An occam Model of XCHANs

Peter Welch (phw@kent.ac.uk)

CPA 2013 Fringe, Napier University, 26 August, 2013

Appeared at CPA 2013 (Communicating Process Architectures 2013) as a fringe lecture by *Peter Welch* of University of Kent. It goes into details of the **XCHAN** by modeling a possible implementation. Welch is showing this with a not yet implemented version of the **occam** programming language:

> https://web.archive.org/web/20160410053657/http://www.wotug.org/paperdb/show_proc.php?f=4&num=30

Restored from (where they would exist forever):

> https://web.archive.org/web/20220527212930/https://www.cs.kent.ac.uk/research/groups/plas/wiki/An_occam_Model_of_XCHANs?action=AttachFile&do=get&target=xchan-slides.pdf

Original not available anymore (Oct2023):

> https://www.cs.kent.ac.uk/research/groups/plas/wiki/An_occam_Model_of_XCHANs?action=AttachFile&do=get&target=xchan-slides.pdf

PDF made from export or print from *Wayback Machine* had each page mirrored (Safari 17.0, macOS 12.7). This new original made from screen clips by *Øyvind Teig*, Trondheim, Norway in Oct2023. Appears at:

> https://www.teigfam.net/oyvind/blog_notes/250/an_occam_model_of_xchans_slides_peter_welch_2013.pdf

XCHAN described at **XCHANs: Notes on a New Channel Type** by *Øyvind Teig*, at CPA 2012 (Communicating Process Architectures 2012), now see:

> XCHANs: Notes on a New Channel Type
> https://www.teigfam.net/oyvind/home/technology/250-xchans-notes-on-a-new-channel-type/

The WoTUG / CPA pages in Oct2013 exist in their original form:

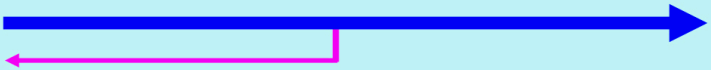
> https://wotug.org/cpa2012/ and https://wotug.org/cpa2013/

XCHANs

An **XCHAN** is a finitely (possibly zero) buffered channel that is *asynchronous* in the sense that it never blocks.

If a writer writes to an **XCHAN** that cannot take the message (e.g. because its buffer is full or, if zero-buffered, because no reader is committed to read), then the write *fails*. The writer gets the success status of each write.

An **XCHAN** also signals on a *feedback* channel when a write will be successful …



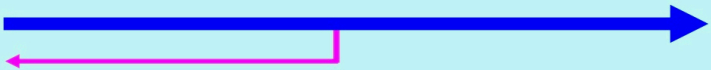
XCHANs

An **XCHAN** is a finitely (possibly zero) buffered channel that is *asynchronous* in the sense that it never blocks.

If a writer writes to an **XCHAN** that cannot take the message (e.g. because its buffer is full or, if zero-buffered, because no reader has committed to read), then the write *fails*. The writer gets the success status of each write.

An **XCHAN** signals on a *feedback* channel when a write is successful …

* Øyvind Teig: "XCHANs: Notes on a New Channel Type", CPA 2012, pp. 155-168, Open Channel Publishing.



XCHANs

An **XCHAN** is a finitely (possibly zero) buffered channel that is *asynchronous* in the sense that it never blocks.

If a writer writes to an **XCHAN** that cannot take the message (e.g. because its buffer is full or, if zero-buffered, because no reader is committed to read), then the write *fails*. The writer gets the success status of each write.

An **XCHAN** also signals on a *feedback* channel when a write will be successful …

We model an **XCHAN** with an **occam-π** process and channels


```
PROTOCOL XCHAN
  CASE
    ready
:
```

```
PROC x.write (VAL DATA d, BOOL status,
              CHAN XCHAN out.x?,
              CHAN DATA out!)
  PRI ALT
    out.x ? ready
      SEQ
        out ! d
        status := TRUE
    SKIP
      status := FALSE
:
```

Non-blocking write to XCHAN (with success status result)

writer


```
PROTOCOL XCHAN
  CASE
    ready
:
```

or

```
PROC x.write.sync (VAL DATA d,
                   CHAN XCHAN out.x?,
                   CHAN DATA out!)
  SEQ
    out.x ? ready
    out ! d
:
```

Synchronous write to XCHAN (blocks until taken)

writer


```
PROTOCOL XCHAN
  CASE
    ready
:
```

or

```
ALT
  out.x ? ready
    out ! d
  ...  other guarded processes
```

Response to XCHAN signal (will not block)

writer


```
PROTOCOL XCHAN
  CASE
    ready
:
```

```
PROC xchan.1 (CHAN DATA in?, out!,
              CHAN XCHAN in.x!)
  WHILE TRUE
    DATA d:
    SEQ
      in.x ! ready
      in ? d
      out ! d
:
```

1-buffered XCHAN

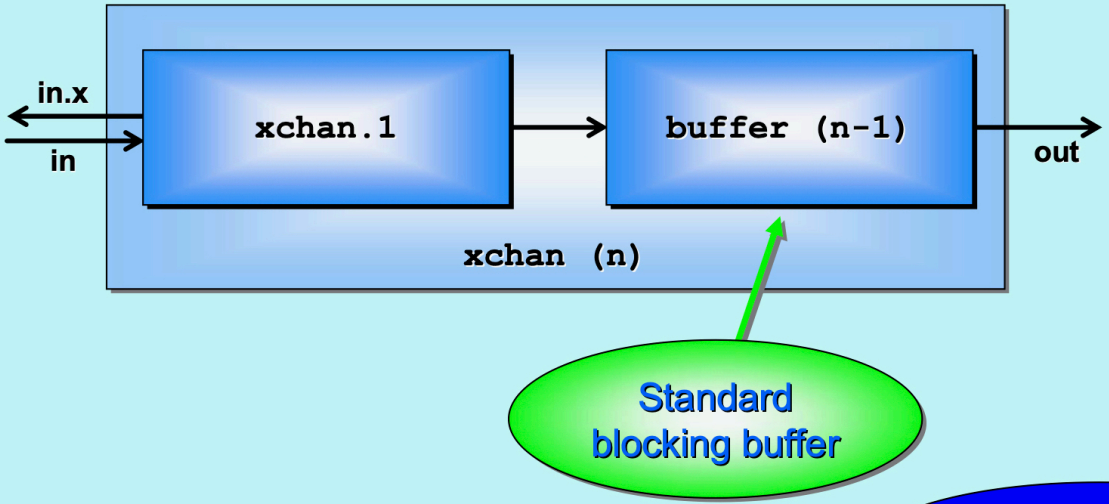
implementation


```
PROTOCOL XCHAN
  CASE
    ready
:
```

n-buffered XCHAN

implementation


```
PROTOCOL XCHAN
  CASE
    ready
  :
```



n-buffered XCHAN

implementation


```
PROTOCOL XCHAN
  CASE
    ready
:
```

```
PROC xchan.0 (CHAN DATA in?, out!,
              CHAN XCHAN in.x!)




:
```

0-buffered XCHAN

implementation


```
PROTOCOL XCHAN
  CASE
    ready
:
```

```
PROC xchan.0 (CHAN DATA in?, out!,
              CHAN XCHAN in.x!)
  WHILE TRUE
    DATA d:
    out !!              -- fish for reader
      !! d
:
```

When and only when a reader commits, this code executes.

Finally, the message is written.

0-buffered XCHAN

implementation


```
PROTOCOL XCHAN
  CASE
    ready
:
```

```
PROC xchan.0 (CHAN DATA in?, out!,
              CHAN XCHAN in.x!)
  WHILE TRUE
    DATA d:
    out !!          -- fish for reader
      SEQ
        in.x ! ready
        in ? d
      !! d
:
```

When and only when a reader commits, this code executes.

Finally, the message is written.

0-buffered XCHAN

implementation


```
PROTOCOL XCHAN
  CASE
    fish
    ready
:
```

```
PROC xchan.0 (CHAN DATA in?, out!,
              CHAN XCHAN in.x!)
  WHILE TRUE
    SEQ
      in.x ! fish          -- fish for writer
      DATA d:
      out !!               -- fish for reader
        SEQ
          in.x ! ready
          in ? d
        !! d
:
```

When and only when a reader commits, this code executes.

Finally, the message is written.

implementation

0-buffered XCHAN (better)


```
PROTOCOL XCHAN
  CASE
    fish
    ready
:
```

```
PROC x.write (VAL DATA d, BOOL status,
              CHAN XCHAN out.x?,
              CHAN DATA out!)
  PRI ALT
    out.x ? fish
      status := FALSE
    out.x ? ready
      SEQ
        out ! d
        status := TRUE
    SKIP
      status := FALSE
:
```

Non-blocking write to XCHAN (with success status result)

writer


```
PROTOCOL XCHAN
  CASE
    fish
    ready
:
```

```
PROC x.write (VAL DATA d,
              CHAN XCHAN out.x?,
              CHAN DATA out!)
  INITIAL BOOL writing IS TRUE:
  WHILE writing
    ALT
      out.x ? fish
        SKIP
      out.x ? ready
        SEQ
          out ! d
          writing := FALSE
:
```

Synchronous write to XCHAN (blocks until taken)

writer


```
PROTOCOL XCHAN
  CASE
    fish
    ready
:
```

```
ALT
  out.x ? fish
    SKIP
  out.x ? ready
    out ! d
  ...  other guarded processes
```

Response to XCHAN signal (will not block)

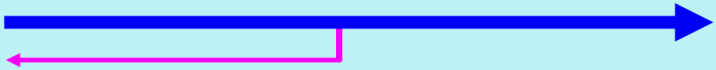
writer

XCHANs

An **XCHAN** is a finitely (possibly zero) buffered channel that is *asynchronous* in the sense that it never blocks.

If a writer writes to an **XCHAN** that cannot take the message (e.g. because its buffer is full or, if zero-buffered, because no reader is committed to read), then the write *fails*. The writer gets the success status of each write.

An **XCHAN** also signals on a *feedback* channel when a write will be successful …



Any questions?