Reversible Delay-Insensitive Distributed Memory Modules

Daniel Morrison & Irek Ulidowski

University of Leicester

April 2013

Overview

Outline

Reversibility

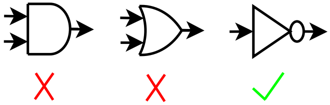
In reversible operations, outputs uniquely determine inputs (bijection). No information loss.

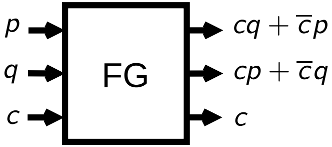
Information loss = energy loss

Typical circuit elements (AND, OR, XOR) other than NOT, are irreversible.

Reversible logic elements were developed by Fredkin and Toffoli.

Fredkin and Toffoli gates are universal for reversible computation.





Asynchronous circuits

Networks of modules connected by wires. Decentralised operation.

Different classes of timing-assumptions:

- Self-timed - Timing assumptions are needed like in synchronous systems. Enforced locally between modules.
- Speed-independent - Operates correctly regardless of arbitrary delays within modules.
- Delay-insensitive - Operates correctly regardless of arbitrary delays within modules and wires.

Full turing-complete delay-insensitivity is not possible in CMOS.

Advantages of asynchronous circuits

Only elements doing useful things are active, leading to lower energy use.

Stronger resistance to environmental variations such as operating temperature, voltage.

Higher reusability of components.

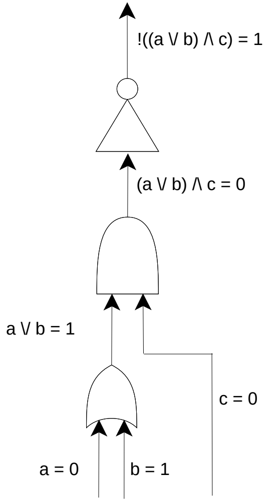
We study the combination of reversibility and asynchrony.

Outline

Synchronous (CMOS) model

A collection of real-time values across a circuit.

The circuit represents a boolean formula.



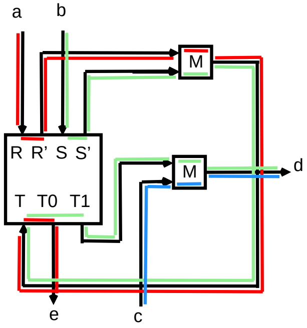
DI model

Signal based, rather than value based like CMOS.

We view a circuit as a set of paths through which signals flow.

A study in causality. Inputs are seen to cause outputs.

The circuit represents a series of events, making it natural for process algebra.



$a.\overline{e} + b.\overline{d} + c.\overline{d}$

Previous Work

Original model formulated by Keller. Standard universal (in terms of arbitrary specifications) sets were given.

Minimal universal sets developed by Patra and Fussell.

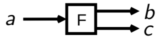
Both reversible and irreversible DI modules exist.

Common elements include Fork, Merge, Join, Select.

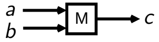
Reversible computation-universal elements such as Rotary Element (Morita) and Reading Toggle & Inverse Reading Toggle (Lee, Peper, Adachi, Morita).

Common modules

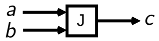
Fork:
$F = (a, \overline{b} | \overline{c}).F$



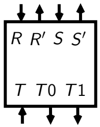
Merge:
$M = (a, \overline{c}).M + (b, \overline{c}).M$



Join:
$J = (a|b, \overline{c}).J$



Select:
$S_0 = (S, \overline{S'}).S_1 + (R, \overline{R'}).S_0 + (T, \overline{T0}).S_0$
$S_1 = (S, \overline{S'}).S_1 + (R, \overline{R'}).S_0 + (T, \overline{T1}).S_1$



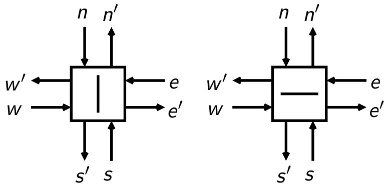
Rotary Element (RE)

$$\begin{aligned} V &= (n,\overline{s'}).V + (s,\overline{n'}).V + \\ &\quad (w,\overline{s'}).H + (e,\overline{n'}).H \\ H &= (n,\overline{w'}).V + (s,\overline{e'}).V + \\ &\quad (w,\overline{e'}).H + (e,\overline{w'}).H \end{aligned}$$

RE is a serial module, so only one signal at a time.

Can be used to build reversible Turing-machines, so RE is reversible Turing-complete.

RE is its own functional inverse - if we reverse its transitions, it behaves as RE.



Outline

Distributed Memory module (DM)

$$S_0 = (q,\overline{0}).S_0 + (p,\overline{1}).S_0 + (r,\overline{s}).S_1 + (c,\overline{s}).S_{\mathrm{a}}$$
$$S_{\mathrm{a}} = (r,\overline{a}).S_1$$
$$S_1 = (q,\overline{1}).S_1 + (p,\overline{0}).S_1 + (r,\overline{s}).S_0 + (c,\overline{s}).S_{\mathrm{b}}$$
$$S_{\mathrm{b}} = (r,\overline{a}).S_0$$

Serial module, holds the value 0 or 1.

S_0, S_1 are steady states; S_{a}, S_{b} are processing states.

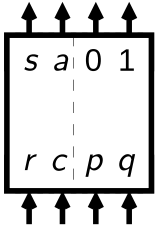
Functionality separated into query (right side) and modification (left side).

Both query (q) and inverse query (p).

r causes a toggle followed by s.

c triggers global toggle.

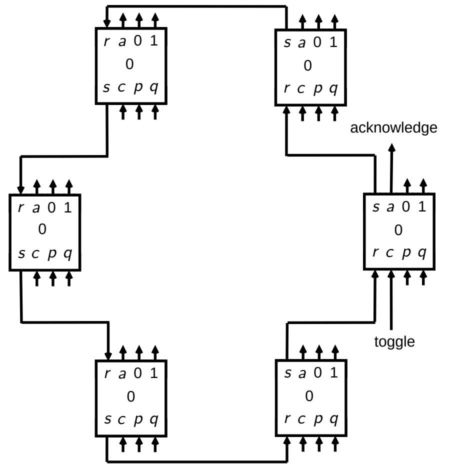
DM is reversible and is its own inverse.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

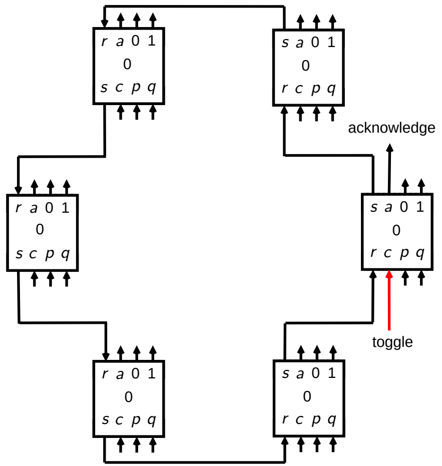
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

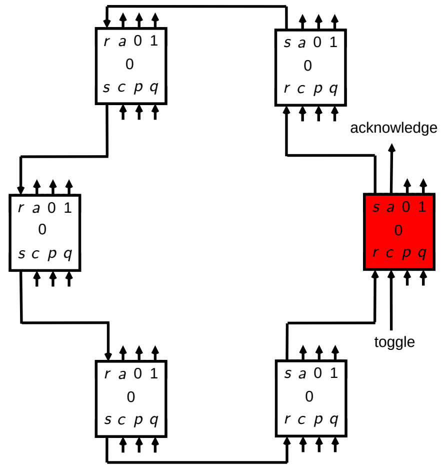
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

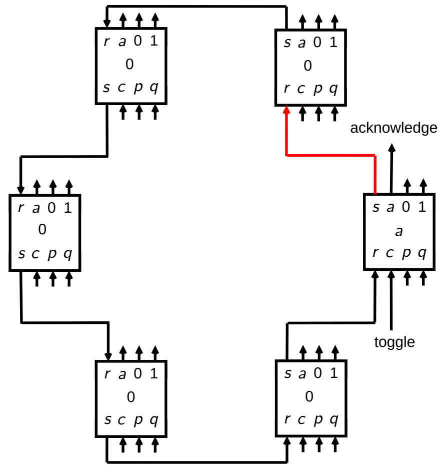
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

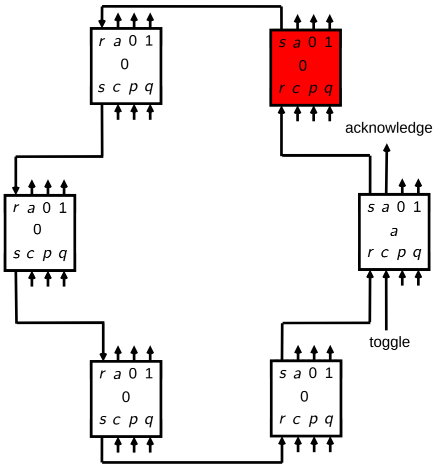
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

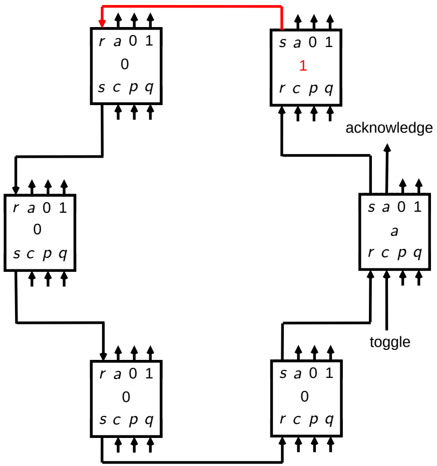
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

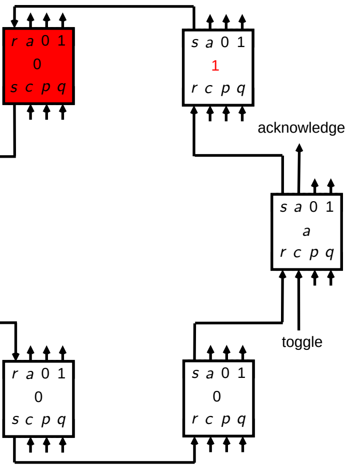
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

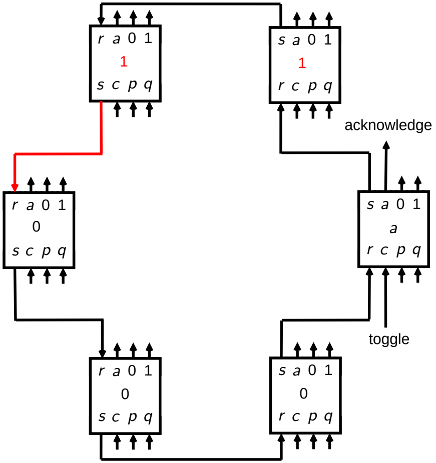
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

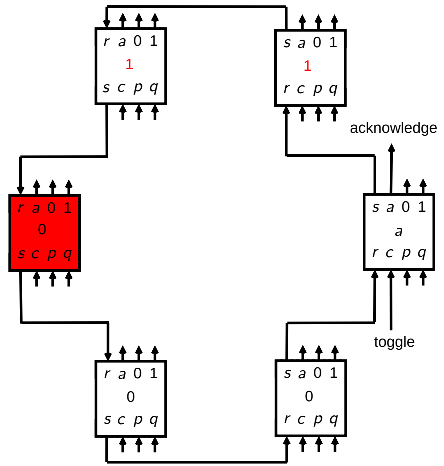
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

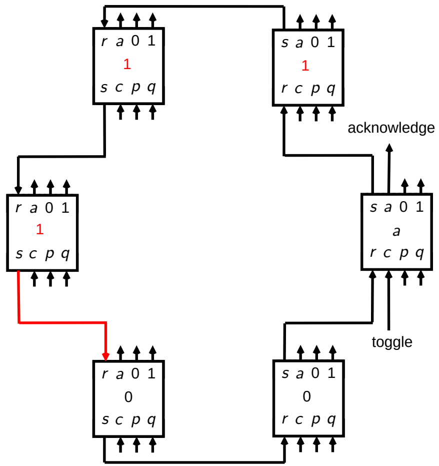
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

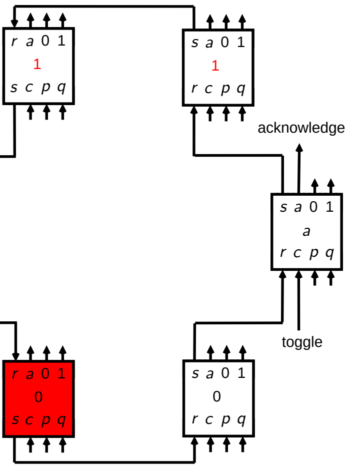
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

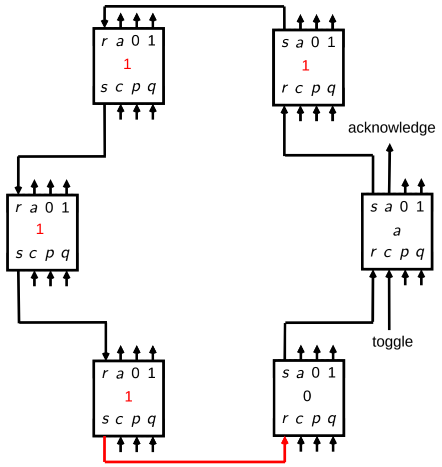
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

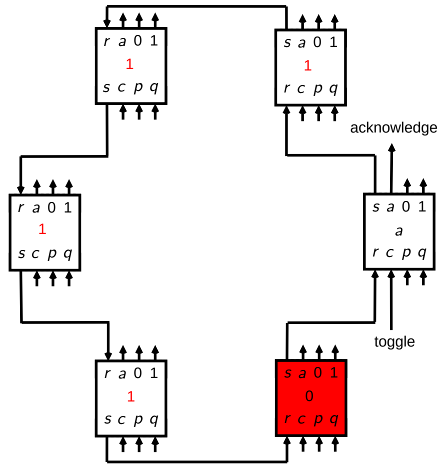
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

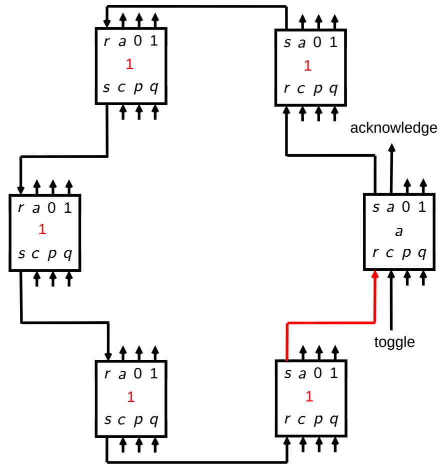
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

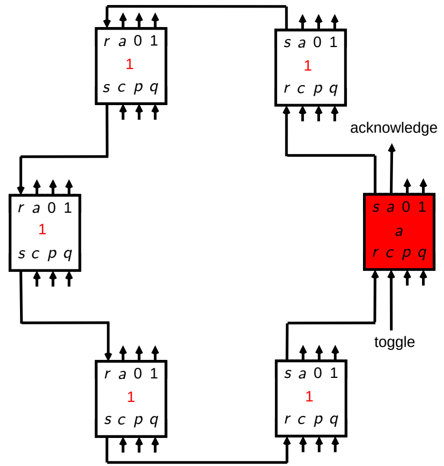
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

Information is naturally mirrored and updated across a network of DM modules.

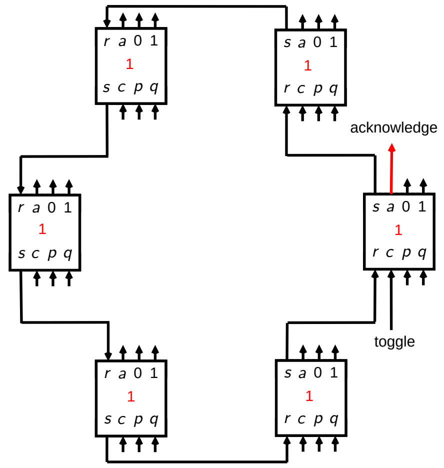
Original module which causes the toggle eventually receives an acknowledge.



Ring of DMs

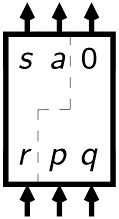
Information is naturally mirrored and updated across a network of DM modules.

Original module which causes the toggle eventually receives an acknowledge.



Reduced DM module (RDM)

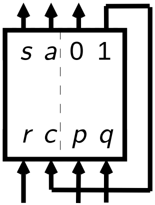
$$S_0 = (q,\overline{0}).S_0 + (p,\overline{s}).S_\mathrm{a} + (r,\overline{s}).S_1$$
$$S_\mathrm{a} = (r,\overline{a}).S_1$$
$$S_1 = (q,\overline{s}).S_\mathrm{b} + (p,\overline{0}).S_1 + (r,\overline{s}).S_0$$
$$S_\mathrm{b} = (r,\overline{a}).S_0$$



RDM is a DM with loop on $1 \to c$.

Toggles the memory with q in state S_1, with p in S_0.

RDM is computationally as powerful as DM. Need five RDMs to realise one DM.



Outline

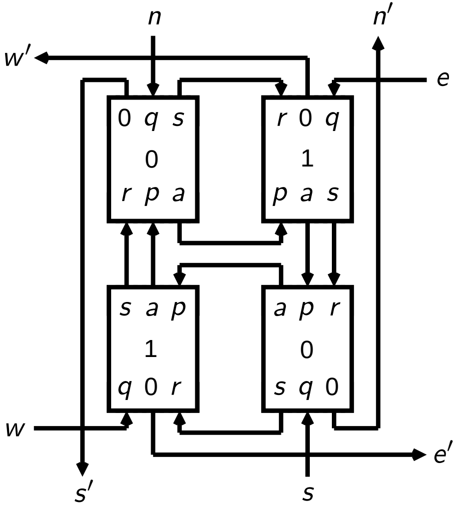
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



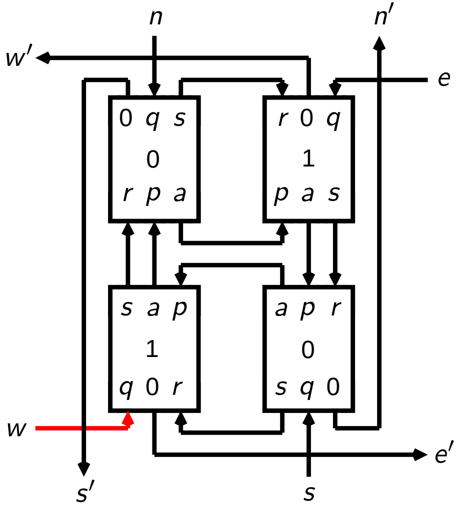
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



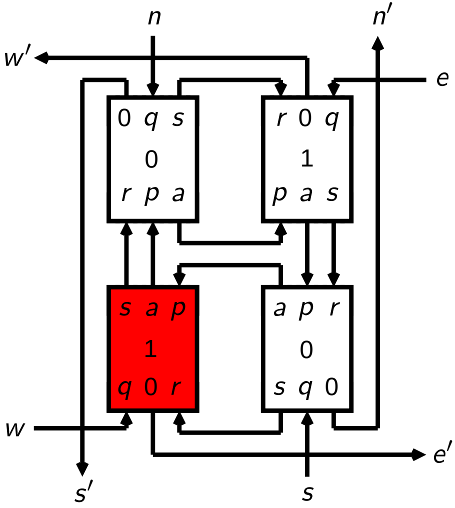
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



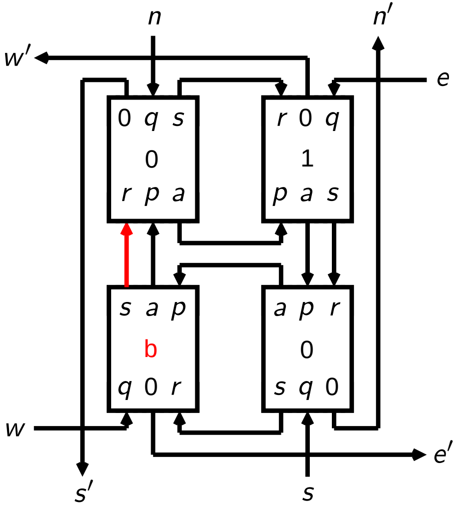
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



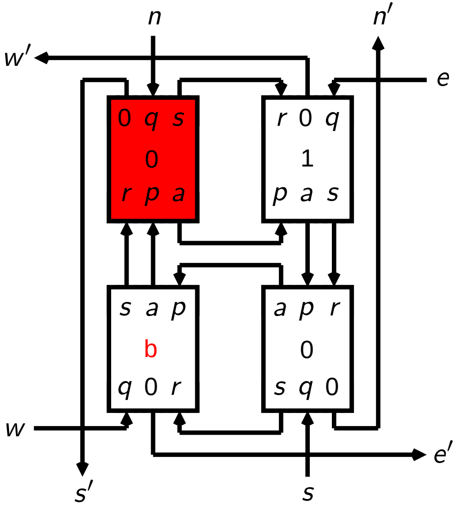
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



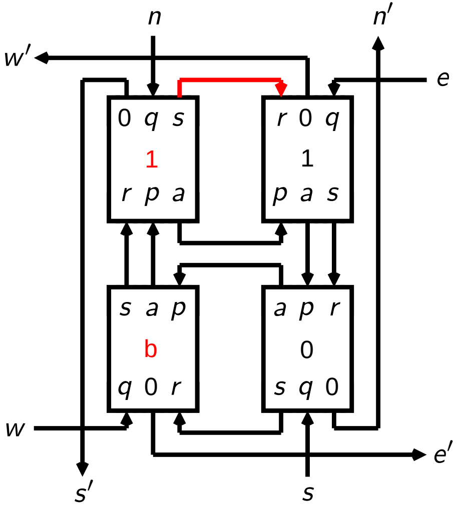
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



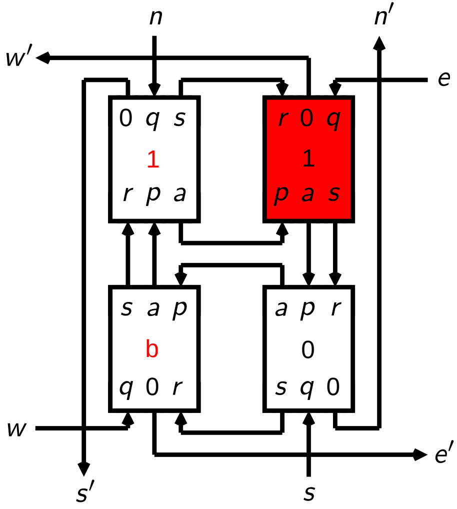
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



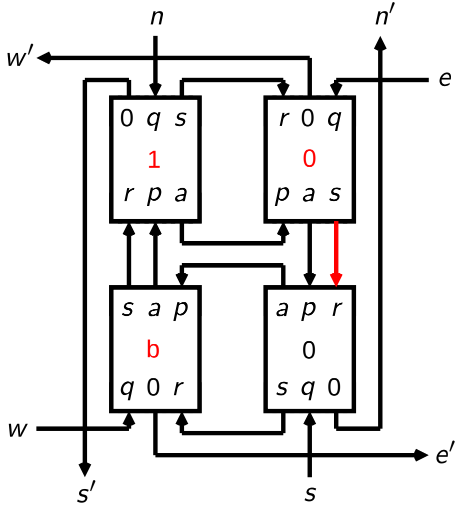
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



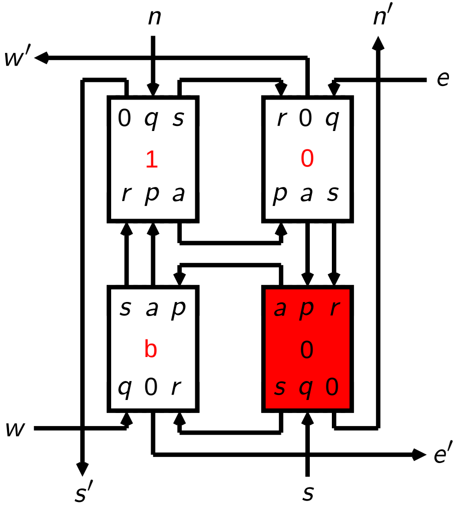
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



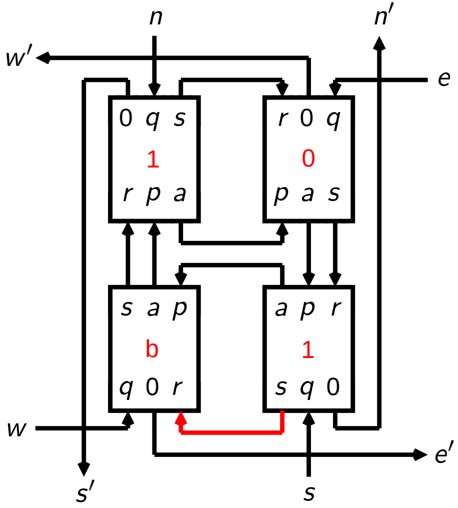
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



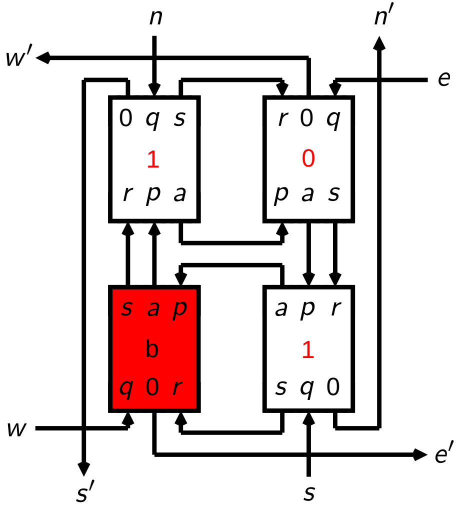
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



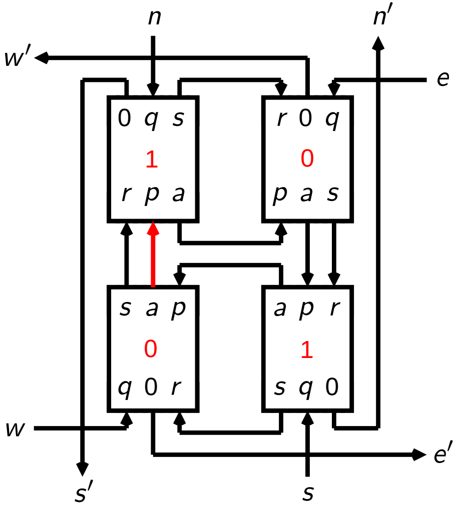
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



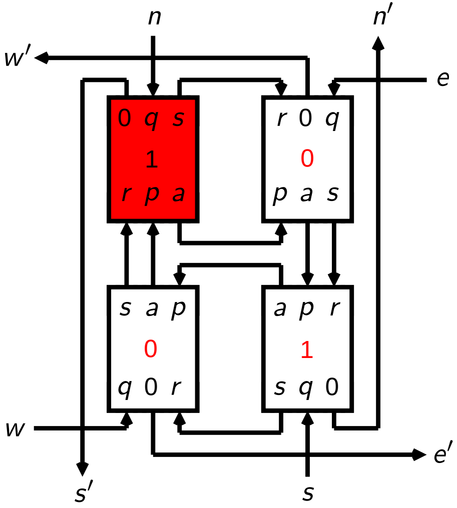
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

Since RE is computation-universal, RDM and DM are also computation-universal.



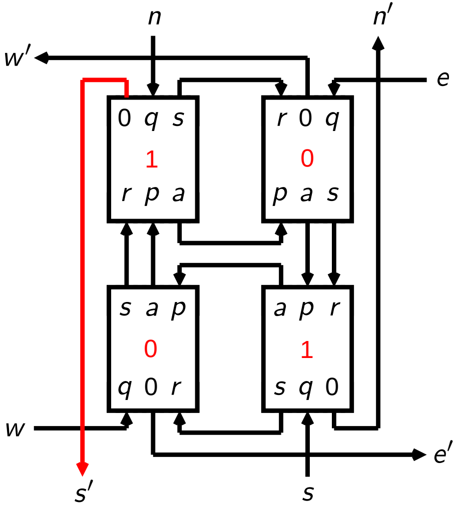
Realisation of RE using RDMs

We need a ring of four RDMs.

Signals pass through or cause a toggle and distributed update.

Maintains reversibility unlike some constructions which use Merge.

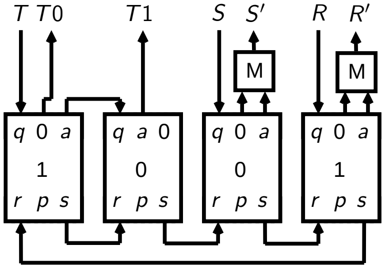
Since RE is computation-universal, RDM and DM are also computation-universal.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

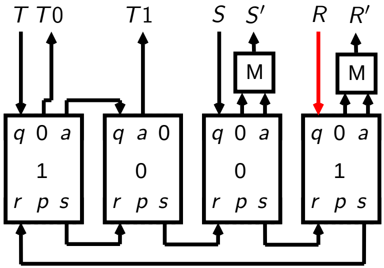
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

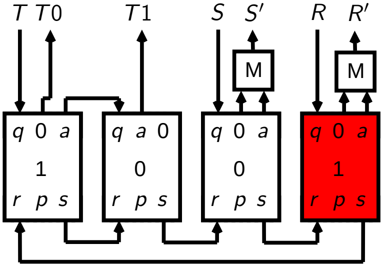
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

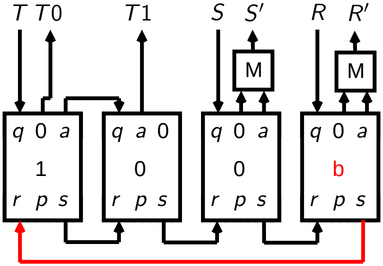
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

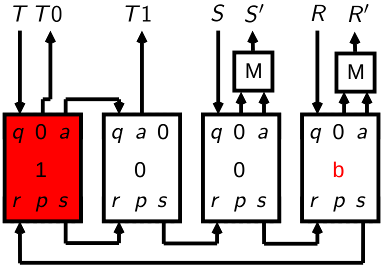
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

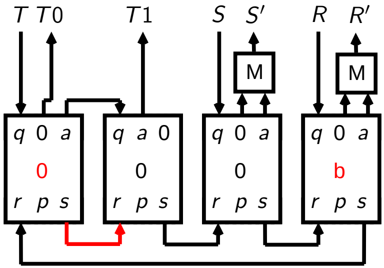
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

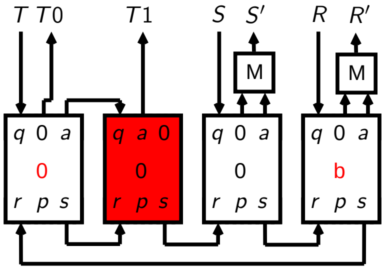
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

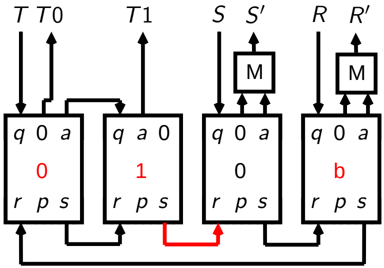
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

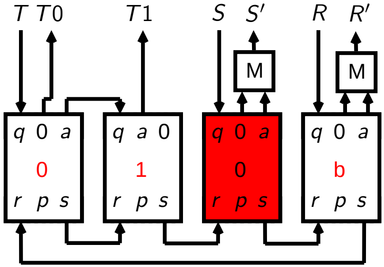
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

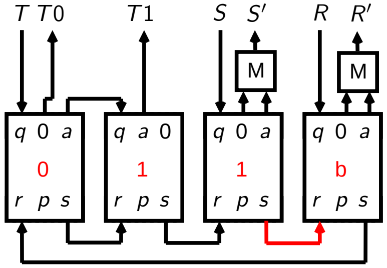
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

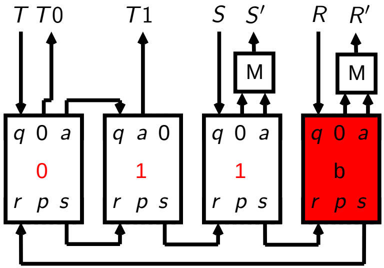
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

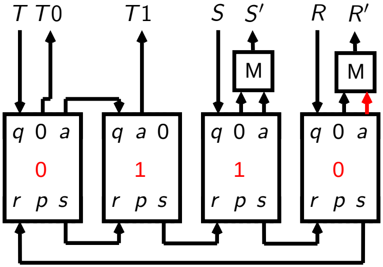
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

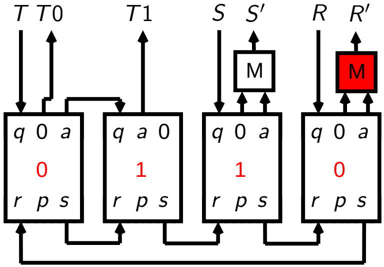
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

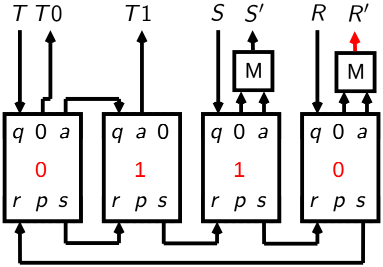
Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Realisation of Select

{Select, Merge} is universal for serial DI modules. This means that any serial module can be built using Selects and Merges.

Can realise Select with four RDMs and two Merges. Hence RDM, Merge is serial-universal. And so DM, Merge is.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

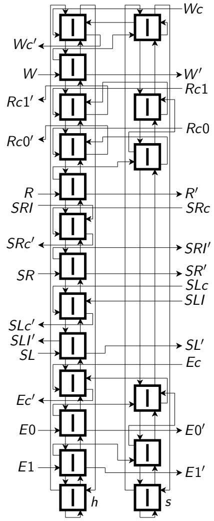
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

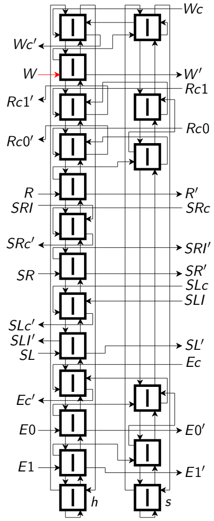
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

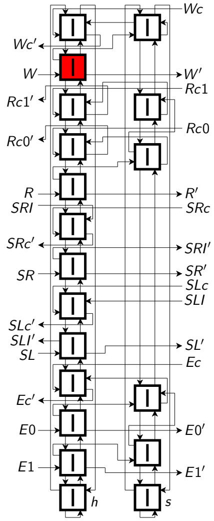
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

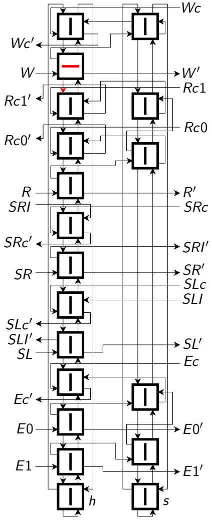
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

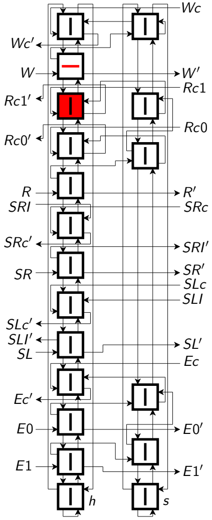
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

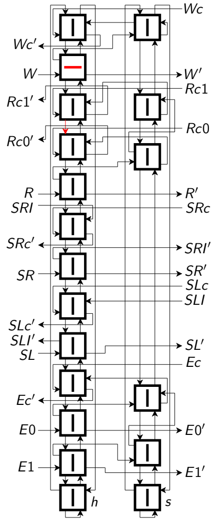
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

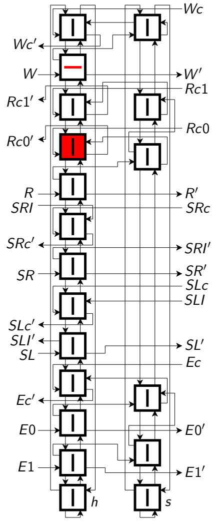
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

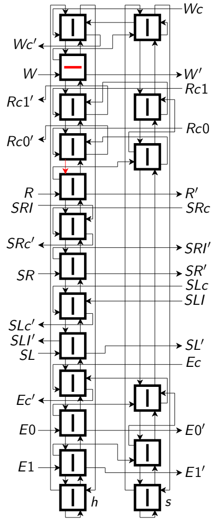
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

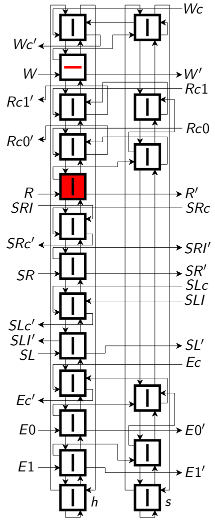
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

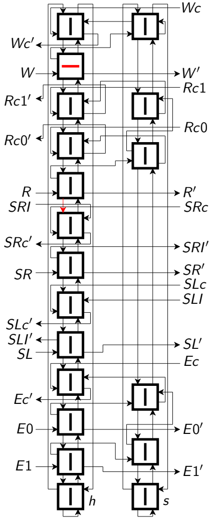
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

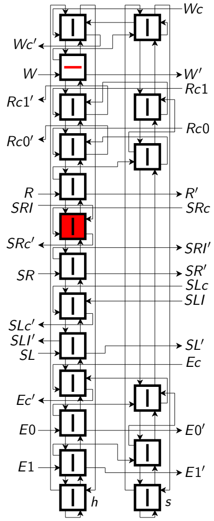
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

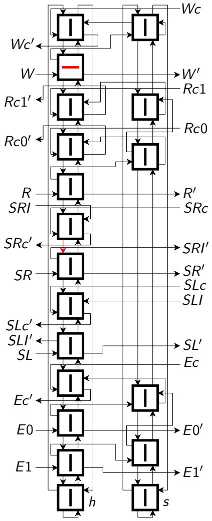
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

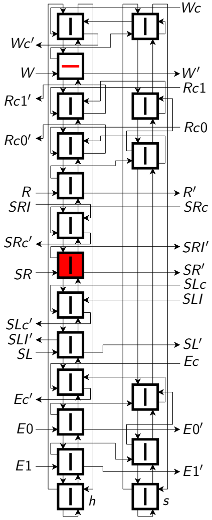
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

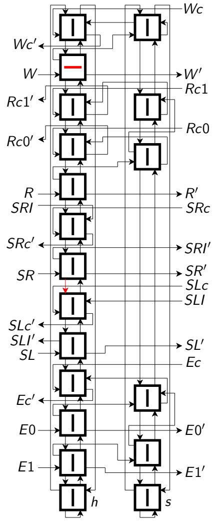
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

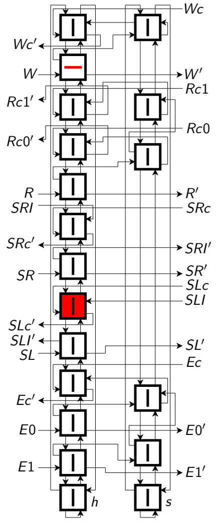
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

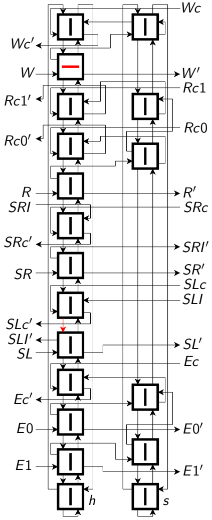
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

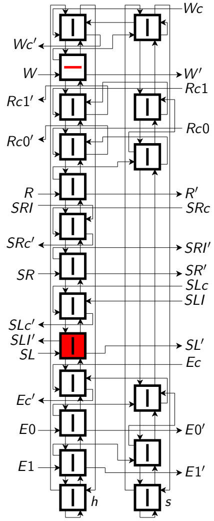
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

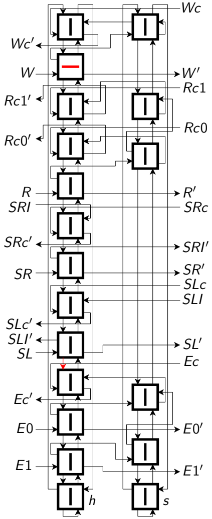
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

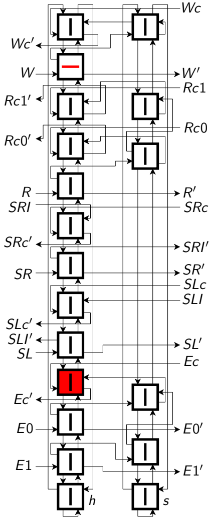
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

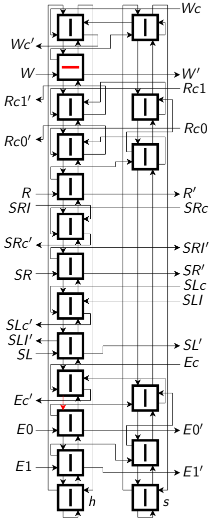
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

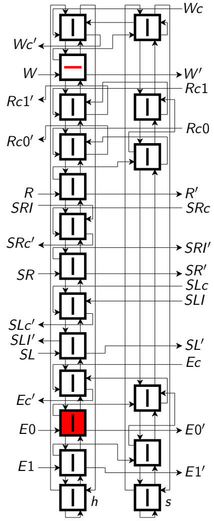
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

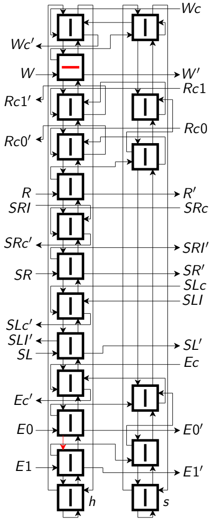
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

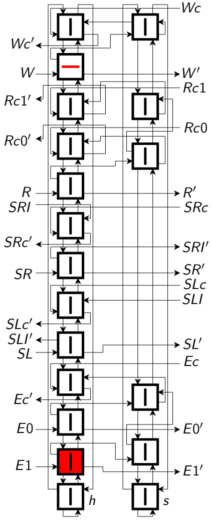
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

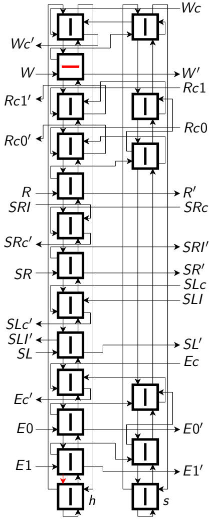
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

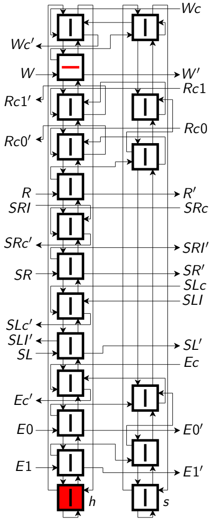
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

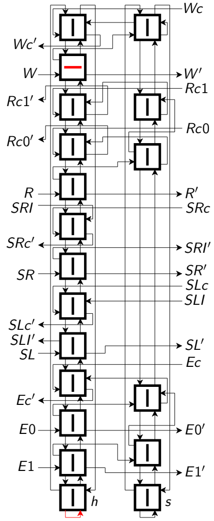
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

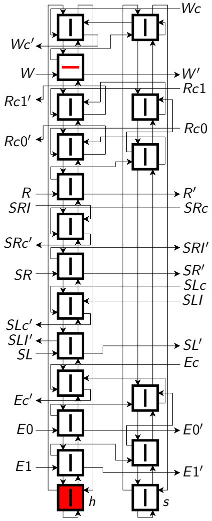
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

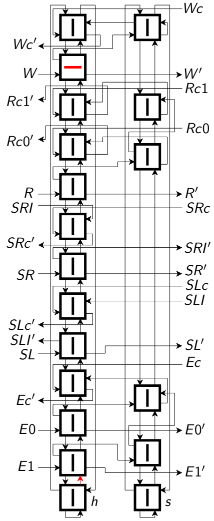
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

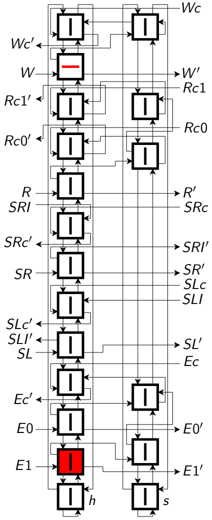
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

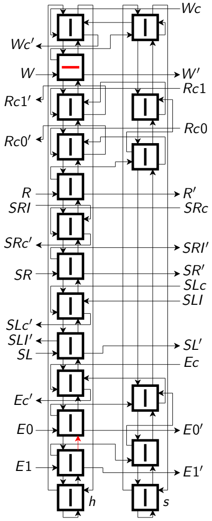
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

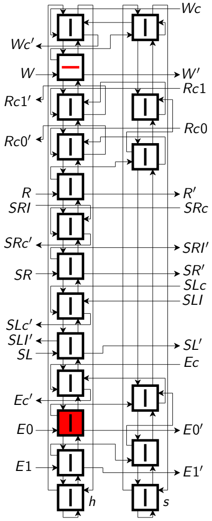
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

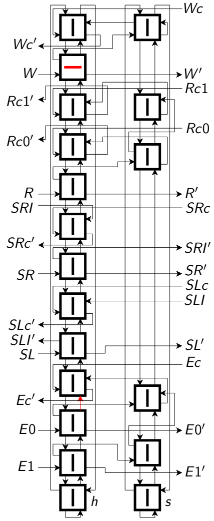
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

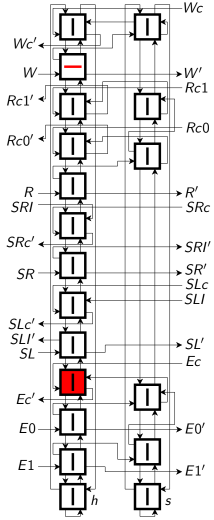
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

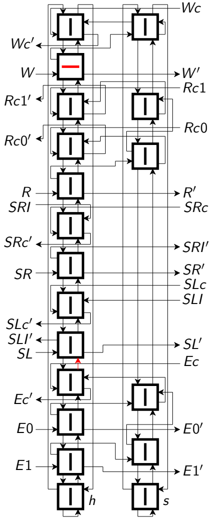
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

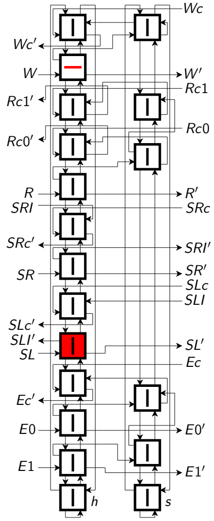
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

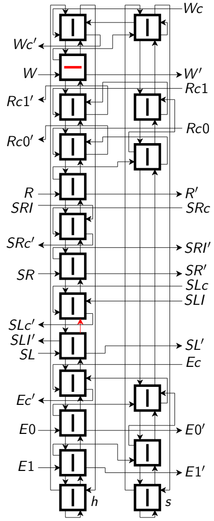
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

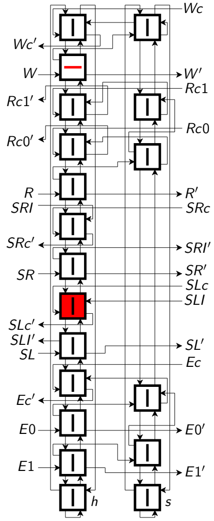
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

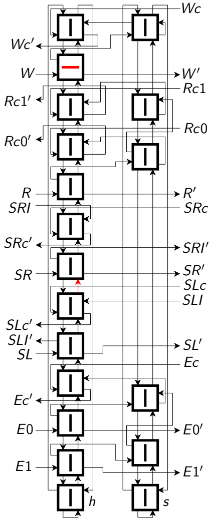
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

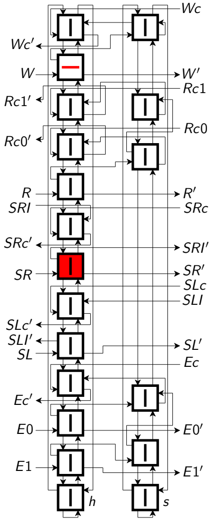
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

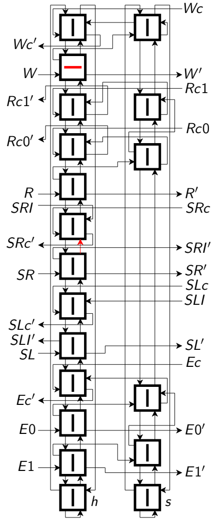
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

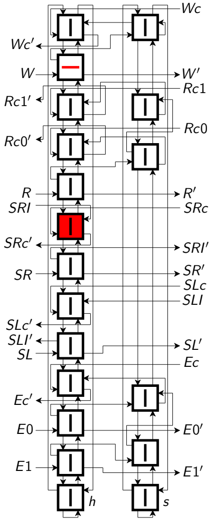
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

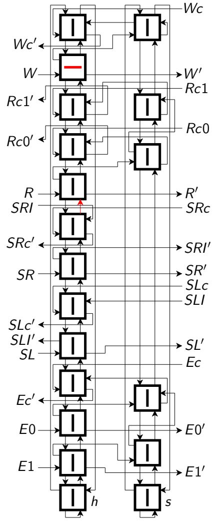
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

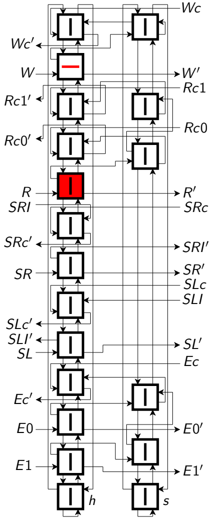
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

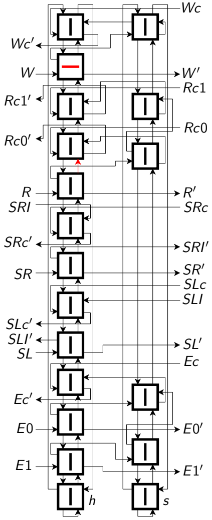
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

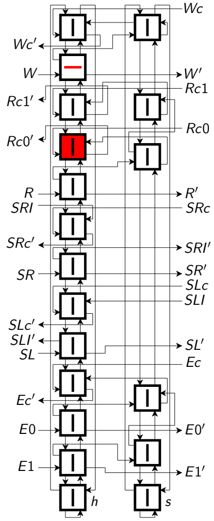
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

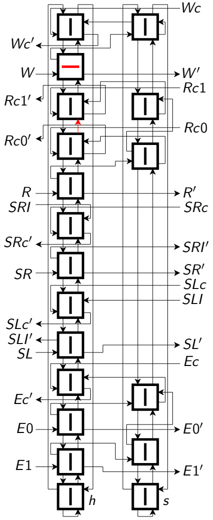
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

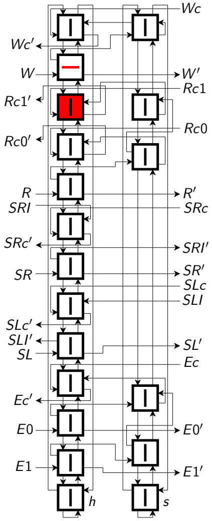
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

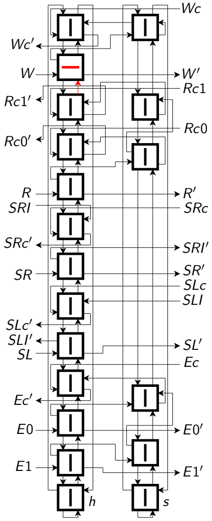
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

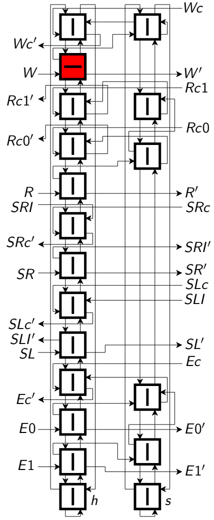
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

This is due to centralised storage of h.



Turing-tape module

Part of Morita's Turing machine construction using REs.

A semi-infinite array of such modules with signals entering from the left.

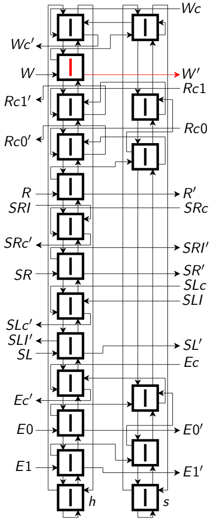
Typically signals are forwarded until reaching the module whose $h = 1$.

Useful operations (read, write etc.) are then performed.

Fairly high transition cost even when only forwarding signals.

When $h = 0$, a signal on W requires 24 transitions before being forwarded to W'.

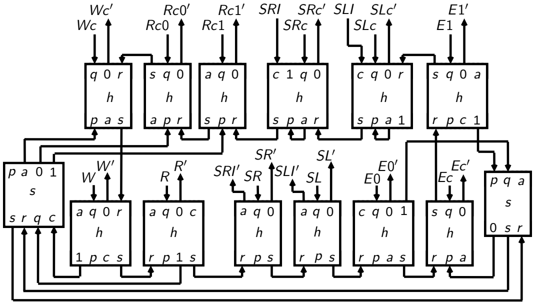
This is due to centralised storage of h.



Turing-tape construction

Distribution of memory reduces interconnection complexity and allows quick checking of h.

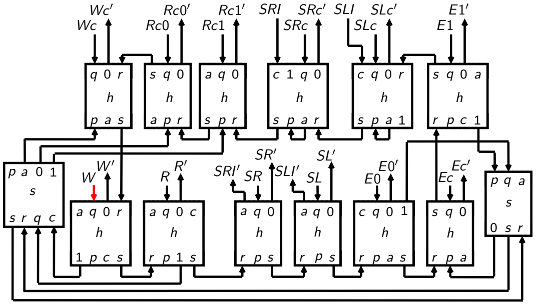
If $h = 0$, a signal on W requires only 1 transition before being output on W'.



Turing-tape construction

Distribution of memory reduces interconnection complexity and allows quick checking of h.

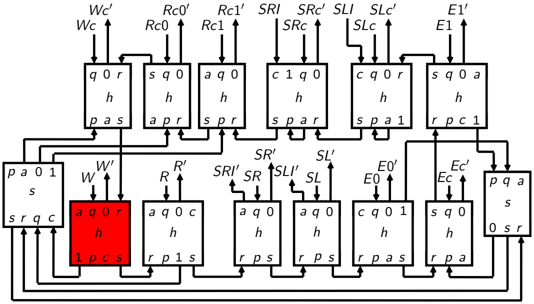
If $h = 0$, a signal on W requires only 1 transition before being output on W'.



Turing-tape construction

Distribution of memory reduces interconnection complexity and allows quick checking of h.

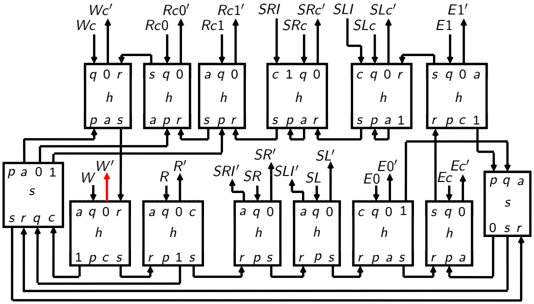
If $h = 0$, a signal on W requires only 1 transition before being output on W'.



Turing-tape construction

Distribution of memory reduces interconnection complexity and allows quick checking of h.

If $h = 0$, a signal on W requires only 1 transition before being output on W'.



Other results

- Simulation of synchronous Fredkin Gate for boolean operations.
- Several parallel-universal sets which include DM/RDM.
- Implementation using 18 Reading Toggles/Inverse Reading Toggles. $<$50% number of modules of RE implementation using RT/IRT.
- Implementation in reversible Cellular Automata. Area only $\approx$5x that of RT/IRT.

Outline

Conclusion

We have:

- Reviewed the importance of reversibility and asynchrony.
- Introduced distributed memory modules DM and RDM.
- Proven their computation-universality.
- Demonstrated their advantages in several constructions.