

An occam Model of XCHANS

Peter Welch (phw@kent.ac.uk)

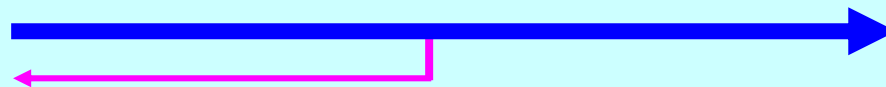
CPA 2013 Fringe, Napier University, 26 August, 2013

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 ...



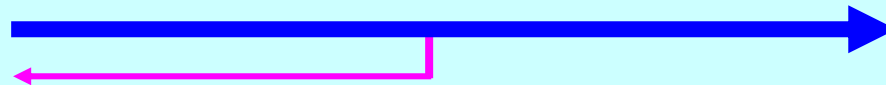
XCHANS

An **XCHAN** is a finitely (possibly zero) buffered channel that is *asynchronous* in the sense that it does not block.

If a writer writes to an **XCHAN** that is buffered, 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** 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.

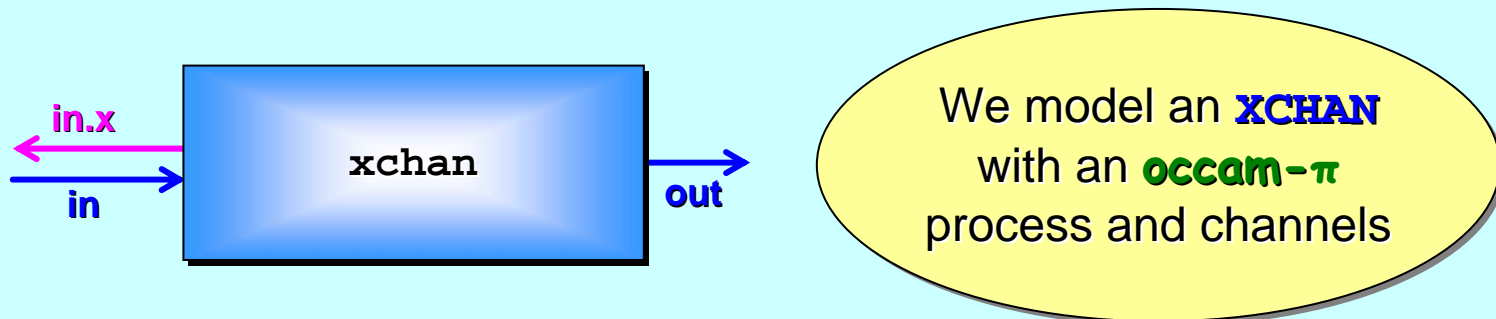


XCHANs

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 ...





```

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

```

```

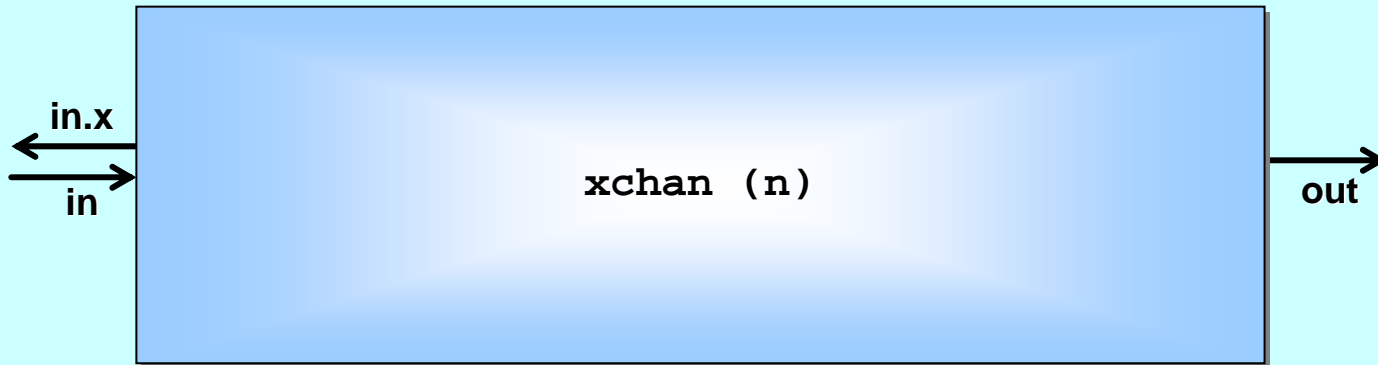
:
```

1-buffered XCHAN

implementation



```
PROTOCOL XCHAN
CASE
  ready
:
```



n-buffered XCHAN

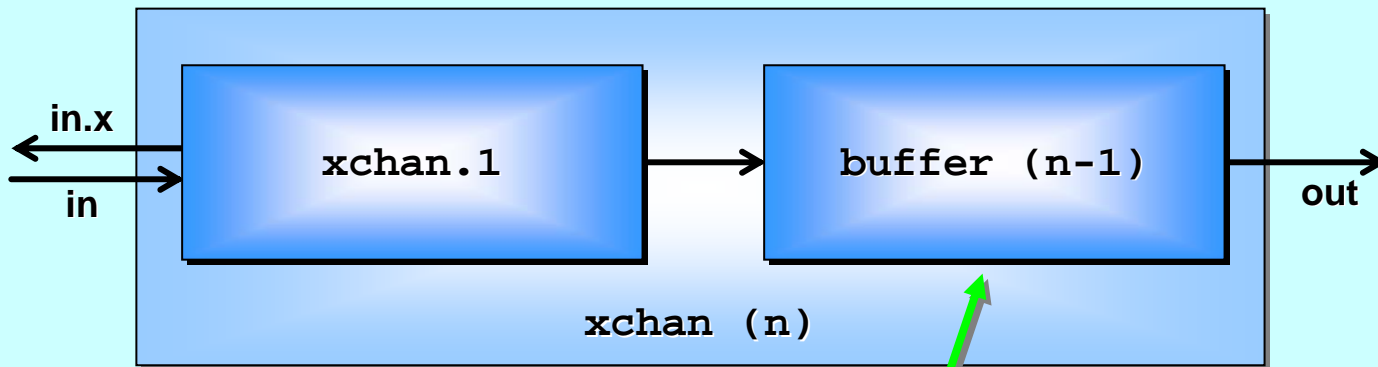
implementation



```

PROTOCOL XCHAN
CASE
  ready
:

```



Standard blocking buffer

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
  [Redacted]
  !! d
:

```

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

Finally, the message is written.

implementation

0-buffered XCHAN



```

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.

implementation

0-buffered XCHAN



```

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 ...

