonstrate *"that a memristor can be used as a memory element in a nonvolatile latch circuit."* (HP)

The procedure, again

The sequence of signals for a single cycle of the program was as follows :

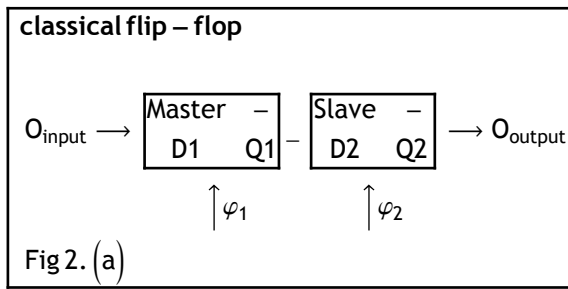
- Power up the circuit.
- Set master flip – flop to state 1.
- Power down the circuit and copy master flip – flop state to memristor.
- Power up the circuit and copy memristor state to slave flip – flop.
- Read and store the voltage outputs of the voltage divider and the slave flip – flop.
- Set master flip – flop to state 0.
- Power down the circuit and copy master flip – flop state to memristor.
- Power up the circuit and copy memristor state to slave flip – flop.
- Read and store the voltage outputs of the voltage divider and the slave flip – flop.

1.2. Memristors as second-order devices

1.2.1. Conceptual analysis

Re-interpretation of the HP memristive flip-flop device construction is focusing on the second-order qualification of memristors and their complementary functionality in respect to classical CMOS devices.

It seems, that HP's circuit is using the memristor in their SAVE and RESTORE circuit like a classical device set between the classical CMOS FF construction albeit with the nice property of memristance.

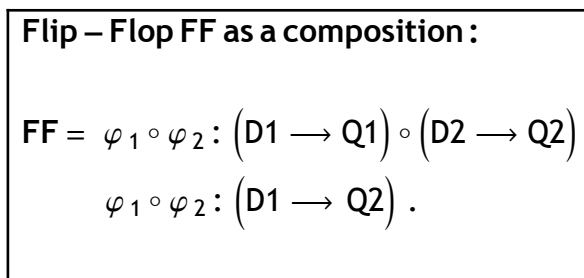


The HP construction shall be transformed into a conceptual device using category – theoretic notions. Hence, the FF appears as a composition of two morphisms defined by $\varphi_1 : D1 \rightarrow Q1$ and $\varphi_2 : D2 \rightarrow Q2$.

$$\varphi_1 : D1 \rightarrow Q1$$

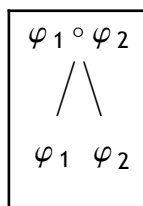
$$\varphi_2 : D2 \rightarrow Q2$$

$$\varphi_1 \circ \varphi_2 \Rightarrow \varphi_3 : O_{input}(D1) \rightarrow O_{output}(Q2).$$

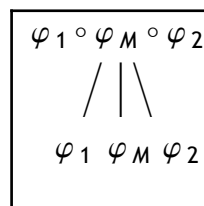


Conceptual graphs

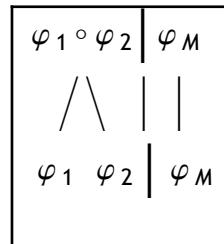
FF –tree



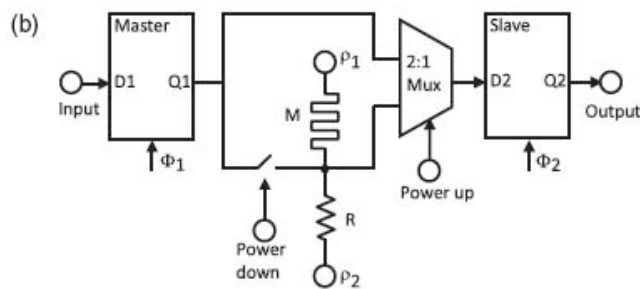
Memristor FF



diamond separation



1.2.2. HP's memristive flip-flop



A 'zoom-in' into the classical flip-flop device FF (Figure 2.(a)) offers a chance for a memristive implementation of the channel (Figure 2.(b)). Hence, the direct channel between Master and Slave gets a more sophisticated definition with a memristive reconfiguration responsible to catch the data after a power crash.

"The nonvolatile latch circuit concept is shown in figure 2. A conventional CMOS master-slave flip-flop circuit (figure 2(a)) is modified to save the master FF state into the memristor when the power-down signal is received. When power is restored, the power-up signal selects the multiplexer (mux) input giving the saved state as the input to the slave FF.

After state restoration and thereafter during normal powered operation, the circuit operates as a conventional CMOS master-slave FF, with all data flowing from the output of the master through the mux to the slave input, and with the memristor unused and sitting dormant.

In the test circuit used for this study, we do not pass data directly from one FF to another since this is a trivial operation; we only test data transfer by means of copying FF state to and from the memristor."

(Robinett et al, HP, Nanotechnology 21 (2010) 235203, p. 3)

Two steps to re-configure Figure 2.

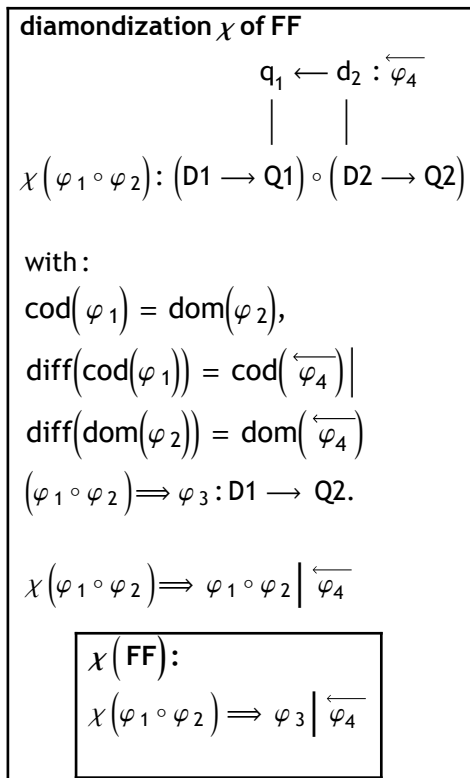
(a) *Conventional* (volatile) design of a CMOS master-slave flip-flop which might be found in a processor register.

(b) HP's *Redesign* of the flip-flop to make it nonvolatile, by providing power-off save/power-on restore, using a memristor *M* to save the state of the flip-flop when power is interrupted. This procedure of reactivating the process is *sequential*, in contrast to the *parallel* processes of the diamond design.

(c) Conceptual *Diamond* re-redesign of HP's re-design of(a).

1.2.3. Diamondization of the classical flip-flop device

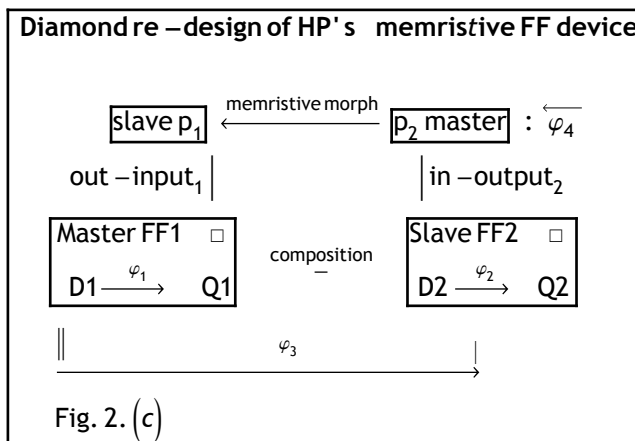
Diamondization is a new technique to 'in-sourcing' the matching conditions of categorical composition (and juxtaposition). Diamondization is uncovering the 'hidden' conditions of a composition in the formalism itself, and is therefore making the conditions of the composition accessible to further transformation or programming. If a composition is representing the rules of a game, diamondization of such a game is offering the possibility to change the rules of the game while running. Hence, diamondization is a second-order approach, unknown to existing mathematical paradigms. Diamondization shouldn't be confused with any hierarchic strategies to reflect with meta-languages on object-languages.



Instead of a 'zoom-in' into the channel and to complement it with an additional device, a diamondization of the master/slave-interactivity is conceptually designed by the 'in-sourcing' of the matching conditions into the formalism (design, device, circuit)of the composition of the FF morphisms between Master- and Slave-FF.

1.2.4. Diamond interpretation of HP's memristive flip-flop device

The wording of HP's construction shall be transformed from a serial strategy into an interventional scenario.



What Q1 delivered D2 by the action φ_3 , i.e. $\varphi_1 \circ \varphi_2$, was monitored and stored before the collapse of φ_3 by p_1 and p_2 of the memristive morphism $\overleftarrow{\varphi_4}$. With the memoryless φ_3 in action, the last values of φ_3 are readable for restauration for Q1 and D2 from $\overleftarrow{\varphi_4}$. Hence, between p_1 , p_2 of $\overleftarrow{\varphi_4}$ and Q1 and D2 of φ_3 , a real read-write relation holds, depending on the on-off-

function of φ_3 .

A re-interpretation of the HP memristive flip-flop device construction is focusing on the second-order qualification of memristors and their complementary functionality in respect to classical CMOS devices.

It seems, that HP's circuit is using the memristor in their SAVE and RESTORE circuit between FF1 and FF2 like a classical device set between the classical CMOS FF construction albeit with the nice property of memristance.

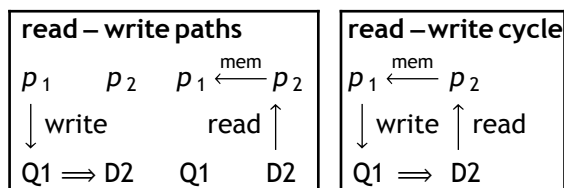
Sequential procedure for HP's solution

FF1 \rightarrow FF2: *serial* composition of two latches FF1 and FF2.

The memristor middle part is added in serial order too.

FF1 \rightarrow (SAVE circuit \rightarrow Memristor \rightarrow Restore circuit) \rightarrow FF2

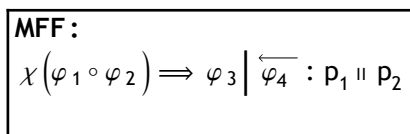
Parallel procedure for the diamond solution

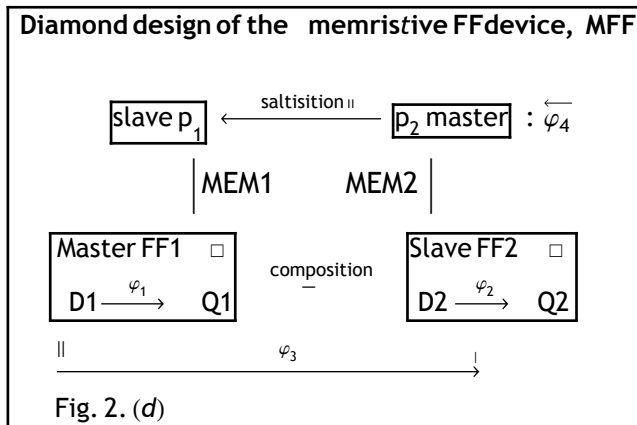


The operation *read* is saving the data of D2 in Memristor as a state of p_2 ; *write* is restoring the data from the Memristor state p_2 to Q1. With the power restored, Q1 then is continuing the interrupted cycle of FF with D2.

1.2.5. Further concretization of the diamond MFF

A further concretization of the construction is necessary to realize a purely interactional definition of a memristive flip-flop MFF. Hence, the out-input₁ and the in-output₂ shall be transformed with the introduction of a memristive implementation as MEM1 and MEM2. As a consequence, the hetero-morphism $\overleftarrow{\varphi}_4$ for the memristive mapping between small master and small slave has to be generalised to a saltatorial jump-operation, “||”, saltisation. Obviously, this construction is dynamizing the relationship between volatile, CMOS, and nonvolatile, MEM, functionalities.





1.3. Translation between HP's construct and the diamond version

1.3.1. Components and I/O Channels (Figure 3)

Components

S1 : write voltage switch

S2: write enable switch :: write, S11, S22 : p1 - Q1, p2 - D2

S3: read enable switch :: S22 read, S11, S22 : p1 - Q1, p2 - D2

FF1: master flip-flop :: ff1, ff2

FF2: slave flip-flop

R1: bridge resistor :: R₁ of MEM1, R₂ of MEM2

M1: storage memristor :: MEM1, MEM2

C0: read comparator

I/O Channels

C1 : master clock

C2: slave clock

D1: master input :: domain of φ_3

Q2: slave output :: codomain of φ_3

VW0: OFF write voltage :: OFF: $\text{cod}(\varphi_3) \neq \text{cod}(\varphi_2)$

VW1: ON write voltage :: ON: $\text{cod}(\varphi_3) = \text{cod}(\varphi_2)$

WE: write enable :: p1 - Q1, p2 - D2

RE: read enable

VT: threshold voltage

1.3.2. Some interpretational combinatorics for MFF

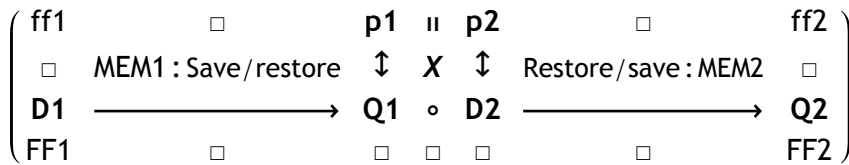
Schemes

The serial HP-scheme: $FF1 \rightarrow (\text{SAVE circuit} \rightarrow \text{Memristor} \rightarrow \text{Restore circuit}) \rightarrow FF2$

becomes a chiasitic memristive scheme:

$(FF1, \text{SAVE}, \text{restore}, \text{Memristor1}) \Leftrightarrow (FF2, \text{save}, \text{Restore}, \text{Memristor2}), \text{ i.e. } (FF1, ff1) \leftrightarrow (FF2, ff2).$

$\overleftarrow{\text{p1}} \parallel \text{p2}$ might function as a saltatorial replacement of a multiplexer Mux 1:2



Procedures

$$1. \left(\begin{array}{ccc} \text{p1} & - & \text{p2} \\ - & X & - \\ \text{Q1} & \Rightarrow & \text{D2} \end{array} \right) :$$

The aim is to save the value of Q1 to continue with D2 by $\text{p2} \leftrightarrow \text{Q1}$ and $\text{p1} \leftrightarrow \text{D2}$. This happens with a cross-wise re-installment of the values of FF. To restore the value of Q1 from FF1 alone wouldn't guarantee the continuation with D2 from FF2 because the matching conditions might possibly not yet been re-established by the re-installment of Q1. Therefore, the cross re-installment of the matching conditions $\text{cod}(\varphi_1) = \text{dom}(\varphi_2)$, i.e. $\text{Q1} \equiv \text{D2}$, shall be re-installed cross-wise by $\text{p2} \leftrightarrow \text{Q1}$ and $\text{p1} \leftrightarrow \text{D2}$. The re-installment of the value of D2 is necessary to fulfil the matching conditions φ_2 with φ_1 , i.e. Q1.

$$2. \left(\begin{array}{ccc} \text{p1} & \leftarrow & \text{p2} \\ - & \nearrow & - \\ \text{Q1} & - & \text{D2} \end{array} \right) : \text{The value of Q1 is stored with p2 and read by p1.}$$

$$3. \left(\begin{array}{ccc} \text{p1} & - & \text{p2} \\ - & \searrow & - \\ \text{Q1} & \rightarrow & \text{D2} \end{array} \right) : \text{The value stored at p1 is read by D2, which is processed by Q1.}$$

$$4. \begin{pmatrix} p1 & \leftarrow & p2 \\ \Downarrow & - & \Downarrow \\ Q1 & \longrightarrow & D2 \end{pmatrix}:$$

The read/write activity between $p1 \Leftrightarrow Q1$ and $p2 \Leftrightarrow Q2$, i.e. between FF1 and FF2, saving and restoring the values of D1 and Q2 as the values p_1 and p_2 .

$$5. \begin{pmatrix} p1 & \leftarrow & p2 \\ \Downarrow & \nearrow & - \\ Q1 & - & D2 \end{pmatrix}:$$

The value of Q1 is stored with p_2 and read by p_1 , accessible to read/write activity between p_1 and Q1.

$$6. \begin{pmatrix} p1 & \leftarrow & p2 \\ - & \nwarrow & \Downarrow \\ Q1 & - & D2 \end{pmatrix}:$$

The value of D2 is stored with p_1 , written by p_2 , accessible to read/write activity between D2 and p_2 .

The read/write activity between D2 and p_2 , $D2 \leftrightarrow p_2$, is restoring the value of p_1 .

$$7. \begin{pmatrix} p1 & \leftarrow & p2 \\ \Downarrow & X & \Downarrow \\ Q1 & \Rightarrow & D2 \end{pmatrix}:$$

The continuation of the cycle $Q1 \Rightarrow D2$ is restored by the interactivity of (5.) and (6.) with the re-installment of the matching conditions of the composition of FF1 and FF2 according to the constellation (1.).

1.3.3. Wordings for interpretations

Therefore, a further concretization of the diamond MFF has to consider more precisely the role of the memristor, i.e. the memristive hetero-morphism of the diamond MFF and its interactivity.

Therefore, a single memristor, as in HP's implementation, might store the data needed as a memory, but will not be able to restore, i.e. to compute, the data needed. Such an interplay of read/write functions seems to be possible with at least two memristors interacting together in the role as a memory function and at once as a computing function.

The read/write-cycles established by the memristive device functions play a double role:

1. p_2 reads (restores) the data of D2 from MEM2 as the results of the composition with FF1, and
2. p_1 writes (saves) the data into Q1 as the results of the interaction with FF2 into FF1 via MEM1.

Both together are defining the functionality of the saltisation $\overleftarrow{\varphi_4}$ between MEM1 of FF2 and MEM1 of FF1.

But the reverse holds too:

1. D2 *writes* (saves) the data p_2 into MEM2 for the interaction with p_1 of MEM1, and
2. Q1 *reads* (re-stores) the data p_1 of MEM1 for the composition with FF2.

Both together are defining the morphism φ_3 between FF1 and FF2 of MFF.

The operation *read* is saving the state of D2 into the memristor MEM2 as the state p_2 ; *write* is restoring the data from the memristor MEM1 as the state p_1 back to Q1. Q1 then is continuing the interrupted cycle with D2.

in reverse

The operation *write* is restoring p_2 in Memristor MEM1 as the state the D2; *read* is restoring the data Q1 from the Memristor MEM2 as the state p_2 to D2. D2 is being continued by the interrupted cycle with Q1.

This double function of active and passive reading and writing between FF and MEM suggests to implement the procedures by two memristors, one for the FF1 and one for the FF2 interaction with MEM, hence MEM1 and MEM2.

Therefore, MEM1 as $(Q1, p_1)$ and MEM2 as $(D2, p_2)$ are getting a technical realization with the help of a memristive interaction between memristor MM1 and memristor MM2 as a concretizations of

$MEM1 = (Q1, p_1)$ and $MEM2 = (D2, p_2)$.

Between MEM1 and MEM2 of MFF, not a composition, but a *saltisation* (jump-operation) holds. Therefore, the full memristive flip-flop MFF seems to be concretized by the *interactivity* of categorical and saltatorial devices of the diamond MFF. Both, categorical composition and saltatorial saltisation are complementary and antidromically interactive.

2. Steps towards concretizations

2.1. Concretized formula of MFF

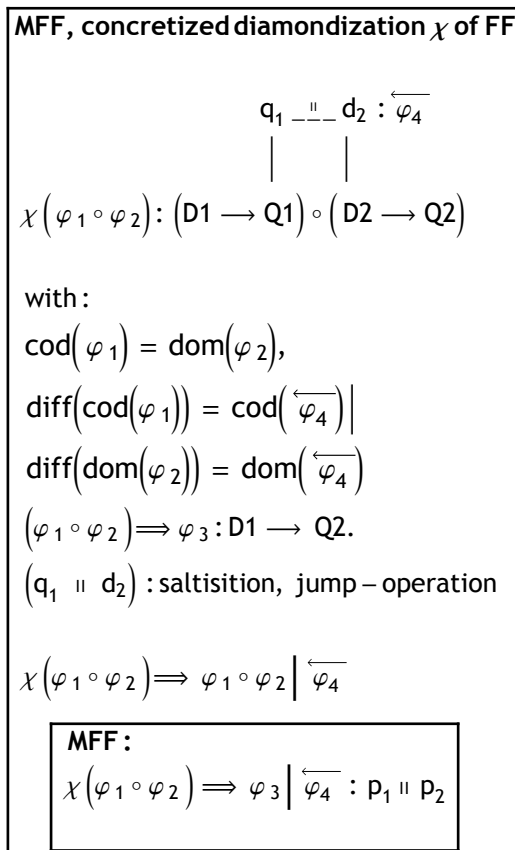
$$\begin{pmatrix} - & FF_2 & - \\ FF_3 & - & MM_4 \\ - & FF_1 & - \end{pmatrix} \Rightarrow \begin{pmatrix} - & FF_2/MM_{FF_1} & - \\ FF_3 & - & MM_4 \\ - & FF_1/MM_{FF_2} & - \end{pmatrix}$$

category = (FF_3, FF_1, FF_2)

saltatory = $(MM_4, MM_{FF_1}, MM_{FF_2})$

diamond = $(\text{category}, \text{saltatory})$

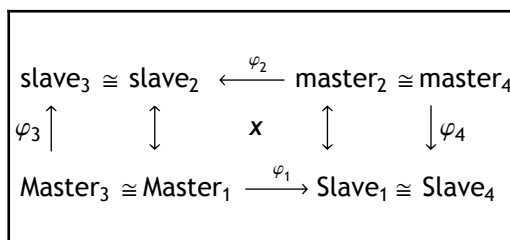
Null



2.2. Conceptual diamond modeling

Diagram of the diamond master – slave relationality

The full diamond scheme for the master – slave relation :



Types of relationships of the master – slave relationality

Null

Diamond relationality

Order relations

Master $\xrightarrow{\varphi_1}$ Slave ,
 master $\xrightarrow{\varphi_2}$ slave,
 Master $\xrightarrow{\varphi_3}$ slave,
 master $\xrightarrow{\varphi_4}$ Slave.

Exchange relations

Master \leftrightarrow slave,
 master \leftrightarrow Slave.

Coincidence relations

Master - master,
 Slave - slave.

Complementarity

φ_3 compl φ_4
 $(\text{Master}_3 \xrightarrow{\varphi_3} \text{slave}_3) \text{ compl } (\text{Slave}_4 \xleftarrow{\varphi_4} \text{master}_4).$

Duality

Master dual Slave,
 master dual slave ,
 $(\text{Master}_1 \xrightarrow{\varphi_1} \text{Slave}_1) \text{ dual } (\text{Slave}_1 \xrightarrow{\varphi_1} \text{Master}_1),$
 $(\text{master}_1 \xrightarrow{\varphi_2} \text{slave}_1) \text{ dual } (\text{slave}_1 \xrightarrow{\varphi_2} \text{master}_1).$

2.3. Interchangeability for MFF

$$\left(\begin{array}{ccc|ccc} - & \text{FF}_2 & - & - & \begin{pmatrix} D_2 \\ Q_2 \end{pmatrix} & - \\ \text{FF}_3 & - & - & \begin{pmatrix} D_3 \\ Q_3 \end{pmatrix} & - & \begin{pmatrix} D_4 \\ Q_4 \end{pmatrix} \\ - & \text{FF}_1 & - & - & \begin{pmatrix} D_1 \\ Q_1 \end{pmatrix} & - \end{array} \right) = \left(\begin{array}{ccc} - & \begin{pmatrix} D_2 \\ Q_2 \end{pmatrix} & - \\ \begin{pmatrix} D_3 \\ Q_3 \end{pmatrix} & - & \begin{pmatrix} D_4 \\ Q_4 \end{pmatrix} \\ - & \begin{pmatrix} D_1 \\ Q_1 \end{pmatrix} & - \end{array} \right)$$

A full interchangeability between the categorical and the saltatorial flip-flops is given by the interchangeability formulas for 3 domains and one environment. This formula is not only considering operations of composition and separation, but of mediation, cross-interchange and saltisition too.

Diamond MFF with 3 layers, 1 environment
 $m = 3, n = 2, \text{env} = 1$

$$\left(\begin{array}{ccc|c} - & \text{FF}_1 & - & \\ \text{FF}_3 & - & & \text{MM}_4 \\ - & \text{FF}_2 & - & \end{array} \right), \left[\begin{array}{ccc|c} D_1 & D_2 & D_3 & D_4 \\ Q_1 & Q_2 & Q_3 & Q_4 \end{array} \right]:$$

$$\left(\begin{array}{c} \left(\begin{array}{c} D_1 \\ \blacksquare 1.2 \times .0 \times .0 \\ D_2 \\ \text{II} 1.2 \times .3 \times .4 \\ D_3 | D_4 \end{array} \right) \left(\begin{array}{c} \circ 1.2 \times .3 \times .0 \\ \diamond 1.2 \times .0 \times .0 \\ \text{II} 0.0 \times .0 \times .4 \end{array} \right) \left(\begin{array}{c} Q_1 \\ \blacksquare 1.2 \times .0 \times .0 \\ Q_2 \\ \text{II} 1.2 \times .3 \times .4 \\ Q_3 | Q_4 \end{array} \right) \right) =$$

$$\left(\begin{array}{c} \left(\begin{array}{c} \left(D_1 \circ_{1.0 \times .0 \times .0} Q_1 \right) \\ \blacksquare 1.2 \times .0 \times .0 \\ \left(D_2 \circ_{0.2 \times .0 \times .0} Q_2 \right) \\ \text{II} 1.2 \times .3 \times .4 \\ \left(D_3 \circ_{0.0 \times .3 \times .0} Q_3 \right) | \left(Q_4 \text{II}_{0.0 \times .0 \times .4} D_4 \right) \end{array} \right) \right)$$

◦ : composition, II : saltisition, II : mediation,
 ◇ : cross – interchange, ■ ≡ (II ◇)

2.4. Misleading modeling

A well known misleading approach is using the concept of inverse morphisms instead of complementary hetero-morphisms of diamond category theory.

The inverse morphism $\overleftarrow{\varphi}_3$ is defined by the input/output values of master FF1 and slave FF2, i.e. D1 and Q2. It sounds intuitively plausible to take those values of the broken cycle as the values to be stored, therefore saved, by the memristive device.

This is a first-order strategy, which is simply repeating the FF definition of the master and slave FF.

And it is not working therefore, because it is just this first-order construct with its results in φ_3 , that is broken. Without escape, the inverse morphism of φ_3 , $\overleftarrow{\varphi}_3$, then is broken too. That is, the inverse of a broken cycle remains a broken cycle albeit in reverse order. That is, $\varphi_3 \circ \overleftarrow{\varphi}_3 = \text{id}_{(D1, p_1)}$.

2.5. Steps towards generalizations

Generalizations are appearing naturally. One is the classical generalization as a *serial* or *parallel* combination of the flip-flop FF and FFM. This happens in a common domain, called ‘*contexture*’. Hence it is mono-contextural. A further generalization is achieved with the introduction of different contexts (sorts) in the mono-contexture. This constellation offers the construction of *multi-layered* crossbar systems.

A very different generalization is opened up by the introduction of different but mediated contextures. Such a *poly-contexture* contains in each contexture the possibility of *multi-layered* crossbar constructions based on multi-contexts, like many-sorted logics. Crossbar systems disseminated over different contextures are leading to the construction of *poly-layered* crossbar systems implying polycontextural logics.

A simple but crucial difference between multi-layered and poly-layered systems is that the composition of the first is additive and associative, while the second is not associative but super-additive and tabular.

Poly-layered memristive systems are realizing the principle of localization for operators. In mono-contextural systems, localization is omitted. Each operator takes place at its locus but all loci are inseparably the same, i.e. they belong to one and only one universe of objects, events, etc.

Operators in poly-contextural systems are qualified by their *place-designator* (Gunther). They take place at their locus but their loci are kenomically different.

R. Kaehr, Memristics: Beyond Memristor Crossbar Systems. Strategies for simplified polycontextural crossbar constructions for memristive computation

3. Discussion and interpretations of the diamond approach

Interpretations

What has to be saved by the memristive device are in fact not the first-order data of FF but the conditions of the possibility of the data, i.e. the data of the matching condition of the composition of the master and the slave flip-flop morphisms (processes). From those conditions, as second-order constructs, the first-order data might be reconstructed on the base of the second-order data saved by the memristive device.

In other words, the history saved by the memristor are not the primary data but the data of the history. Historical data are data of data. Those second-order data might then be used to continue processing on the first-order level of the flip-flop.

As a metaphor, the data of an observer of a data processing system are not the data of the observed system. But such observer-depending data of second-order might be given 'back' to the observed, i.e. first-order system to continue its game. Hence, the memristor is playing the game of an observer which is lending or giving away his data to the observed system.

The memristive system is primarily storing the *rules* of the observed game and only secondarily the data involved.

Slogans

If the "Big Masters"(Master/Slave)fail, the wee masters (master/slave) are in charge, delivering the(ir) carried (stored) information, collected by permanent second-order observations, about the last cycle(s) of the big game. Their role, thus, plays in inverse temporal order (history) and on a second-order level in the tectonics of the system in respect to the big masters. The big masters are playing in CMOS, the wee masters, complementary, in memristors.

The play of the master and the slave as such is, if realized, represented by the compositional play, i.e. in the third system as the composition of system1, FF1, and system2, FF2.

If the play fails, the wee masters are still in charge because they are representing with their memristive capability the history of the ended game represented in the matching conditions of the big game. Hence, if the big masters enters the game again by the power renewed, the wee masters are offering their data to continue at the same level, where their game got interrupted. This is possible by the interplay of the volatile CMOS flip-flop FF and the nonvolatile memristive flip-flop ff functionalities.

Hence again, what is the crucial difference of the proposed sketch for an interactional approach to the buffer-like implementation of HP's circuit?

Diamondization

Diamondization is reflecting the matching conditions (MC) of the composition of the flip-flop construction. Without the MCs, the construction is not working. In a technical model, if the power, which obviously is part of the MCs too, breaks down for the first-order device, the matching conditions as such remains and are represented by their last status, i.e. stored, in the complementary mapping of the second-order level, which is technically realized by a memristive element.

A distinction of “activated” and “non-activated” first- and second-order levels of permanently installed devices, CMOS and memristive, are in order.

This interactional modeling tries to avoid the disadvantages of a channel modeling of the interactions between Master FF and Slave FF by introducing their *double* role in an interaction, i.e. as ‘big’ Masters and ‘wee’ masters and respectively as ‘big’ Slaves and ‘wee’ slaves.

In other words, if an interaction shall happen between two agents, then both are simultaneously in an active and in a passive role.

Master-slave

The Master is able to act on a Slave only if the Slave is enabling this approach. Thus, as an enabler, the Slave acts as an active master and the Master becomes a slave.

Slave-master

The other way round, the Slave is able to be addressed by a Master as a Slave only if the Slave is accepting this approach to be addressed. Thus, as an enabler to be addressed, the slave acts actively as a Master.

Both turns are pre-installed by the designer of a classical FF and are not realized by the interaction of the device.

Critics

HP’s construction is using the flip-flop channel as a memristive *buffer*, and is not yet exploiting the possibilities of the *interactivity* of the master/slave relationship by the involvement of the memristor.

The concepts of complementarity, simultaneity and antidromicity are not yet used in the construction of HP’s memristive FF device.

Nevertheless, it is the merit of HP’s research team to have opened up unforeseen possibilities for new developments in computing in the large, in hardware, software and architectonics of computing systems.