Weryfikacja Futurebus+

Marcin Poturalski
m.poturalski@students.mimuw.edu.pl

12 kwietnia 2005

PLAN

- Kontekst
- Opis protokołu
- Modelowanie
- Weryfikacja
- Metoda niezmiennika

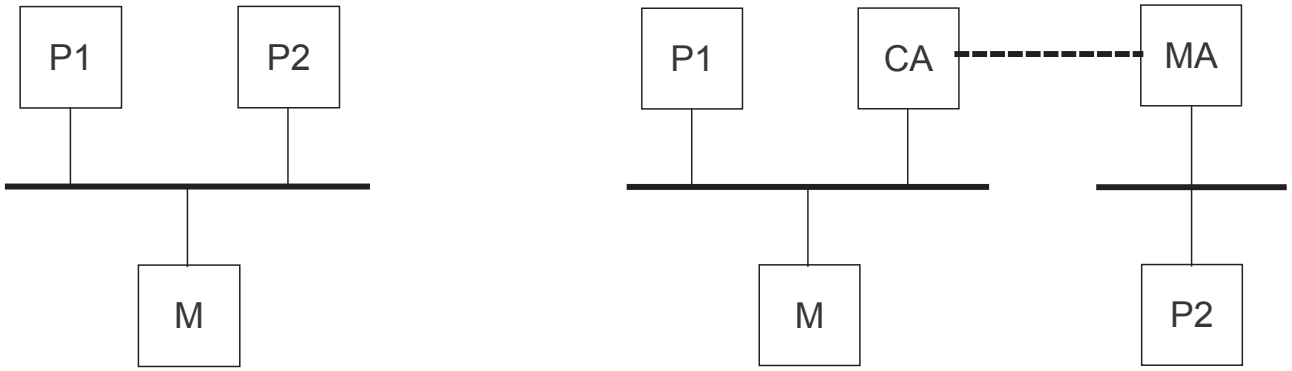
KONTEKST

Futurebus+ to standard opisujący wydajną, asynchroniczną szynę pamięci, zaproponowany przez IEEE w 1991 roku. Projektantom przyświecał cel stworzenia "ostatecznego" standardu szyny, który nie zostawiłby dużego pola do popisu ich następnikom.

W ramach standardu zdefiniowano protokół odpowiadający za zachowanie spójności pamięci cache procesorów podłączonych do szyny. Po opublikowaniu propozycji standardu IEEE 896.1-1991, autorzy (1) zajeli się zweryfikowaniem poprawności tego właśnie protokołu. Udało im sie znaleźć istotne błędy, a także odkryć pewne niejasności w specyfikacji. Było to pierwsze zakończone tak znacznym sukcesem zastosowanie formalnej weryfikacji do propozycji standardu IEEE.

OPIS PROTOKOŁU

Protokół działa w środowisku jednej lub wielu połączonych mostami szyn. Do każdej szyny podłączone może być wiele procesorów i modułów pamięci, oraz moduły odpowiedzialne za komunikację między szynami (mosty).



OPIS PROTOKOŁU

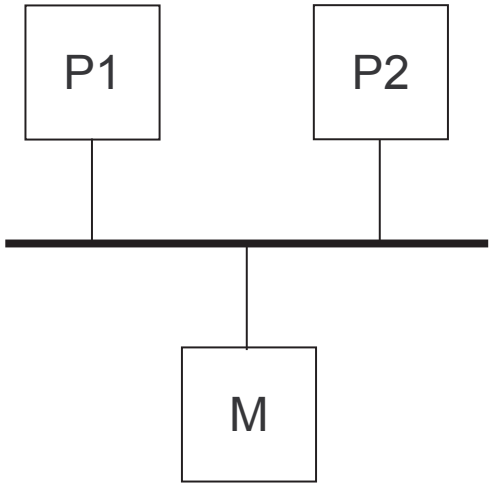
Aby zachować spójność w ramach pojedynczej szyny, wszystkie moduły mają obowiązek nieustannie nasłuchiwać co się dzieje na szynie (*snoop*). Za zachowanie zgodności pomiędzy szynami odpowiadają mosty. Moduły komunikują się ze sobą za pomocą transakcji. W celach wydajnościowych dopuszczono możliwość podzielenia transakcji (*split*). Transakcja jest przerywana, a szyna zwalniana. Musi być jednak dokończona w ramach dodatkowej transakcji.

Za niepodzielna jednostkę pamięci przyjęto *cache line*. Dla każdej takiej linii każdy moduł cache posiada stan, który mówimy o tym, jakie ma do niej prawa:

invalid, *shared-unmodified*, *exclusive-unmodified* lub *exclusive-modified*.

OPIS PROTOKOŁU

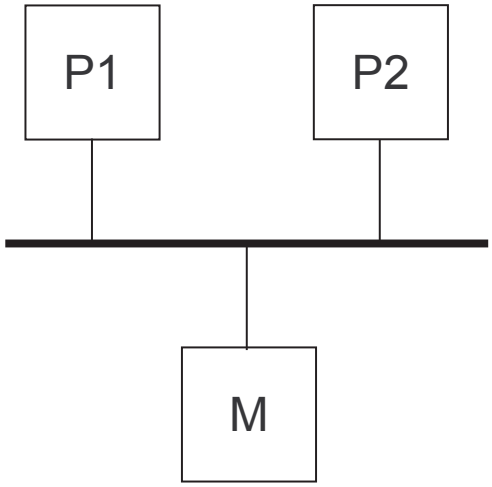
Przykład dla pojedynczej szyny:



1. P1 i P2 są w stanie *invalid*.

OPIS PROTOKOŁU

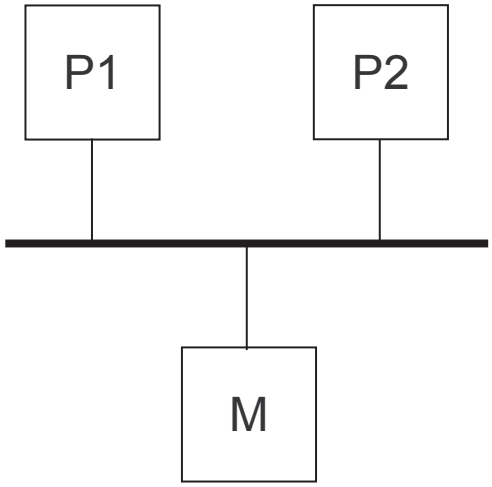
Przykład dla pojedynczej szyny:



1. P1 i P2 są w stanie *invalid*.
2. P1 rozpoczyna transakcję *read-shared*, aby otrzymać kopię do odczytu.

OPIS PROTOKOŁU

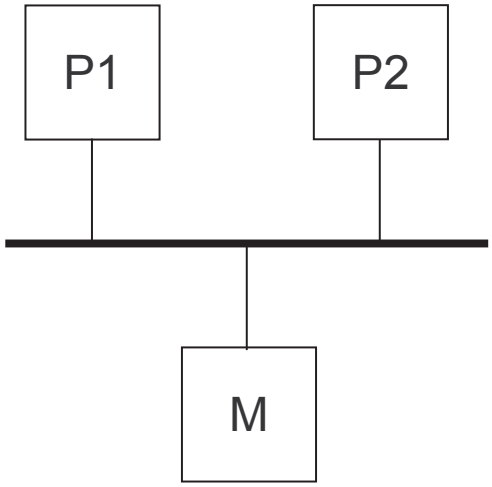
Przykład dla pojedynczej szyny:



1. P1 i P2 są w stanie *invalid*.
2. P1 rozpoczyna transakcję *read-shared*, aby otrzymać kopię do odczytu.
3. P2 nasłuchuje i decyduje się tę kopię pozyskać (*snarfing*).

OPIS PROTOKOŁU

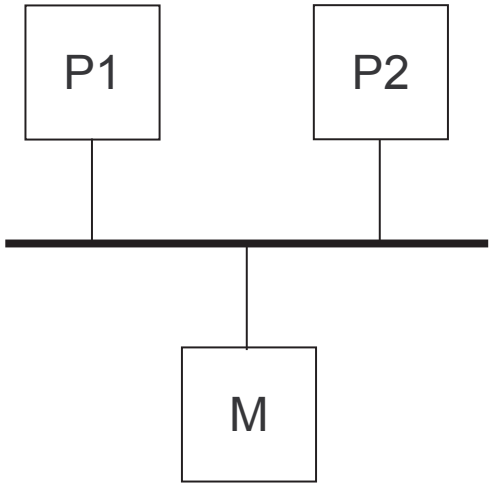
Przykład dla pojedynczej szyny:



1. P1 i P2 są w stanie *invalid*.
2. P1 rozpoczyna transakcję *read-shared*, aby otrzymać kopię do odczytu.
3. P2 nasłuchuje i decyduje się tę kopię pozyskać (*snarfing*).
4. P1 i P2 otrzymują kopię linii, oba są w stanie *shared-unmodified*. Koniec transakcji.

OPIS PROTOKOŁU

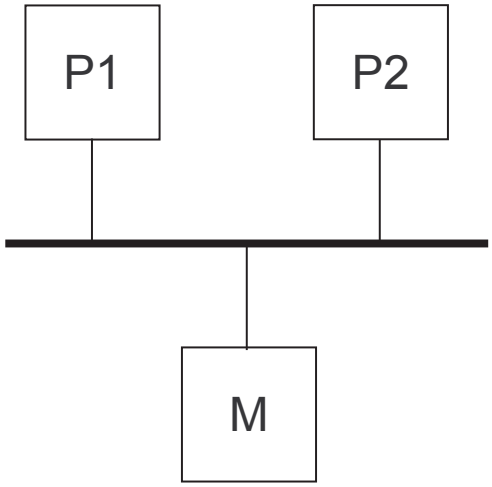
Przykład dla pojedynczej szyny:



1. P1 i P2 są w stanie *invalid*.
2. P1 rozpoczyna transakcję *read-shared*, aby otrzymać kopię do odczytu.
3. P2 nasłuchuje i decyduje się tę kopię pozyskać (*snarfing*).
4. P1 i P2 otrzymują kopię linii, oba są w stanie *shared-unmodified*. Koniec transakcji.
5. P1 chce pisać. Rozpoczyna transakcję *invalidate*.

OPIS PROTOKOŁU

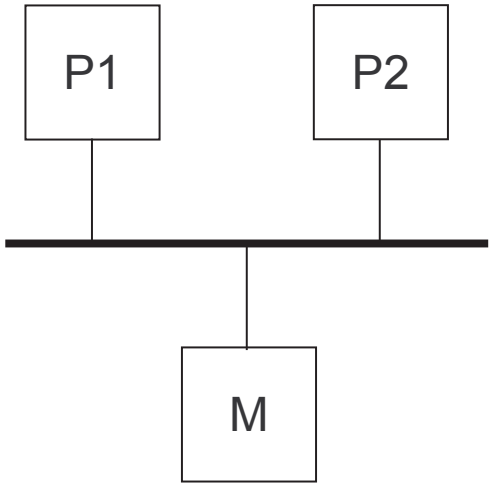
Przykład dla pojedynczej szyny:



1. P1 i P2 są w stanie *invalid*.
2. P1 rozpoczyna transakcję *read-shared*, aby otrzymać kopię do odczytu.
3. P2 nasłuchuje i decyduje się tę kopię pozyskać (*snarfing*).
4. P1 i P2 otrzymują kopię linii, oba są w stanie *shared-unmodified*. Koniec transakcji.
5. P1 chce pisać. Rozpoczyna transakcję *invalidate*.
6. P2 nasłuchuje i usuwa linię (stan *invalid*).

OPIS PROTOKOŁU

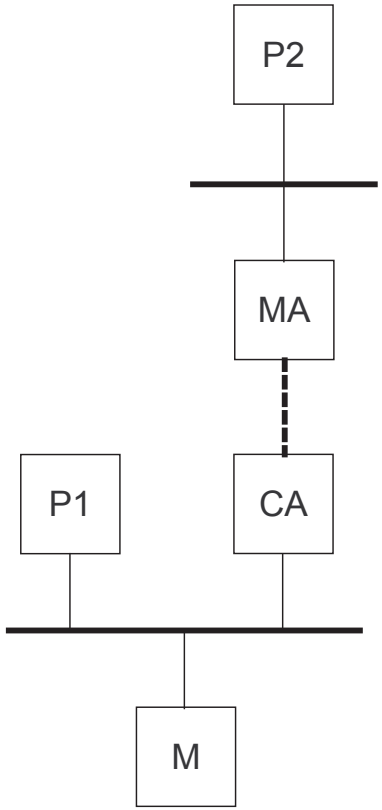
Przykład dla pojedynczej szyny:



1. P1 i P2 są w stanie *invalid*.
2. P1 rozpoczyna transakcję *read-shared*, aby otrzymać kopię do odczytu.
3. P2 nasłuchuje i decyduje się tę kopię pozyskać (*snarfing*).
4. P1 i P2 otrzymują kopię linii, oba są w stanie *shared-unmodified*. Koniec transakcji.
5. P1 chce pisać. Rozpoczyna transakcję *invalidate*.
6. P2 nasłuchuje i usuwa linię (stan *invalid*).
7. P1 przechodzi do stanu *exclusive-modified*. Koniec transakcji.

OPIS PROTOKOŁU

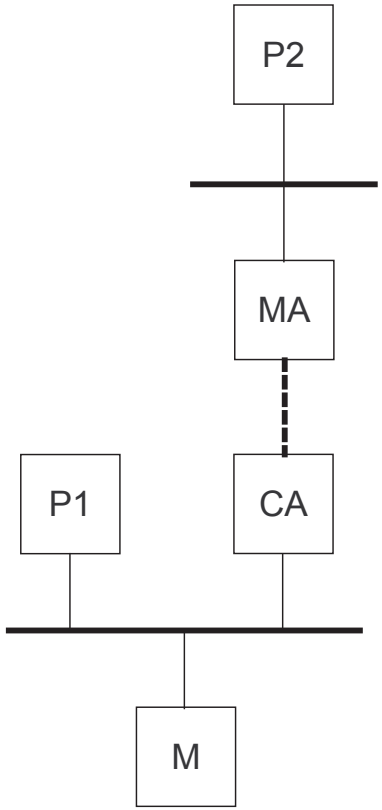
Przykład dla dwóch szyn:



1. P1 i P2 są w stanie *invalid*.

OPIS PROTOKOŁU

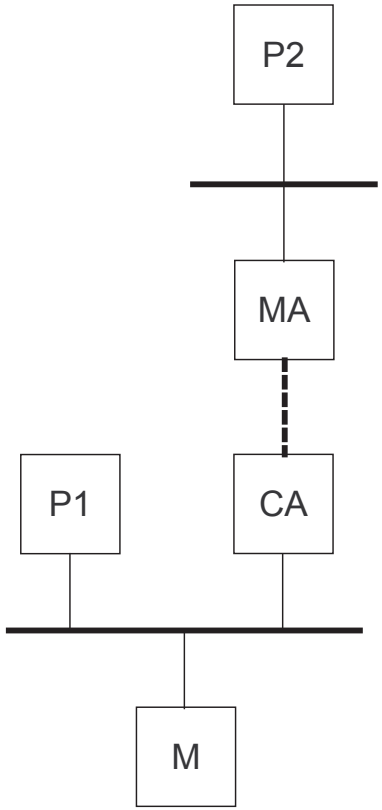
Przykład dla dwóch szyn:



1. P1 i P2 są w stanie *invalid*.
2. P2 rozpoczyna transakcję *read-modified*, aby otrzymać kopię do zapisu.

OPIS PROTOKOŁU

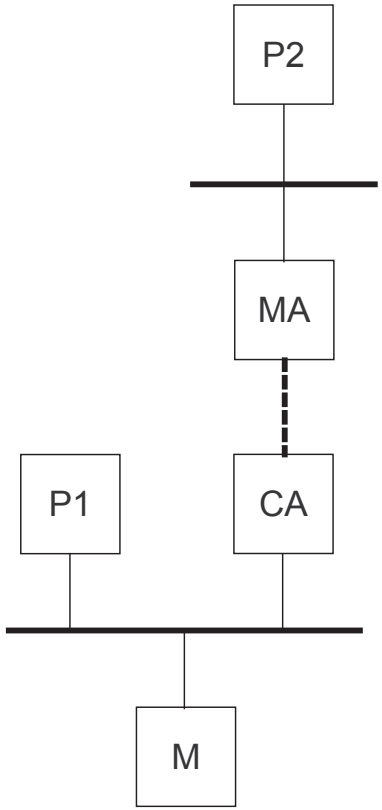
Przykład dla dwóch szyn:



1. P1 i P2 są w stanie *invalid*.
2. P2 rozpoczyna transakcję *read-modified*, aby otrzymać kopię do zapisu.
3. MA dzieli tę transakcję, aby tę kopię pozyskać. Przekazuje zapytanie do CA.

OPIS PROTOKOŁU

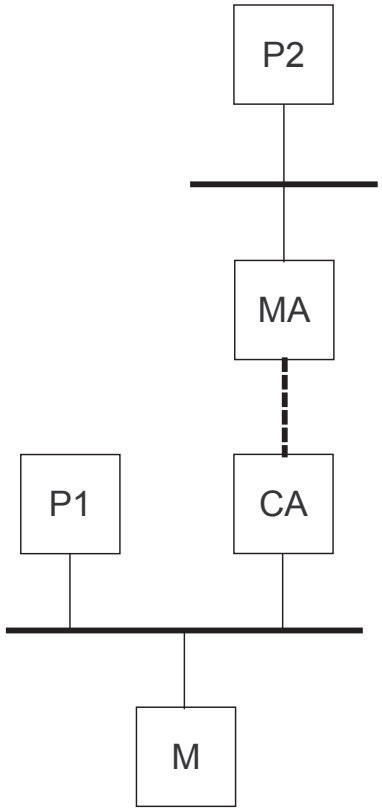
Przykład dla dwóch szyn:



1. P1 i P2 są w stanie *invalid*.
2. P2 rozpoczyna transakcję *read-modified*, aby otrzymać kopię do zapisu.
3. MA dzieli tę transakcję, aby tę kopię pozyskać. Przekazuje zapytanie do CA.
4. CA rozpoczyna *read-modified* na swojej szynie.

OPIS PROTOKOŁU

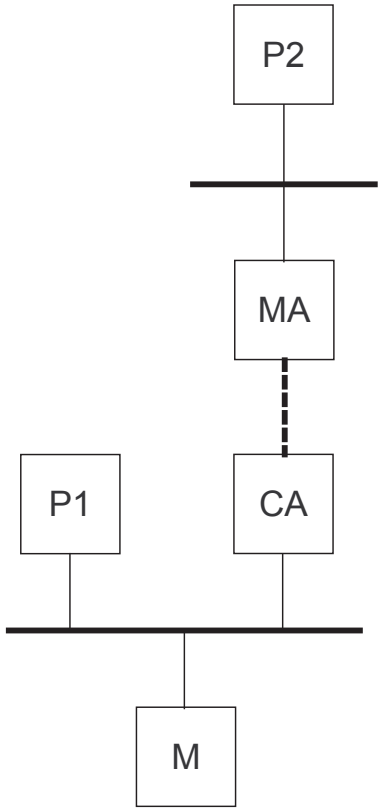
Przykład dla dwóch szyn:



1. P1 i P2 są w stanie *invalid*.
2. P2 rozpoczyna transakcję *read-modified*, aby otrzymać kopię do zapisu.
3. MA dzieli tę transakcję, aby tę kopię pozyskać. Przekazuje zapytanie do CA.
4. CA rozpoczyna *read-modified* na swojej szynie.
5. M dostarcza linię, CA przekazuje linię do MA. P nie wolno danych przechwycić.

OPIS PROTOKOŁU

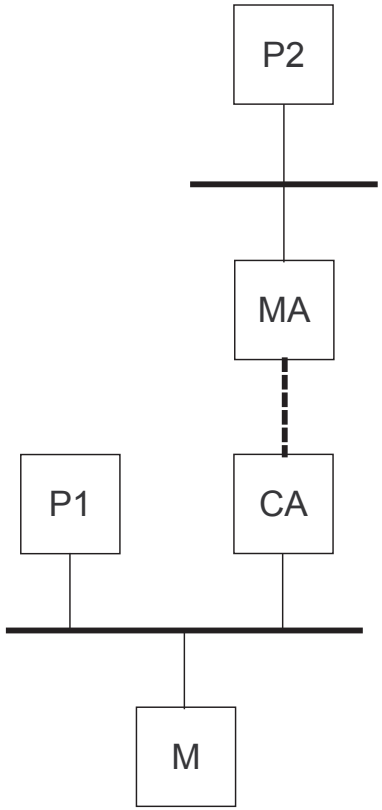
Przykład dla dwóch szyn:



1. P1 i P2 są w stanie *invalid*.
2. P2 rozpoczyna transakcję *read-modified*, aby otrzymać kopię do zapisu.
3. MA dzieli tę transakcję, aby tę kopię pozyskać. Przekazuje zapytanie do CA.
4. CA rozpoczyna *read-modified* na swojej szynie.
5. M dostarcza linię, CA przekazuje linię do MA. P nie wolno danych przechwycić.
6. MA rozpoczyna *modified-response* aby dokończyć przerwaną transakcję. P2 otrzymuje kopię i przechodzi do *exclusive-modified*.

OPIS PROTOKOŁU

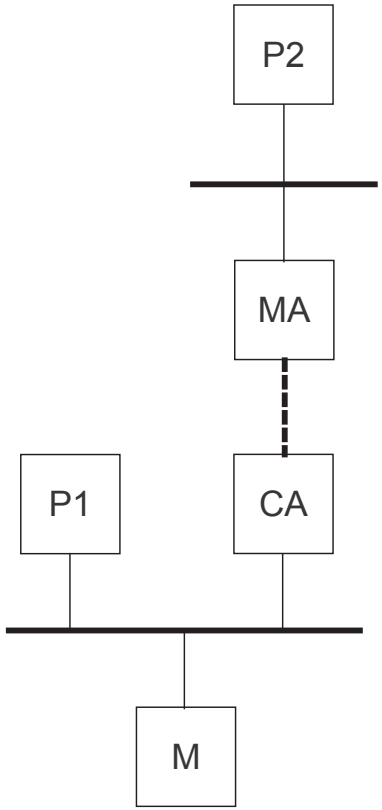
Przykład dla dwóch szyn:



2. P2 rozpoczyna transakcję *read-modified*, aby otrzymać kopię do zapisu.
3. MA dzieli tę transakcję, aby tę kopię pozyskać. Przekazuje zapytanie do CA.
4. CA rozpoczyna *read-modified* na swojej szynie.
5. M dostarcza linię, CA przekazuje linię do MA. P nie wolno danych przechwycić.
6. MA rozpoczyna *modified-response* aby dokończyć przerwaną transakcję. P2 otrzymuje kopię i przechodzi do *exclusive-modified*.
7. P1 ropoczyna *read-shared*.

OPIS PROTOKOŁU

Przykład dla dwóch szyn:



3. MA dzieli tę transakcję, aby tę kopię pozyskać. Przekazuje zapytanie do CA.

4. CA rozpoczyna *read-modified* na swojej szynie.

5. M dostarcza linię, CA przekazuje linię do MA. P nie wolno danych przechwycić.

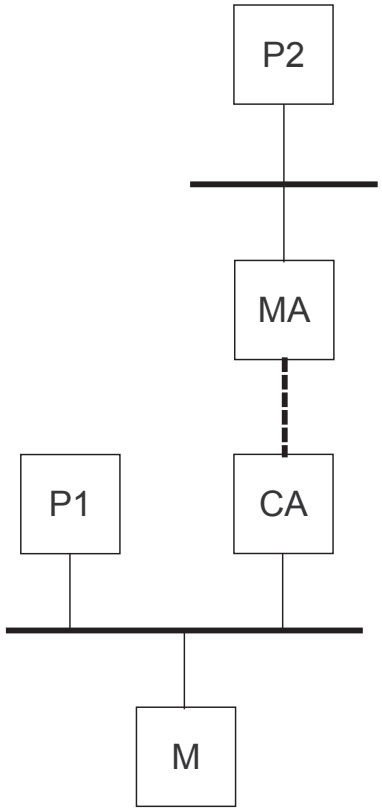
6. MA rozpoczyna *modified-response* aby dokończyć przerwaną transakcję. P2 otrzymuje kopię i przechodzi do *exclusive-modified*.

7. P1 ropoczyna *read-shared*.

8. CA interweniuje i dzieli transakcję. Wystosowuje zapytanie do MA.

OPIS PROTOKOŁU

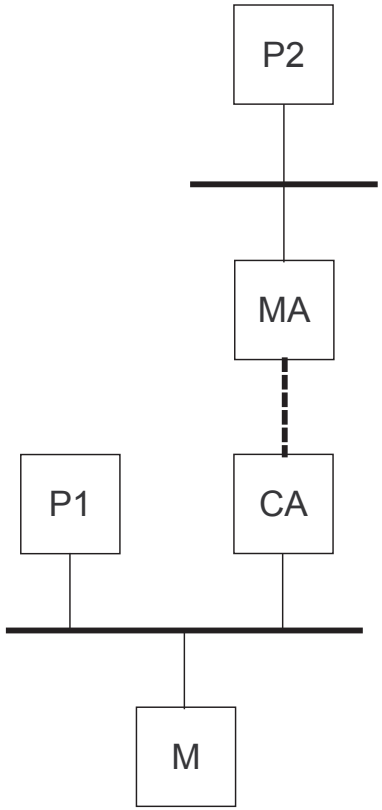
Przykład dla dwóch szyn:



4. CA rozpoczyna *read-modified* na swojej szynie.
5. M dostarcza linię, CA przekazuje linię do MA. P nie wolno danych przechwycić.
6. MA rozpoczyna *modified-response* aby dokończyć przerwaną transakcję. P2 otrzymuje kopię i przechodzi do *exclusive-modified*.
7. P1 ropoczyna *read-shared*.
8. CA interweniuje i dzieli transakcję. Wystosowuje zapytanie do MA.
9. MA rozpoczyna *read-shared*. P2 interweniuje i dostarcza linię do MA, które przekazuje ją do CA.

OPIS PROTOKOŁU

Przykład dla dwóch szyn:



5. M dostarcza linię, CA przekazuje linię do MA. P nie wolno danych przechwycić.

6. MA rozpoczyna *modified-response* aby dokończyć przerwaną transakcję. P2 otrzymuje kopię i przechodzi do *exclusive-modified*.

7. P1 ropoczyna *read-shared*.

8. CA interweniuje i dzieli transakcję. Wystosowuje zapytanie do MA.

9. MA rozpoczyna *read-shared*. P2 interweniuje i dostarcza linię do MA, które przekazuje ją do CA.

10. CA rozpoczyna *shared-response*, aby zakończyć przerwaną transakcję.

OPIS PROTOKOŁU

Podsumowując.

Stany, w których może się znajdować cache (odnośnie każdej linii pamięci):

invalid, *shared-unmodified*, *exclusive-unmodified* lub *exclusive-modified*.

Transakcje, które mogą się w ramach szyny odbywać to:

read-shared, *read-modified*, *invalidate*, *copyback*, *shared-response* i *modified-response*.

MODELOWANIE

Opis protokołu zawarty w dokumencie IEEE 896.1-1991 składa się z dwóch części.

Pierwsza, nieformalna, opisuje prozą zachowanie protokołu, ale nie omawia wszystkich scenariuszy.

Druga, która jest faktyczną specyfikacją, do opisu używa tzw. atrybutów. Są to w istocie flagi binarne, wraz z regułami, które mowią w jakich okolicznościach (tzn. stanu innych atrybutów) je ustawiać i zerować, oraz nie do końca formalnym wyjaśnieniem, za te flagi odpowiadają.

Nie wszystkie aspekty zostały porządnie wyspecyfikowane. W szczególności nie jest opisany sposób w jaki mają działać mosty.

Aby przystąpić do formalnej weryfikacji, trzeba było zbudować formalny model. Dzięki temu udało się wykryć wszelkie niejasności w specyfikacji i już samo to było pewnym sukcesem autorów.

MODELOWANIE

W czasie modelowania autorzy wielokrotnie upraszczali i opuszczali różne aspekty protokołu.

- Usunięto niskopoziomowe zachowanie szyny. W zbudowanym modelu jednej transkacji odpowiada jeden krok (wszystkie moduły jednej szyny złożone są synchoronicznie).
- Uproszczono przydzielanie szyny. Moduł, który ma być *master*, jest wybierany niedeterministycznie.
- Jest tylko jeden moduł pamięci, który posiada tylko jedną linię. Linia ta składa się z pojedynczego bitu.
- Zrezygownano z transakcji *read-invalid* i *write-invalid* zaprojektowanych z myślą o DMA.
- Wiele innych rzeczy uproszczono, korzystając gęsto z niedeterminizmu.
- Mosty zostały wymodelowane mocno niedeterministycznie, a szyny złożone są asynchronicznie.

MODELOWANIE

Fragment kodu modułu procesora w SMV:

```
next(state) :=
  case
  ...
  master:
    case
    CMD = read-shared:
      case
      state = invalid:
        case
        !SR & !TF:  exclusive-unmodified;
        !SR:  shared-unmodified;
        1:  invalid;
        esac;
      ...
      esac;
    ...
    esac;
```

state – stan procesora

CMD – transakcja, która odbywa się w tej turze

SR – czy transakcja jest dzielona

TF – czy transakcja jest *snarf*'owana przez jakiś moduł

tf – czy transakcja jest *snarf*'owana przez ten moduł

MODELOWANIE

Fragment kodu modułu procesora w SMV c.d.:

```
CMD = read-shared:
  case
  state in { invalid, shared-unmodified }:
    case
    !tf:  invalid;
    !SR:  shared-unmodified;
    1:  state
    esac;
  ...
  esac;
...
esac;
```

state – stan procesora

CMD – transakcja, która odbywa się w tej turze

SR – czy transakcja jest dzielona

TF – czy transakcja jest *snarf*'owana przez jakiś moduł

tf – czy transakcja jest *snarf*'owana przez ten moduł

WERYFIKACJA

W ramach uzyskania spójności autorzy specyfikują kilka formuł mówiących o ogólnej poprawności protokołu (nie tylko spójności cache'y).

Dla każdego modułu d ma zachodzić:

$$\mathbf{AG}(\neg\ d.bus\_error\ \wedge\ \neg\ d.error)$$

bus_error jest ustawiane, gdy moduł zauważy nielegalną kombinację sygnałów na szynie, $error$ zaś gdy zaobserwuje transakcję, która nie ma prawa zajść zgodnie z jego stanem.

Dla dowolnych różnych procesorów p_1 i p_2:

$$\mathbf{AG}(p_1.writable \Rightarrow \neg\ p_2.readable)$$

$$\mathbf{AG}(p_1.readable \wedge p_2.readable \Rightarrow p_1.data = p_2.data)$$

WERYFIKACJA

Dla dowolnego procesora p i pamięci m:

$$\mathbf{AG}(p.readable \wedge \neg\ m.memory\_line\_modified \Rightarrow p.data = m.data)$$

I wreszcie dla dowolnego procesora p:

$$\mathbf{AG\ EF}\ p.readable \wedge \mathbf{AG\ EF}\ p.writable$$

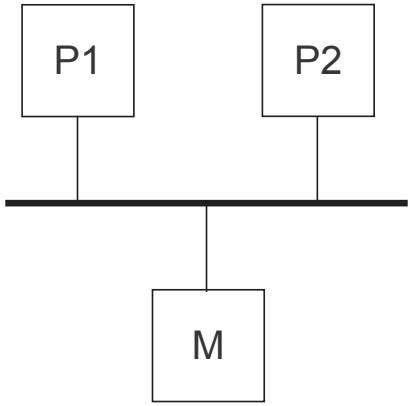
WERYFIKACJA

W ramach weryfikacji autorzy sprawdzali konfiguracje zawierające do 8 procesorów i do 3 szyn. Dla największej sprawdzonej konfiguracji SMV zużył około 150.000 węzłów.

Wykryto szereg błędów, zarówno w konfiguracji z pojedynczą szyną, jak i z wieloma szynami. W każdym przypadku autorzy zaproponowali taką modyfikację standardu, która znaleziony błąd eliminuje. W rezultacie uzyskali nową wersję standardu, która wydaje się być dużo bardziej pewna, niż pierwotna.

WERYFIKACJA

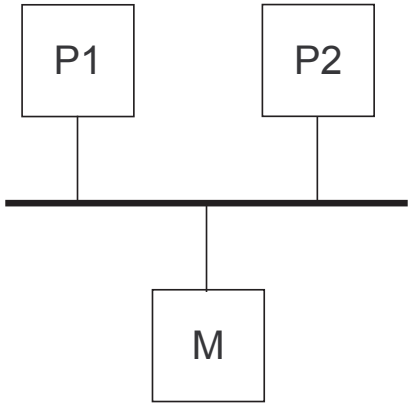
Przykładowy błąd dla konfiguracji jednoszynowej:



1. P1 posiada kopię *exclusive-unmodified*.

WERYFIKACJA

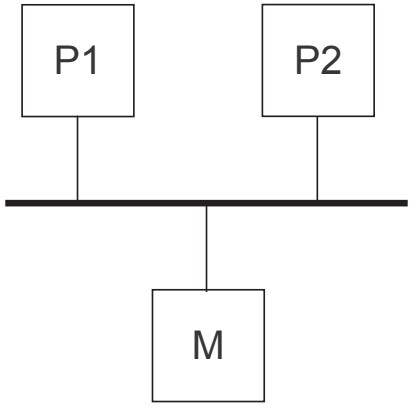
Przykładowy błąd dla konfiguracji jednoszynowej:



1. P1 posiada kopię *exclusive-unmodified*.
2. P2 rozpoczyna transakcję *read-shared*.

WERYFIKACJA

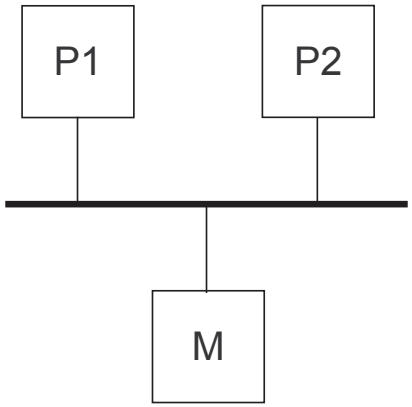
Przykładowy błąd dla konfiguracji jednoszynowej:



1. P1 posiada kopię *exclusive-unmodified*.
2. P2 rozpoczyna transakcję *read-shared*.
3. M dzieli transakcję.

WERYFIKACJA

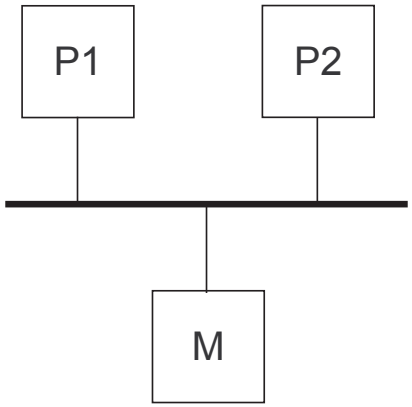
Przykładowy błąd dla konfiguracji jednoszynowej:



1. P1 posiada kopię *exclusive-unmodified*.
2. P2 rozpoczyna transakcję *read-shared*.
3. M dzieli transakcję.
4. P1 widzi to, ale pozostaje w stanie *exclusive-unmodified*.

WERYFIKACJA

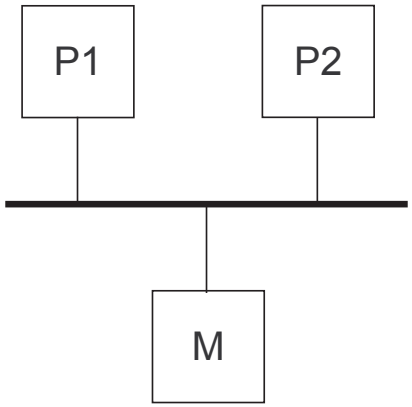
Przykładowy błąd dla konfiguracji jednoszynowej:



1. P1 posiada kopię *exclusive-unmodified*.
2. P2 rozpoczyna transakcję *read-shared*.
3. M dzieli transakcję.
4. P1 widzi to, ale pozostaje w stanie *exclusive-unmodified*.
5. P1 pisze do pamięci i zmienia stan na *exclusive-modified*. Robi to bez wykonywania transakcji.

WERYFIKACJA

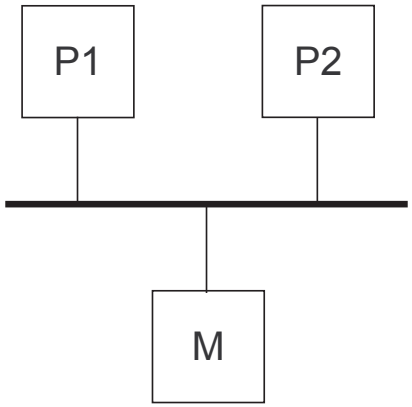
Przykładowy błąd dla konfiguracji jednoszynowej:



1. P1 posiada kopię *exclusive-unmodified*.
2. P2 rozpoczyna transakcję *read-shared*.
3. M dzieli transakcję.
4. P1 widzi to, ale pozostaje w stanie *exclusive-unmodified*.
5. P1 pisze do pamięci i zmienia stan na *exclusive-modified*. Robi to bez wykonywania transakcji.
6. M kończy rozpoczętą transakcję poprzez *shared-response*.

WERYFIKACJA

Przykładowy błąd dla konfiguracji jednoszynowej:



1. P1 posiada kopię *exclusive-unmodified*.
2. P2 rozpoczyna transakcję *read-shared*.
3. M dzieli transakcję.
4. P1 widzi to, ale pozostaje w stanie *exclusive-unmodified*.
5. P1 pisze do pamięci i zmienia stan na *exclusive-modified*. Robi to bez wykonywania transakcji.
6. M kończy rozpoczętą transakcję poprzez *shared-response*.
7. P2 otrzymuje nieaktualny stan linii i przechodzi do *shared-unmodified*.

WERYFIKACJA

Autorzy proponują rozwiązać ten problem poprzez nakazanie procesorowi zmianę stanu na *shared-unmodified*, kiedy widzi, że pamięć dzieli transakcję.

1. P1 posiada kopię *exclusive-unmodified*.
2. P2 rozpoczyna transakcję *read-shared*.
3. M dzieli transakcję.
4. P1 widzi to, ale pozostaje w stanie *exclusive-unmodified*. $\Leftarrow$ NIEMOŻLIWE

METODA NIEZMIENNIKA

Przedstawię teraz zaczerpniętą z książki (2) metodę niezmiennika, która pozwala dowodzić własności dla dowolnie dużych klas modeli i jej zastosowanie do *Future-bus+*.

Jeżeli mamy rodzinę struktur $\mathcal{M}$ oraz zwrotną i przechodnią relację $\succeq$ to *niezmiennikiem* nazywamy taką strukturę $\mathcal{I}$, że dla dowolnego $M \in \mathcal{M}$ zachodzi $\mathcal{I} \succeq M$.

Jeśli $\succeq$ jest równoważnością bisymulacyjną, to:

$$\mathcal{I} \models \varphi \iff M \models \varphi$$

gdzie φ jest dowolną formułą $CTL^\star$.

Jeśli $\succeq$ jest quasi-porządkiem symulacyjnym, to:

$$\mathcal{I} \models \varphi \Rightarrow M \models \varphi$$

gdzie φ jest dowolną formułą $ACTL^\star$.

METODA NIEZMIENNIKA

Nas będzie interesował pewien szczególny przypadek rodziny struktur: $\{P^i\}_{i=1}^{\infty}$.

Podnoszenie do potęgi odnosi się do operatora dwuargumentowego $\|$. Ma być on monotoniczny ze względu na $\succeq$, tzn.

$$P_1 \succeq Q_1 \wedge P_2 \succeq Q_2 \;\Rightarrow\; P_1 \| P_2 \succeq Q_1 \| Q_2$$

Przez łatwą indukcję można pokazać, że $\mathcal{I}$ jest dobrym niezmiennikiem dla rodziny $\{P^i\}_{i=1}^{\infty}$, jeśli $\mathcal{I} \succeq P$ oraz $\mathcal{I} \succeq \mathcal{I} \| P$.

(1) $\mathcal{I} \succeq P$

(2) $\mathcal{I} \succeq P^{n-1} \Rightarrow \mathcal{I} \| P \succeq P^n \Rightarrow \mathcal{I} \succeq P^n$

METODA NIEZMIENNIKA

Jak to zastosować do Futurebus+ ?

Można w SMV zdefiniować moduł P procesora tak, aby n-krotne złożenie synchroniczne odpowiadało konfiguracji z jedną szyną i n procesorami. Jeśli uda nam się znaleźć niezmiennik dla klasy struktur $\{P^i\}_{i=1}^{\infty}$ (ze względu na quasi-porządek symulacyjny) to możemy sprawdzając formuły dla niezmiennika wnioskować o konfiguracji jednoszynowej dla dowolnej liczby procesorów.

METODA NIEZMIENNIKA

Fragment kodu dla P dotyczący `cmd`:

```
next(cmd) :=
  case
    state = invalid & !master:
      { read-shared, read-modified, idle };
    state = exclusive-modified & !master:
      { copy-back, idle };
    state = exclusive-unmodified & !master:
      { copy-back, idle };
    state = shared-unmodified & !master:
      { invalidate, copy-back, idle };
    master: cmd;
    1: idle;
  esac;
```

P nie jest dobrym niezmiennikiem, ponieważ $P \| P$ ze stanu (`exclusive-modified, invalid`) może przejść zarówno do stanu, gdzie `cmd = copy-back`, jak i do stanu, gdzie `cmd = read-shared`, czego P nie potrafi dla żadnego stanu.

METODA NIEZMIENNIKA

Fragment kodu dla P' dotyczący `cmd`:

```
next(cmd) :=
  case
    state = invalid & !master:
      { copy-back, read-shared, read-modified, idle };
    state = exclusive-modified & !master:
      { copy-back, read-shared, read-modified, idle };
    state = exclusive-unmodified & !master:
      { copy-back, read-shared, read-modified, idle };
    state = shared-unmodified & !master:
      { invalidate, copy-back, idle };
    master: cmd;
    1: idle;
  esac;
```

P' jest już dobrym niezmiennikiem. Oczywiście $P' \succeq P$. Trzeba jeszcze pokazać $P' \succeq P' \| P$.

METODA NIEZMIENNIKA

Stan s (z P') jest w relacji z (s_1, s_2) (z $P' \| P$) jeśli zachodzą następujące warunki:

- jeśli s jest *invalid* to s_1 i s_2 są *invalid*
- jeśli s jest *shared-unmodified* to jeden z s_1, s_2 jest w *shared-unmodified* a drugi w *shared-unmodified* lub *invalid*
- jeśli s jest *exclusive-(un)modified* to jeden z s_1, s_2 jest *exclusive-(un)modified*, a drugi *invalid*

Sprawdzenie, że jest to w istocie quasi-porządek symulacyjny jest proste, choć żmudne.

BIBLIOGRAFIA

(1) E. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D. Long, K. McMillan, L. Ness, *Verification of the Futurebus+ Cache Coherence Protocol*, CMU-CS-92-206, 1992.

(2) Edmund M. Clarke, Orna Grumberg, Doron A. Peled, *Model Checking*, Ch. 15.2-3, MIT Press, 1999.

(3) K. Williams, R. Esser, *Verification of the Futurebus+ Cache Coherence protocol: A case study in model checking*, 2004.

(4) *http://granite.sru.edu/~stringer/fb.html*