# Multiprocessor Shared-Memory Information Exchange (MSMIE)

Submitted by Savi Maharaj

July 12, 1996

## 1 Introduction

The Multiprocessor Shared-Memory Information Exchange (MSMIE) is a protocol for "inter-processor communications in distributed, microprocessor-based nuclear safety systems" which has been used in the embedded software of Westinghouse nucear systems designs. This small note contains three level of abstract descriptions expressed using VDM-SL (and one alternative modelling strategy).

More information about the protocol can be found in "Multiprocessor Shared-Memory Information Exchange" by L.L. Santoline et al. This was published in IEEE Transactions on Nuclear Science, 16(1), 1989. More information about the VDM model (and a corresponding PVS model) and the analysis of it can be found in "Proof in VDM: Case Studies" edited by J.C. Bicarregui. This is published by Springer in the FACIT series with ISBN 3540761861.

## 2 First specification

This is the first specification given in the report.

functions

```
1.0  count[@T] : @T × @T* → ℕ
 .1  count (s, ss) △
 .2    cases ss :
 .3      [] → 0,
 .4      others → if hd ss = s
 .5               then 1 + count[@T] (s, tl ss)
 .6               else count[@T] (s, tl ss)
 .7    end
```

types

```
2.0  Status = S | M | N | I;

3.0  MName = token
```

The slave operation is specified as follows:

```
4.0  state Σ of
 .1    b : Status*
 .2    ms : MName-set
```

```
.3    inv mk-Σ (b, ms) ≜
.4        len b = 3 ∧
.5        count[Status] (S, b) = 1 ∧
.6        count[Status] (M, b) ∈ {0, 1} ∧
.7        count[Status] (N, b) ∈ {0, 1} ∧
.8        (count[Status] (M, b) = 0  ⇔  ms = {})
.9    init s ≜ s = mk-Σ ([S, I, I], {})
.10 end
```

operations

```
5.0  slave ()
.1   ext wr b : (Status*)
.2   pre  true
.3   post ∀ i ∈ {1, 2, 3} ·
.4             (b̵ (i) = S  ⇒  b (i) = N) ∧
.5             (b̵ (i) = M  ⇒  b (i) = M) ;
```

Next is the acquire operation:

```
6.0  acq (l : MName)
.1   ext wr b : (Status*)
.2       wr ms : (MName-set)
.3   pre  (¬ (l ∈ ms)) ∧
.4       (∃ i ∈ {1, 2, 3} · b (i) = N ∨ b (i) = M)
.5   post ms = m̵s̵ ∪ {l} ∧
.6        ∀ i ∈ {1, 2, 3} ·
.7             if b̵ (i) = N ∧ m̵s̵ = {}
.8             then b (i) = M
.9             else b (i) = b̵ (i) ;
```

Now comes the release operation.

```
7.0  rel (l : MName)
.1   ext wr b : (Status*)
.2       wr ms : (MName-set)
.3   pre  l ∈ ms
.4   post ms = m̵s̵ \ {l} ∧
.5        ∀ i ∈ {1, 2, 3} ·
.6             if b̵ (i) = M ∧ ms = {}
.7             then b (i) ∈ {N, I} ∧ count[Status] (N, b) = 1
.8             else b (i) = b̵ (i)
```

# 3 Second specification

This is the more abstract specification.

types

```
8.0  Status1 = SII | SIN | SIM | SNM
```

The operations are specified as follows:

```
9.0  state Σ1 of
.1     bs : Status1
.2     ms : MName-set
.3     inv mk-Σ1 (bs, ms) ≜
.4         ms = {}  ⇔  bs ∈ {SII, SIN}
.5     init s ≜ s = mk-Σ1 (SII, {})
.6   end
```

operations

```
10.0 slave ()
.1   ext wr bs : Status1
.2       rd ms : (MName-set)
.3   pre true
.4   post (←bs ∈ {SII, SIN}  ⇒  bs = SIN) ∧
.5        (←bs ∈ {SIM, SNM}  ⇒  bs = SNM) ;
```

```
11.0 acq (l : MName)
.1   ext wr bs : Status1
.2       wr ms : (MName-set)
.3   pre (¬ (l ∈ ms)) ∧ (¬ (bs = SII))
.4   post ms = ←ms ∪ {l} ∧
.5        if ←ms = {}
.6        then bs = SIM
.7        else bs = ←bs ;
```

```
12.0 rel (l : MName)
.1   ext wr bs : Status1
.2       wr ms : (MName-set)
.3   pre l ∈ ms
.4   post ms = ←ms \ {l} ∧
.5        if ms = {}
.6        then bs = SIN
.7        else bs = ←bs
```

# 4 Third specification

This is the most abstract specification.

The operations are specified as follows:

```
13.0 state Σ0 of
.1     b : B
.2     ms : MName-set
.3     inv mk-Σ0 (b, ms) ≜
.4         b = false  ⇒  ms = {}
.5     init s ≜ s = mk-Σ0 (false, {})
.6   end
```

operations

14.0 $slave\,()$
.1 ext wr $b : \mathbb{B}$
.2 pre true
.3 post $b =$ true ;

15.0 $acq\,(l : MName)$
.1 ext rd $b : \mathbb{B}$
.2 wr $ms : (MName\text{-set})$
.3 pre $b =$ true $\wedge \neg\,(l \in ms)$
.4 post $ms = \overleftarrow{ms} \cup \{l\}$ ;

16.0 $rel\,(l : MName)$
.1 ext wr $ms : (MName\text{-set})$
.2 pre $l \in ms$
.3 post $ms = \overleftarrow{ms} \setminus \{l\}$

# 5 Alternative view of MSMIE

types

17.0 $BName = \mathrm{N1} \mid \mathrm{N2} \mid \mathrm{N3};$

18.0 $MName =$ token

functions

19.0 $nil\text{-}or\text{-}different[@A] : [[@A]]^* \to \mathbb{B}$
.1 $nil\text{-}or\text{-}different\,(l) \triangleq$
.2 $\forall\, i \in$ inds $l \cdot$
.3 $l\,(i) =$ nil $\vee$
.4 $(\forall\, j \in$ inds $l \cdot l\,(i) = l\,(j) \;\Rightarrow\; i = j)$

The operations are specified as follows:

20.0 state $\Sigma 3$ of
.1 $s : BName$
.2 $m : [BName]$
.3 $n : [BName]$
.4 $ms : MName\text{-set}$
.5 inv mk-$\Sigma 3\,(s, n, m, ms) \triangleq$
.6 $(m =$ nil $\;\Leftrightarrow\; ms = \{\}) \wedge$
.7 $nil\text{-}or\text{-}different[BName]\,([s, m, n])$
.8 init $s \triangleq s =$ mk-$\Sigma 3\,(\mathrm{N1},$ nil , nil , $\{\})$
.9 end

operations

21.0 $slave\,()$
.1 ext rd $m : ([BName])$
.2 wr $n : ([BName])$
.3 wr $s : BName$
.4 pre true

.5 post $n = \overleftarrow{s}$ ;

22.0 $acq\,(l : MName)$
.1 ext wr $ms : (MName\text{-set})$
.2 wr $n, m : ([BName])$
.3 pre $(\neg\,(l \in ms)) \wedge \neg\,(n = \mathsf{nil} \wedge m = \mathsf{nil})$
.4 post $ms = \overleftarrow{ms} \cup \{l\} \wedge$
.5 $(\neg\,(\overleftarrow{ms} = \{\}) \Rightarrow m = \overleftarrow{m} \wedge n = \overleftarrow{n}) \wedge$
.6 $(\overleftarrow{ms} = \{\} \Rightarrow m = \overleftarrow{n} \wedge n = \mathsf{nil})$ ;

23.0 $rel\,(l : MName)$
.1 ext wr $ms : (MName\text{-set})$
.2 wr $n, m : ([BName])$
.3 pre $l \in ms$
.4 post $ms = \overleftarrow{ms} \setminus \{l\} \wedge$
.5 $(\neg\,(ms = \{\}) \Rightarrow m = \overleftarrow{m} \wedge n = \overleftarrow{n}) \wedge$
.6 $(ms = \{\} \wedge \neg\,(n = \mathsf{nil}) \Rightarrow n = \overleftarrow{n} \wedge m = \mathsf{nil}) \wedge$
.7 $(ms = \{\} \wedge n = \mathsf{nil} \Rightarrow n = \overleftarrow{m} \wedge m = \mathsf{nil})$

# Index