ASYNCHRONOUS CIRCUITS

(Robert Meolic, 2002, 2007, 2013)

There are 4 directories for this subject. Directory **circuits** contains definition of basic elements and some tests, directory **circuits-dme** contains files for verification of distributed mutual-exclusion circuits.

Directory **circuits-iscas** is about verification of ISCAS85 and ISCAS89 benchmark circuits. This work is very incomplete. Here are some links:
`http://www.cbl.ncsu.edu/benchmarks/Benchmarks-upto-1996.html`
`http://www.cse.nd.edu/courses/cse598m/www/projects/benchmarks.html`
`http://web.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html`

Directory **circuits-dat** contains old specifications of elements (not using CCS parser) and some circuits composed of them. Some elements in **circuits-dat** have been prepared also for input-receptive and semi-modular modelling, but this work is very incomplete.

Directory **circuits**

Here are definitions of basic elements and some tests. File **init.tcl** will load all elements from directory and is thus used in the beginning of every circuit.

```
# BASIC GATES

ccs_read wire1ab.ccs
ccs_read inv1ab.ccs
ccs_read fork1abc.ccs
ccs_read and2abc.ccs
ccs_read or2abc.ccs
ccs_read nand2abc.ccs
ccs_read nor2abc.ccs
ccs_read xor2abc.ccs

# ADITIONAL GATES

ccs_read and3abcd.ccs
ccs_read or3abcd.ccs
ccs_read nand3abcd.ccs
ccs_read nor3abcd.ccs

ccs_read and4abcde.ccs
ccs_read or4abcde.ccs
ccs_read nand4abcde.ccs
ccs_read nor4abcde.ccs

ccs_read fork1abcd.ccs

# ELEMENTS FOR ASYNCHRONOUS CIRCUITS
```

```
ccs_read c2abc.ccs
ccs_read me2abcd.ccs
ccs_read mez3abcde.ccs

# FLIP-FLOPS

ccs_read d1ab.ccs
ccs_read dqb1abc.ccs
ccs_read rs2abc.ccs
ccs_read rsqb2abcd.ccs
```

EST has special composition for circuits (well, nothing will prevent you to use other types of composition). There, a special minimisation is performed during the composition. As a result, the resulting process represents asynchronous circuit under the fundamental mode of operation.

EST also contains parser for simple Verilog-style descriptions. Currently, it is implemented as a convertor producing new file with CCS description.

Here is an example of recognisable Verilog-style description:

```
// THIS IS OSCILATOR
module OSC_V();
input x;
output y;

  wire A,B,C;

  // inputs
  assign A = x;

  // circuit
  and AND2_0(C,A,B);
  not NOT_0(B,C);

  // outputs (internal wires must not be used as outputs)
  assign y = C;

endmodule
```

Here is log from executing test-oscilator.tcl:

```
Reading file: test-oscilator.ccs
  Circuit OSC
    Multi-way composition ... OK
    Minimization ... (2s, 2t, 2v) OK

Reading file: oscilator.v

Reading file: tmpccs.ccs
  Circuit OSC_V
    Multi-way composition ... OK
```

```
    Minimization ... (2s, 2t, 2v) OK

PROCESS OSC
INITIAL STATE s1
s1 = x? . s2
s2 = y! . s2

PROCESS OSC_V
INITIAL STATE s1
s1 = x? . s2
s2 = y! . s2

Strong observational equivalence checking between OSC and OSC_V... OK
```

Here is log from executing test-andor.tcl:

```
Reading file: test-andor.ccs
  Circuit AND3
    Multi-way composition ... OK
    Minimization ... (12s, 28t, 28v) OK
  Circuit OR3
    Multi-way composition ... OK
    Minimization ... (12s, 28t, 28v) OK
  Circuit AND4
    Multi-way composition ... OK
    Minimization ... (21s, 69t, 69v) OK
  Circuit OR4
    Multi-way composition ... OK
    Minimization ... (21s, 69t, 69v) OK

Strong observational equivalence checking between AND-000 and AND3... OK
Strong observational equivalence checking between OR-000 and OR3... OK
Strong observational equivalence checking between AND-0000 and AND4... OK
Strong observational equivalence checking between OR-0000 and OR4... OK
```

File test-hazards.ccs includes some simple circuits and ACTLW formulae to detect static, dynamic, and steady-state hazards.

```
circuit STATIC = /// (
  OR-00,
  OR-01 [y1/c][f/b][c/a],
  AND-11 [f/c][e/b][d/a],
  INV [d/b],
  INV [e/b][b/a]
) \c\d\e\f\x\y2

circuit DYNAMIC = /// (
  INV [c/b],
  INV [e/b],
  OR-01 [f/c][e/b][d/a],
  AND-00 [d/c],
  AND-11 [y1/c][f/b][c/a]
) \c\d\e\f\x\y2

circuit TRANSIENT = /// (
  INV [xinv/b][x/a],
  AND-10 [y1/c][y2a/b][xinv/a],
  OR-00 [y2/c][y2b/b][x/a],
  FORK [y2b/c][y2a/b][y2/a]
) \xinv\y2\y2a\y2b
```

```
circuit STEADY = /// (
  INV [xinv/b][x/a],
  AND-10 [z/c][y2a/b][xinv/a],
  OR-00 [z1/c][y1a/b][z/a],
  OR-00 [z2/c][y2a/b][x/a],
  FORK [y1a/c][y1/b][z1/a],
  FORK [y2a/c][y2/b][z2/a]
) \xinv\z\z1\z2\y1a\y2a

circuit STEADY1 = /// (
  INV [xinv/b][x/a],
  AND-10 [z/c][y2a/b][xinv/a],
  OR-00 [z1/c][y1a/b][z/a],
  OR-00 [z2/c][y2a/b][x/a],
  FORK [y1a/c][y1/b][z1/a],
  FORK [y2a/c][y2/b][z2/a]
) \xinv\z\z1\z2\y1a\y2a\y2

circuit STEADY2 = /// (
  INV [xinv/b][x/a],
  AND-10 [z/c][y2a/b][xinv/a],
  OR-00 [z1/c][y1a/b][z/a],
  OR-00 [z2/c][y2a/b][x/a],
  FORK [y1a/c][y1/b][z1/a],
  FORK [y2a/c][y2/b][z2/a]
) \xinv\z\z1\z2\y1a\y2a\y1

/* STATIC HAZARD */
property STATIC1 == EEF {NOT y1!} <y1!> <y1!> <NOT y1!> true;
property STATIC2 == EEF {NOT y2!} <y2!> <y2!> <NOT y2!> true;

/* DYNAMIC HAZARD */
property DYNAMIC1 == EEF {y1!} <y1!> <y1!> true;
property DYNAMIC2 == EEF {y2!} <y2!> <y2!> true;

/* STEADY-STATE HAZARD */
property STEADY1 == EEF {x?} ((<x?> TRUE) AND (<y1!> <x?> TRUE));
property STEADY2 == EEF {x?} ((<x?> TRUE) AND (<y2!> <x?> TRUE));
```

Here is log from executing test-hazards.tcl:

```
Reading file: test-hazards.ccs
  Circuit STATIC
    Multi-way composition ... OK
    Minimization ... (10s, 20t, 20v) OK
  Circuit DYNAMIC
    Multi-way composition ... OK
    Minimization ... (6s, 10t, 10v) OK
  Circuit TRANSIENT
    Multi-way composition ... OK
    Minimization ... (4s, 5t, 5v) OK
  Circuit STEADY
    Multi-way composition ... OK
```

```
    Minimization ... (6s, 8t, 8v) OK
  Circuit STEADY1
    Multi-way composition ... OK
    Minimization ... (3s, 2t, 2v) OK
  Circuit STEADY2
    Multi-way composition ... OK
    Minimization ... (4s, 3t, 3v) OK
  Property STATIC1
    STATIC1 = EEF {NOT y1!} <y1!> <y1!> <NOT y1!> true;
  Property STATIC2
    STATIC2 = EEF {NOT y2!} <y2!> <y2!> <NOT y2!> true;
  Property DYNAMIC1
    DYNAMIC1 = EEF {y1!} <y1!> <y1!> true;
  Property DYNAMIC2
    DYNAMIC2 = EEF {y2!} <y2!> <y2!> true;
  Property STEADY1
    STEADY1 = EEF {x?} ((<x?> true) AND (<y1!> <x?> true));
  Property STEADY2
    STEADY2 = EEF {x?} ((<x?> true) AND (<y2!> <x?> true));


============================
 LOOKING FOR STATIC HAZARDS
============================

ACTL/ACTLW model checking on process STATIC
EEF {NOT y1!} <y1!> <y1!> <NOT y1!> true ==> TRUE

ACTL/ACTLW model checking on process STATIC
EEF {NOT y2!} <y2!> <y2!> <NOT y2!> true ==> FALSE

ACTL/ACTLW model checking on process DYNAMIC
EEF {NOT y1!} <y1!> <y1!> <NOT y1!> true ==> FALSE

ACTL/ACTLW model checking on process DYNAMIC
EEF {NOT y2!} <y2!> <y2!> <NOT y2!> true ==> FALSE

ACTL/ACTLW model checking on process TRANSIENT
EEF {NOT y1!} <y1!> <y1!> <NOT y1!> true ==> TRUE

ACTL/ACTLW model checking on process TRANSIENT
EEF {NOT y2!} <y2!> <y2!> <NOT y2!> true ==> FALSE

ACTL/ACTLW model checking on process STEADY1
EEF {NOT y1!} <y1!> <y1!> <NOT y1!> true ==> FALSE

ACTL/ACTLW model checking on process STEADY1
EEF {NOT y2!} <y2!> <y2!> <NOT y2!> true ==> FALSE

ACTL/ACTLW model checking on process STEADY2
EEF {NOT y1!} <y1!> <y1!> <NOT y1!> true ==> FALSE

ACTL/ACTLW model checking on process STEADY2
EEF {NOT y2!} <y2!> <y2!> <NOT y2!> true ==> FALSE


============================
 LOOKING FOR DYNAMIC HAZARDS
============================

ACTL/ACTLW model checking on process STATIC
EEF {y1!} <y1!> <y1!> true ==> FALSE

ACTL/ACTLW model checking on process STATIC
EEF {y2!} <y2!> <y2!> true ==> FALSE

ACTL/ACTLW model checking on process DYNAMIC
EEF {y1!} <y1!> <y1!> true ==> TRUE

ACTL/ACTLW model checking on process DYNAMIC
EEF {y2!} <y2!> <y2!> true ==> FALSE

ACTL/ACTLW model checking on process TRANSIENT
EEF {y1!} <y1!> <y1!> true ==> FALSE
```

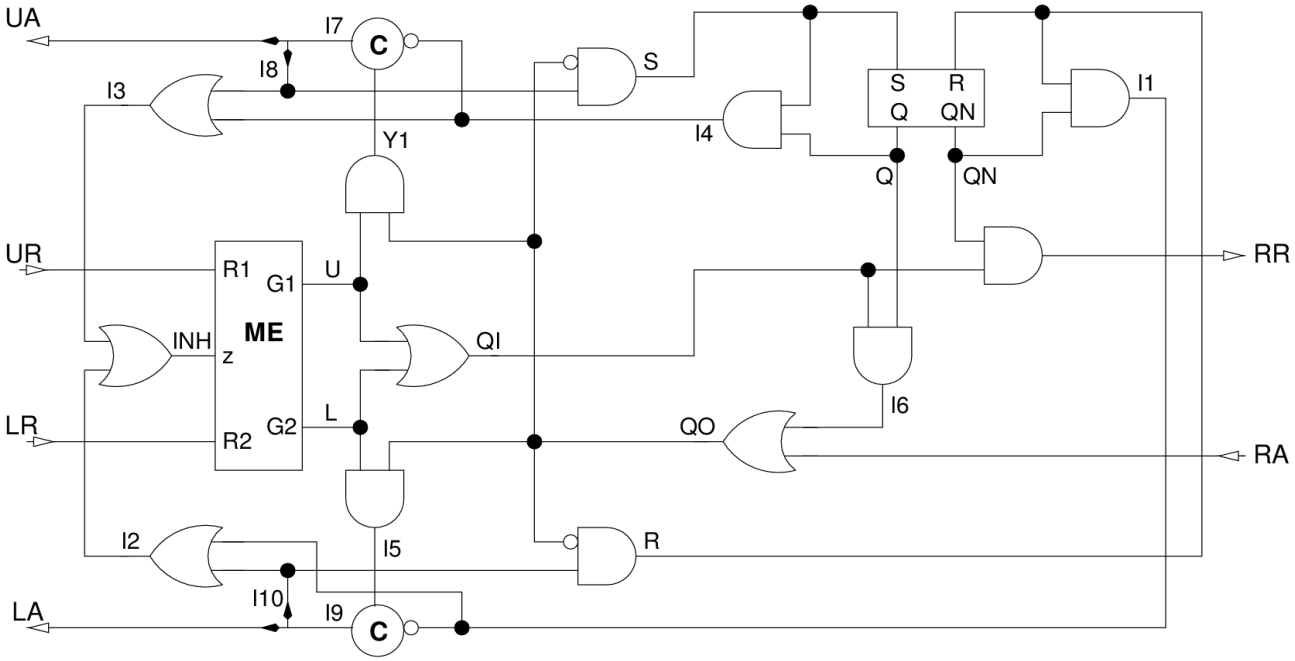
```
ACTL/ACTLW model checking on process TRANSIENT
EEF {y2!} <y2!> <y2!> true ==> FALSE

ACTL/ACTLW model checking on process STEADY1
EEF {y1!} <y1!> <y1!> true ==> FALSE

ACTL/ACTLW model checking on process STEADY1
EEF {y2!} <y2!> <y2!> true ==> FALSE

ACTL/ACTLW model checking on process STEADY2
EEF {y1!} <y1!> <y1!> true ==> FALSE

ACTL/ACTLW model checking on process STEADY2
EEF {y2!} <y2!> <y2!> true ==> FALSE


===========================
 LOOKING FOR STEADY HAZARDS
===========================

ACTL/ACTLW model checking on process STATIC
EEF {x?} ((<x?> true) AND (<y1!> <x?> true)) ==> FALSE

ACTL/ACTLW model checking on process STATIC
EEF {x?} ((<x?> true) AND (<y2!> <x?> true)) ==> FALSE

ACTL/ACTLW model checking on process DYNAMIC
EEF {x?} ((<x?> true) AND (<y1!> <x?> true)) ==> FALSE

ACTL/ACTLW model checking on process DYNAMIC
EEF {x?} ((<x?> true) AND (<y2!> <x?> true)) ==> FALSE

ACTL/ACTLW model checking on process TRANSIENT
EEF {x?} ((<x?> true) AND (<y1!> <x?> true)) ==> FALSE

ACTL/ACTLW model checking on process TRANSIENT
EEF {x?} ((<x?> true) AND (<y2!> <x?> true)) ==> FALSE

ACTL/ACTLW model checking on process STEADY1
EEF {x?} ((<x?> true) AND (<y1!> <x?> true)) ==> FALSE

ACTL/ACTLW model checking on process STEADY1
EEF {x?} ((<x?> true) AND (<y2!> <x?> true)) ==> FALSE

ACTL/ACTLW model checking on process STEADY2
EEF {x?} ((<x?> true) AND (<y1!> <x?> true)) ==> FALSE

ACTL/ACTLW model checking on process STEADY2
EEF {x?} ((<x?> true) AND (<y2!> <x?> true)) ==> FALSE
```

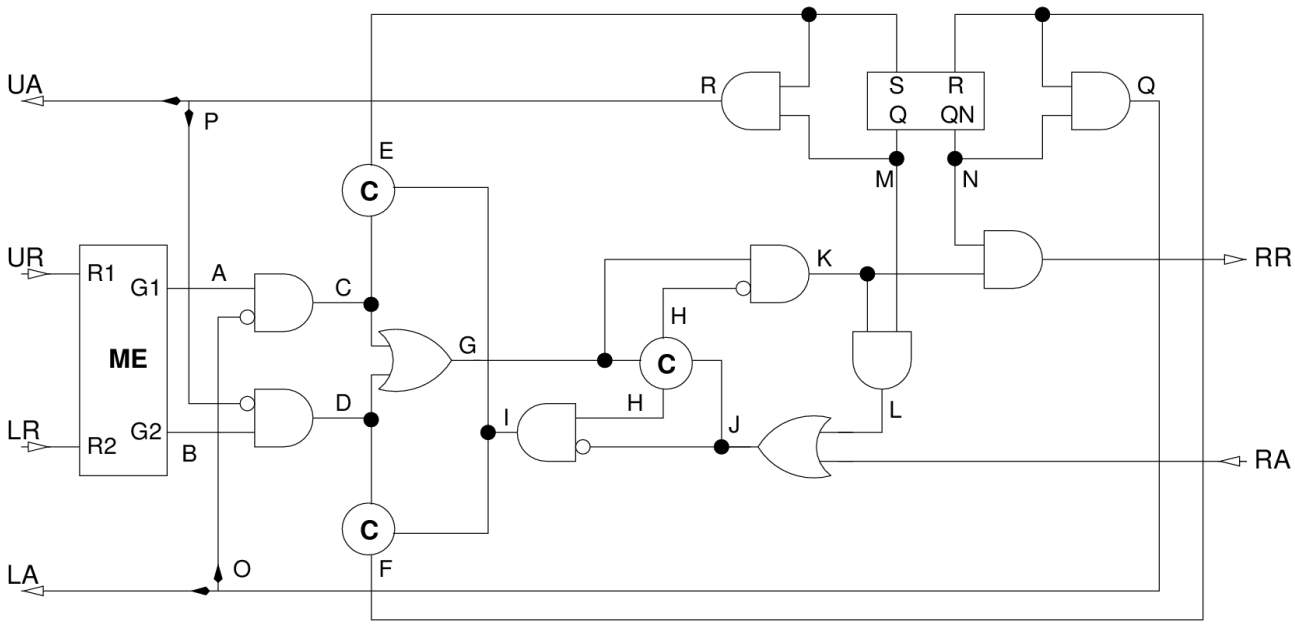
Directory **circuit-dme**

Distributed mutual-exclusion (DME) circuits are asynchronous circuits composed of identical DME cells connected in a ring of arbitrary size. Each DME cell provides a connection point for one user, and all users compete for exclusive access to a shared resource. We are able to detect hazards and verify correctness of external behaviour of the circuits under the fundamental mode of operation.

Hazards reflecting in glitches are classified with regard to their shape into static and dynamic hazards. Static hazard occurs when the signal is momentarily changed al-though it should remain the same. Dynamic hazard occurs if the signal oscillates before changing its value. A steady-state hazard is present in the circuit if after a particular sequence of input signals with some arrangements of delays an output signal appears, but with other arrangements of delays it does not.

Here are two well-known implementations of DME cell (both proposed by A. J. Martin).





Files dme-martin.ccs and dme-smv.ccs contain description of the two circuits. There are two versions of each circuit which differs in initial state. Circuits DME-MARTIN-FALSE and DME-SMV-FALSE initially do not possess a token, while DME-MARTIN-TRUE and DME-SMV-TRUE starts with a token.

```
circuit DME-MARTIN-FALSE = /// (
  MEZ-000 [l/e][u/d][inh/c][lr/b][ur/a],
  OR-00 [qi/c][l/b][u/a],
  AND-00 [y1/c][qo/b][u/a],
  AND-00 [i5/c][qo/b][l/a],
  C-01 [i7/c][i4/b][y1/a],
```

```
  C-01 [i9/c][i1/b][i5/a],
  OR-00 [i3/c][i4/b][i8/a],
  OR-00 [i2/c][i10/b][i1/a],
  OR-00 [inh/c][i2/b][i3/a],
  AND-10 [s/c][i8/b][qo/a],
  AND-10 [r/c][i10/b][qo/a],
  OR-00 [qo/c][ra/b][i6/a],
  FORK [ua/c][i8/b][i7/a],
  FORK [la/c][i10/b][i9/a],
  RSQB-000 [qb/d][q/c][s/b][r/a],
  AND-00 [i4/c][q/b][s/a],
  AND-00 [i6/c][q/b][qi/a],
  AND-01 [i1/c][qb/b][r/a],
  AND-10 [rr/c][qi/b][qb/a]
) \l\u\inh\q\qn\qi\qo\y1\r\s\i1\i2\i3\i4\i5\i6\i7\i8\i9\i10

circuit DME-MARTIN-TRUE = /// (
  MEZ-000 [l/e][u/d][inh/c][lr/b][ur/a],
  OR-00 [qi/c][l/b][u/a],
  AND-00 [y1/c][qo/b][u/a],
  AND-00 [i5/c][qo/b][l/a],
  C-01 [i7/c][i4/b][y1/a],
  C-01 [i9/c][i1/b][i5/a],
  OR-00 [i3/c][i4/b][i8/a],
  OR-00 [i2/c][i10/b][i1/a],
  OR-00 [inh/c][i2/b][i3/a],
  AND-10 [s/c][i8/b][qo/a],
  AND-10 [r/c][i10/b][qo/a],
  OR-00 [qo/c][ra/b][i6/a],
  FORK [ua/c][i8/b][i7/a],
  FORK [la/c][i10/b][i9/a],
  RSQB-001 [qb/d][q/c][s/b][r/a],
  AND-01 [i4/c][q/b][s/a],
  AND-01 [i6/c][q/b][qi/a],
  AND-00 [i1/c][qb/b][r/a],
  AND-00 [rr/c][qi/b][qb/a]
) \l\u\inh\q\qn\qi\qo\y1\r\s\i1\i2\i3\i4\i5\i6\i7\i8\i9\i10

circuit DME-SMV-FALSE = /// (
  ME-00 [B/d][A/c][lr/b][ur/a],
  AND-01 [C/c][O/b][A/a],
  AND-10 [D/c][B/b][P/a],
  C-00 [E/c][I/b][C/a],
  C-00 [F/c][I/b][D/a],
  OR-00 [G/c][D/b][C/a],
  C-00 [H/c][J/b][G/a],
  AND-01 [I/c][J/b][H/a],
  OR-00 [J/c][ra/b][L/a],
  AND-01 [K/c][H/b][G/a],
  FORK [la/c][O/b][Q/a],
```

```
  FORK [ua/c][P/b][R/a],
  RSQB-000 [N/d][M/c][E/b][F/a],
  AND-00 [R/c][M/b][E/a],
  AND-00 [L/c][M/b][K/a],
  AND-01 [Q/c][N/b][F/a],
  AND-10 [rr/c][K/b][N/a]
) \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R

circuit DME-SMV-TRUE = /// (
  ME-00 [B/d][A/c][lr/b][ur/a],
  AND-01 [C/c][O/b][A/a],
  AND-10 [D/c][B/b][P/a],
  C-00 [E/c][I/b][C/a],
  C-00 [F/c][I/b][D/a],
  OR-00 [G/c][D/b][C/a],
  C-00 [H/c][J/b][G/a],
  AND-01 [I/c][J/b][H/a],
  OR-00 [J/c][ra/b][L/a],
  AND-01 [K/c][H/b][G/a],
  FORK [la/c][O/b][Q/a],
  FORK [ua/c][P/b][R/a],
  RSQB-001 [N/d][M/c][E/b][F/a],
  AND-01 [R/c][M/b][E/a],
  AND-01 [L/c][M/b][K/a],
  AND-00 [Q/c][N/b][F/a],
  AND-00 [rr/c][K/b][N/a]
) \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R
```

Files dme-martin-abstract.ccs and dme-smv-abstract.ccs are used to create abstracted versions of circuits where only one output is preserved. For example:

```
circuit DME-SMV-TRUE-UA = ///(DME-SMV-TRUE [tau/la][tau/rr])
circuit DME-SMV-TRUE-LA = ///(DME-SMV-TRUE [tau/ua][tau/rr])
circuit DME-SMV-TRUE-RR = ///(DME-SMV-TRUE [tau/ua][tau/la])
```

Here are the properties from dme-cell-properties.actl, dme-cell-static.actl, dme-cell-dynamic.actl, and dme-cell-steady.actl.

```
/* Macros */
#define UA {NOT ua!} W {ua!}
#define LA {NOT la!} W {la!}

#
## Sistem vsebuje zagatno stanje
AAX {false} OR EEF AAX {false};

#
## Sistem vsebuje divergentno stanje
(EEX {true} AND EEG {tau} EEX {true}) OR (EEF (EEX {true} AND EEG
{tau} EEX {true}));
```

```
#
## Dana zahteva bo ali odobrena ali pa umaknjena
AA[{NOT ur?} W {ur?} NOT EEG {(NOT ua!) AND (NOT ur?)}];
AA[{NOT lr?} W {lr?} NOT EEG {(NOT la!) AND (NOT lr?)}];

#
## Mutual exclusion = UA in LA ne smeta biti istocasno na 1
## ... ob prvi spremembi signalov
AA[$UA AA[{NOT la!} W {ua!}]];
AA[$LA AA[{NOT ua!} W {la!}]];

#
## ... ob drugi spremembi signalov
AA[$UA AA[$UA AA[$UA AA[{NOT la!} W {ua!}]]]];
AA[$LA AA[$LA AA[$LA AA[{NOT ua!} W {la!}]]]];

/* Macros */
#define IN (ur? OR lr? OR ra?)
#define OUT (ua! OR la! OR rr!)

/* Mozna sta dva izhodna dogodkov zaporedoma */
EEF {$IN} <$OUT> <$OUT> <$IN> true;

/* Mozni so vsaj trije izhodni dogodki zaporedoma */
EEF {$IN} <$OUT> <$OUT> <$OUT> true;

/* Obstaja stanje, v katerem je hkrati mozen vhodni dogodek */
/* in zaporedje izhodni - vhodni dogodek                    */
EEF {$IN} ((<$IN> TRUE) AND (<$OUT> <$IN> TRUE));
```

Here is the log from executing dme-martin.tcl:

```
Reading file: dme-martin.ccs
  Circuit DME-MARTIN-FALSE
    Multi-way composition ... OK
    Minimization ... (78s, 161t, 161v) OK
  Circuit DME-MARTIN-TRUE
    Multi-way composition ... OK
    Minimization ... (78s, 161t, 161v) OK

Reading file: dme-martin-abstract.ccs
  Circuit DME-MARTIN-FALSE-UA
    Multi-way composition ... OK
    Minimization ... (38s, 100t, 100v) OK
  Circuit DME-MARTIN-FALSE-LA
    Multi-way composition ... OK
    Minimization ... (38s, 98t, 98v) OK
  Circuit DME-MARTIN-FALSE-RR
    Multi-way composition ... OK
    Minimization ... (67s, 184t, 184v) OK
  Circuit DME-MARTIN-TRUE-UA
    Multi-way composition ... OK
    Minimization ... (38s, 100t, 100v) OK
  Circuit DME-MARTIN-TRUE-LA
    Multi-way composition ... OK
    Minimization ... (38s, 98t, 98v) OK
  Circuit DME-MARTIN-TRUE-RR
```

```
    Multi-way composition ... OK
    Minimization ... (67s, 184t, 184v) OK


=====================
INFO ABOUT PROCESSES
=====================

DME-MARTIN-FALSE has 78 states / 161 transitions / 431 BDD nodes.
DME-MARTIN-TRUE has 78 states / 161 transitions / 446 BDD nodes.

DME-MARTIN-FALSE-UA has 38 states / 100 transitions / 232 BDD nodes.
DME-MARTIN-FALSE-LA has 38 states / 98 transitions / 232 BDD nodes.
DME-MARTIN-FALSE-RR has 67 states / 184 transitions / 410 BDD nodes.
DME-MARTIN-TRUE-UA has 38 states / 100 transitions / 242 BDD nodes.
DME-MARTIN-TRUE-LA has 38 states / 98 transitions / 234 BDD nodes.
DME-MARTIN-TRUE-RR has 67 states / 184 transitions / 397 BDD nodes.

=====================================
CHECKING PROPERTIES OF DME-MARTIN-FALSE
=====================================

ACTL/ACTLW model checking on process DME-MARTIN-FALSE
# Sistem vsebuje zagatno stanje

AAX {false} OR EEF AAX {false} ==> FALSE

# Sistem vsebuje divergentno stanje

(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})) ==> FALSE

# Dana zahteva bo ali odobrena ali pa umaknjena

AA[{NOT ur?} W {ur?} NOT EEG {(NOT ua!) AND (NOT ur?)}] ==> FALSE

AA[{NOT lr?} W {lr?} NOT EEG {(NOT la!) AND (NOT lr?)}] ==> FALSE

# Mutual exclusion = UA in LA ne smeta biti istocasno na 1
# ... ob prvi spremembi signalov

AA[{NOT ua!} W {ua!} AA[{NOT la!} W {ua!}]] ==> FALSE

AA[{NOT la!} W {la!} AA[{NOT ua!} W {la!}]] ==> FALSE

# ... ob drugi spremembi signalov

AA[{NOT ua!} W {ua!} AA[{NOT ua!} W {ua!} AA[{NOT ua!} W {ua!} AA[{NOT la!} W {ua!}]]]] ==> FALSE

AA[{NOT la!} W {la!} AA[{NOT la!} W {la!} AA[{NOT la!} W {la!} AA[{NOT ua!} W {la!}]]]] ==> FALSE

================================================
LOOKING FOR HAZARDS in DME-MARTIN-FALSE on line UA
================================================

ACTL/ACTLW model checking on process DME-MARTIN-FALSE-UA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
TRUE
Witness: (ra?)(ur?)(ua!)(ra?)(ua!)(ua!)(lr?)


ACTL/ACTLW model checking on process DME-MARTIN-FALSE-UA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-MARTIN-FALSE-UA
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> TRUE
Witness: (lr?)(ra?)(lr?)(lr?)(ur?)(ra?)(ra?)


================================================
LOOKING FOR HAZARDS in DME-MARTIN-FALSE on line LA
================================================
```

```
ACTL/ACTLW model checking on process DME-MARTIN-FALSE-LA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-MARTIN-FALSE-LA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-MARTIN-FALSE-LA
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> TRUE
Witness: (ra?)(ur?)(lr?)(ur?)(ur?)(ra?)


================================================
LOOKING FOR HAZARDS in DME-MARTIN-FALSE on line RR
================================================

ACTL/ACTLW model checking on process DME-MARTIN-FALSE-RR
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
TRUE
Witness: (lr?)(rr!)(ur?)(lr?)(rr!)(rr!)(lr?)


ACTL/ACTLW model checking on process DME-MARTIN-FALSE-RR
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-MARTIN-FALSE-RR
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> TRUE
Witness: (ra?)(ur?)(rr!)(lr?)(ur?)(rr!)(ur?)(ra?)(lr?)


================================================
LOOKING FOR HAZARDS in DME-MARTIN-TRUE on line UA
================================================

ACTL/ACTLW model checking on process DME-MARTIN-TRUE-UA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
TRUE
Witness: (lr?)(lr?)(ra?)(ur?)(ua!)(ra?)(ua!)(ua!)(lr?)


ACTL/ACTLW model checking on process DME-MARTIN-TRUE-UA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-MARTIN-TRUE-UA
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> TRUE
Witness: (lr?)(ra?)(lr?)(lr?)(ur?)(ra?)(ra?)


================================================
LOOKING FOR HAZARDS in DME-MARTIN-TRUE on line LA
================================================

ACTL/ACTLW model checking on process DME-MARTIN-TRUE-LA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-MARTIN-TRUE-LA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
```

```
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-MARTIN-TRUE-LA
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> TRUE
Witness: (ra?)(ur?)(lr?)(ur?)(ur?)(ra?)


================================================
LOOKING FOR HAZARDS in DME-MARTIN-TRUE on line RR
================================================

ACTL/ACTLW model checking on process DME-MARTIN-TRUE-RR
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
TRUE
Witness: (lr?)(lr?)(lr?)(rr!)(ur?)(lr?)(rr!)(rr!)(lr?)


ACTL/ACTLW model checking on process DME-MARTIN-TRUE-RR
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-MARTIN-TRUE-RR
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> TRUE
Witness: (ra?)(ur?)(lr?)(ur?)(ur?)(ra?)(lr?)
```

Here is the log from executing dme-smv.tcl:

```
Reading file: dme-smv.ccs
  Circuit DME-SMV-FALSE
    Multi-way composition ... OK
    Minimization ... (50s, 108t, 108v) OK
  Circuit DME-SMV-TRUE
    Multi-way composition ... OK
    Minimization ... (50s, 108t, 108v) OK

Reading file: dme-smv-abstract.ccs
  Circuit DME-SMV-FALSE-UA
    Multi-way composition ... OK
    Minimization ... (26s, 66t, 66v) OK
  Circuit DME-SMV-FALSE-LA
    Multi-way composition ... OK
    Minimization ... (28s, 72t, 72v) OK
  Circuit DME-SMV-FALSE-RR
    Multi-way composition ... OK
    Minimization ... (32s, 80t, 80v) OK
  Circuit DME-SMV-TRUE-UA
    Multi-way composition ... OK
    Minimization ... (26s, 66t, 66v) OK
  Circuit DME-SMV-TRUE-LA
    Multi-way composition ... OK
    Minimization ... (28s, 72t, 72v) OK
  Circuit DME-SMV-TRUE-RR
    Multi-way composition ... OK
    Minimization ... (32s, 80t, 80v) OK


======================
INFO ABOUT PROCESSES
======================

DME-SMV-FALSE has 50 states / 108 transitions / 288 BDD nodes.
DME-SMV-TRUE has 50 states / 108 transitions / 297 BDD nodes.

DME-SMV-FALSE-UA has 26 states / 66 transitions / 159 BDD nodes.
DME-SMV-FALSE-LA has 28 states / 72 transitions / 171 BDD nodes.
DME-SMV-FALSE-RR has 32 states / 80 transitions / 176 BDD nodes.
DME-SMV-TRUE-UA has 26 states / 66 transitions / 166 BDD nodes.
```

```
DME-SMV-TRUE-LA has 28 states / 72 transitions / 178 BDD nodes.
DME-SMV-TRUE-RR has 32 states / 80 transitions / 178 BDD nodes.

==================================
CHECKING PROPERTIES OF DME-SMV-FALSE
==================================

ACTL/ACTLW model checking on process DME-SMV-FALSE
# Sistem vsebuje zagatno stanje

AAX {false} OR EEF AAX {false} ==> FALSE

# Sistem vsebuje divergentno stanje

(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})) ==> FALSE

# Dana zahteva bo ali odobrena ali pa umaknjena

AA[{NOT ur?} W {ur?} NOT EEG {(NOT ua!) AND (NOT ur?)}] ==> FALSE

AA[{NOT lr?} W {lr?} NOT EEG {(NOT la!) AND (NOT lr?)}] ==> FALSE

# Mutual exclusion = UA in LA ne smeta biti istocasno na 1
# ... ob prvi spremembi signalov

AA[{NOT ua!} W {ua!} AA[{NOT la!} W {ua!}]] ==> FALSE

AA[{NOT la!} W {la!} AA[{NOT ua!} W {la!}]] ==> FALSE

# ... ob drugi spremembi signalov

AA[{NOT ua!} W {ua!} AA[{NOT ua!} W {ua!} AA[{NOT ua!} W {ua!} AA[{NOT la!} W {ua!}]]]] ==> FALSE

AA[{NOT la!} W {la!} AA[{NOT la!} W {la!} AA[{NOT la!} W {la!} AA[{NOT ua!} W {la!}]]]] ==> FALSE

================================================
LOOKING FOR HAZARDS in DME-SMV-FALSE on line UA
================================================

ACTL/ACTLW model checking on process DME-SMV-FALSE-UA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-FALSE-UA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-FALSE-UA
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> FALSE
Counterexample not generated.


================================================
LOOKING FOR HAZARDS in DME-SMV-FALSE on line LA
================================================

ACTL/ACTLW model checking on process DME-SMV-FALSE-LA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-FALSE-LA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-FALSE-LA
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
```

```
ra?)> true)) ==> FALSE
Counterexample not generated.


==========================================
LOOKING FOR HAZARDS in DME-SMV-FALSE on line RR
==========================================

ACTL/ACTLW model checking on process DME-SMV-FALSE-RR
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
TRUE
Witness: (ra?)(lr?)(rr!)(rr!)(lr?)


ACTL/ACTLW model checking on process DME-SMV-FALSE-RR
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-FALSE-RR
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> FALSE
Counterexample not generated.


==========================================
LOOKING FOR HAZARDS in DME-SMV-TRUE on line UA
==========================================

ACTL/ACTLW model checking on process DME-SMV-TRUE-UA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-TRUE-UA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-TRUE-UA
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> FALSE
Counterexample not generated.


==========================================
LOOKING FOR HAZARDS in DME-SMV-TRUE on line LA
==========================================

ACTL/ACTLW model checking on process DME-SMV-TRUE-LA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-TRUE-LA
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-TRUE-LA
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> FALSE
Counterexample not generated.


==========================================
LOOKING FOR HAZARDS in DME-SMV-TRUE on line RR
==========================================

ACTL/ACTLW model checking on process DME-SMV-TRUE-RR
```

```
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ur? OR lr? OR ra?)> true ==>
TRUE
Witness: (lr?)(lr?)(ra?)(lr?)(rr!)(rr!)(lr?)


ACTL/ACTLW model checking on process DME-SMV-TRUE-RR
EEF {(ur? OR lr? OR ra?)} <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> <(ua! OR la! OR rr!)> true ==>
FALSE
Counterexample not generated.


ACTL/ACTLW model checking on process DME-SMV-TRUE-RR
EEF {(ur? OR lr? OR ra?)} ((<(ur? OR lr? OR ra?)> true) AND (<(ua! OR la! OR rr!)> <(ur? OR lr? OR
ra?)> true)) ==> FALSE
Counterexample not generated.
```

Files dme-2el.ccs, dme-3el.ccs, dme-4el.ccs, and dme-2el.ccs are compositions of DME cells into a ring. For example:

```
/* DME composed of 3 cells */
circuit DME = /// (
  dme-true  [ur1/ur][ua1/ua][r12/lr][a21/la][r31/rr][a13/ra],
  dme-false [ur2/ur][ua2/ua][r23/lr][a32/la][r12/rr][a21/ra],
  dme-false [ur3/ur][ua3/ua][r31/lr][a13/la][r23/rr][a32/ra]
) \r12\r23\r31\a13\a21\a32
```

Here are the properties from dme2.actl (for each number of cells, there are different formulae):

```
/* Macros */
#define UA1 {NOT ua1!} W {ua1!}
#define UA2 {NOT ua2!} W {ua2!}
#define NOTUA (NOT ua1! AND NOT ua2!)

#
#Sistem vsebuje zagatno stanje
AAX {false} OR EEF AAX {false};

#
#Sistem vsebuje divergentno stanje
(EEX {true} AND EEG {tau} EEX {true}) OR (EEF (EEX {true} AND EEG
{tau} EEX {true}));

#
#Dana zahteva bo ali odobrena ali pa umaknjena
AA[{NOT ur1!} W {ur1!} NOT EEG {(NOT ua1!) AND (NOT ur1?)}];
AA[{NOT ur2!} W {ur2!} NOT EEG {(NOT ua2!) AND (NOT ur2?)}];

#
## After an acknowledge is sent (removed),
## it will not be removed (sent) before the user requests this.
AAG [ua1!] AA[{NOT ua1!} W {ur1?}];
AAG [ua2!] AA[{NOT ua2!} W {ur2?}];

#
```

```
## After an user gets acknowledge for the first time,
## others will not get acknowledge until his ack is removed.
AA[$UA1 AA[{$NOTUA} W {ua1!}]];
AA[$UA2 AA[{$NOTUA} W {ua2!}]];

#
## After an user gets acknowledge for the second time,
## others will not get acknowledge until his ack is removed.
AA[$UA1 AA[$UA1 AA[$UA1 AA[{$NOTUA} W {ua1!}]]]];
AA[$UA2 AA[$UA2 AA[$UA2 AA[{$NOTUA} W {ua2!}]]]];
```

Here is the log from executing dme-2el.tcl:

```
Reading file: dme-martin.ccs
  Circuit DME-MARTIN-FALSE
    Multi-way composition ... OK
    Minimization ... (78s, 161t, 161v) OK
  Circuit DME-MARTIN-TRUE
    Multi-way composition ... OK
    Minimization ... (78s, 161t, 161v) OK

Copying process:  DME-MARTIN-FALSE --> dme-false... OK
Copying process:  DME-MARTIN-TRUE --> dme-true... OK
Reading file: dme-2el.ccs
  Circuit DME
    Multi-way composition ...
    WARNING: Circuit dme-true#1 has hazards!
    WARNING: Replaced with process dme-true#1H ...
    WARNING: Circuit dme-false#1 has hazards!
    WARNING: Replaced with process dme-false#1H ...OK
    Minimization ... (13s, 20t, 20v) OK


======================
INFO ABOUT PROCESSES
======================

DME circuit has 13 states / 20 transitions / 79 BDD nodes.

====================
CHECKING PROPERTIES
====================

ACTL/ACTLW model checking on process DME
Sistem vsebuje zagatno stanje

AAX {false} OR EEF AAX {false} ==> FALSE

Sistem vsebuje divergentno stanje

(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})) ==> FALSE

Dana zahteva bo ali odobrena ali pa umaknjena

AA[{NOT ur1!} W {ur1!} NOT EEG {(NOT ua1!) AND (NOT ur1?)}] ==> TRUE

AA[{NOT ur2!} W {ur2!} NOT EEG {(NOT ua2!) AND (NOT ur2?)}] ==> TRUE

# After an acknowledge is sent (removed),
# it will not be removed (sent) before the user requests this.

AAG [ua1!] AA[{NOT ua1!} W {ur1?}] ==> TRUE

AAG [ua2!] AA[{NOT ua2!} W {ur2?}] ==> TRUE

# After an user gets acknowledge for the first time,
# other users will not get acknowledge until his acknowledge is removed.
```

```
AA[{NOT ua1!} W {ua1!} AA[{(NOT ua1! AND NOT ua2!)} W {ua1!}]] ==> FALSE

AA[{NOT ua2!} W {ua2!} AA[{(NOT ua1! AND NOT ua2!)} W {ua2!}]] ==> FALSE

# After an user gets acknowledge for the second time,
# other users will not get acknowledge until his acknowledge is removed.

AA[{NOT ua1!} W {ua1!} AA[{NOT ua1!} W {ua1!} AA[{NOT ua1!} W {ua1!} AA[{(NOT ua1! AND NOT ua2!)} W
{ua1!}]]]] ==> FALSE

AA[{NOT ua2!} W {ua2!} AA[{NOT ua2!} W {ua2!} AA[{NOT ua2!} W {ua2!} AA[{(NOT ua1! AND NOT ua2!)} W
{ua2!}]]]] ==> FALSE
```

PLEASE NOTE: There is an assumption that the composition of already minimised processes does not represent the circuit trustworthy. Thus, the theoretical aspects of this work should be revised.

Files dme-martin.v and dme-smv.v are alternative description of DME cells. Here is the log from executing test-dme-verilog.tcl:

```
Reading file: dme-martin.ccs
  Circuit DME-MARTIN-FALSE
    Multi-way composition ... OK
    Minimization ... (78s, 161t, 161v) OK
  Circuit DME-MARTIN-TRUE
    Multi-way composition ... OK
    Minimization ... (78s, 161t, 161v) OK

Reading file: dme-smv.ccs
  Circuit DME-SMV-FALSE
    Multi-way composition ... OK
    Minimization ... (50s, 108t, 108v) OK
  Circuit DME-SMV-TRUE
    Multi-way composition ... OK
    Minimization ... (50s, 108t, 108v) OK

Reading file: dme-martin.v

Reading file: tmpccs.ccs
  Circuit DME-MARTIN
    Multi-way composition ... OK
    Minimization ... (78s, 161t, 161v) OK

Reading file: dme-smv.v

Reading file: tmpccs.ccs
  Circuit DME-SMV
    Multi-way composition ... OK
    Minimization ... (50s, 108t, 108v) OK

Trace equivalence checking between DME-MARTIN-FALSE and DME-MARTIN... OK
Testing equivalence checking between DME-MARTIN-FALSE and DME-MARTIN... OK
Strong observational equivalence checking between DME-MARTIN-FALSE and DME-MARTIN... OK
Trace equivalence checking between DME-SMV-FALSE and DME-SMV... OK
Testing equivalence checking between DME-SMV-FALSE and DME-SMV... OK
Strong observational equivalence checking between DME-SMV-FALSE and DME-SMV... OK
Trace equivalence checking between DME-MARTIN-TRUE and DME-MARTIN... NOT EQUIVALENT
Testing equivalence checking between DME-MARTIN-TRUE and DME-MARTIN... NOT EQUIVALENT
Strong observational equivalence checking between DME-MARTIN-TRUE and DME-MARTIN... NOT EQUIVALENT
Trace equivalence checking between DME-SMV-TRUE and DME-SMV... NOT EQUIVALENT
Testing equivalence checking between DME-SMV-TRUE and DME-SMV... NOT EQUIVALENT
Strong observational equivalence checking between DME-SMV-TRUE and DME-SMV... NOT EQUIVALENT
```