"A software development methodology for distributed real-time systems that shares the simplicity and applicability of SpaceWire"

Kazuto Matsui
Prominent Network LP

SpaceWire WG Meeting 5
16th/2005

Space System Requirements

- Decentralized data acquisition or control are required when SpaceWire is used.
- Some applications need high speed I/O control.
- Distributed computing with 10-100(or maybe more) of nodes can be made when cascading routing switches.
- Fault tolerance of the network.
- But a safety programming methodology still remains!!

Possible Solutions for Concurrency

- RTOS(TRON, VxWorks, RT-Linux, etc) is everybody’s favourite.
- Use of the conventional programming languages such as C/C++ with multi-threading is alternative choice.
- But how do we know that both cases are correctly working for the real-time distributed computing over the SpaceWire ?
- So a cleaver methodology is plausible !!

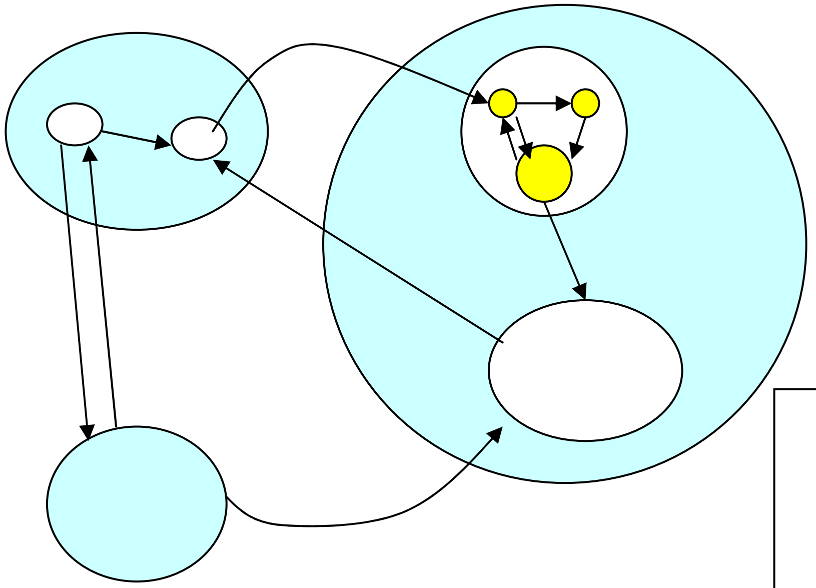
What is CSP ?

- CSP (Communicating Sequential Processes) is a process algebra based on mathematical foundation developed at Oxford.
- Over 20 years many experiences are being cumulated.
- occam and Transputer were the embodiments of the CSP model in the past.
- Believe or not, CSP concept is still existing !!

Notation of CSP processes

$P \ ::= \ STOP$	Stop
$\mid SKIP$	Successful termination
$\mid channel!e \rightarrow Q$	Channel output
$\mid channel?e \rightarrow Q$	Channel input
$\mid a \rightarrow Q$	Prefix Event
$\mid Q;\ R$	Sequence
$\mid Q \ \lvert[alpha_1 \parallel alpha_2]\rvert \ R$	Parallel
$\mid Q \sqcap R$	Internal Choice
$\mid Q \,\square\, R$	External Choice
$\mid Q \mid\mid\mid R$	Interleave
$\mid \mu X.P$	Recursion
$\mid P \lhd b \rhd Q\,(P\ if\ \ b\ else\ Q)$	Condition

Occam Primitives



Skip	*SKIP*
Stop	*STOP*
Assignment	*Var := Exp*
Input	*Chan ? Var*
Output	*Chan ! Exp*
Procedure call	*Name(Exp0, .., Expn)*
Sequential Composition	*SEQ(P0, …, Pn)*
Conditional branching	*IF((b0, P0), …(bn,Pn))*
Iteration	*WHILE(b, P)*
Parallel composition	*PAR(P0, …, Pn)*
Alternation	*ALT((g0, P0), …(gn, Pn))*
Priority	*PRI(P1, P2, ..Pn)*

Why CSP model is useful ?

- Scheduler is very compact and very fast than using “monitor”.
- Event driven is the main feature.
- Channel based I/O control is easy to implement PAR I/O.
- Non-deterministic process is important for distributed I/O control.
- Building block structure can create a test module starting from small to a large area.
- About reliability, formal methods can validate the system in early stage of the programming.

Open source

GNU Lesser GPL

1. JCSP (Java CSP Library) Network Edition

http://www.cs.kent.ac.uk/projects/ofa/jcsp/ -- University of Kent

Quickstone Technologies

2. C++CSP(The Kent C++CSP Library)

http://www.cs.kent.ac.uk/projects/ofa/c++csp/

KENT
UNIVERSITY OF KENT
AT CANTERBURY

3. KRoC(Kent Retargetable occam Compiler)

http://www.cs.kent.ac.uk/projects/ofa/kroc/

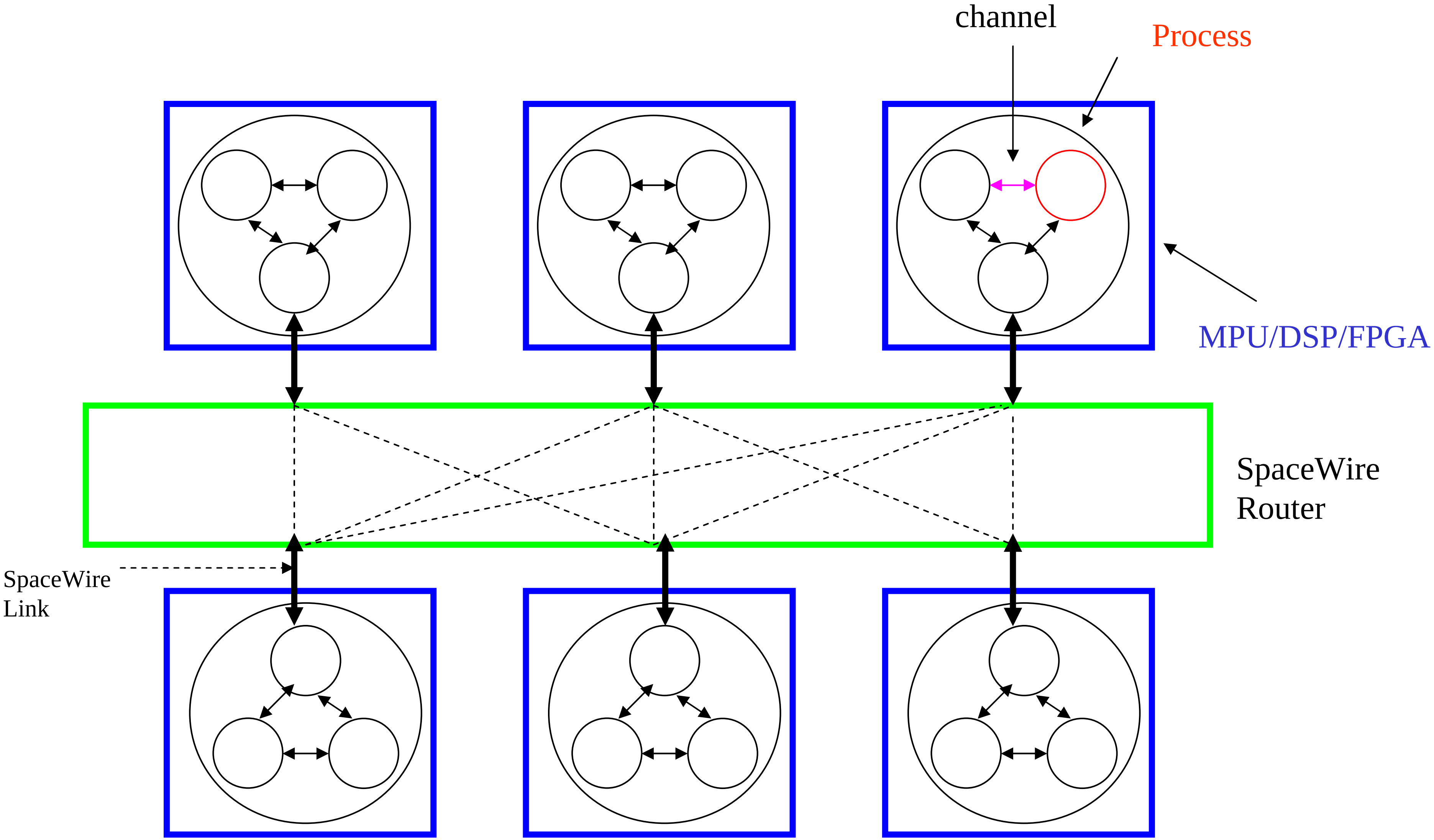
WoTUG

Context switching time is 67nsec(P3@800MHz)

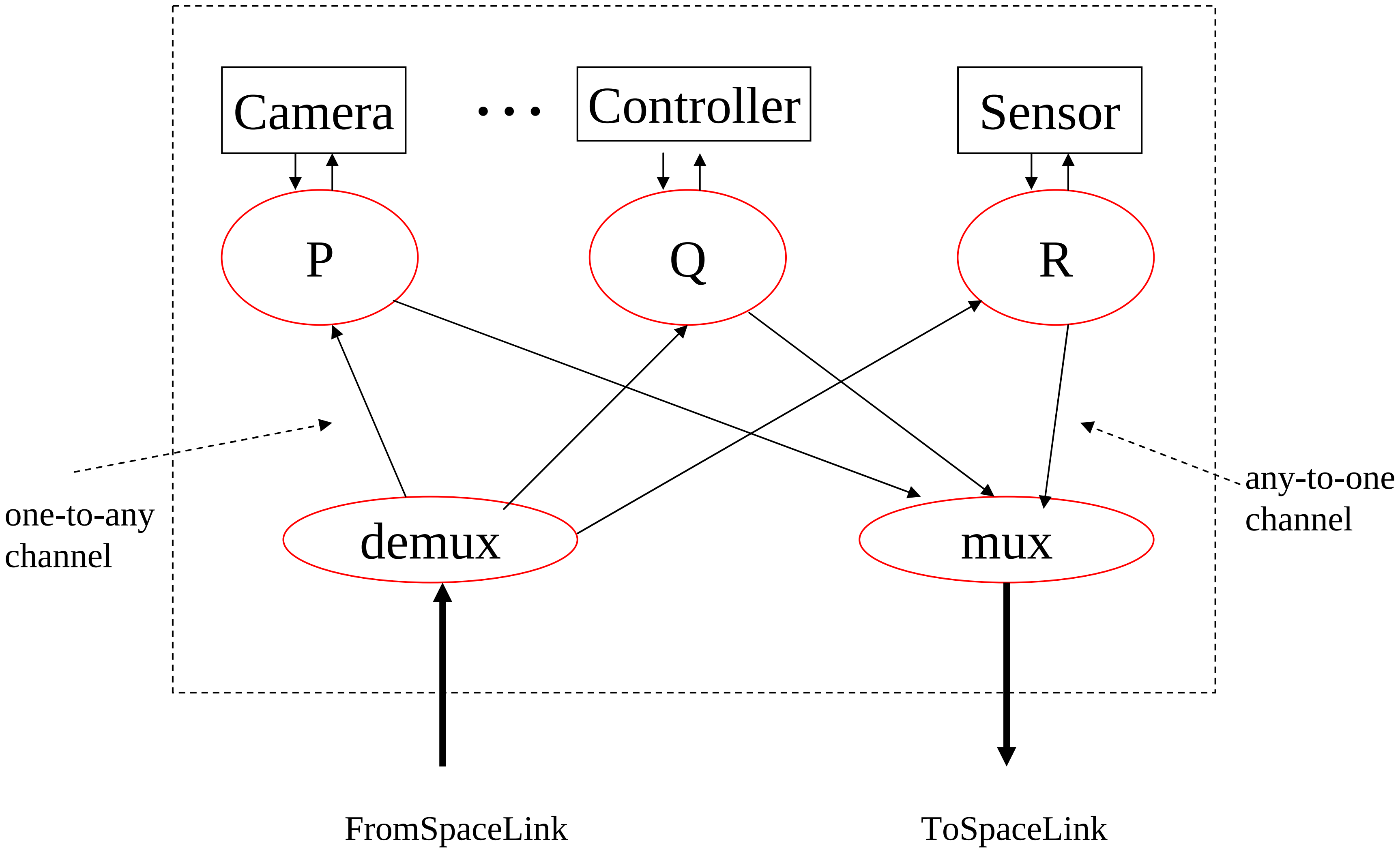
How can we use CSP model with SpaceWire ?

- Use occam channels to map point-to-point communication to the Spacewire links.
- In a large number of nodes, create harnesses (such as pipeline, MUX,DEMUX, etc) to access I/O modules. Then link occam channels to the SpaceWire links.
- Handel-C (Celoxica) is an EDA tool to compile occam model to the FPGA.

Process Harness



Deadlock-free Multiple Controllers



Development Environments

- JCSP (MS-Windows, Linux, Solaris, Mac, etc)
  - (Eclipse, Struts, Microsoft Visual .NET, DOS)
- C++CSP (MS-Windows, Linux, Solaris, etc)
  - Libraries are provided with C++ source code.
- KRoC(occam)
  - Linux, Cygwin, etc