Tangram Handshake Channels

Ad Peeters

Ad.Peeters@philips.com

To be covered

• Tangram handshake channels at three levels
  – Programming language (Tangram)
  – Intermediate architecture (Handshake circuits)
  – VLSI realization
• How new channel notion (and its associated compilation scheme) has drastically improved modularity and expressive power
Traditional Programming View

- Inspired by and based on Hoare’s CSP
- Channel composed of input and output channels
- Parallel composition:
  - parallel outputs not allowed
  - parallel inputs synchronize

Compilation: box view

This compilation scheme implies broadcast
Compilation: handshake view

Three passivators, three 2-way copies: six C-elements

Optimization: One passivator, one 3-way copy: three C-elements
Tangram View

- Channels are explicitly declared (and implemented)
- Separation in input and output ports (also for nonputs)
- Inputs and outputs are treated similarly
  - Parallel outputs are allowed
  - Parallel inputs other than broadcast
  - Both have optional arbitration
- Probed channels

Handshake Components Involved

- Passivator
- Copy
- Multiplexer
- Demultiplexer
- Arbiter
Passivator

- Synchronization of input (receivers) and output (senders)
- Alignment of data-valid schemes

Passivator

- Aligning data-valid schemes often requires latching of data inside passivator
  - $2\phi$: no options but latching
  - $4\phi$: early sender, early receiver
- Not required in some 4-phase schemes
  - $4\phi$: broad sender, early receiver
  - $4\phi$: late sender, early receiver
Copy

- Data wires fork to each destination
- Handshakes are synchronized
  - Push: fork for request (data-valid), C-element for ack
  - Pull: C-element for request, fork for ack (data-valid)

(De)multiplexer

- Control handled by mixer (call), which merges handshakes, provided mutual-exclusive activation
- Data is either forked (dmx) or multiplexed (mux)
  - multiplex select signals from mixer
Arbiter

- Data wires are copied from input to output
- Control is managed by control arbiter, which assures mutual exclusion on active ports

Link to Tangram: channels

- implemented by passivator component
- partitioned nonput, coded in input/output
- probe: status of channel near passivator
Link to Tangram: input

- **Sequential composition**
  - receivers are by definition mutual-exclusive
  - demultiplexer

- **Parallel comp., broadcast**
  - all receivers participate in each communication
  - copy component

- **Parallel comp., narrowcast**
  - one receiver participates in each handshake
  - guaranteed by program(mer) or (handshake) circuit
  - mutual-exclusive receivers
    - no arbitration required
  - possible conflicts
    - arbitration required
Link to Tangram: output

- Multiple senders
  - one sender participates in each communication
  - arbitration ...
    - may be guaranteed by the program(mer), or
    - must be assured by the (handshake) circuit

Tangram Example 1 (Classic)

```csp
begin
  a: chan T
  | a!E || a?x
end
```

- Classic example in CSP, Occam
- Distributed implementation of \( x := E \)
Tangram Example 2 (Classic)

begin
a: chan T
| (a!E ; a!F) || (a?x ; a?y)
end

- Sequential senders
- Sequential receivers
- Equivalent to x:=E ; y:=F

Tangram Example 3 (Broad)

begin
a: chan T broad
| a!E || a?x || a?y
end

- Classic example in CSP, Occam
- Synchronized receivers
- Channel a is used to broadcast E
- Equivalent to x:=E || y:=E
Tangram Example 4 (Narrow)

begin
  a: chan T narrow arb
  | (a!E ; a!F) || (a?x || a?y)
end

• Sequential senders
• Competing (arbitrated) receivers
• Outcome depends on arbitration
• Results in either $x:=E ; y:=F$ or $y:=E ; x:=F$

Tangram Example 5 (Multi)

begin
  a: chan T multi arb
  | (a!E || a!F) || (a?x ; a?y)
end

• Competing (arbitrated) senders
• Sequential receivers
• Results in either $x:=E ; y:=F$ or $x:=F ; y:=E$
• Outcome depends on arbitration
Tangram Example 6 (Multi, Narrow)

begin
  a: chan T multi arb narrow arb
  | (a!E || a!F) || (a?x || a?y)
end

- Competing (arbitrated) senders
- Competing (arbitrated) receivers
- Results in \( x:=E \); \( y:=F \) or \( x:=F \); \( y:=E \) ...
- … or \( y:=E \); \( x:=F \) or \( y:=F \); \( x:=E \)

Tangram Example 7 (Multi, Broad)

begin
  a: chan T multi arb broad
  | (a!E || a!F) || ((a?x || a?y) ; a?z)
end

- Competing (arbitrated) senders
- Synchronized receivers
- Results in either \( (x:=E || y:=E) ; z:=F \)
  or \( (x:=F || y:=F) ; z:=E \)
Tangram Example 8 (Surprise)

begin
  a: chan T multi narb narrow narb
  | (a!E ; a?y) || (a?x ; a!F)
end

• Sequential send and receive access
• Yes, it is allowed in Tangram
• Results in \( x:=E \ ; y:=F \)
• Would deadlock for broadcast channel

Classic FIFO Buffer

• One-place FIFO buffer
  
  \[
  \text{Fifo1} = \text{proc}(\text{a?chan T } \& \text{ b!chan T}).
  \begin{array}{l}
  \text{begin } x:\text{var T} \mid \text{forever do } a?x \ ; b!x \text{ od end}
  \end{array}
  \]

• It takes three for a three-place FIFO
  
  \[
  \text{Fifo3} = \text{proc}(\text{a?chan T } \& \text{ b!chan T}).
  \begin{array}{l}
  \text{begin } m0,m1:\text{chan T}
  \mid \text{Fifo1 (a,m0)} \mid \text{Fifo1 (m0,m1)} \mid \text{Fifo1 (m1,b)}
  \text{end}
  \end{array}
  \]
Stock Holder - Farming

- Tangram allows for unforeseen solutions
- Starting from classic one-place FIFO buffer
  \[
  \text{Fifo1} = \text{proc}(a?\text{chan }T \& b!\text{chan }T).
  \text{begin } x:\text{var }T \mid \text{forever do } a?x ; b!x \text{ od end}
  \]
- Three in parallel form a three-place-holder
  \[
  \text{Fifo3} = \text{proc}(a?\text{chan }T \text{ narrow arb}
  & b!\text{chan }T \text{ multi arb}).
  \text{Fifo1 } (a,b) || \text{Fifo1 } (a,b) || \text{Fifo1 } (a,b)
  \]

Semaphores

- & P : chan ~ narrow arb
- & V : chan ~ multi narb
- & Semaphore : proc().
  \text{forever do } P!~ ; V?~ \text{ od}
- & ProcessA : proc(). P?~ ; CriticalSectionA() ; V!~
- & ProcessB : proc(). P?~ ; CriticalSectionB() ; V!~
- & ProcessC : proc(). P?~ ; CriticalSectionC() ; V!~

- Arbitration on P in channel declaration
- Arbitration implicit in rest of program
- Program text independent of number of processes
Conclusion

• Powerful channel notion in Tangram
  – parallel outputs: multiple senders
  – parallel inputs: broadcast, narrowcast
  – arbitration is optional (for multi and narrow)
  – probe is supported (non-committing inspection)
• Examples from applications in Joep’s talk

Conclusion

• Implicit arbitration in channels leads to programs that are more modular than the explicit arbitration programmed in sel/alt constructs
• Implementation of constructs is efficient
  – no latching in communication paths
  – no redundant return-to-zero phase