CSP、一体誰が使っているの？

CSP研究会

松井和人

matsui@csp-consortium.org

まえがき

Transputerが消えてからもう14年になる。CSP/occamは過去のものと葬られている。特に大学での教育、学会では甚だしい誤解がある。

CSP/occamモデルは数学的法則に裏づけられており、普遍的な法則である。

不思議なことに、欧米ではCSPモデルの継承と進化に努力を払っている。

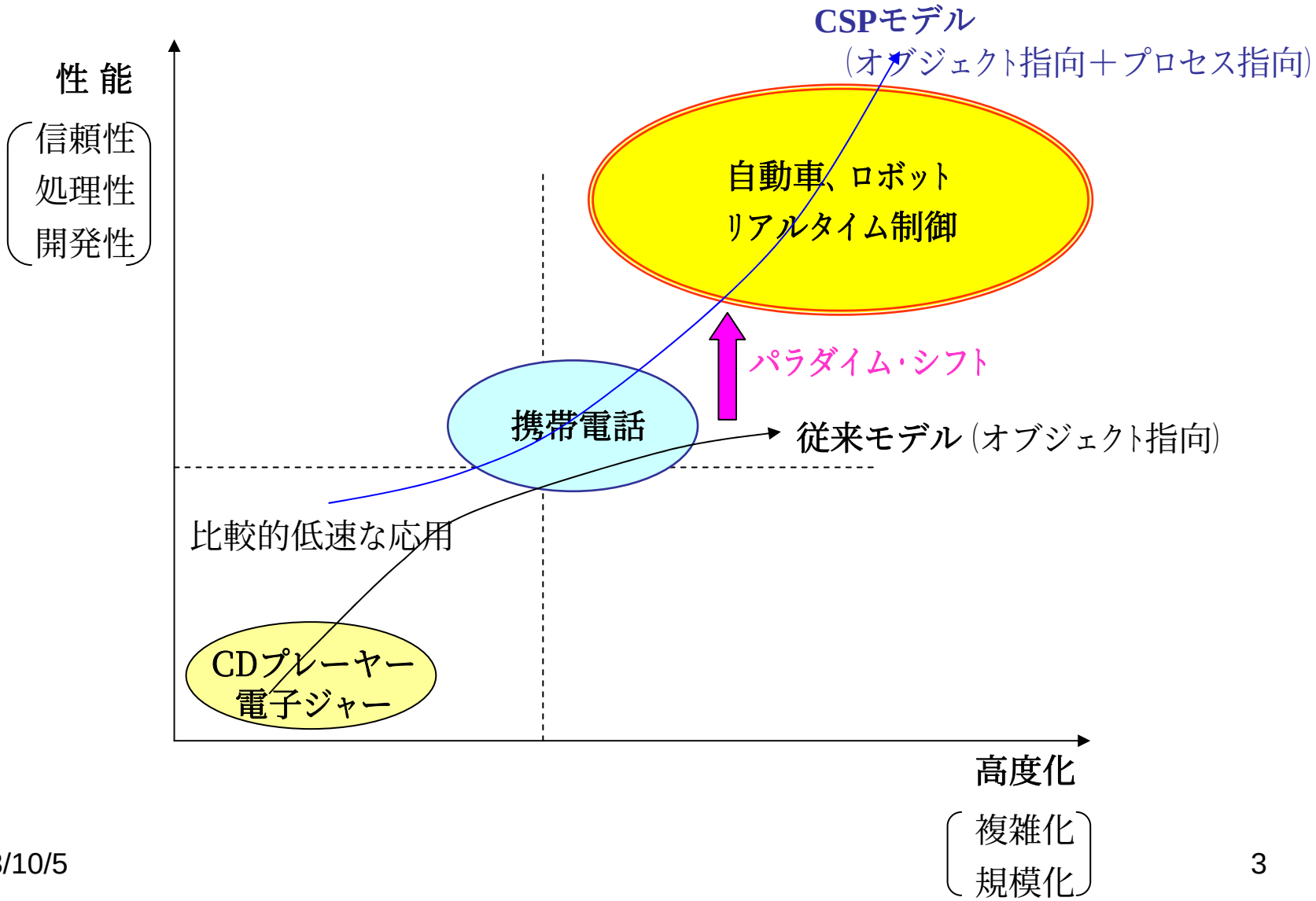
大学、産業界では形式手法の必要性が高まっているが、関心は検証ツールだけであり、プログラミング言語とのギャップは縮まらない。

CSPモデルは形式手法とプログラミング言語がリンクできる。

アプリケーションはマルチコアチップにも対応しつつあり、並列処理は簡単に手が届くようになっているが、マルチスレッドの問題は未解決である。

この資料は初心者の方々にCSPの具体的な事例を知るべく、多くのアプリケーションの中に隠された事例について、まとめたものである。

高度システムへの対応


CSPモデル
(オブジェクト指向＋プロセス指向)
性 能
信頼性
処理性
開発性
自動車、ロボット
リアルタイム制御
パラダイム・シフト
携帯電話
従来モデル (オブジェクト指向)
比較的低速な応用
CDプレーヤー
電子ジャー
高度化
複雑化
規模化

組み込みソフトの課題

- C/C++/Javaのような逐次処理プログラミング技法では複雑でリアルタイム性の高い並列処理（特に入出力）、イベント駆動、時間駆動等を実現するのは非常に困難である。
- POSIX Threadによるマルチスレッド並列プログラミングは排他制御するための各スレッドはメモリー間で競合が生じ、異なった結果を発生させる。
- RTOS (iTRON, RT-Linux等)を使っても上記の問題点は今だに解決されていない。
- 並列処理、リアルタイム処理のアプリケーションではオブジェクト指向プログラミングは混乱が生ずる。プロセス指向の方が安全である。
- マルチコアでサポートされているOpenMPはFortranの流れを受け継いでおり、組み込みシステムの要求に十分対応できない。
- モデル検証ツールと実装するプログラミング言語(C/C++/Javaなど)の間には大きなギャップがある。つまり検証されていない。
- IEC61508(機能安全規格)、ISO26262(自動車)などの上位レベルでは形式手法が要求されている。

マルチスレッドの問題

どの様にモデリングして、安全に動作する事が検証できるのか？

共有メモリを使った並列処理から開放されない限り、メモリアクセスの競合の問題は解決されない。

これでよいのか？55ページにマルチスレッドの安全性は証明されていないと記載されている！！

複雑な並列処理を**java.util.concurrent**パッケージでやるのは**CHAOS !!**

Channel, Guard, PipeなどCSPと似たものが用意されているが、プログラミングのパターンを暗記しても駄目！！

Java言語で学ぶデザインパターン入門
マルチスレッド編（増補改訂版）

結城　浩［著］
ソフトバンククリエイティブ（*2006/03/31* **出版**）

Java Monitors - CONCERNS

<java.sun.com/products/jfc/tsc/articles/threads/threads1.html>

- ***“ If you can get away with it, avoid using threads. Threads can be difficult to use, and they make programs harder to debug. ”***

- ***“ Component developers do not have to have an in-depth understanding of threads programming: toolkits in which all components must fully support multithreaded access, can be difficult to extend, particularly for developers who are not expert at threads programming. ”***

Java Monitors - CONCERNS

<java.sun.com/products/jfc/tsc/articles/threads/threads1.html>

- ***" It is our basic belief that extreme caution is warranted when designing and building multi-threaded applications … use of threads can be very deceptive … in almost all cases they make debugging, testing, and maintenance vastly more difficult and sometimes impossible. Neither the training, experience, or actual practices of most programmers, nor the tools we have to help us, are designed to cope with the non-determinism … this is particularly true in Java … we urge you to think twice about using threads in cases where they are not absolutely necessary …"***

Java Monitors - CONCERNS

- **So, Java monitors are not something with which we want to think - certainly not on a daily basis.**
- **If we have to think with them at all, we want some serious help!**
- **The first step in getting that help is to build a formal model of what we think is happening …**

Peter Welch

T-Kernelの問題点

この検証は非常に難しい！！

SMP型プログラム構築時の注意点

SMP型のプログラムは、全体の平均スループットを最大限にできるメリットがありますが、そのコンセプトから、いくつか注意するべき点があります。

SMP型プログラムは、プログラムが複数のCPUコアに動的に分割して実行されるため、個々の処理の実行順序や割り当てられるCPUコアの予測が不可能になり、処理によってはリアルタイム性を確実に保証できない場合があります。また、従来のシングルプロセッサ向け既存プログラムでは、優先度ベースにスケジューリング方式に頼った排他を行っている場合、低優先度プログラムが、CPUコアが空いているときに実行されてしまうなど、**同期･排他のやり方に起因する問題が発生する場合があります。**

http://www.esol.co.jp/embedded/et-kernel_multicore-edition.html#smpprogramming

Javaプログラマーのための CSP

CSPの利点

CSPは、並行プログラミングのための仕組みとして、成熟した数学的理論の上に構成されています。そのためCSPは、マルチスレッドにおける一般的な落とし穴、つまり競合の危険性、デッドロック、ライブロック、リソース不足などを防ぐための豊富な設計パターンやツールを備えています。このように数学的に洗練された機能がJCSPライブラリーには組み込まれているため、このライブラリーを直接使ったアプリケーション・プログラムを、指示されたガイドラインに基づいて書くことができるのです。(つまり、必ずしもJCSPライブラリーの基礎となる理論を理解しなくても、このライブラリーは利用できるのです。もちろん、明確に理解しておけば、さらに有利なことは当然です。) またJava用の正式なCSPモデルができているため、JCSP構成体を使って構築された任意のマルチスレッドJavaアプリケーションを、分析し、正式に検証することもできます。

先にも触れたように、AOPの利点の多くは、CSPベースのプログラムにも当てはまります。CSPベースのプログラムの基本的な利点は、コンサーンが分離できることです。CSPでは、プロセスは直接のメソッド呼び出しではなく、チャネルを介してのみ対話動作を行うため、CSPをプログラムの基礎として使う (例えばJCSPライブラリーを使う) ことによって、プロセスとプロセスの間を明確に切り離すことができます。またCSPでは、チャネル・インターフェースの後ろにあるプロセスの中に、データやアプリケーション・ロジックを完全にカプセル化することができます。全ての副作用はプロセス境界の内側のみに制限刺され、プログラム的な実体との対話動作は明確に公開されます。JCSPベースのプログラムには、隠れた対話動作というものはありません。ネットワークの各レベルで、すべての構造が見えるのです。

http://www.ibm.com/developerworks/jp/java/library/j-csp3/

Javaプログラマーのための CSP

CSPの利点として2番目は、構成の容易さです。プロセスは、より複雑なプロセスを形成するように構成することができ（この複雑なプロセス自体を使って、さらに複雑なプロセスを形成することもできます）、時間の経過と共に修正したり拡張したりすることが容易にできるのです。そのため、CSPに基づくアプリケーション設計は、非常に単純化され、理解しやすく、しかも維持管理という面から見ると、非常に堅牢なのです。また、構成が容易というCSPの性格から、プログラムの再利用も容易です。第2回でのプログラミング例でも見た通り、1つのJCSPプロセスを、幾つもの異なる設定で使うことができます。しかも不要な出力は、場合によってはブラック・ホールにリダイレクトできるのです。

CSPによる並行プログラミングの利点の最後は、分散システムを構築する開発者にとって非常に明確なものです。この記事では、並行性を実現するための、様々な選択肢を説明しました（つまり、複数スレッドを1つのプロセスで実行する、複数プロセスを1つのプロセッサーで実行する、複数プロセスを複数プロセッサーで実行する、など）。通常、こうした並行性機構では、それぞれ全く異なる実行モデルやプログラミング・インターフェース、設計方式などを使う必要があります。ところがCSPは、学ぶべき、そして使用すべきプログラミング・インターフェースは1つ、設計方式も1つという、統一された設計モデルを提供しているという点で、大きく異なります。CSPを使ってアプリケーションを開発しておけば、そのアプリケーションを実行するプラットフォームをどうするか（つまりマルチスレッドのプラットフォームか、単一プロセッサー上での複数プロセスによるプラットフォームか、分散プロセッサー上で実行する複数プロセスによるプラットフォームか）という判断を、アプリケーションの設計やコードに影響を与えず（少なくとも大きな影響は与えずに）下すことができるのです。

http://www.ibm.com/developerworks/jp/java/library/j-csp3/

JCSPとjava.util.concurrent

ほとんどのJava開発者は、J2SE 1.5クラス・ライブラリーの標準的な一部として導入されたjava.util.concurrentパッケージのことを知っているでしょう。JCSPライブラリーは、このパッケージがJavaプラットフォームに追加される前から存在していたものなので、両方とも学ぶ必要があるのかと疑問に思う人、あるいは、既にjava.util.concurrentに慣れてしまった場合にもJCSPを学ぶ必要があるのか、と思う人も多いと思います。

…
…
…

しかし、この記事での議論から理解して頂けたと思いますが、(JCSPライブラリーを使うことによる) CSPベースの手法には、java.util.concurrentの想定をはるかに上回る見返りがあるのです。通信し合うプロセスのネットワークとしてシステムを構成し、それを使ってネットワーク内にネットワークを構成するという考え方は、自然であり、かつ単純です。これによってマルチスレッド・アプリケーションの書き方が改善されるだけではなく、製品の品質も改善されるのです。JCSPを使ってアプリケーションを構築すると、インターフェースが非常にきれいになり、隠れた副作用も無くなります。その結果、できあがる製品も、維持管理や技術変更による影響を受けにくくなるのです。正式に検証できること (つまり、どの程度アプリケーションが安全で、どの程度健全なものかを、きちんと理由を付けて説明できること) も、安全性が致命的重要性を持ち、また経済的価値の高い金融アプリケーションなどにとって、大きな強みと言うことができます。

http://www.ibm.com/developerworks/jp/java/library/j-csp3/

Java並行性のバグのパターン

http://www.ibm.com/developerworks/jp/java/library/j-concurrencybugpatterns/

第１６回(２０００年）稲盛財団京都賞受賞

http://www.inamori-f.or.jp/laureates/k16_a_antony/ctn.html

アントニー・ホーア (Anthony Hoare)

贈賞理由　*CSP(Communicating Sequential Processes)*理論の考案者

「ソフトウェア科学に対する先駆的かつ根幹的な貢献」

ホーア教授は、1960年代初期より、ソフトウェアの信頼性向上のために公理論的アプローチを中心に種々の提案を行い、ソフトウェア科学の発展に根幹的な貢献をした。

教授は1960年代初め、今日最もよく使われる効率の良い整列アルゴリズムである「クイック・ソート」を考案した。これは、その基本アルゴリズムを簡潔な再帰的記述によって表現する画期的なもので、当時の計算機の性能を最大限に使った高速で巧妙な手法であった。

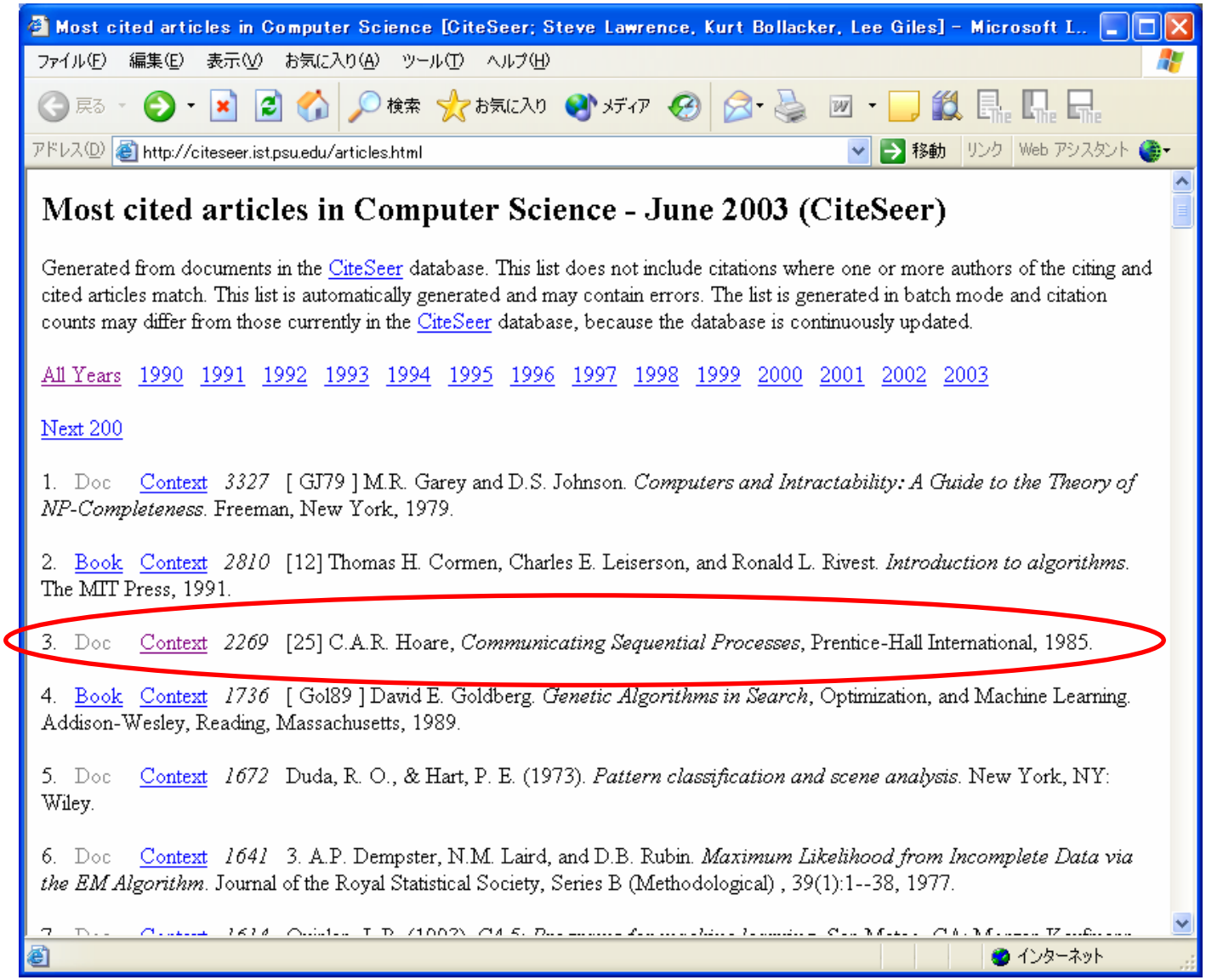
1969年、教授はまた、ホーア論理と呼ばれている公理論的意味論を用いて、プログラミング言語の定義付けと設計を行った。これは、教授の深い洞察力と独創性に裏打ちされ、それまでのソフトウェアの歴史に一線を画す強力かつ優美なものである。ここに、教授はプログラミングを厳密な科学として確立した。1960年代後半、コンピュータの大型化、大容量化に伴い、大規模ソフトウェアの開発が要求される中、その開発の困難や、システムの信頼性の問題からソフトウェア危機が叫ばれたとき、**教授はホーア論理を基礎に、プログラムの正しさは論理的に検証されるべきものであるとの指針を明確に打ち出した。**

ここで教授は、構築されたソフトウェアの正当性、信頼性を保証するための方法論を与えるべく、データ型の概念を明確にし、データの構造化の重要性を主張するとともに、階層化と抽象化を中心に据え、プログラムを段階的に詳細化していく構造化プログラミングの体系を提唱した。この教授のデータ型の概念と検証規則は、良質なプログラム設計のための根本的な指針となり、ソフトウェア危機を克服するための大きな原動力になった。また、**1972年には、オペレーティング・システムの抽象化と構造化の研究を通じてモニターという概念を提案、さらに、その後提唱された並列処理システムの挙動を記述する理論的枠組みである相互通信逐時システム（CSP）は、並列コンピュータの実用化に基礎を与え、今日の普及に大きく寄与した。**

ホーア教授のソフトウェア科学に対する業績は、いずれもが高度化、多様化するコンピュータソフトウェアの根幹にかかわるもので、分野の広さ、またその理論の深さのすべてにおいても比類のないものであり、安全な情報化社会を構築するために不可欠な技術的基盤を与えるものであった。今日のソフトウェアにかかわる科学と技術の発展は、教授の業績に負うところがきわめて大きい。

よって、ホーア教授に先端技術部門における2000年京都賞を贈呈する。

2003年においてCSPは世界の研究の第３位

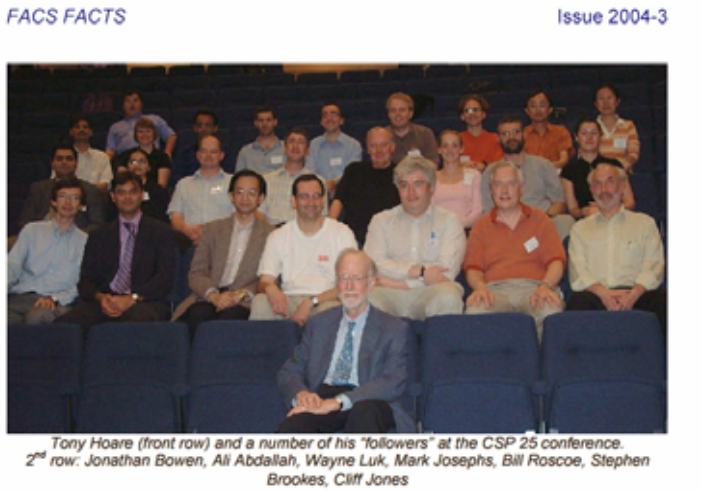


http://citeseer.ist.psu.edu/articles.html

CSPは過去のものではない。

Sir Tony Hoare の近況

- 2000年に先端技術部門で第16回稲盛財団京都賞を授与される。
- 2000年3月7日に女王からナイトの称号を授与される。
- ケンブリッジ大学内のマイクロソフト研究所。
- UK Grand Challengesプロジェクトを推進中 (http://www.nesc.ac.uk/esi/events/Grand_Challenges/proposals/)。
- 2004年7月7日、8日にCSP25周年会議を行う。
- 詳しくは：http://research.microsoft.com/~thoare/

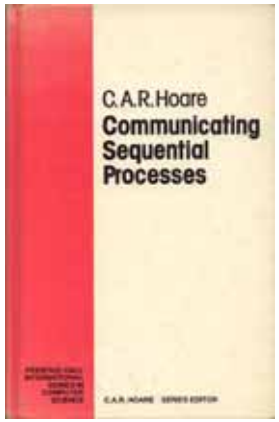


Tony Hoare (front row) and a number of his "followers" at the CSP 25 conference.
2nd row: Jonathan Bowen, Ali Abdallah, Wayne Luk, Mark Josephs, Bill Roscoe, Stephen Brookes, Cliff Jones

CSPの教科書

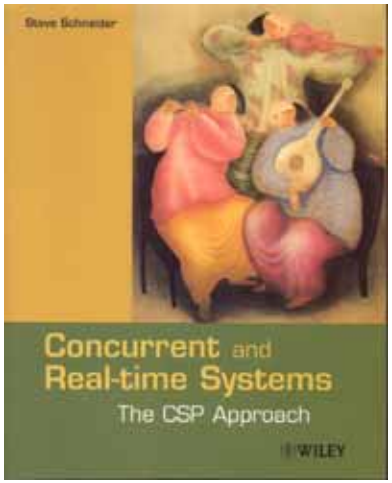
以下にpdfファイルが用意されています！！

http://www.usingcsp.com/



(Prentice-Hall International, ISBN 0-13-153271-5)
日本語版は丸善から出版されています。
（ホーアCSPモデルの理論 吉田信博訳 ISBN4-621-03683-1）

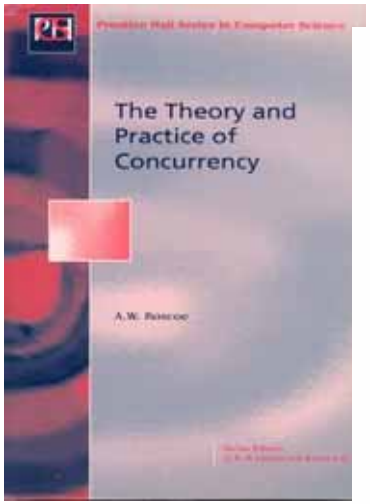
これは分かり易く書かれています。



Concurrent and Real-time Systems
The CSP Approach
Steve Schneider
Wiley
ISBN 0-471-62373-3

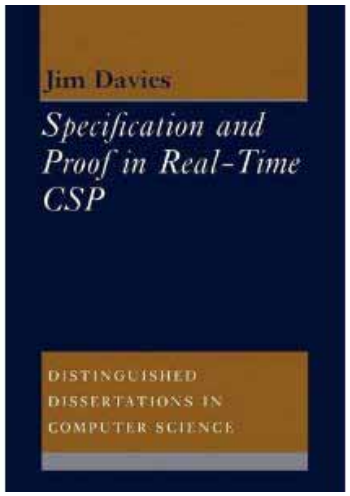
以下にpdfファイルが用意されています！！

http://www.fsel.com/



The Theory and Practice of Concurrency
A.W. Roscoe
Prentice Hall
ISBN 0-13-674409-5

Real-Time -CSP



Specification and Proof in Real-Time CSP
Jim Davies
Cambridge University Press
ISBN 0-521-45055-1

プロセス指向

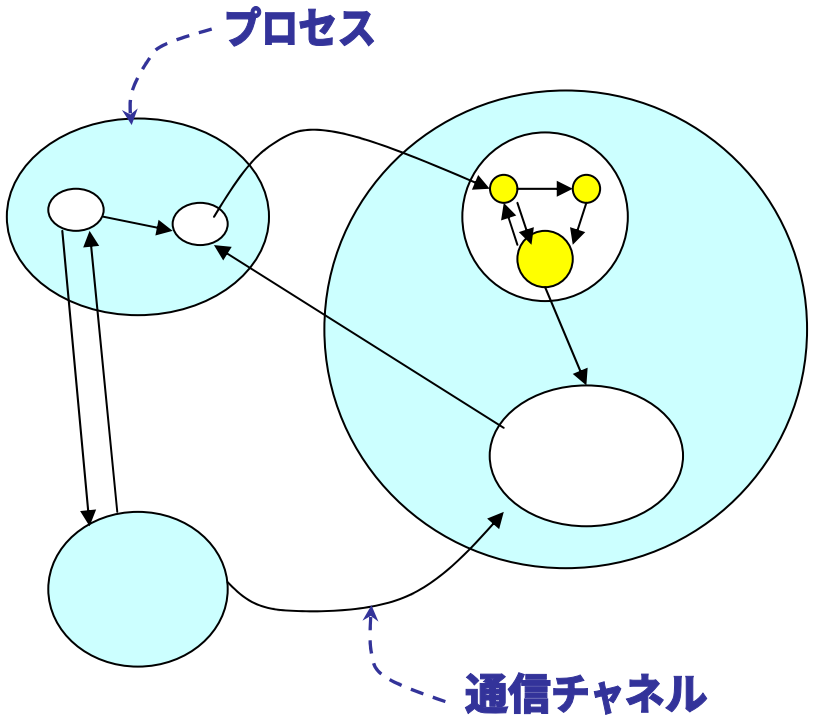
丸い円はプロセス、矢印は通信チャネル

Communicating Sequential Processes では、

プロセスが逐次型であっても相互に通信し合うことで、並列処理が実現。

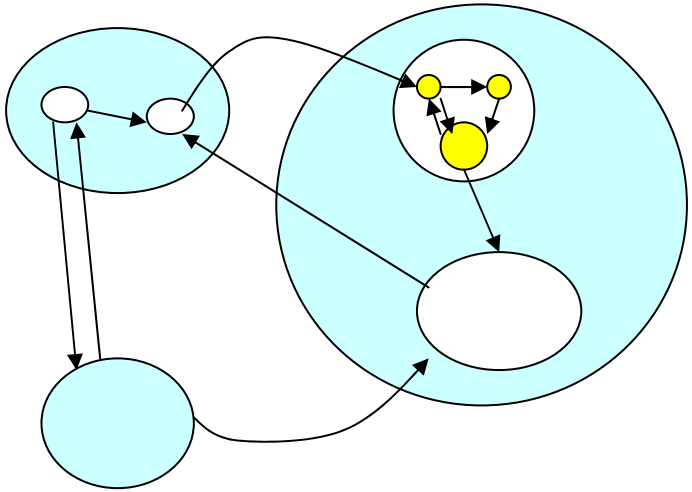
数学的体系も持っており、コンパイラの動作が明確であり、安全に動作する。

プロセスとはプログラムの集合体を表す。



CSPモデルをダイレクトに翻訳したプログラミング言語としてoccamがある。

occam Primitives

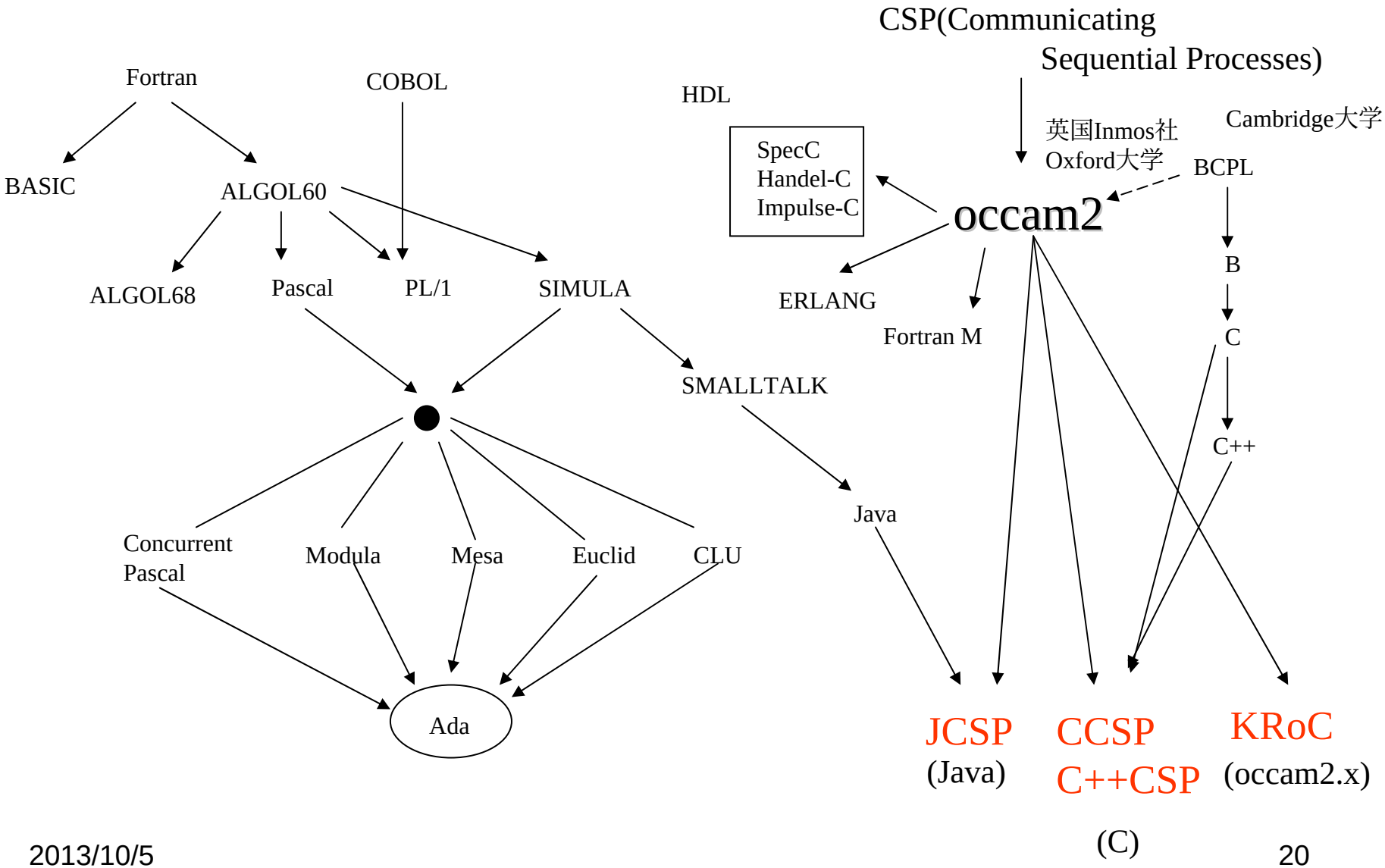


Skip	*SKIP*
Stop	*STOP*
Assignment	*Var := Exp*
Input	*Chan ? Var*
Output	*Chan ! Exp*
Procedure call	*Name(Exp0, .., Expn)*
Sequential Composition	*SEQ(P0, …, Pn)*
Conditional branching	*IF((b0, P0), …(bn,Pn))*
Iteration	*WHILE(b, P)*
Parallel composition	*PAR(P0, …, Pn)*
Alternation	*ALT((g0, P0),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

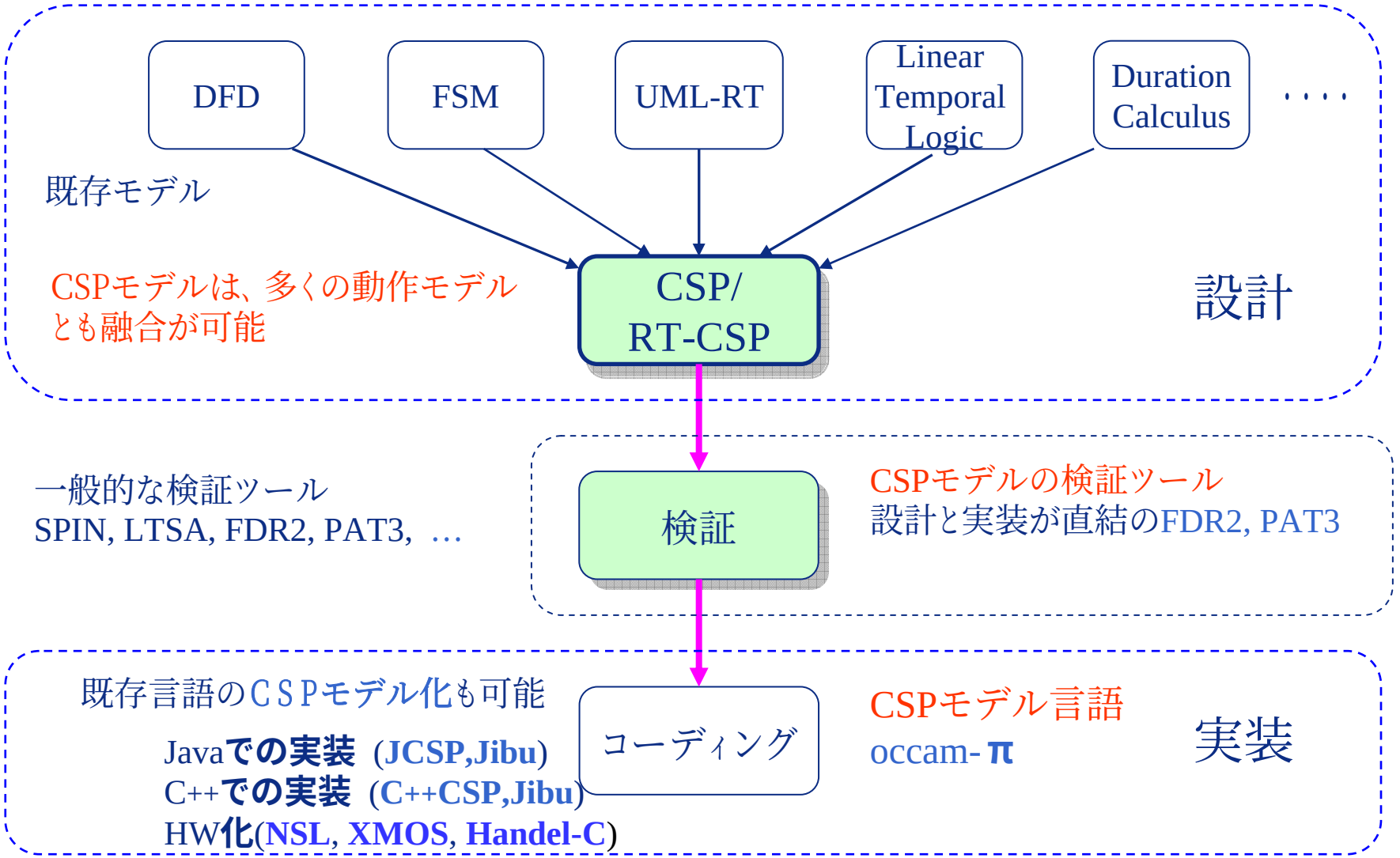
1. *SEQ* 逐次処理
2. *PAR* 並列処理
3. *WHILE* ループ
4. *IF* 条件
5. *CASE* 選択
6. *ALT* プロセスの選択
7. *PRI* 優先度
8. *SKIP*
9. *STOP*
10. *c ! e* チャンネルへの出力
11. *c ? x* チャンネルからの入力

コンパクトなコードサイズで高性能！！

プログラミング言語の派生



CSPモデルでは、設計と実装が直結



occam-for-all プロジェクト

(Transputing without Transputers!!)

occamはSTMicroelectronics社の登録商標

"Entia non sunt multiplicanda praeter necessitatem."

1978年
Oxford

CSP

16/32bit 並列プロセッサー

Transputer

occam1

1980年代前半
英国Inmos社

1980年
代後半

SpaceWire(IEEE1355)

University of Twente
The Netherlands

occam2

Visual-CSP

Jibu

AXON7

JCSP

C++CSP

CCSP

1990年代

Handel-C

XMOS

TPCore

2008年に製品化

首都大学東京
福永研究室

occam2.1

2000年以降

1992年からKent大学

occam3

1991年に仕様
だけ決まる

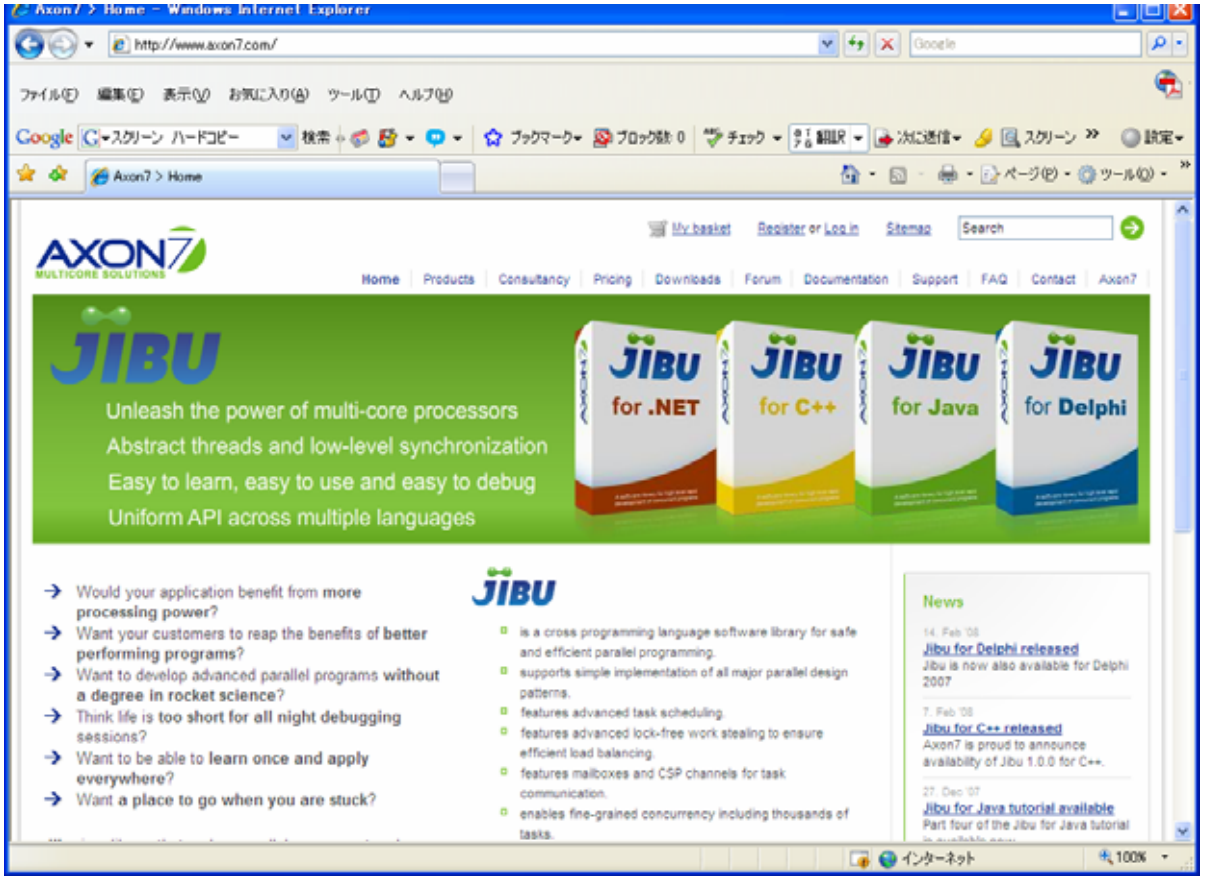
KRoC

Kent Retargetable occam Compiler

occam-π

Transterpreter

Jibu: AXON7

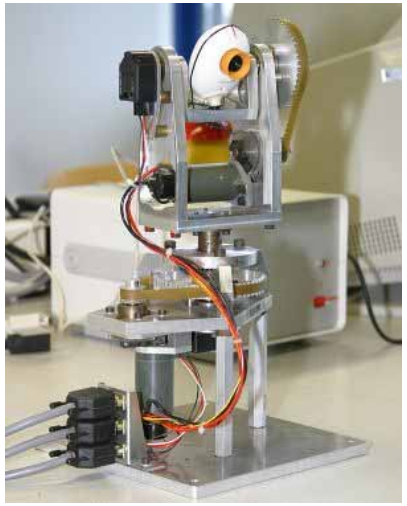


デンマークのコペンハーゲン大学で開発された。

Delphi (OO-Pascal)、Java、C++、C# のCSPライブラリーがVisual Studio .NET上で用意され、安全な並列処理プログラミングが可能となる。更にx86マルチコア対応となっている。AXON7社から発売中。(http://www.axon7.com/)

*Twente*大学でのツール開発

Tripod
A mechatronic set up with three linear motors.
Experimental setup for:
Concurrent (CT) control implementation
CT/CSP safety patterns
Formal checking of CSP software designing
Learning feed-forward control, being done in the PhD project of Bas de Kruijf
Control hardware platform:
Industrial PC under DOS
Parallel PC/104's (planned)



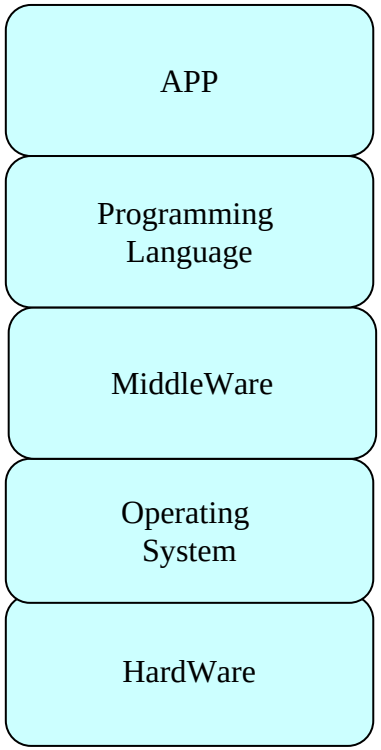
JIWY
2DOF end-effector for driving a camera
Experimental setup for:
Refinement of control laws
Concurrent (CT) control implementation
Fault tolerance by CT exception handling
Control hardware platform:
PC under DOS and RTAI Linux
Analog Devices ADSP 21992 embedded processors

University of Twente
The Netherlands

http://www.ce.utwente.nl/designtools/

LINIX
Simple motion control demonstrator
Experimental setup for:
Integrated control laws and control software automatic generation
Basic & Adaptive control (done by other projects at our group)
Control hardware platform:
Analog Devices ADSP 21990/21992 embedded processors

HWからSWまで



Microsoft Research

FormalSystems

KENT UNIVERSITY OF KENT

ERLANG

University of Twente The Netherlands

AXON7 MULTICORE SOLUTIONS

CCSP, C++CSP, JCSP, occam-pi, Transterpreter,
Jibu(Java, C++, C#, Delphi), Fortran-M, Erlang, FDR2, CSP-Prover
Visual-CSP, CHP(Haskell), PyCSP(Python), CSP++, CL(Common Lisp)
LibCSP(I, Newsqueak, Alef, Limbo, Go

adapteva

Grid-occa, OAP, LEGO Mindstorms
Microsoft Singularity OS, RMoX,
Open Licence Society

RMoX

OPEN LICENSE SOCIETY

HPI Hasso Plattner Institut

Tangram, Verilog+, VerilogCSP, Handel-C , XMOS,
TPCore, ARM, STBus, STNoC, SpaceWire(IEEE1355)

PHILIPS sense and simplicity

EPSON EXCEED YOUR VISION

USC

ARM

KALRAY

ASIC ASSP FPGA XMOS SDS

NASA

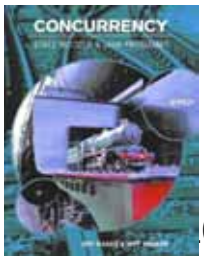
esa

JAXA

NEC/TOSHIBA

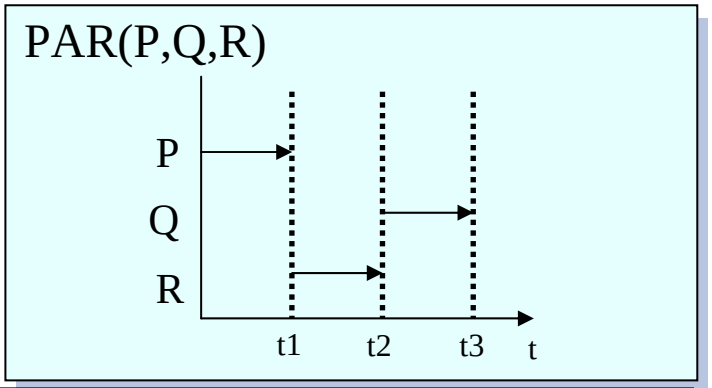
CSPモデルを採用しているJava

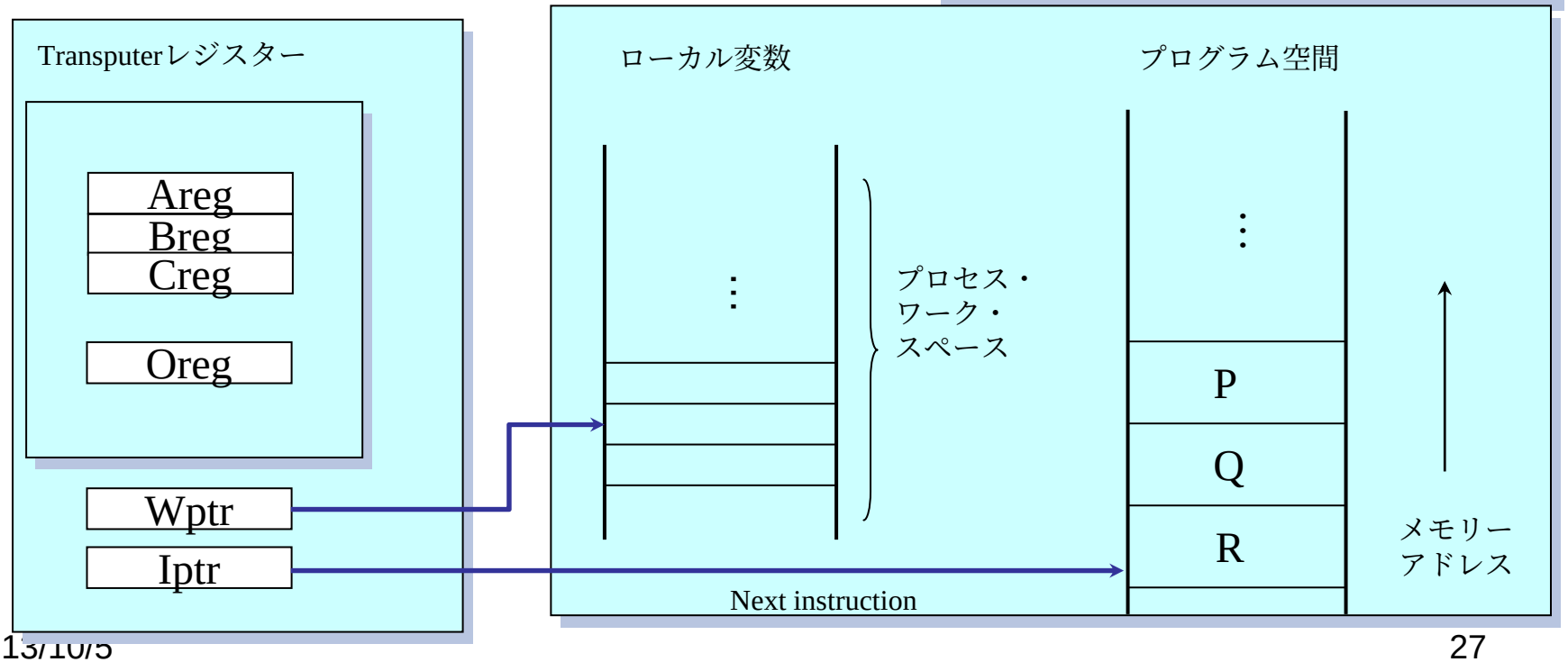
1. JCSP Basic & Network Edition（英国Kent大学
   -- http://www.cs.kent.ac.uk/projects/ofa/jcsp/）
2. CTJ（オランダTwente大学）
3. Ptolemy II（米国UCB -- http://ptolemy.eecs.berkeley.edu/）
4. Jassda（ドイツOldenburg大学）
5. Distributed Real-Time Java（ドイツPaderborn大学）
6. Ecole Polytechnique Fédérale de Lausanne
   -- (http://ltiwww.epfl.ch/sJava)
7. Oasis（フランスINRIA -- http://www.inria.fr/oasis ）
8. SCOOP（ポルトガルUniversidade do Minho）
9. Higher-Order-Concurrency in Java（カナダWaterloo大学）
10. Concurrency: State Models & Java Programs(Jeff Magee & Jeff Kramer)
    http://www.doc.ic.ac.uk/~jnm/book/ （英国インペリアルカレッジ）
11. CSP-based object-oriented description of parallel reconfiguration architectures
    http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=47542
    （米国イリノイ大学）
12. JACK: Java Architecture with CSP Kernel (Informatics Center, Federal University of Pernambuco)
13. CSP.NET(University of Copenhagen,
    http://www.cspdotnet.com/Home/tabid/36/Default.aspx)
14. Grid-occam(http://www.grid-occam.org/moin/StartSeite)



CSPモデルは非常に高速！！

Linux上で動作する**KRoC(occam)**を例に、CSPライブラリ内でどのように高速性を実現しているかを紹介します。**KRoC(occam)**は**ETC**(Extended Transputer Code)を採用し、**IA32**のコードを最適化しています。x86であっても基本的なプロセス制御は**Transputer**の方法を受け継いでいます。従ってプロセスの切換は非常に**高速**に実行されます。**50nsec**(P4@2.4GHz)のcontext switch時間と、**20nsec**のプロセスのstartup/shutdown時間が得られています。

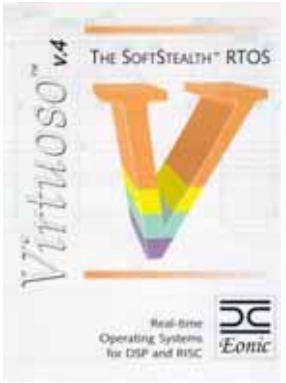




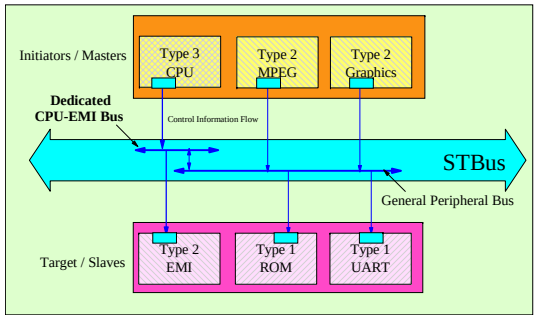
CSPモデルの応用例
（～1995年頃まで）

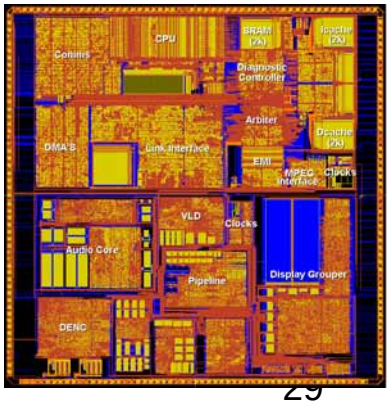
CSPモデルのアプリケーション例(1)

1. Virtuoso RTOS, EONIC SYSTEMS (http://www.eonic.com/)
2. Celoxica -- Occam compilation to FPGA (http://www.celoxica.com)
3. ARM社 – ARM プロセッサー (http://www.arm.com)
4. SpaceWire 宇宙産業ネットワーク (http://www.estec.esa.nl/tech/spacewire/)
5. Grid Computing (Fortran-M) (http://www-fp.mcs.anl.gov/~foster/)
6. STMicroelectronics社のSoCチップの内部バス(STB, DSC, DTV, GPS等)
7. 32 ports Packet Routing Switch Chip STC104 (occamで動作モデル記述)
8. 8 ports Packet Routing Switch (FPGA, UbiSwitch481 – JCSPで動作モデル記述) (http://www.4links.co.uk)
9. TCP/IPプロトコルの並列化(IBMチューリッヒ研究所)
10. SpecC System(http://www1.ics.uci.edu/~specc/)
11. 探索船Twin-Burger (http://underwater.iis.u-tokyo.ac.jp/robot/tb/tb-chp1-e.html)
12. ロボット制御 (http://www.mel.go.jp/soshiki/robot/jiritsu/project/malec-j.html)
13. Concurrent Java (JCSP, CTJ) – (http://www.cs.ukc.ac.uk/projects/ofa/jcsp/)
14. Concurrent Java (http://www-dse.doc.ic.ac.uk/concurrency/)
15. マルチエージェントシステム
16. N社衛星通信
17. KDDI高速Fax
18. レーザプリンター
19. 画像処理、グラフィックス



ST20





STBのチップ

CSPモデルのアプリケーション例(2)

20. 富士通論理LSI設計用エキスパート･システム（日経エレクトロニクス記事 1985.11.4）
21. ディスクネットワークDR-net(http://yokota-www.cs.titech.ac.jp/~yokota/drnet/exsys-j.html)

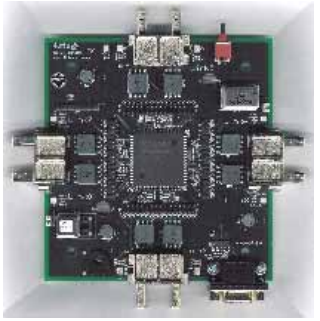
22. MIMD型並列コンピュータ- GC-MPC-128 (http://www.jaist.ac.jp/iscenter/mpc/old-machines/Parsytec/)
23. BSP(Bulk Synchronous Process)のモデル
24. Ada, occam, TRONでも採用されているRendezvous通信モデル
25. SPIN Protocol Validator (Gerald J. Holzmann, AT&T Bell Lab)
26. PROMELA(PROcess MEta LAnguage)

27. Parallel CAMAC Project （高エネルギー加速機構 – http://www-online.kek.jp/~yasu/Parallel-CAMAC/）
28. UML2.0のセマンティックス
29. クライアント･サーバシステムにおける信頼性向上の一手法 –UNISYS (http://www.unisys.co.jp/tec_info/tr52/5202.htm)

Twin-Burger

GC-MPC-128

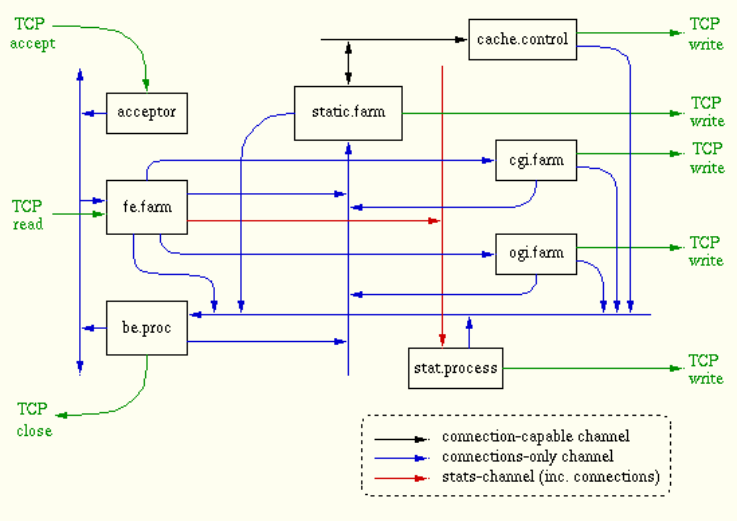
1355 Routing Switch



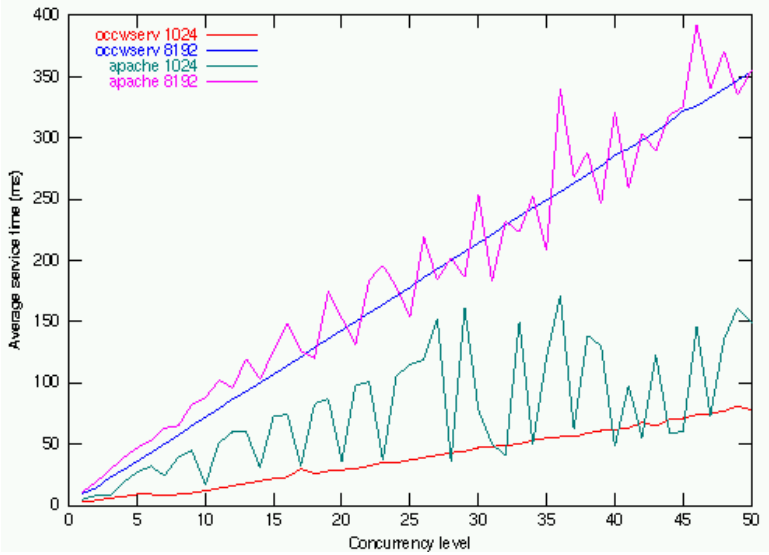
4Links社(http://4Links.co.uk/)

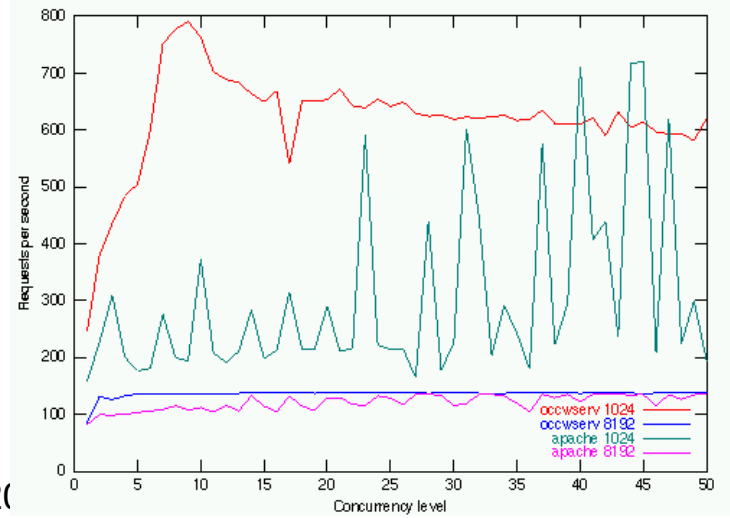
CSPモデルのアプリケーション例(3)

31. occwserv: An occam Web-Server (http://wotug.ukc.ac.uk/ocweb/)

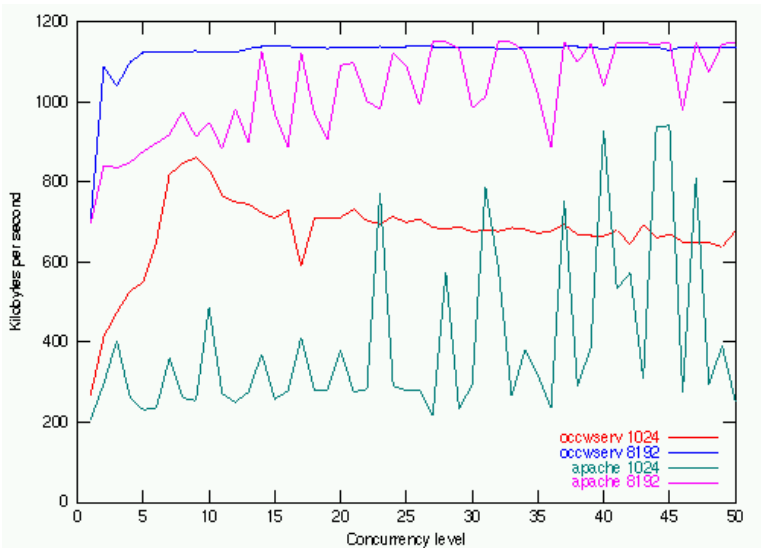


http://wotug.ukc.ac.uk/ocweb/





http://wotug.ukc.ac.uk/ocweb/perf.html



CSPモデルのアプリケーション例(4)

32. TransputerのFPGA化 (首都大学東京　福永研究室)
http://tmubsun.center.metro-u.ac.jp/index_j.html

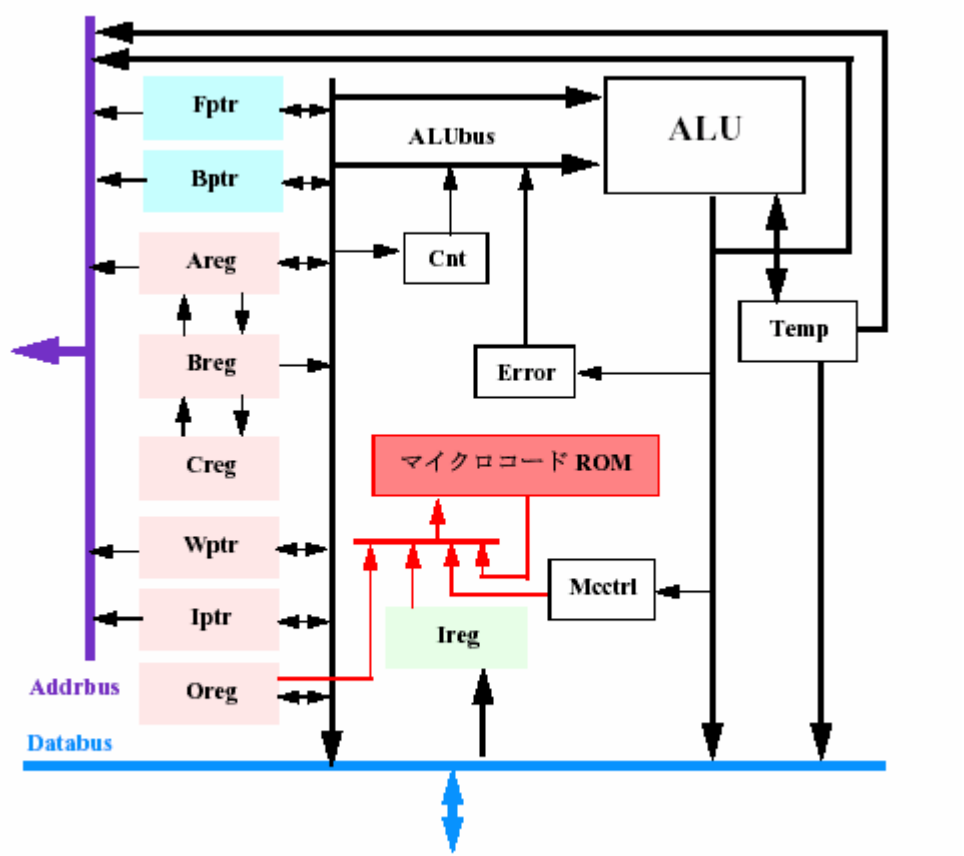


図 4.1: CPU 部の概要図

CSPモデルのアプリケーション例(5)

33. Concurrent Programming ERLANG
(http://www.erlang.org/ , http://www.erlang-projects.org/)
当初ERICSSONで開発されテレコムで使用

34. Concurrent System Interface Modeles – VME (IMEC、**ベルギー**)

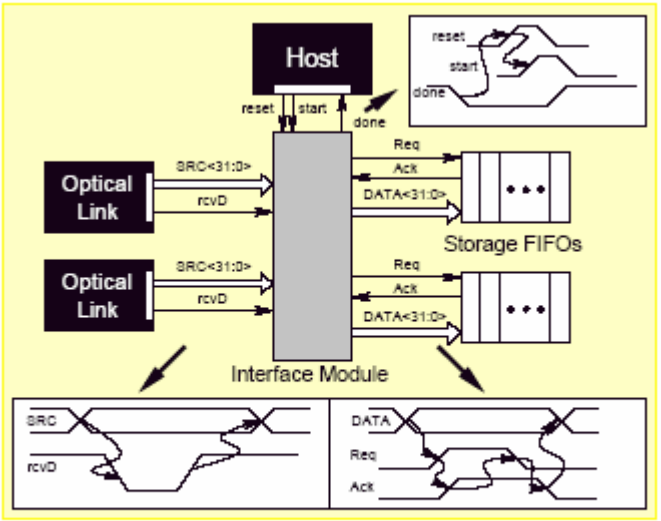


Figure 1: Interface for dual-optical source to FIFO buffers.

```
extern function decode (address); /* addr dec */
protocol vme; /* VME protocol */
protocol mem; /* memory protocol */

process vme_to_mem :
channel addr1⟨14:1⟩, data1⟨7:0⟩, write : vme;
channel addr2⟨10:1⟩, data2⟨7:0⟩, we : mem;
{
   boolean mode, x⟨14:1⟩, y⟨7:0⟩, z⟨7:0⟩;
   write?*mode || addr1?*x ;
   if (decode (x⟨14:11⟩) == 1) {
      if (mode == 0) { /* read mode */
         addr2!x || we!mode || { data1?y ; data2!y }
      } else { /* write mode */
         addr2!x || we!mode || { data2?z ; data1!z }
} } }
```

Figure 11: Specification of VMEbus to processor memory interface.

35. IMPLEMENTING APPLICATION FRAMEWORKS: OBJECT-ORIENTED FRAMEWORKS AT WORK (http://www.cse.unl.edu/~fayad/Books/ImplemB2/preface.htm) -- IBM

CSPモデルのアプリケーション例(6)

36. SuperH CPU Core
マルチメディアSoC(System-on Chip)デバイス用のRISC CPUコアファミリ。32ビットのSH-4ファミリおよび64ビットのSH-5ファミリをリリースしています。SH-4 CPUコアには、音声および画像応用に開発されたベクタ浮動小数点演算器が搭載されています。SH-5ファミリは64ビットのデータパスで、SIMD(Single Instruction, Multiple Data)命令を用いたマルチメディア処理が可能です。それぞれのCPUコアファミリは、シンセサイザブルコアおよびハードマクロの両方を含む幅広い製品形態を用意しています。 詳細はアーキテクチャのページを参照ください。
また、コア外のSoC設計をサポートするため、CSP(Core Support Peripherals)と呼ぶ周辺デバイスのセットを、ライセンシのSoC設計用に提供します。
SuperHyway Bus™
SuperHHyway Bus™ はVSIA/VCI準拠のSoC用オンチップバスです。スケーラブルな高バンド幅と低レイテンシを実現するバスです。SuperHywayをベースとした設計を行うライセンシに対して構築ツールが提供されます。このツールはSEDK(Soc Evaluation Design Kit)に含まれます。AMBAやCoreConnect等の他の標準バスへのインタフェースにより異なるIPライブラリの周辺が接続可能となります。詳細は、SuperHywayのページを参照ください。

これはESPRITのChameleonプロジェクトの成果である。David MayがPACTで指揮をした。SuperHyway Busはクロスバスイッチを使った方式で、内部バスはCSPモデルを使っている。その後日立とSTの共同プロジェクトが実現した。

*1355(HIC)*ネットワークとは

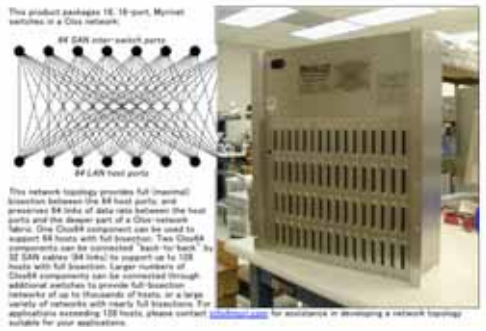
- HIC(Heterogeneous InterConnect)
- ST(Inmos)とBullが中心になって標準化した。
- IEEE1355-1995が標準化された。
- CPU/Board間を相互接続する技術。
- あらゆる異なったプロトコルがネットワーク上に相互乗り入れが可能となる。
- CSPモデルによるイベント駆動による大規模な並列化
- ST(Inmos)社のT9000の通信リンク(DS-Link)とRouter C104がリファレンスのモデルとされた。
- 低価格、高性能、高信頼性の実現。
- 現在はSpaceWireとして宇宙産業機器のネットワークと規格化されている。

IEEE1355の他のネットワークへのインパクト

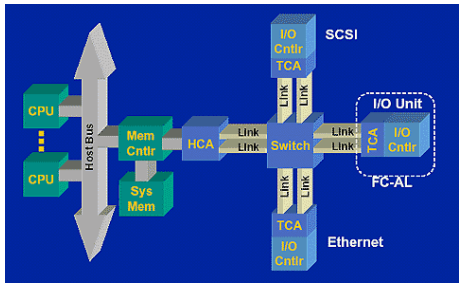
MPU間のシリアル相互接続はInmos社のT800が最初で、その後大規模なネットワークはT9000, C104で実現される（1991年）。

DS-LinkはSTMicroelectronics(旧Inmos)の特許。

- FireWire(IEEE1394)
  - 1394a のPHY部はApple社のMike Teenerが1355のDS-Linkの性能に注目して採用する。
  - 1394b(Long Haul/High Speed) はSTのColin Whitby-Strevensが1355のHS-Linkの経験を導入した。
- USB – 1394に対抗してキーボード、マウス周りのシリアルリンクが出現。
- Myrinet(1997)
- PCI-Express(2002)
- InfiniBand(1999)
- NGIO(Next Generation IO-1999)



いずれも1355のパケット・スイッチングと類似した相互の接続手法を採用している



*CSP*モデルの応用例（*1995*年以降）

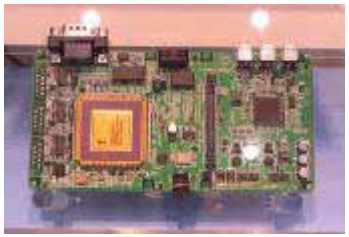
SpaceWire製品

- NASA(http://ipp.gsfc.nasa.gov/ft-tech-spacewire.html)
- ESA(http://spacewire.esa.int/content/Home/HomeIntro.php)
- JAXA(http://spacewire.esa.int/content/Missions/JAXA.php)
- NEC SpaceCube(http://techon.nikkeibp.co.jp/article/NEWS/20051214/111641/)
- 4Links(http://www.4links.co.uk)
- STAR-Dundee(http://www.star-dundee.com)
- Aeroflex(http://www.aeroflex.com/aboutus/pressroom/newsrelease/2006/071706
- ATMEL(http://www.atmel.com/)
- Gaisler Research (http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=262&Itemid=179)
- Dynamic Engineering(http://www.dyneng.com/spacewire.html)
- Saab Ericsson Space (http://www.evertiq.com/news/read.do?news=1043&cat=2)

- Aurelia Microlettronica (www.caen.it/micro/pdf/Products_Services_Overview_2006-2007.pdf)

【TRONSHOW】宇宙分野でもT-Engine，その名も「SpaceCube」

http://techon.nikkeibp.co.jp/article/NEWS/20051214/111641/



「Teacube」や「EvaCube」など立方体状のT-Engine評価システムが，ついに宇宙分野にも進出する。NECソフトは，Teacubeを基にして宇宙機用に変更を加えたT-Engine評価システム「SpaceCube」を，2005年12月14日から開催中のTRON関連の展示会「TRONSHOW 2006」で展示した。人工衛星などの宇宙機で用いられるインタフェース「SpaceWire」を備えるのが特徴である。NECグループでは，NEC東芝スペースシステムが人工衛星の開発などを手掛けており，NECソフトがそのソフトウエア開発などを受託している関係から，SpaceCubeの開発に至ったという。

今回のSpaceCubeは，あくまでソフトウエア開発の支援用であり，このまま人工衛星などに搭載するわけではないが，「プリント基板がむき出しの評価ボードより，立方体状でスマートな筐体に収まっている方が，開発作業も楽しい」（NECソフトの説明員）とする。今後，人工衛星の組み込みソフトウエア開発現場などに売り込んでいくという。

SpaceCubeは，FPGAを搭載するボードをTeacubeに追加することで，3チャネルのSpaceWireインタフェースに対応した。搭載するマイコンはNECエレクトロニクスのVR5701であり，Teacubeと同じである。

SpaceWireは，物理層に差動伝送方式の「LVDS」を採用し，送受信用のFIFOを組み合わせた簡素なインタフェースである。IEEE1355を基にしている。FPGAなどに搭載されているLVDSの送受信回路さえ用意すれば，特別なインタフェースLSIは必要ないため，安価にシステムを構成できるという。データ伝送速度は最大200Mビット/秒で，ネットワークのトポロジーもスター型やツリー型，リング型などを選択可能で自由度が高い。宇宙航空研究開発機構（JAXA）や米NASA，欧州のESAなどが共同で策定したものである。

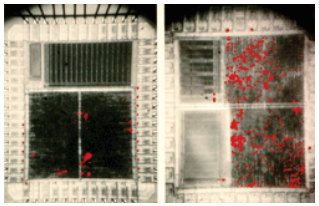
会場ではSpaceCubeと接続して使う拡張ボード「SpaceWire Extension for T-Engine」も参考出展した。SpaceWireを3チャネル，CAN 2.0Bを1チャネル，IEEE1394a-2000を3チャネル搭載しており，これらの間でのプロトコル変換やゲートウエイ用途などに向ける。NECソフトとNEC東芝スペースシステムが共同開発した。

NEC東芝スペース（株）以外に6社が真剣に研究開発をしている。
経済産業書、NEDO、JAXA、SJAC、USEF等も注目しており、今後の展開が期待される。

ハードウェア記述言語(*HDL*)

- Tangram (Philips)
  - http://www.eetimes.com/story/OEG20030331S0020

左側が非同期型で、発熱は少ない。



右側が同期型で、赤い点が発熱している場所を表す。

- Verilog+（セイコー・エプソン）
  - http://www.epson.co.jp/e/newsroom/tech_news/tnl0505single.pdf
- VerilogCSP (University of Southern California)
  - http://jungfrau.usc.edu/new/research/current/verilogcsp/
- Handel-C(Agility)
  - http://www.celoxica.co.jp/
- XMOS
  - http://www.xmos.com/
- Basla (University of Manchester)

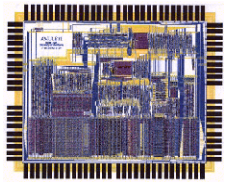
これらはCSPモデルを直接実装している。

- SpecC
- SystemC

channel, par, pipe, protocol, synchronize、階層化、タイミングなど CSPモデルの影響を強く受けている。

非同期回路(*Asynchronous Circuit*)

MPUの消費電力低下のために必要な技術



- Tangram (Handshake Circuit)
  - Philips社でのEDAツール
  - http://www.eetimes.com/story/OEG20030331S0020
- occarm
  - http://www.cs.bham.ac.uk/~gkt/Research/occarm/occarm_home.html
- ARM
  - http://www.arm.com
- Seiko EPSON
  - CQ出版Design Wave, p63-p84, 2005 July

イベント駆動型による低消費電力化の実現!!

イベント駆動型チップ*ARM*プロセッサー

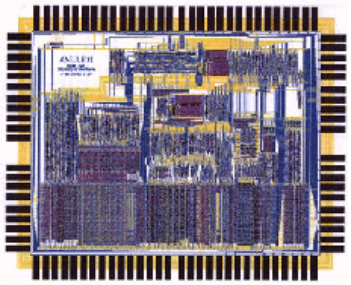
OCCARM is a distributed discrete event simulator of the AMULET1 microprocessor, an asynchronous version of the ARM RISC processor which has been developed by the AMULET Group, in the Department of Computer Science, University of Manchester within the ESPRIT OMI-MAP project.

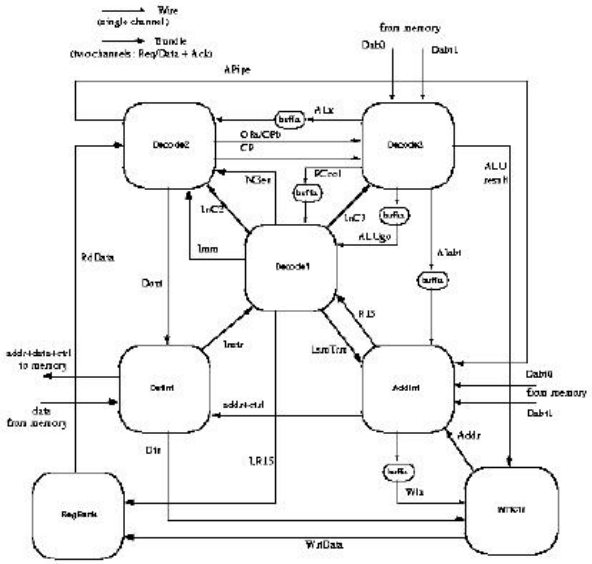
The motivation for developing OCCARM has been the increasing debate regarding the potential use of **CSP** for modeling the non-deterministic parallel behaviour and structure of asynchronous hardware. The main results of this endeavour have been presented in various publications.

OCCARM is written in occam (the name OCCARM derives from the combination of **occ**am and **ARM**). It consists of more than 15.000 lines of occam code and describes AMULET1 at the Register Transfer Level. It is instruction driven. It executes ARM6 machine code produced by a standard ARM compiler. Instructions enter the simulator as 32-bit quantities in hexadecimal format. Instruction decoding is performed by means of PLA models implemented as two dimensional arrays of boolean values; the model makes use of a library of occam functions developed to allow instructions to be treated both as integer values and as one dimensional boolean arrays.

OCCARM code and sources are available upon request.

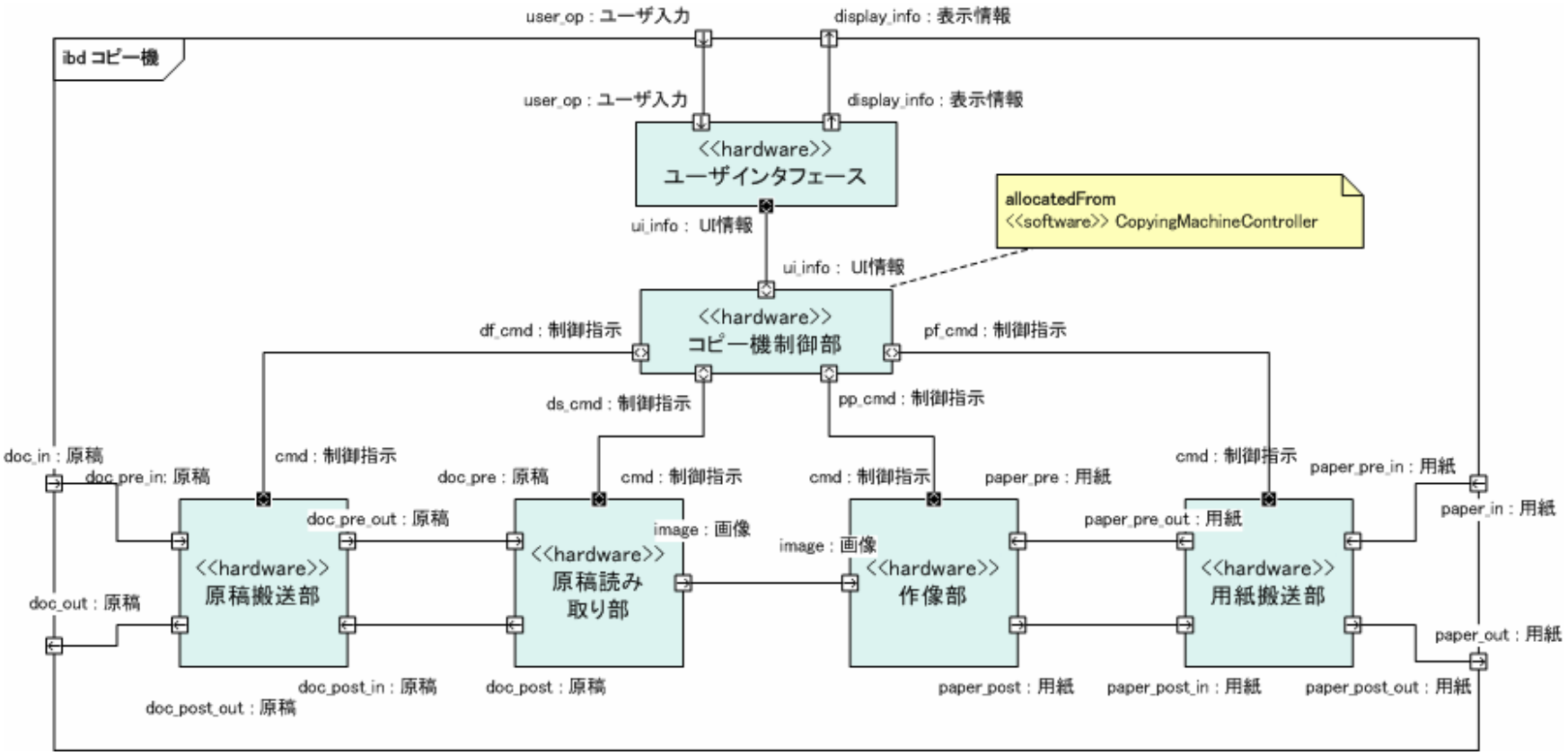
OCCARM was initially configured to run on the ParSiFal computer, a 64-Transputer multiprocessor machine developed in the Department of Computer Science, University of Manchester. Both, single and multi-transputer configurations havebeen developed. A number of techniques have been devised for collecting and analysing monitoring information. OCCARM is currently being ported on a workstation network using the **KRoC** system.





http://www.cs.bham.ac.uk/~gkt/Research/Occarm/occarm_home.html

アーキテクチャ記述言語

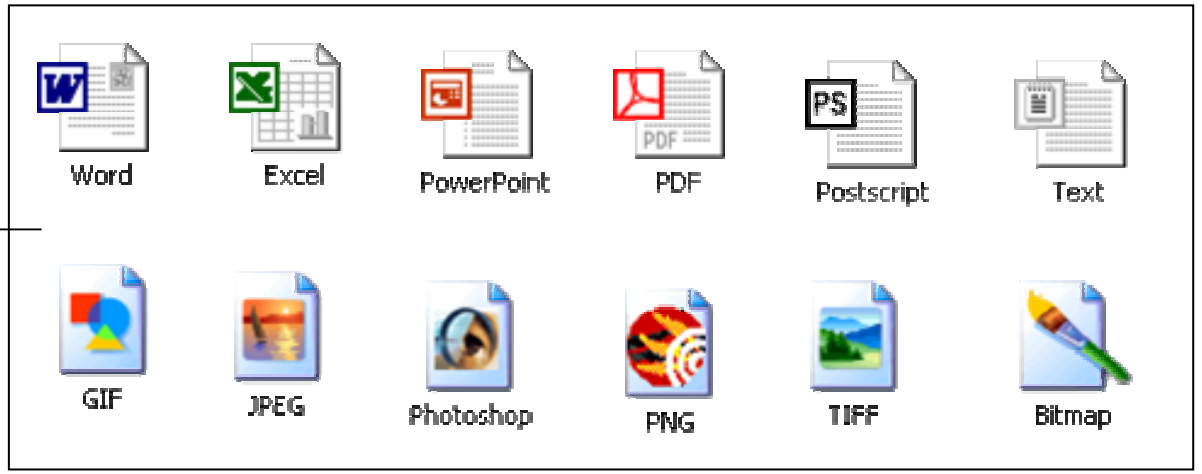


SysML、リアルタイム処理をサポートしたAADLのコンポーネント間には
相互作用があり、CSP/Timed CSPモデルを導入する事ができる。

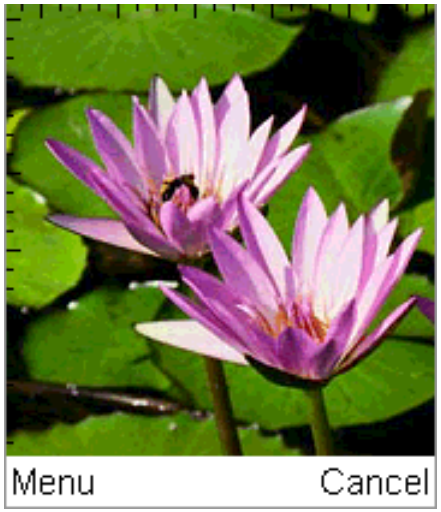
(Ultra Literally Air)

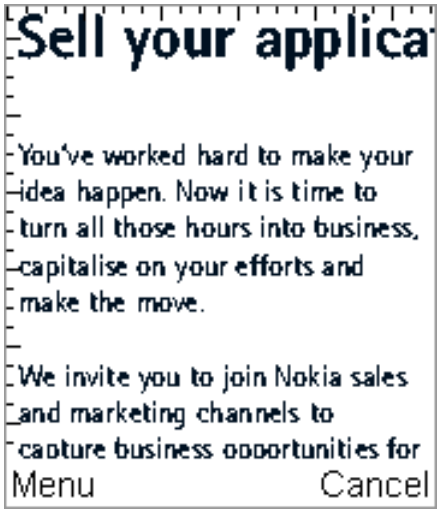
NOKIA携帯電話

文書、Fax, emailが携帯、PDAで見ることができます。



JCSPはノキアの携帯電話に搭載されている。

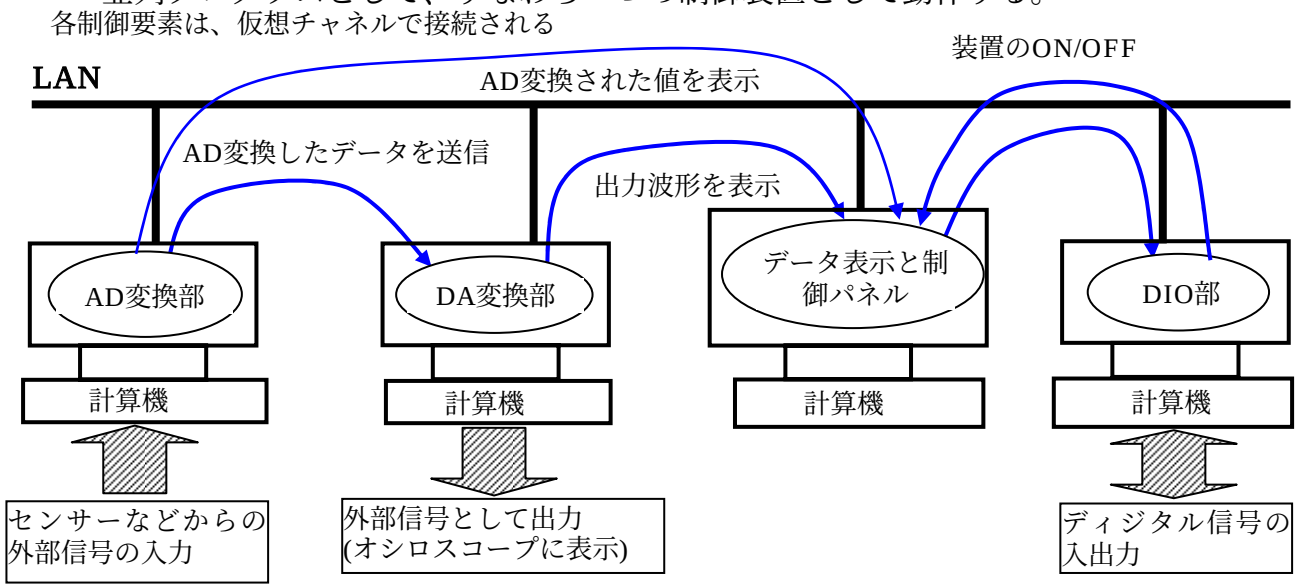




CSPモデルに基づいた分散型制御装置の開発

計算機を用いた制御装置は、一般にDA変換部、AD変換部、データ表示部、制御部などに分けて考えることができる。通常、これらは制御ループに組み込まれるので物理的に近い位置にあり頻度の高い通信が要求されるが、本研究ではJCSPと呼ばれる並列／分散処理ツールを用い、DA変換部、AD変換部あるいはデータ表示部などの制御要素を複数のコンピュータに配置することができる分散型の制御装置の構成を試みた。

JCSPは、CSP(Communicating Sequential Processes)モデルと呼ばれる並列処理モデルをJava言語で利用できるように開発されたクラスライブラリーである。これを用いることにより互いに独立しているプロセスがチャネルを介して互いに通信しあいながら同期して動作する並列プログラムを作成することができる。さらに、JCSPのプロセスはLAN上の仮想チャネルで接続することができるので離れた場所でも容易にデータの送受信を行うことが出来る。試作した制御装置は下図に示すように４台のコンピュータにそれぞれDA変換ボード、AD変換ボード、データ表示部およびDIOボードを配置し、全体で一つの制御装置として機能するように各部に対応するDAプロセス、ADプロセス、データ表示プロセスおよびDIOプロセスを開発した。これらのプロセスは、いずれもチャネルを入出力とするCSPモデルになっており、LAN上の仮想チャネルで通信しながら同期して動作し、一つの並列プログラムとして、すなわち一つの制御装置として動作する。



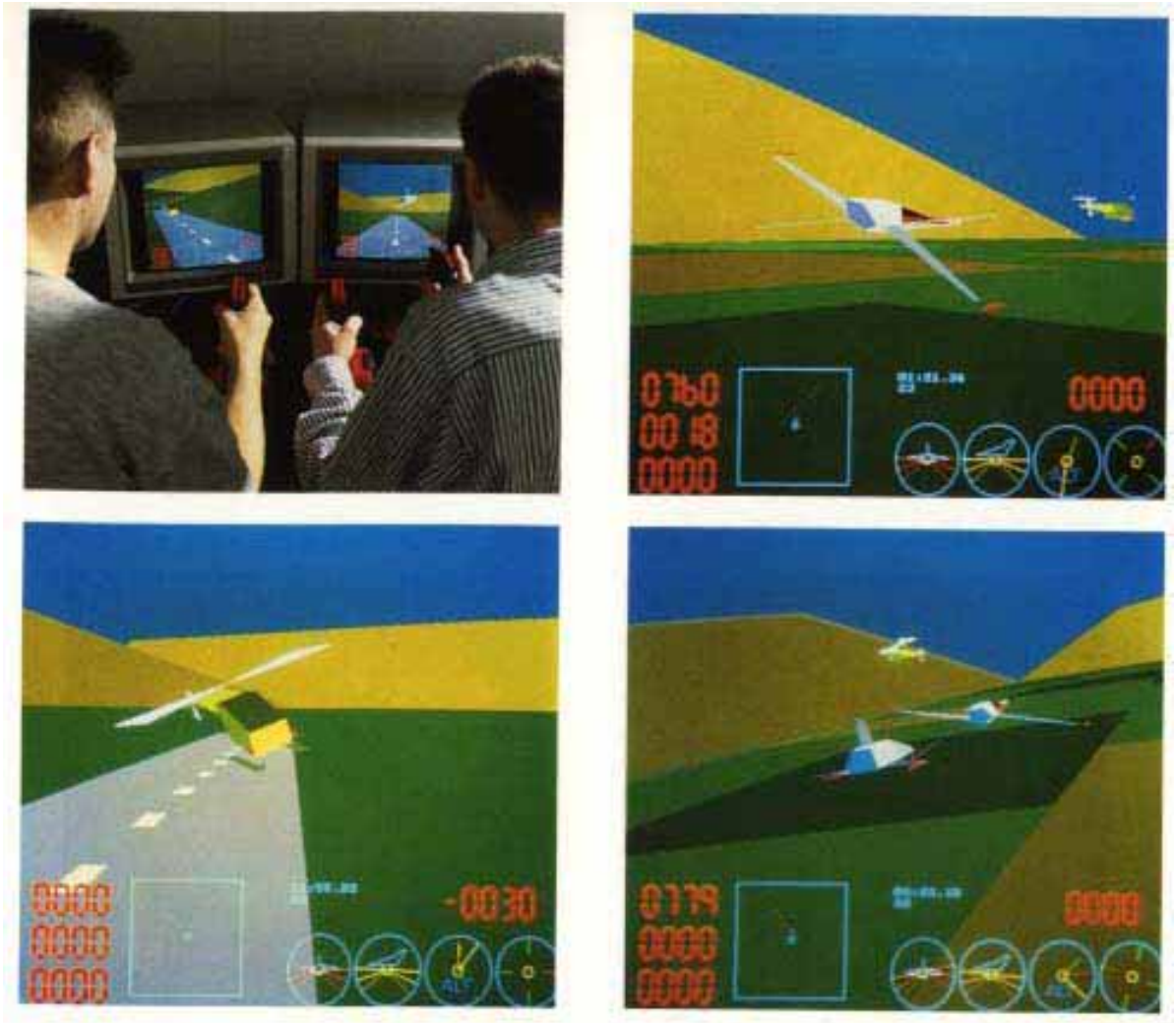
分散型制御装置の構成例

GPars(Groovy Parallel Systems)

- Concurrent collection processing
- Composable asynchronous functions
- Fork/Join
- Actor programming model
- Dataflow concurrency constructs
- **CSP**
- Agent
- 他

http://gpars.codehaus.org/

Inmos Flight Simulator (1987)



Erlang(Ericsson Language)

- Ericsson社において考案され、テレコム分野の開発ツールとして使われている。
  - http://www.erlang.org/
  - http://ja.wikipedia.org/wiki/Erlang
  - http://www.ericsson.com/technology/opensource/erlang/index.shtml
  - 分散化された環境
  - フォルトトレラント
  - リアルタイム性
  - 無停電で稼動
- CSP/occamのモデルを強く受けている。

Esterel

- 同期型並列処理言語(Lustre)
- リアクティブなシステムに適用
- 宇宙、航空、自動車で多くの実績あり
- CSPモデルはSDF(Synchronous Data Flow)の仕組みとして実装されている。
  - http://ja.wikipedia.org/wiki/Esterel
  - http://www.esterel-technologies.com/

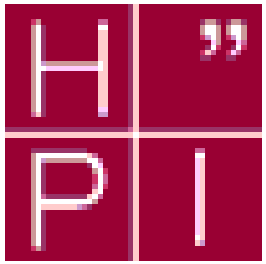
CSPモデルによるOperating System

- RMoX(Raw-Metal occam Experiment)
  - http://rmox.net/
- Microsoft Singularity OS
  - http://research.microsoft.com/os/singul
  - http://en.wikipedia.org/wiki/Singularity_(operating_system)
  - RMoXの影響を強く受けている。
- Open License Society
  - http://www.openlicensesociety.org/drupal55/index.php
  - RTOSはEONIC社の頃から開発されており、CSPモデルを採用している。
  - Eric Verhulstは1989年頃から活動をしている。
  - 応用はオートモーティブにターゲットを置いている。

*occam*コンパイラ

- KRoC(Kent Retargetable occam-(pi) Compiler)
  - http://www.cs.kent.ac.uk/projects/ofa/kroc/
- Transterpreter(Transputer Interpreter)
  - Transputerのbyteコードを翻訳(interprete)する。
  - http://www.transterpreter.org/
  - Transputer VMとも呼ばれる。
  - Lego Mindstorm, Cellチップ、mobile robotに移植されている。
- SPoC(Southampton Portable occam Compiler)
  - http://www.hpcc.ecs.soton.ac.uk/hpci/tools/index.htm
  - http://www.itk.ntnu.no/ansatte/Hendseth_Sverre/occam/index.html
  - occam2.5からCコードが出力される。

The Grid-occam Project



Dipl.-Inf. Bernhard Rabe
Dipl.-Inf. Peter Tröger
Dr. Martin von Löwis
Prof. Dr. rer. nat. habil. Andreas Polze

Operating Systems & Middleware Group
Hasso-Plattner-Institute, University of Potsdam

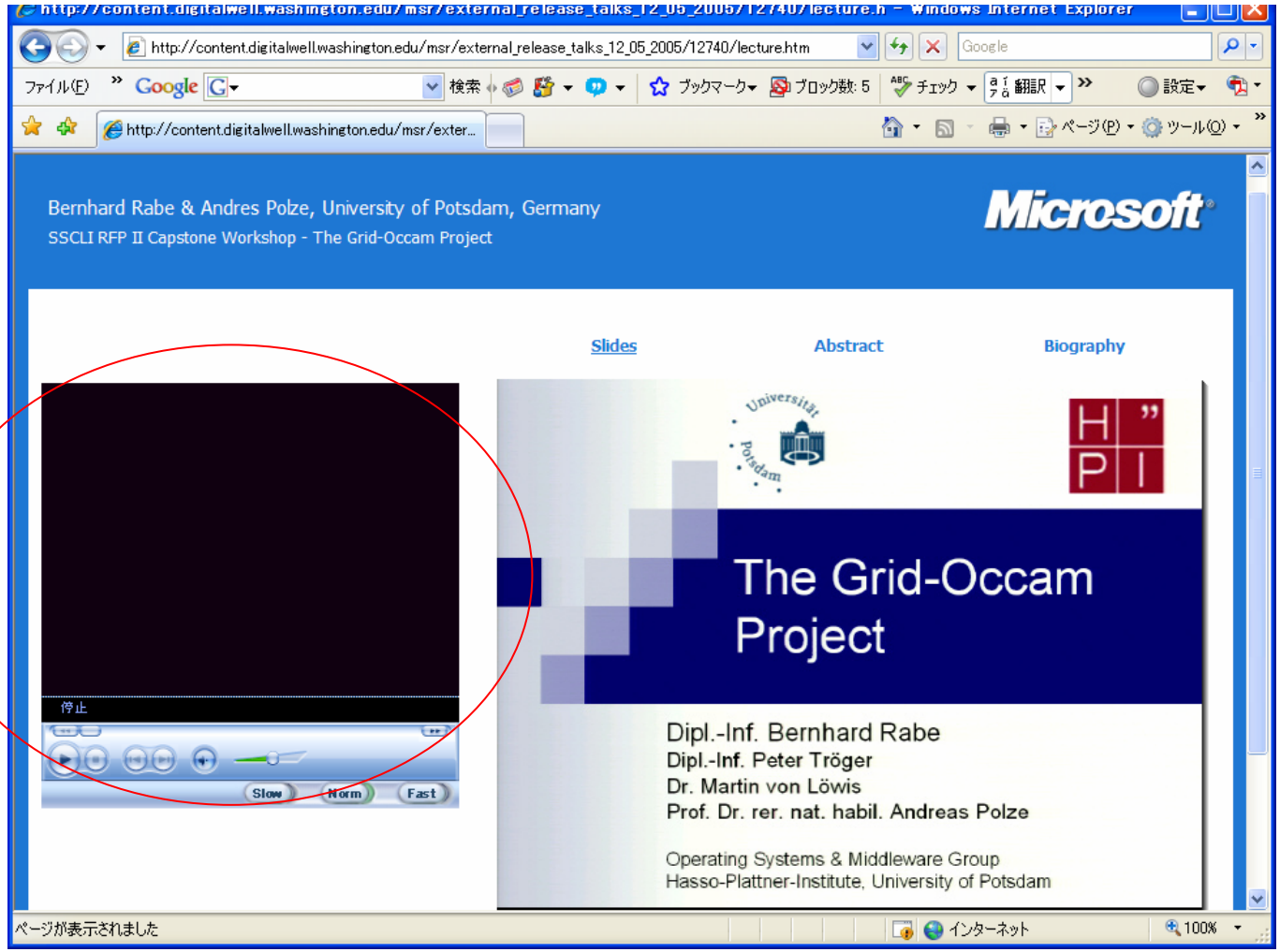
マイクロソフト社が開発資金を出している。
商用化される可能性は高い。

http://research.microsoft.com/collaboration/university/europe/RFP/rotor/2004Projects.aspx

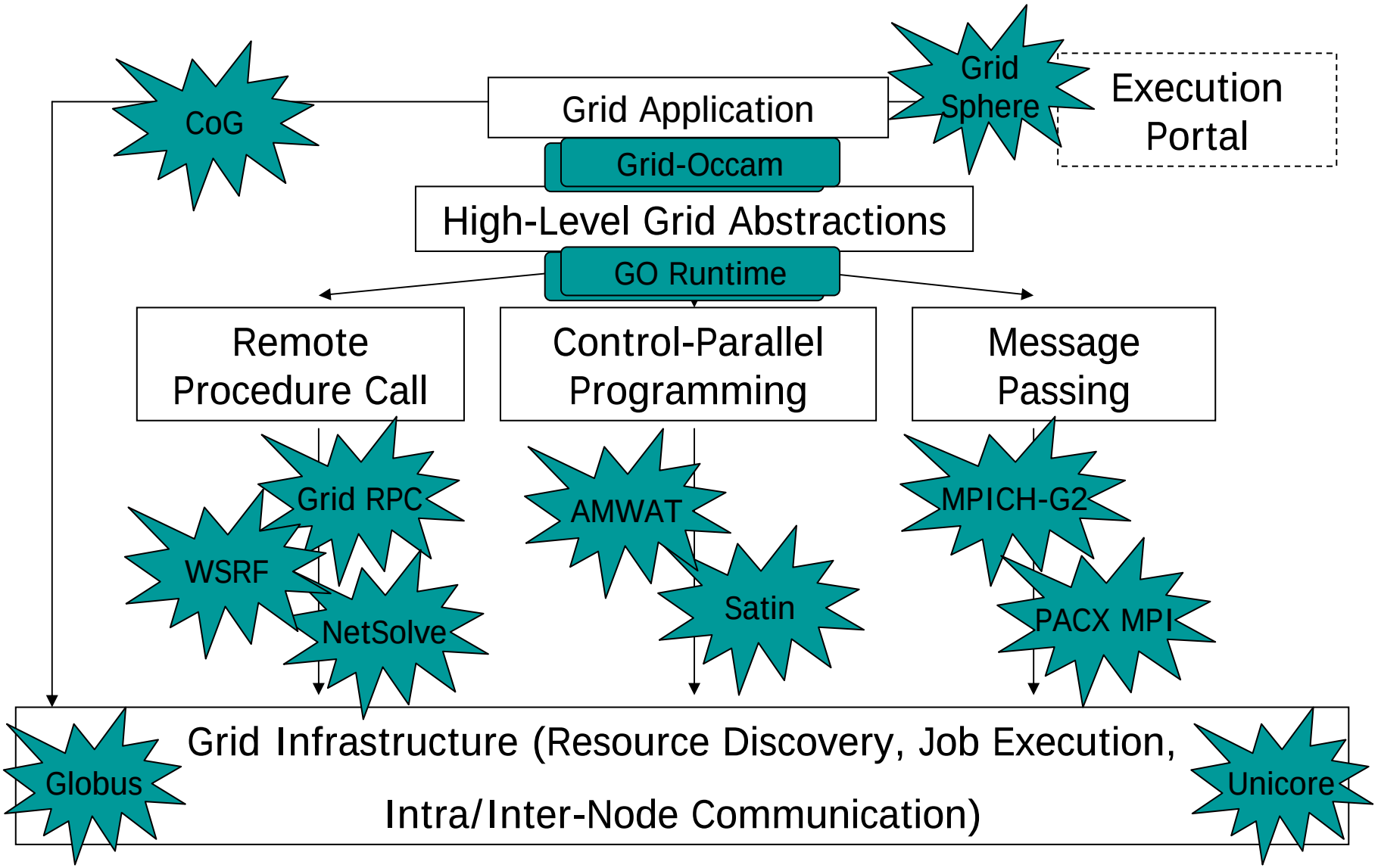
Southampton大学にいたTony Heyはマイクロソフト社の副社長に抜擢される。
彼はCSP/occam/Transputerの啓蒙者であった。

Grid-occam Project

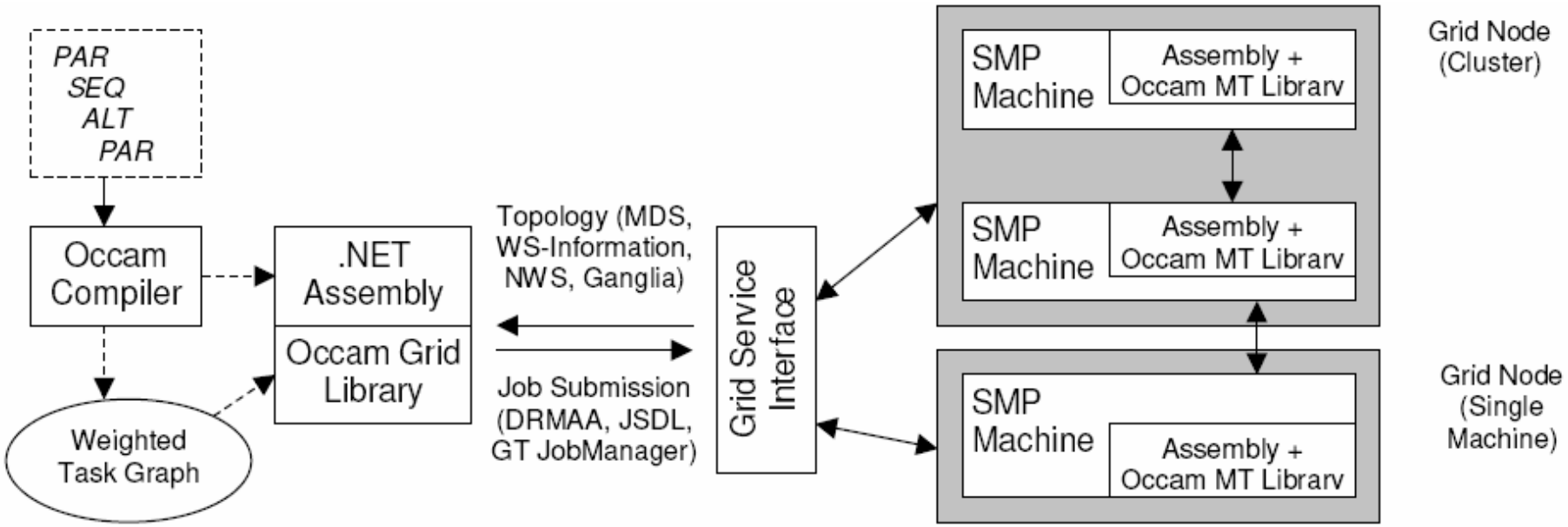
Grid-occam プロジェクトに関する簡単なビデオ映像が見られる。



Building a Grid Application



Grid-occam Execution Environment



SSDL(SOAP Service Description Language)

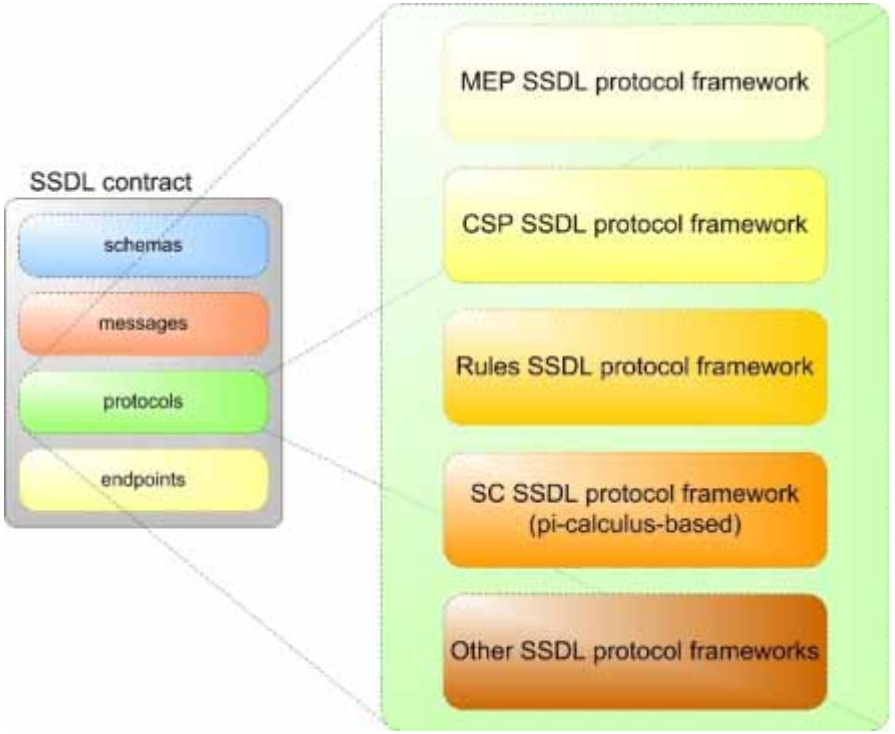
MEP - The Message Exchange Patterns (MEP) Framework defines a collection of XML Infoset element information items that represent commonly used simple exchange patterns. The current set of message exchange patterns supported by the MEP framework is a superset of that found in the latest draft of the WSDL specification.

CSP – The Communicating Sequential Processes (CSP) Framework defines a collection of XML Infoset element information items for defining a multi-message exchange using sequential process semantics, based on basic CSP semantics. Work is currently underway to make use of the full strength of the CSP in describing contracts.

Rules - The Rules-based SSDL Protocol Framework defines a collection of XML Infoset element information items that can be used to describe a multi-message exchange protocol using conditions. The protocols captured using the Rules-based SSDL protocol framework can be validated for correctness, liveness, and other properties.

SC - The Sequencing Constraints (SC) Protocol Framework defines a collection of XML Infoset element information items that can be used to describe a multi-party, multi-message exchange protocol using notations based on the pi-calculus. Protocols in the framework are specified using a sequential technique, specifying the legal set of actions at each stage of the protocol. The framework is intended to provide a simple way of specifying protocols but also have a formal basis to allow properties of the protocols to be determined.

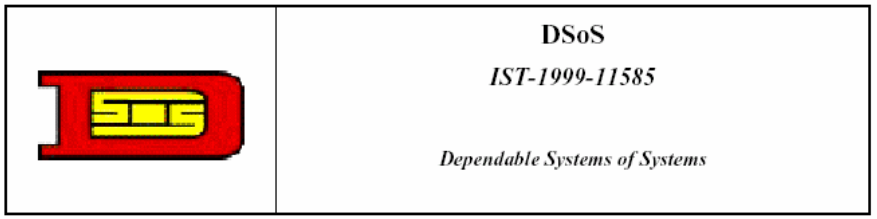
SOAP**プロトコルに**CSP**モデルが採用されている**



現在.NET対応のソフトウェアツールを準備中。

CSPモデルのWebへの適用

```
<xsd:complexType name=``statement''>
<xsd:sequence>
<xsd:choice maxOccurs=unbounded/>
<xsd:element name=``Assign'' type=``assign''/>
<xsd:element name=``Sequence'' type=``sequence''/>
<xsd:element name=``Par'' type=``par''/>
<xsd:element name=``Choice'' type=``choice''/>
<xsd:element name=``Iteration'' type=``while''/>
<xsd:element name=``Call'' type=``call''/>
<xsd:element name=``Returns'' type=``return''/>
<xsd:element name=``Send'' type=``send''/>
<xsd:element name=``OnInput'' type=``onInput''/>
<xsd:element name=``Wait'' type=``wait''/>
<xsd:element name=``Raise'' type=``raise''/>
<xsd:element name=``Try'' type=``try''/>
</xsd:choice>
</xsd:sequence>
</xsd:complexType>
```



WSCAL statements are quite similar to the ones introduced by XML-based languages for the specification of composite services (see Section 1.2.1 for references). We more specifically base the definition of WSCAL on the **CSP** language for the base statements, providing a sound basis towards reasoning about WSCAL specifications. We thus detail here only the statements that are specific to Web Services composition, i.e., handling of interactions with participants and Web Services, and of exceptions (element given in bold face in the above definition of Statement).

セマンティック・ウェブ

W3Cのサイトに CSP/Pi-Calculusのガイドラインが記されている。

http://www.w3.org/2001/sw/BestPractices/SE/ODA/

4.2 Ontologies as Formal Model Specifications and the Incorporation of Such Models in Semi-Formal Languages Example C: A cooperative Semantic Web-based environment for semantic link among models According to [Brown 2004], a formal underpinning for describing models facilitates meaningful integration and transformation among models, and is the basis for automation through tools. However, in existing MDA practice, semi-formal metamodels instead of formal specification languages are used as such formal underpinnings for describing models. An obvious reason is that, unlike UML, the industrial effort for standardising diagramatic notations, a single dominating formal specification language does not exist. **Furthermore, different specifaction languages are designed for different purposes; e.g., B/VDM/Z* are designed for modelling data and states, while CSP/CCS/π-calculus** are designed for modelling behaviors and interactions.**

[Wang 2004] briefly describes such an Semantic Web-based enviroment for semantic link among models, using DAML+OIL (instead of OWL). Examples of semantic links include assertions that Object-Z classes are (treated as) equivalent to CSP processes and that Object-Z operations are equivalent to CSP events.

Bell研究所

Bell研究所では分散コンピューティングに際してCSPモデルのプログラミング言語(Squeak, Newsqueak, Alef, Linbo)を開発しています。 Squeak, Newsqueak, Alef, LinboはCSP/occamの影響を受けています！！

http://en.wikipedia.org/wiki/Plan_9_from_Bell_Labs

http://swtch.com/~rsc/thread/

http://video.google.com/videoplay?docid=8102320126179653 44#

Google Go 言語

What are Go's ancestors?
Go is mostly in the C family (basic syntax), with significant input from the Pascal/Modula/Oberon family (declarations, packages), plus some ideas from languages inspired by Tony Hoare's CSP, such as Newsqueak and Limbo (concurrency). However, it is a new language across the board. In every respect the language was designed by thinking about what programmers do and how to make programming, at least the kind of programming we do, more effective, which means more fun.
…
…

Why build concurrency on the ideas of CSP?
Concurrency and multi-threaded programming have a reputation for difficulty. We believe the problem is due partly to complex designs such as pthreads and partly to overemphasis on low-level details such as mutexes, condition variables, and even memory barriers. Higher-level interfaces enable much simpler code, even if there are still mutexes and such under the covers.
One of the most successful models for providing high-level linguistic support for concurrency comes from Hoare's Communicating Sequential Processes, or CSP. Occam and Erlang are two well known languages that stem from CSP. Go's concurrency primitives derive from a different part of the family tree whose main contribution is the powerful notion of channels as first class objects.

http://golang.org/
http://golang.org/doc/go_lang_faq.html
http://newsnidea.com/19752/googles-go-programming-language-tutorial-video/

(3)の並列処理の記述は，CSP (communicating sequential process)と呼ぶプロセス代数のモデルに基づいている。CSPは1978年にTony Hoare氏が提案した考え方で，データの入出力を一つのアトミックな操作として定義するというもの。共有データを持たないことで，ロックやセマフォといった同期のメカニズムを使わずに並列処理を記述できる。

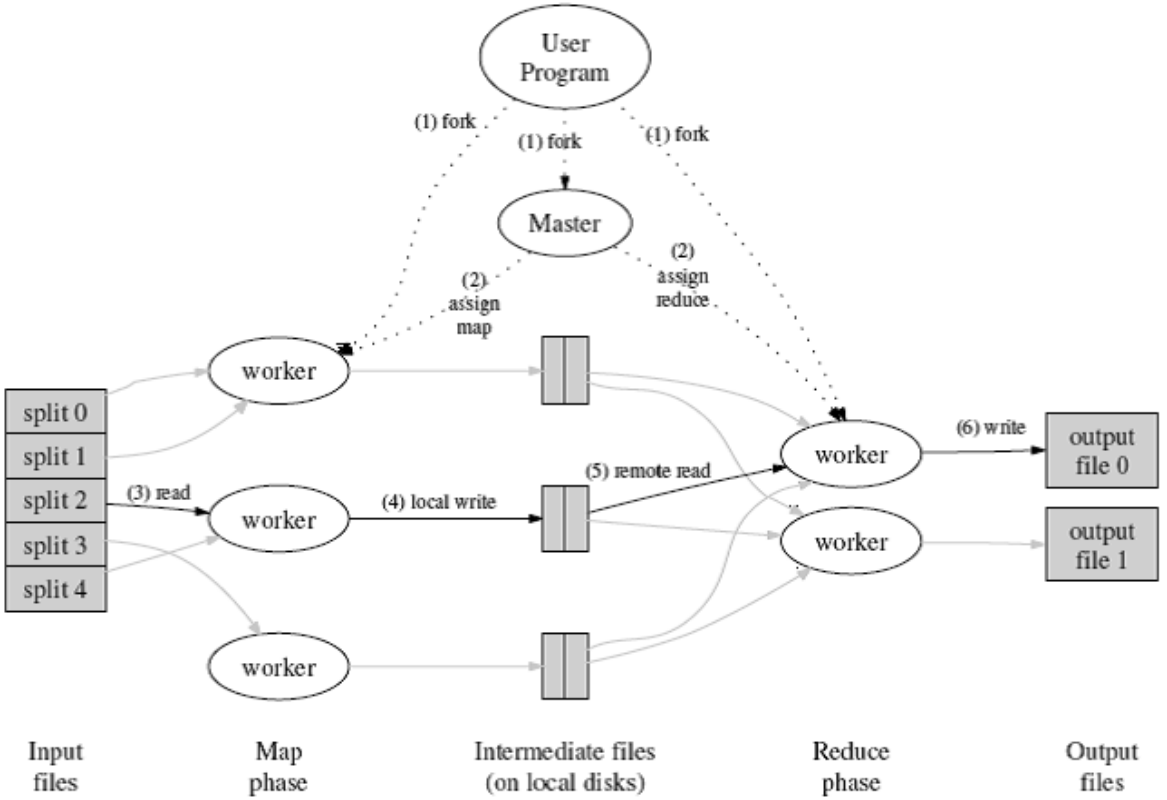
http://techon.nikkeibp.co.jp/article/NEWS/20091111/177527/?ref=rss

並列処理記述言語「*Axum*」

http://techon.nikkeibp.co.jp/article/NEWS/20091120/177846/

マイクロソフト社もCSPモデルを採用！！
残念ながらプロジェクトは2010年頃に中止されている。

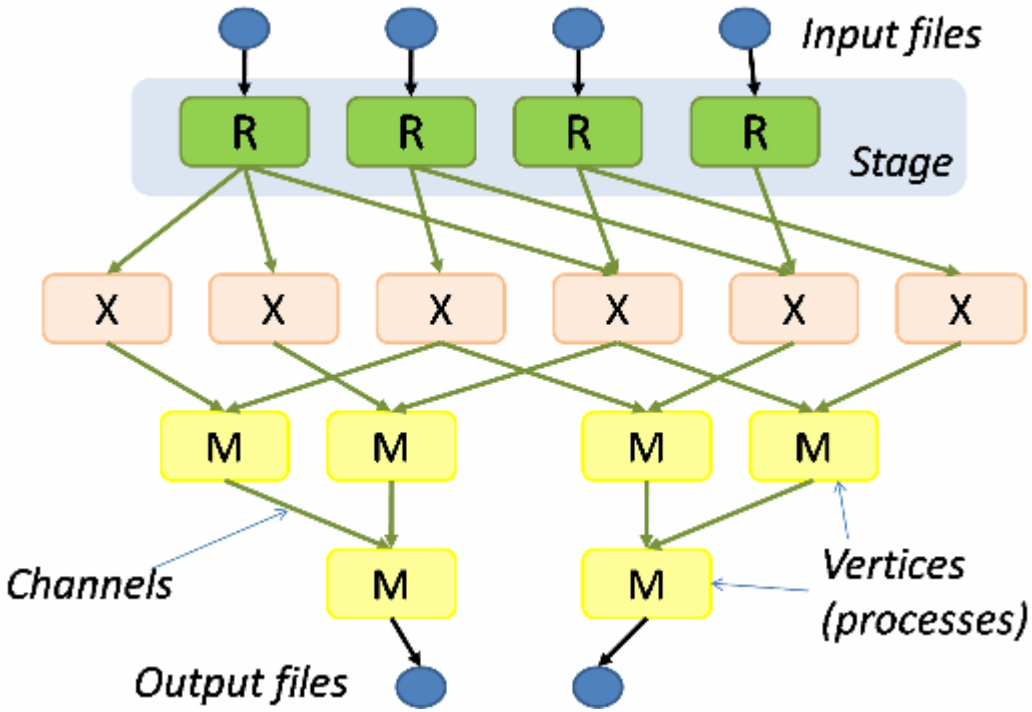
MapReduce



GoogleのCloud ComputingのMapReduceはCSP/occam
のFarmingの並列アルゴリズムに類似している。
しかしマルチスレッドのバグはApacheのサイトに報告されている。

Microsoft Research

Dryad



A Dryad programmer writes several sequential programs and connects them using one-way channels. The computation is structured as a directed graph: programs are graph vertices, while the channels are graph edges. A Dryad job is a graph generator which can synthesize any directed acyclic graph. These graphs can even change during execution, in response to important events in the computation.

プロセスとチャネルでMapReduceと似た事をしている。

http://research.microsoft.com/en-us/projects/dryad/

PyCSP

PythonにCSPモデルの開発はノルウェーのTomoso大学のJohn Markus Bjorndalenを中心に進められており、Googleもスポンサーになっている。

http://www.cs.uit.no/~johnm/code/PyCSP/

http://code.google.com/p/pycsp/ -- ここにpycsp-0.7.0.tar.gzファイルあり

The PyCSP project is an on-going project to bring
CSP (Communicating Sequential Processes) to Python.

UNIVERSITY OF COPENHAGEN

eScience Center

Communicating Processes in Python

The main goal of this project is to enable non-computer scientists to execute their scientific simulations efficiently on the next generation processors having 16+ cores. PyCSP and the associated graphical user interface CSPBuilder are an implementation of Hoares CSP algebra that dates back to 1978. By using CSP the programmer can prove a set of correctness criteria in their program that could not otherwise be shown. A CSP application is concurrrent in structure and can easily model data-flows from scientific applications, thus if non-computer scientists create their scientific simulations using PyCSP, they might be able to utilize multi-core systems.
Facts: The project has one PhD student at the eScience centre, Rune Friborg Møllegaard and is funded through a project that has a total of 3 PhD's and one Postdoc.

Contact: PhD Rune Møllegaard Friborg, runef@diku.dk

python-csp

http://code.google.com/p/python-csp/

```
>>> @process
... def writer(channel, n, _process=None):
...     for i in xrange(n):
...             channel.write(i)
...     channel.poison()
...     return
...
>>> @process
... def reader(channel, _process=None):
...     while True:
...             print channel.read()
...
>>> chan = Channel()
>>> Par(reader(chan), writer(chan,5)).start()
0
1
2
3
4
>>>
```

S. Mount, M. Hammoudeh, S. Wilson, R. Newman (2009) *CSP as a Domain-Specific Language Embedded in Python and Jython.* In Proceedings of Communicating Process Architectures 2009. Eindoven, Netherlands. 1St -- 4th November 2009. Published IOS Press.

http://code.google.com/p/python-csp/downloads/list -- ダウンロード

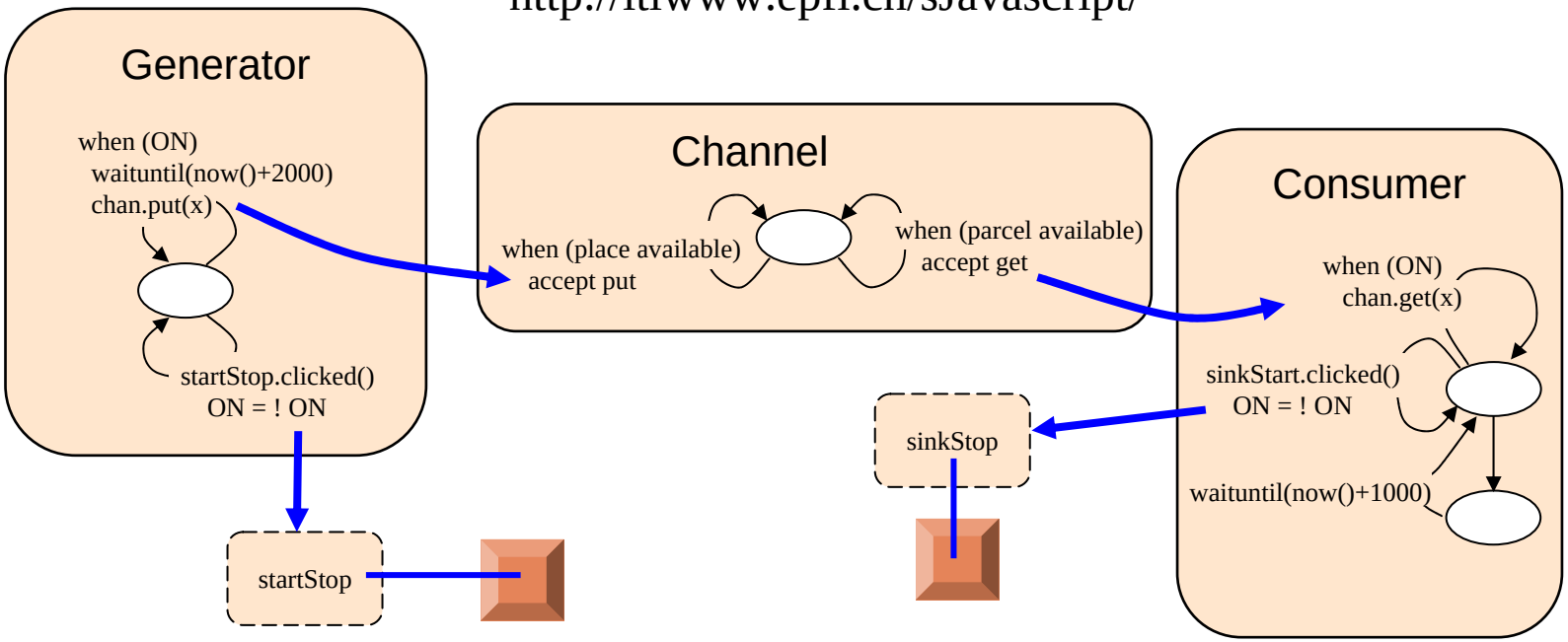
Synchronous Javascript

(= Javascript-CSP)

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

C. Petitpierre, EPFL, Switzerland

http://ltiwww.epfl.ch/sJavascript/



まだ開発の途中の様であるが、**sync channel, select**の機能が用意されている。

CHP

University of Kent

CHP: Communicating Haskell Processes http://www.cs.kent.ac.uk/projects/ofa/chp/

Grid Computing

Ian Foster

Associate Division Director
Senior Scientist
Head, Distributed Systems Lab
Mathematics & Computer Science
Argonne National Laboratory
9700 S. Cass Ave., MCS/221
Argonne, IL 60439
ph. +1-630-252-4619
fax +1-630-252-9556
foster@mcs.anl.gov

http://www-fp.mcs.anl.gov/~foster/

Professor of Computer Science
The University of Chicago
1100 E. 58th Street
Ryerson Hall
Room 155
Chicago, IL 60637
ph. +1-773-702-3487
fax +1-773-702-8487
foster@cs.uchicago.edu

Fortran M, CC++にはCSP/occamのチャンネルが採用されている　！！
1990年代後半にSun MicrosystemsがP2PをJavaに導入する。

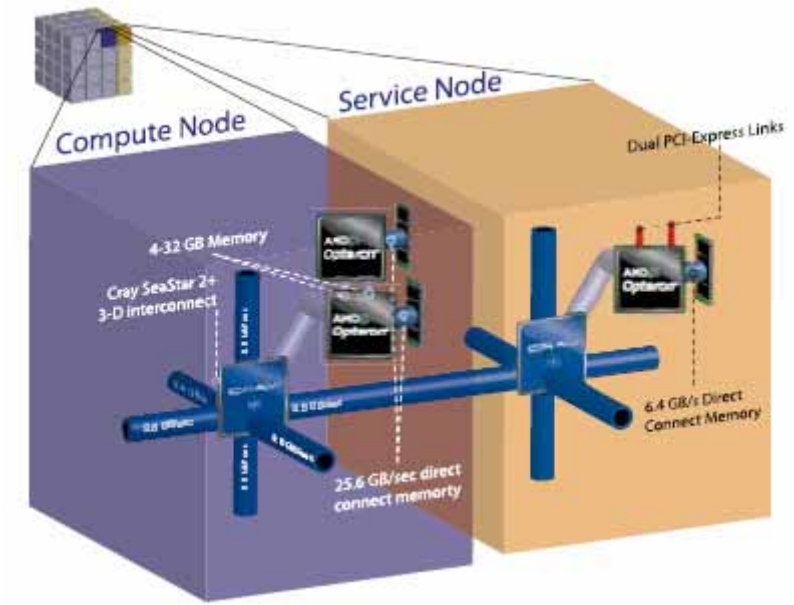
The basic abstractions used to describe parallel algorithms have been developed in the course of many years of research in operating systems, distributed computing, and parallel computation. The use of channels was first explored by Hoare in Communicating Sequential Processes (**CSP**) [154] and is fundamental to the **occam** programming language [231,280]. However, in CSP the task and channel structure is static, and both sender and receiver block until a communication has completed. This approach has proven too restrictive for general-purpose parallel programming. The task/channel model introduced in this chapter is described by Chandy and Foster [102], who also discuss the conditions under which the model can guarantee deterministic execution [51].

*MPI*の通信方式

- Point-to-Point通信
  - 同期(MPI_Ssend)
  - 非同期(MPI_Send, MPI_Recv)
  - バッファ付(MPI_Bsend)
- Collective通信
  - barrier(MPI_Barrier)
  - broadcast(MPI_Bcast)
  - scatter(MPI_Scatter)
  - gather(MPI_Gather)
  - gather-to-all (MPI_Allgather)
  - all-to-all(MPI_Alltoall)

APIはoccamと類似しているが、マルチスレッドの数は少ない。
複雑系の計算には、計算モデルが検証されているかは？

Cray XT5

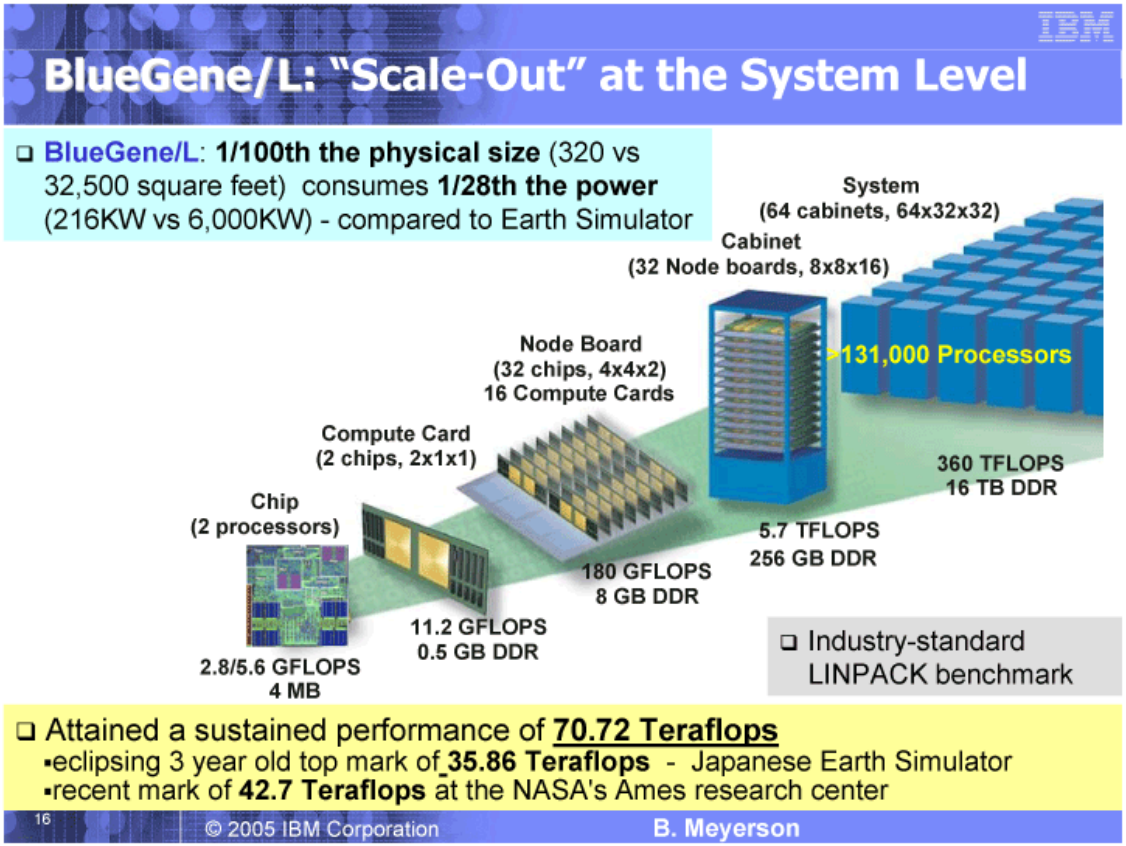


Designed for MPI message passing, the Cray SeaStar2+ chip combines communications processing and high-speed routing on a single device. Each communications chip is composed of aHyperTransport™ link, a Direct Memory Access (DMA) engine, a communications and management processor, a high-speed interconnect router and a service port.

http://www.cray.com/Assets/PDF/products/xt/CrayXT5Brochure.pdf

Transputerのアーキテクチャーと類似している。
ネットワークはHandel-CによるFPGAを使っている。

IBM BlueGene



http://ja.wikipedia.org/wiki/Blue_Gene

アーキテクチャーはTransputerの影響を強く受けている！！

組み込み設計フローを一変！！

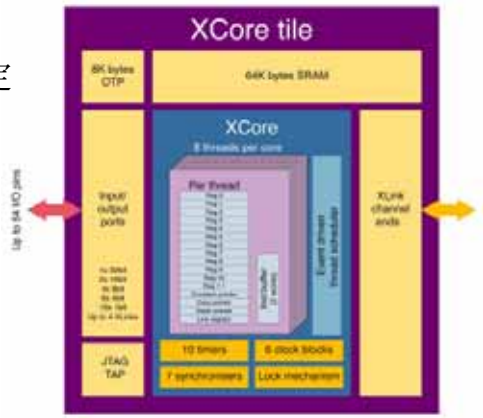
XMOS semiconductor

Transputer考案者、David May(Bristol大学教授) が設計した。

http://www.xmos.com/

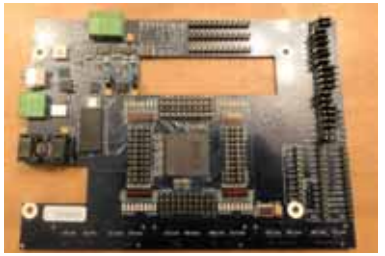
http://www.cs.bris.ac.uk/~dave/

2008年2Qに出荷予定



マルチコアと並列言語で実現

同社は今回の記者会見で、これまで公表していなかったXCoreの詳細についても説明した。これによれば、通常はハードウエアとして実装する処理もソフトウエアとして記述できるという特徴は、外部入出力端子(I/Oピン)の論理レベル遷移に対して、プロセッサLSI上で稼働する複数のスレッドがリアルタイムに応答する仕組みを用意することで実現した。具体的には、実時間管理に向けてマルチコア・プロセッサのハードウエア構成を工夫した上で、ソフトウエア処理の並列性や動作タイミングを記述できるようにC言語を拡張した独自のプログラミング言語「XC」を組み合わせた。以下で順番に説明する。



まずはXCoreの物理的なアーキテクチャである。32ビットRISCプロセッサとスレッド・スケジューラ回路、タイマー、クロック生成回路、同期/ロック機構をまとめたコアを用意し、このコアと、I/Oピンとのインターフェース回路や、RISCプロセッサで実行中の命令と処理対象データを置くSRAM、XC言語で開発したソフトウエアを格納するOTP (One Time Programmable) メモリー、コア間のインターコネクト回路(「XLink」と呼ぶ)で単位プロセッサ・コア(「タイル」と呼ぶ)を構成した上で、複数のタイルをアレイ状に接続した。クロック周波数は最大400M～500MHz。1個のタイル当たり400～500MIPSの処理性能が得られる。例えば4つのタイルを集積すれば、処理性能は最大1600～2000MIPSに達する。

http://www.eetimes.jp/contents/200712/29105_1_20071210205510.cfm

注目：**T9000、C104ルーティング・スイッチを意識したアーキテクチャー！！**

XMOS

Elektra 07 European Electronics Industry Awards WINNER

ACE EETIMES AWARDS

DATE 08

Panel: Thurs 13th Mar, 2:45pm
HDL or Embedded Design Methods

Embedded Systems CONFERENCE SILICON VALLEY

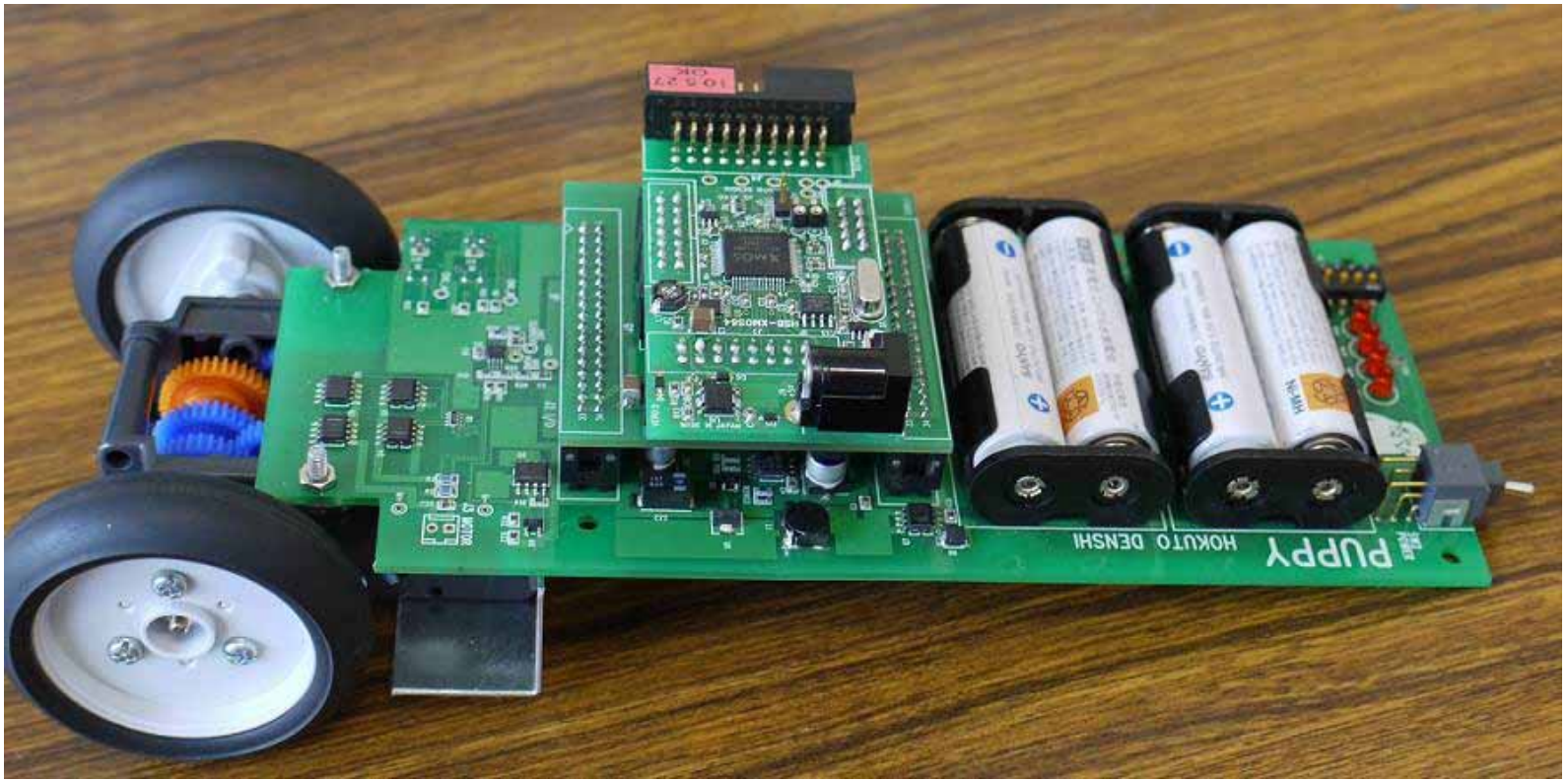
See us at booth 3034

コンパイル時間はXMOSが早い、コードサイズも小さい！！

Design Tradeoffs & Comparisons

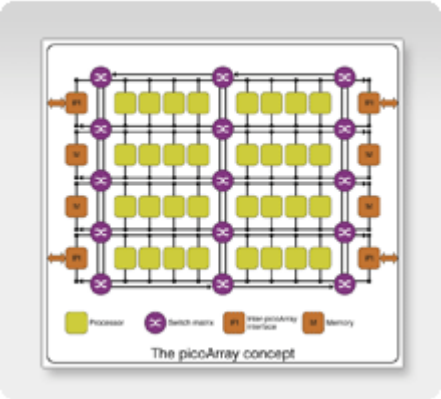
	XMOS SDS	FPGA	ASIC
Design Entry	*C/C++/XC (behavioural)*	*Verilog/VHDL (HW descriptions)*	
"Compile" Times	*Seconds*	*Hours*	Days
Design Changes	*Recompile single source*	*Repeat chip level P&R, timing closure and verification*	
IP Migration	*Rapid retarget*	*Often redesign to new rules & specs*	
"Program" Data	*~few Kbytes*	*~megabits*	0
Achieving necessary Performance	*Thread scheduler ensures guaranteed performance of function*	*Hand-fitting and low-level manipulation of the silicon and the design tools. Subsequent design changes can force repeating the hand-fitting process*	

NPO法人が学習キットを投入



http://techon.nikkeibp.co.jp/article/NEWS/20100727/184537/?SS=imgview&FD=1276070522

picoChip

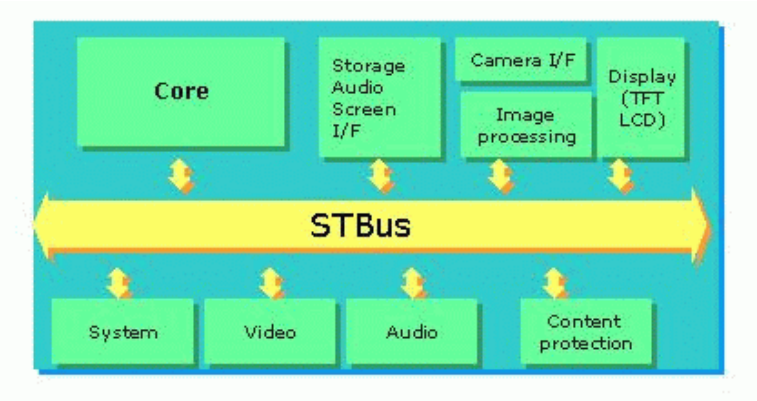


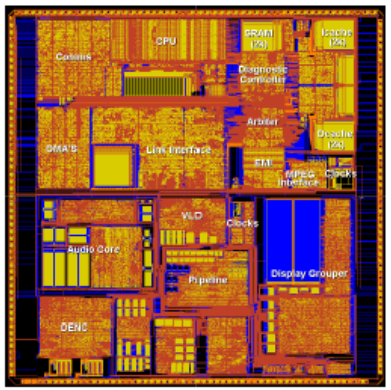
MIMO技術を採用し無線の超並列信号処理を実現したチップ。
WiMAXなどで実用化されている。マルチコアの数は256個！！
David MayがpicoChip社の技術顧問となっている。

http://www.picochip.com/

System-on-Chip

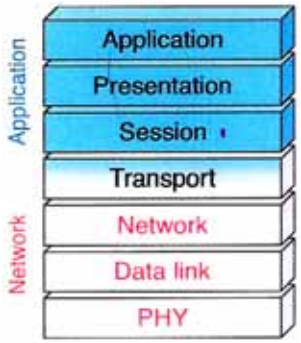
・ASIC/FPGAにおいて各種IPコアと相互接続をさせる手法。
・Busは入出力処理をするために、IPコアからのデータの流れの調停を行う。
・マスター・スレーブ方式でペリフェラルの速度に応じてグループ化する。
・CSPモデルはCrossbarスイッチでもって入出力の制御に適応された。
・DSCにおいて他社のPCIバス構成に比べて50倍の性能が得られた。
・他にAMBA Bus、IBM CoreConnect Bus等がある。
・**Handel-C**はVerilog/VHDLをimport/exportできるのでシリコンIP、SoCのBus設計に使える。





STBチップ

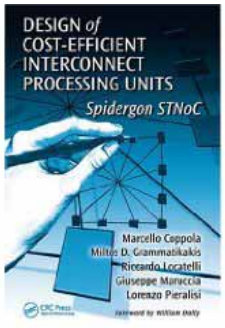
STBusはSTMicroelectronics社が開発したOn-Chip-Bus。



STNoC

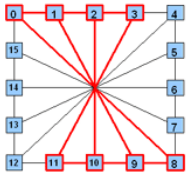
STMicroelectronics

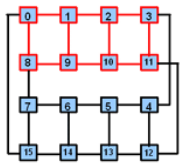
http://www.st-japan.co.jp/data/press/t1741t.html

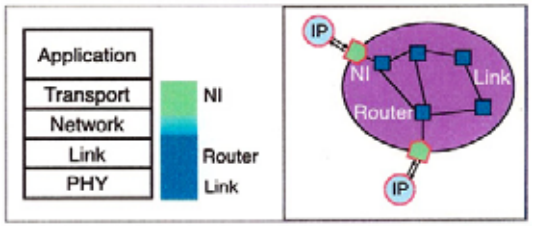


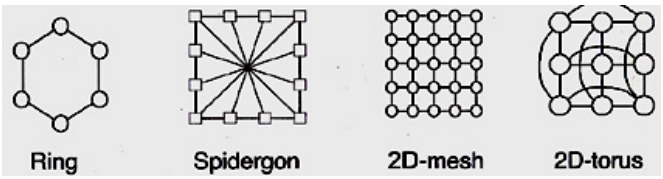
伝統的なオン・チップ・バス・アーキテクチャは、2つの理由からボトルネックになりつつあります。第1に、絶えず複雑化しているSoCデバイスに追随するため、バス・アーキテクチャも常に進化し続ける必要があります。したがって、各IPブロック内のバス・インタフェースを頻繁に変更しなければならず、新しいSoCソリューションのタイム・ツー・マーケットが増大することになります。第2に、トランジスタと同じ振る舞いでムーアの法則に従ってスケール・ダウンする代わりに、より多くのオンチップ機能を接続する必要が生じるため、相互接続は新しいテクノロジー世代ごとに複雑化します。その結果、シリコン面積、オンチップ通信速度、全体の電力消費といったコスト/パフォーマンス要素に対してバスが占める割合が次第に増加します。長期的には、すでにSTが世界をリードする研究開発の成果を発表しているチップ内光通信のような手法によって、この問題はなくなるかも知れません。中期的には、顧客が求める価格/性能/消費電力の改善の組み合わせを維持するために、新しいチップ内相互接続テクノロジーが必要になります。

業界の専門家の多くは、NoCテクノロジーがその解だと考えています。NoCテクノロジーの本質は、伝統的なサーキット・スイッチ型バスの代わりに、非常に単純化したネットワーク・パラダイムに似た階層型プロトコル・スタックを取り入れたパケット・ベースのパラダイムを使おうというものです。このシナリオでは、プロセッサ・コア、キャッシュ・メモリ、I/O機能、あるいはオーディオ/ビデオコーデックのような専用化されたIPブロックなどの実証済みIPが、単純にライブラリから取り出され、SoCの設計に追加されて、高速で低消費電力、シリコン面積の小さいパケット・ベースの通信プロトコルを使って、相互に通信を行います。



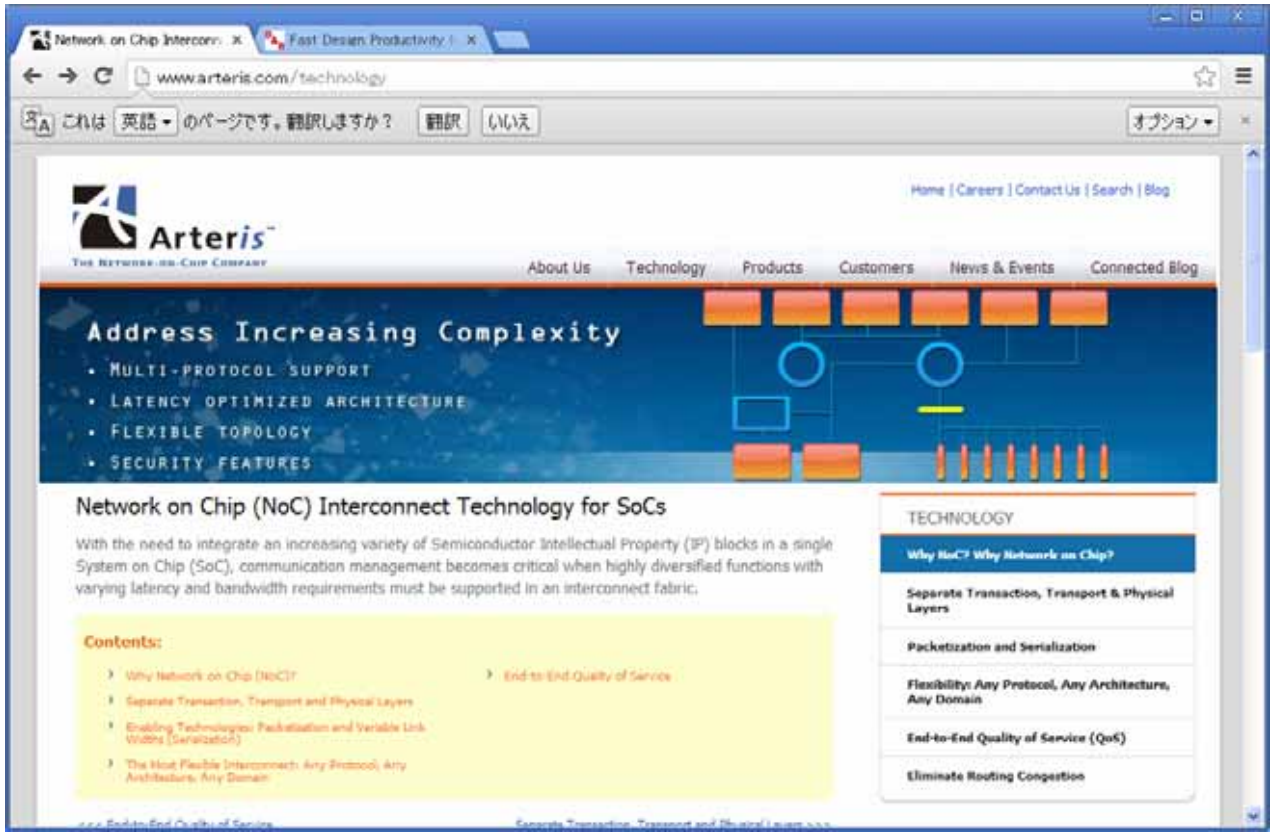






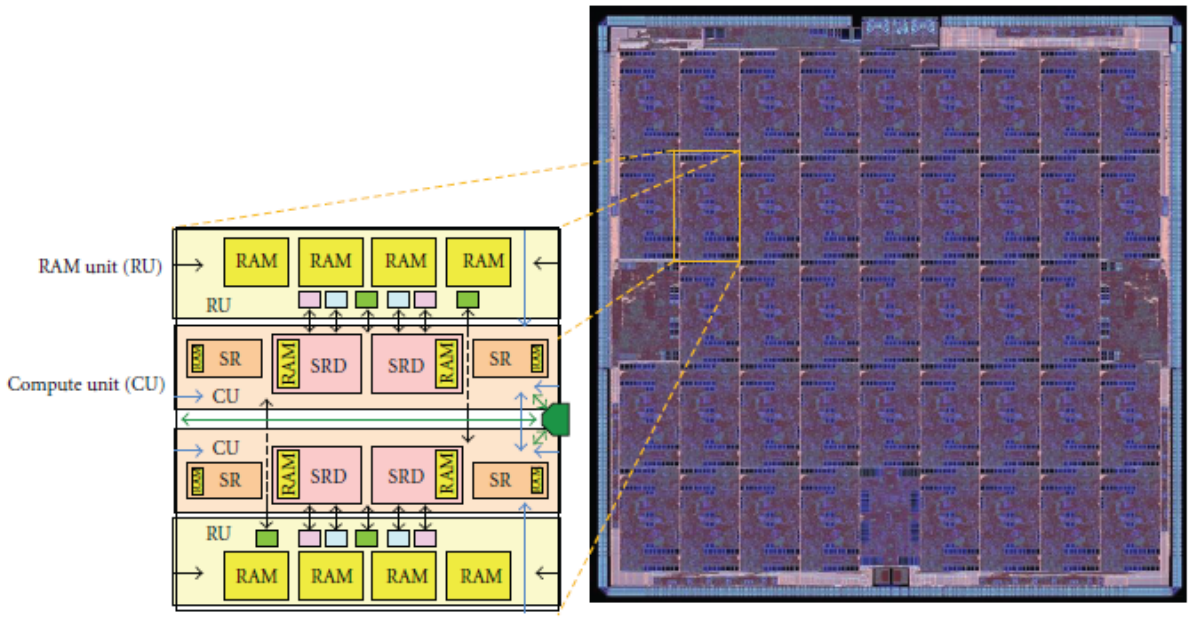
http://www.embedded.com/columns/technicalinsights/197005068?_requestid=270065

半導体メーカのSoCにNoC技術が使われる



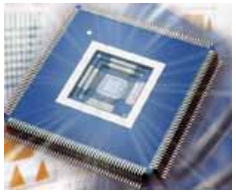
Arteris社のNoC技術は既に多くの半導体メーカーのSoCで採用されています。
IPコア間のパケット通信の構造はIEEE1355(T9000/C104)と類似しています。
C言語だけでは、アプリケーションは構築できません。チャネル通信と並行処理が必要です。Renesas, MegaChips, NTT Electronics, Toshibaなどの社名が発表されています。

MPPAチップにoccam-pi



スウェーデンのHalmstad大学ではAmbric社（Nethra Imaging社に吸収される）のMPPA(Massively Parallel Processor Arrays)に超並列処理言語occam-piをマッピングさせています。コアの数は360個です。

マルチコア

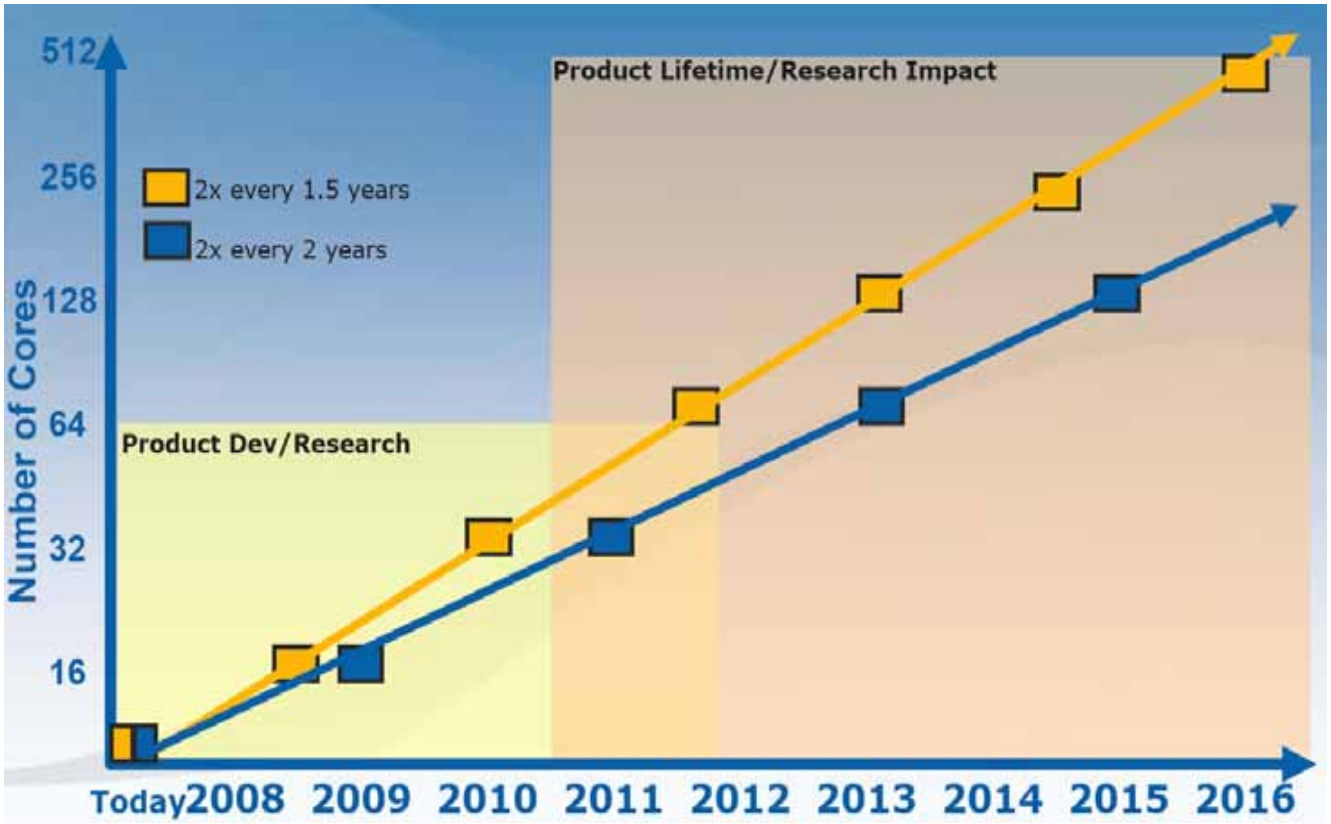


- SALSA (Service Aggregated Linked Sequential Activities)
  - http://www.infomall.org/multicore/index.php/Main_Page

We generalize the well known **CSP** (Communicating Sequential Processes) of Hoare to describe the low level approaches to area a) as “Linked Sequential Activities” in ***SALSA***. We use activities in ***SALSA*** to allow one to build services from either threads, processes (usual MPI choice) or even just other services. We choose linkage to denote the different ways of synchronizing the parallel activities that may involve shared memory rather than some form of messaging or communication

G. Foxが牽引している。ActivityはCSPのプロセス、Serviceはイベントに対応するように思われる。

Intel's Projection

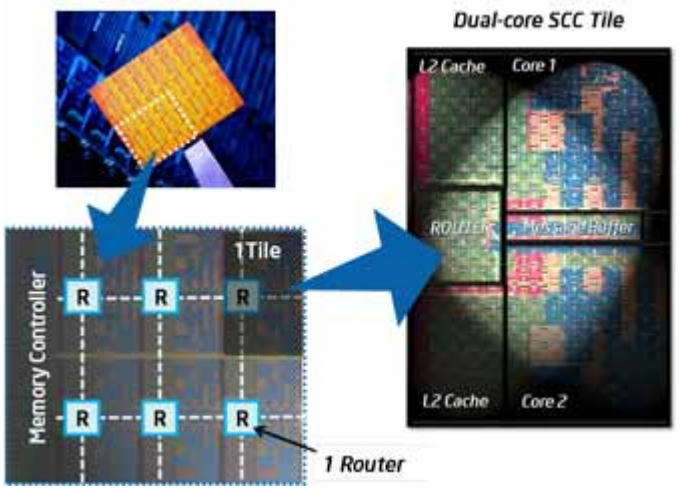


今後サーバー、Web系、組み込み系、全ての分野において超並列のプログラミング技術が必要とされる。

これに伴って、益々安全な並列処理が必要となる。

occam-pi/JCSP/C++CSPはマルチコアの対応を準備中。
CSP.NET(Java, C++, Delphi)は既に対応しており、現在Axon7から発売されている。

OpenComRTOS のSCCへの移植



SCCの詳細は以下のサイトを見てください。

Altreonic has signed an agreement with Intel Labs to port its multicore capable OpenComRTOS to the Single-chip Cloud Computer (SCC) experimental processor is a 48-core ‘concept vehicle’ created by Intel Labs as a platform for many-core software research. It incorporates technologies intended to scale multi-core processors to 100 cores and beyond, such as an on-chip network, advanced power management technologies and support for “message-passing.”

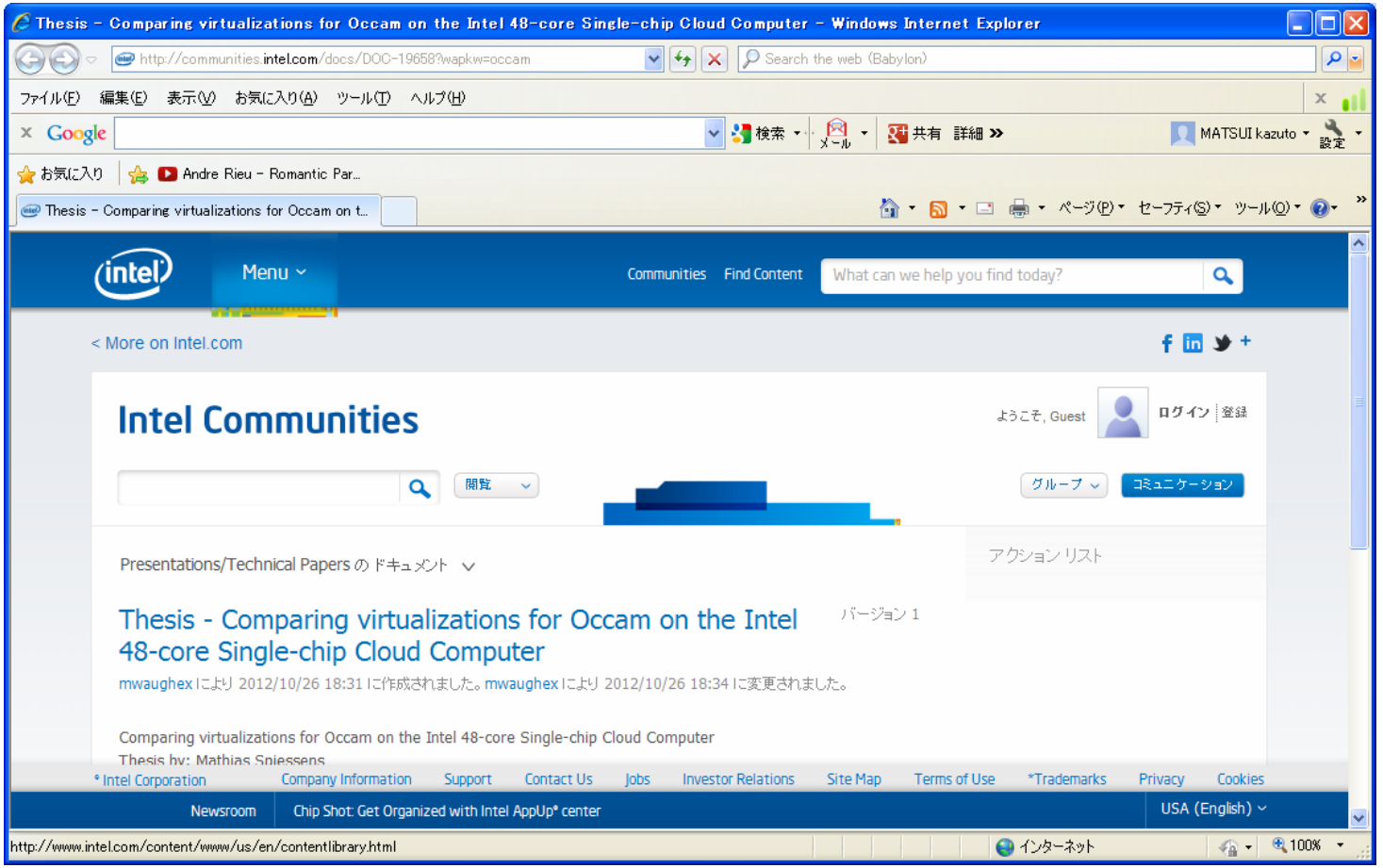
While the hardware scaling seems to have no limits thanks to Moore’s law (and Intel is well placed through its access to the most advanced and dense silicon technology), programming modern manycore and multicore systems is still perceived as a serious challenge and software productivity has been lagging far behind the progress made on the hardware side. Therefore Intel Labs has created the Many-core Applications Research Community (MARC) and invited academics and industry to join the effort making serious progress in this direction.

Programming parallel systems with many processor cores has been at the core of Altreonic’s technology for many years and hence it was natural for Altreonic to join in the effort as it already has proven technology.

http://www.altreonic.com/

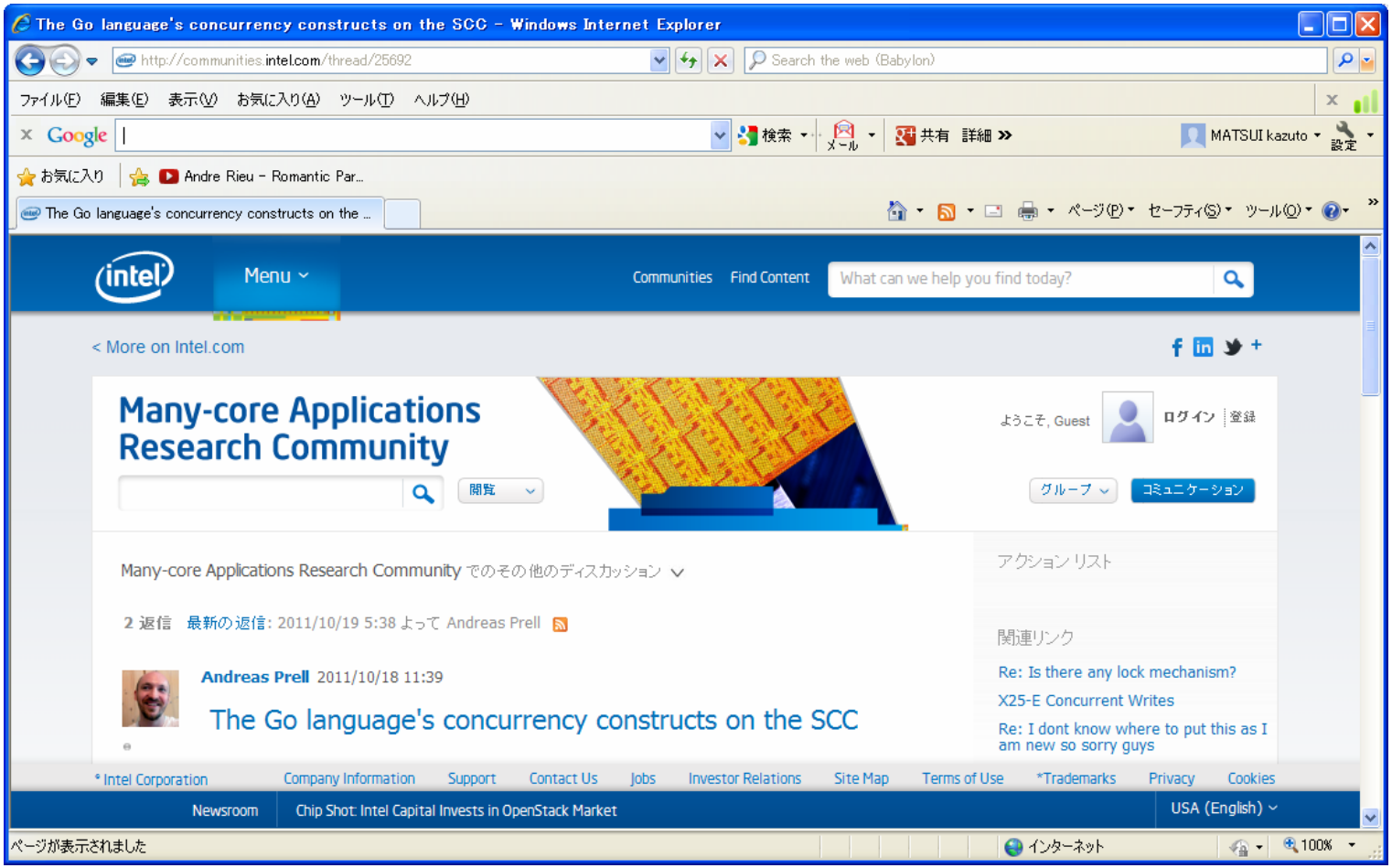
http://techresearch.intel.com/ProjectDetails.aspx?Id=1

Intel SCCチップへのoccamの移植

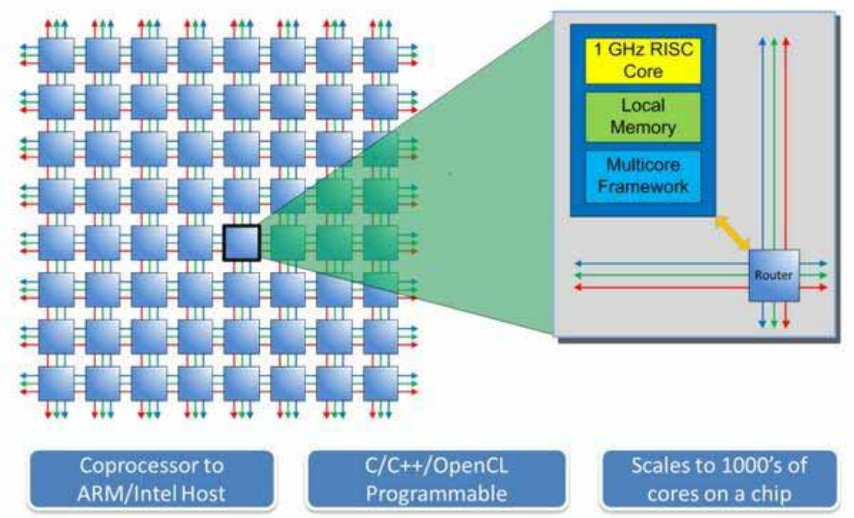


http://communities.intel.com/docs/DOC-19658?wapkw=occam

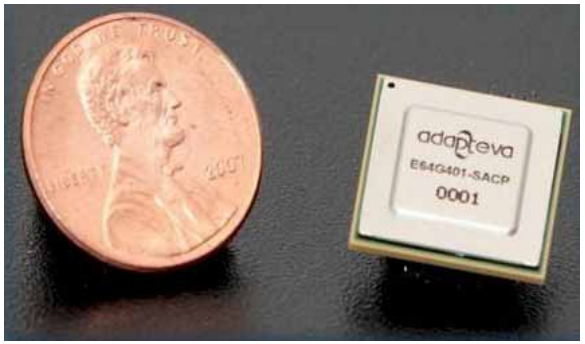
Intel SCCチップへのGo言語の移植

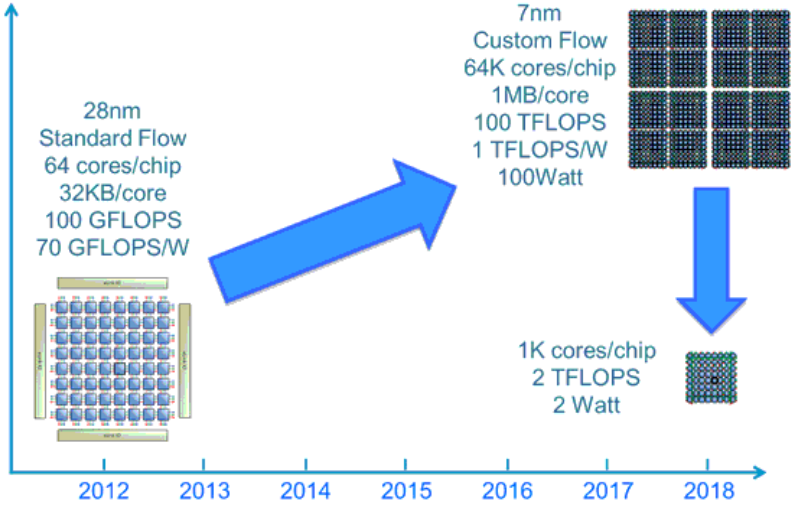


*Adapteva's Epiphany-*Ⅳ *chip*



T9000/C104の様な構造



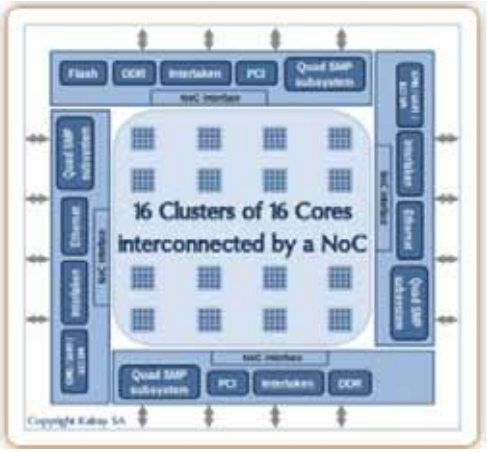


http://www.theregister.co.uk/2012/10/07/adapteva_epiphany_parallel_computing_kickstarter/

Adapteva $99 parallel processing boards targeted for summer

http://phys.org/news/2013-04-adapteva-parallel-boards-summer.html

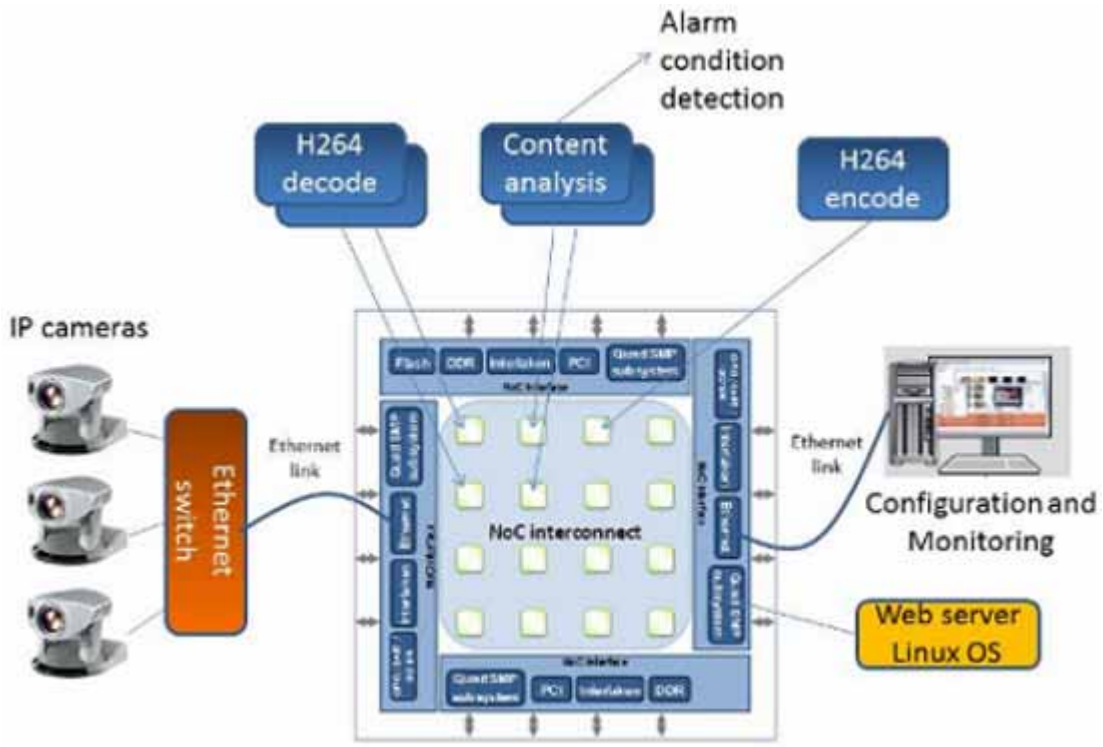
KALRAY社のMPPA



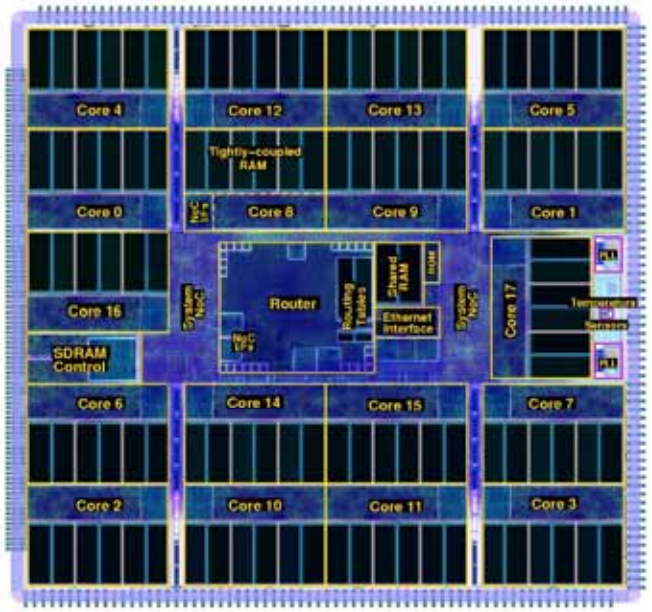
256個のCPUコアを内臓。NoCはTransputer/occamとよく似た構造である。
Sigma-Cはデータ・フローに特化したプログラミング言語で、Agentモデルである。
社長のJ. MonnierはSTMicroelectronicsのCentral R&DでVLIWのチップを開発していた。
Transputer/occamはよく理解している。

http://www.kalray.eu/products/mppa-manycore/mppa-256/

Part Number	Number OF PROCESSORS	GFLOPS (32-BIT) @400MHz	TOP @400MHz	Internal Memory Size (MBytes)
MPPA-256	256	230	0,7	32+2
MPPA-512	512	460	1,4	64+8
MPPA-1024	1024	920	2,8	128+8

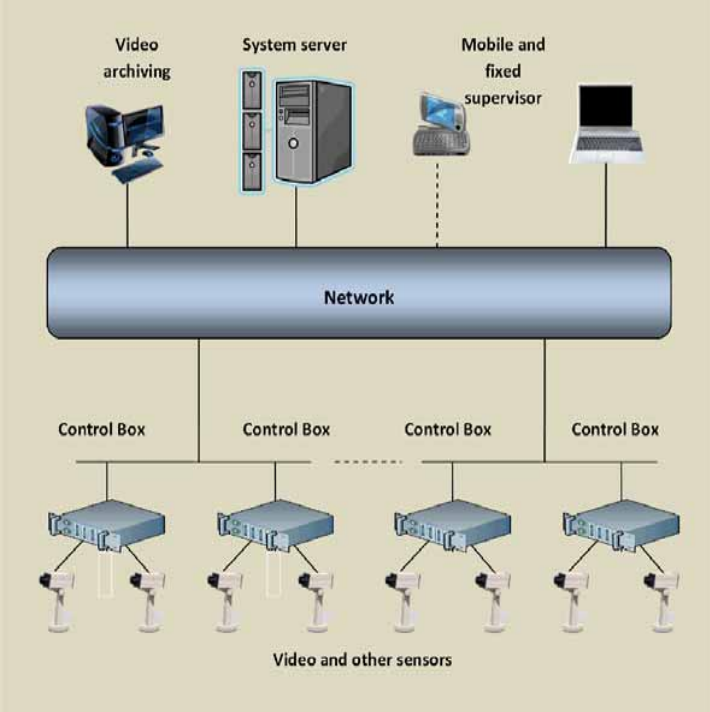


SpiNNakerチップ



SpiNNaker chipは人工知能に特化したNoC(Network-on-Chip)で、ニューラルネットを基本とした超並列コンピューティングです。コアにはARM9のMPUが採用されています。ワンチップに18個のコアが搭載されています。ノード数が４つのボードで72個のARMプロセッサが搭載されています。将来的にはこのモジュールが他のモジュールと接続され、筐体の中に収められる予定で、最大で何と**1,036,800個**のプロセッサコアが搭載されるようです。これだけ大規模なシステムのボード間、筐体間の通信は同期通信だと困難です。同一のクロック領域だけに同期通信を採用しています。異なったクロック領域はチャネルで接続されます。ここでは**GALS** (Globally Asynchronous Locally Synchronous)の通信方式を採用しています。GALSは一般的にCSPモデルの拡張として知られています。

SystemJ言語

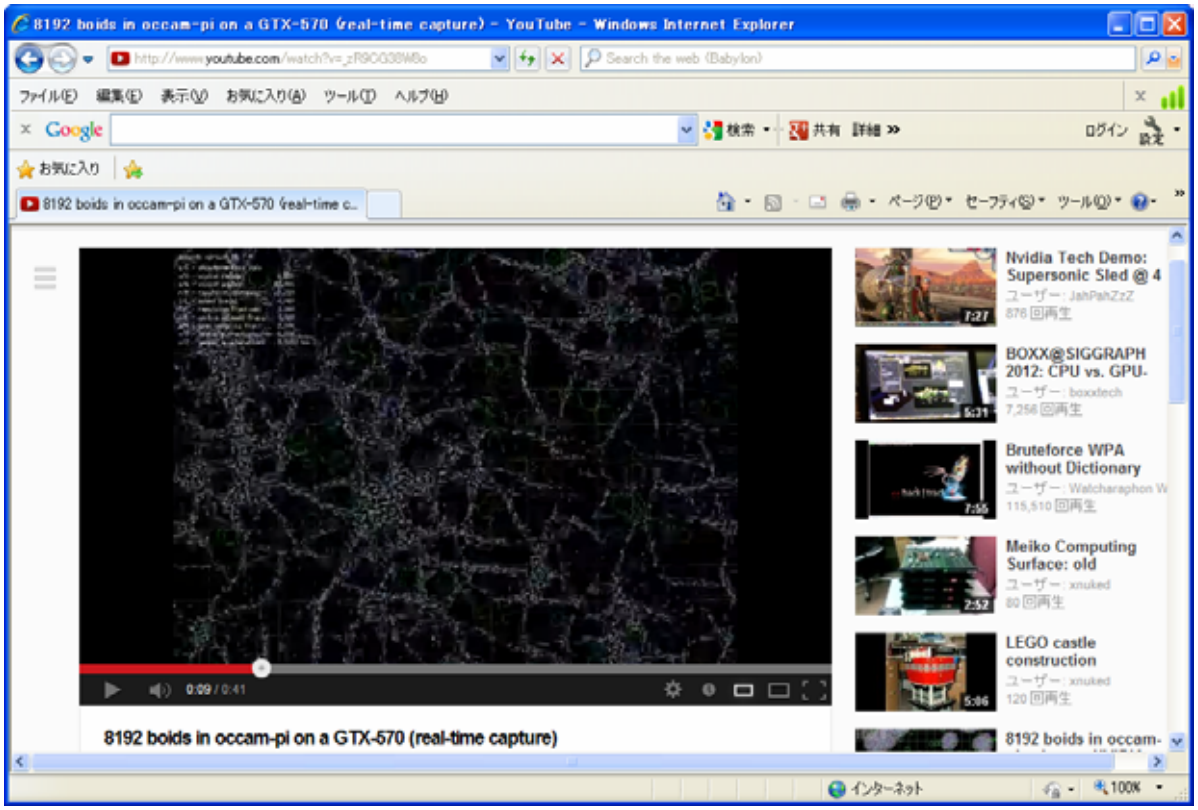


SystemJはGALS(Globaly Asynchronous Locally Synchronous)をサポートするプログラミング言語です。
EsterelとCSPモデルが融合したプログラミング言語です。SystemJはニュージーランドの Auckland大学で開発されています。

http://systemjtechnology.com/

occam-piのCUDAへの移植

occam-piがGPU上で走ります！！高度な並行処理が使えます。
OpenCLは使っていません。



http://www.youtube.com/watch?v=_zR9CG38W8o

セキュリティ・プロトコル

Modelling and Analysis of Security Protocols

Peter Ryan, Steven Schneider, M. H. Goldsmith, G. Lowe and A. W. Roscoe

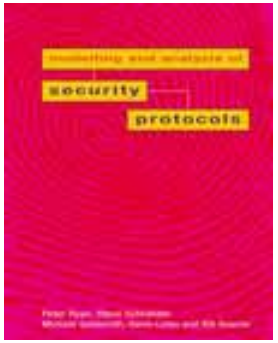
Addison-Wesley 2001

ISBN 0-201-67471-8. 300 pages. Index. Bibliography. Three appendices. $35.00

セキュリティ・プロトコルの検証は数多く見られる。日本でもこの本をお手本にしている。

Reviewed by Robert Bruen August 1, 2003

As any field matures, theory and mathematics appear. Security is no exception. Areas such as cryptology have been mathematical for a long time, in fact, it is quite near impossible to any serious work in cryptology without good math knowledge. Other areas in security are in various stages of catch up mode. Protocols are still a bit to low level for many security professionals, but they are the basic building blocks of security. The folks who work with protocols every day understand them in a way that the rest of us do not. The demand for protocol improvements has been increasing for a while, however, there are not a lot of books explaining protocols and even fewer that try to model them. Ryan and Schneider have helped to fill that gap with this book.



The authors use CSP, Communicating Sequential Processes, "...a mathematical framework for the description and analysis of systems consisting of components (processes) interacting via the exchange of as interprocess messages." It is process algebra which operating systems folks will recognize as interprocess communications or task-to-task communication, not a new concept [1, 2], but a somewhat different use. The principle of correct message delivery is at the heart of it.

They also use use a commercial model checker FDR from Formal Systems and a compiler called Casper, written by the authors. All three of these tools are available on the Web. Their process is to describe a protocol in a script written in an abstract notation for Casper. Casper translates the script into code that CSP can read and then can be checked by FDR. The Casper script has several sections that model the protocol, such its description, the variables, the processes, specifications, functions systems and the intruder. Each section is written in a formal manner with a specific syntax. Naturally, you must have thought it through beforehand. Throughout the book, the Yahalom protocol is used as the object of analysis. Using the same protocol helps to provide a little more clarity, unlike using a different protocol for different examples. Yahalom is well known enough to make it a good choice to tie everything together. There is a good chapter on writing the Casper code with enough detail to allow the reader to learn it such that it can be useful. A completed scripted is contained in an appendix, along with the output.

The authors cover fundamental principles of security, like authentication, non-repudiation, integrity, and so forth with bridges to the modeling aspects. The explanations are understandable, which is not always the case with mathematical works. The book adds to the security field by making the deeper levels of protocol modeling and analysis more accessible.

[1]Communicating Sequential Processes, C.A.R. Hoare. Communications of the ACM 21(8):666-677, August 1978.
[2]Communicating Sequential Processes, C.A.R. Hoare. Prentice Hall International Series in Computer Science, 1985.

http://www.ieee-security.org/Cipher/BookReviews/2003/Ryan_by_bruen.html

プロトコルの検証

- オープンソースXen Hypervisorではセキュリティ確保の保証にCSPモデルを採用(http://www.xen.org/)。
- Xenはクラウド･コンピューティングに使用されている様です(http://www.xen.org/products/cloudxen.html)。
- CSPモデルによる開発はNaval Researchで開発されたものと関係があるようである。
- XMLのアクセス制御モデル(Korea Information Security Agency)
- その他

*BPM,SOA*はCSPモデルに適する

図解･上級SEのためのビジネスモデリング
テクニック―機能ユニットモデルのBPM方
法論
芳賀正彦著
日刊工業新聞社

詳説 ビジネスプロセスモデリング
オライリー･ジャパン

SOAアーキテクチャーは
イベント駆動モデルである。

*IBM*社内で*CSP*モデルを採用した研究開発例

1. K. Chandy, J. Misra and Laura Haas. "**Distributed Deadlock Detection**", ACM Transactions on Computer Systems, Vol. 1, No. 2, May 1983, Pages 144-156.

2. J. C. P. Woodcock. "**Transaction processing primitives and CSP**". *The IBM Journal of Research & Development* 31(5):535–545 1987.

3. Matthias Kaiserswerth. "**The Parallel Protocol Engine**", IEEE/ACM TRANSACTIONS ON NETWORKING, Vol.1, NO. 6, DECEMBER 1993.
(IBM Researeh Division, Zurich Research Laboratory)

Transputer**採用**

4. D. G. Shea. "**The IBM Victor V256 partitionable multiprocessor**",
(IBM J. RES. DEVELOP. VOL. 35 NO. *5/6* SEFTEMBERINOVEMBER 1991)

5. Jochen M. K¨uster and Shane Sendall and Michael Wahler.
"**Comparing Two Model Transformation Approaches**", -- UML**から**CSP**変換への検討**
(IBM Zurich Research Laboratory)

6. "**Specification and Verification of Selected Intrusion Tolerance Properties Using CSP and FDR**"
Security protocol in MAFTIA middleware.(Project IST-1999-11583)

7. "**Security in Business Process Engineering**"
(IBM Zurich Research Laboratory, R¨uschlikon, Switzerland)

High-Integrity Component-Based Engineering for Enterprise Systems-(1)

更に最近ではSouthampton大学と共同でCSPモデルを導入したe-Businessの開発に力を注いでいる

Background

The rapidly expanding and changing world of Enterprise Computing requires that developers be able to respond to customer needs as quickly and effectively as possible. To be cost effective, these services need to be implemented using as many existing components as possible rather than being constructed from scratch. In addition, the business-critical and security-critical nature of Enterprise Computing requires the systems providing the services to have very high integrity levels.

Typically, the development of a new service involves the construction of business process models which describe the service at a high level of abstraction and which can be agreed upon with the customer. These models are then translated into more detailed architectural designs which can be deployed onto existing enterprise systems such as Enterprise Java Beans (EJB).

Unfortunately there is still a large deployment gap between the high-level business process models and the target enterprise systems. Although there is considerable on-going work in developing tools and techniques to support this translation and mapping, the existing techniques are incomplete and not fully effective. This proposal aims to help bridge this gap by combining the systems development expertise of **IBM UK Laboratories** with the advanced software engineering methods expertise of DSSE.

High-Integrity Component-Based Engineering for Enterprise Systems-(2)

Programme

This proposal seeks funding for an initial investigation of how existing industrial strength development techniquesmay be enhanced to meet the demands of High Integrity Enterprise Computing. One of the deliverables of the initial investigation will be a more comprehensive programme of collaborative research in this area.Currently, IBM UK Laboratories make use of XML for specifying architectural models for implementing Enterprise Computing. While XML is very effective at describing structural aspects of component relationships, it is weak at describing the behavioural aspects of components. **It is critical to be able to reason about the eventual behaviour of an enterprise system from the behaviours of the components used to build it.**

For the initial investigation we propose to look at how the XML design technique may be enhanced with ideas from behavioural specification languages such as B, CSP and the UML Object Constraint Language (OCL). These languages provide rich notations for describing service and component behaviour at appropriate levels of abstraction and provide refinement techniques for ensuring consistency between models at different levels. The languages have the added advantage of a formal underpinning which supports the attainment of high integrity levels.

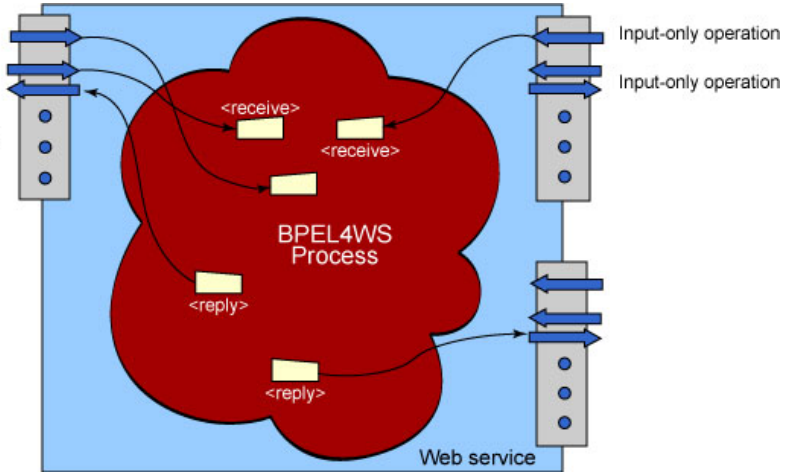
IBM UK Laboratories will provide DSSE with access to their XML tool, as well as some specifications of services and EJB-based architectural components. They will also provide extensive on-going feedback on the investigation.

Business Process with BPEL4WS

The BPEL4WS process itself is basically a flow-chart like expression of an algorithm. Each step in the process is called an activity. There are a collection of primitive activities: invoking an operation on some Web service (<invoke>), waiting for a message to operation of the service's interface to be invoked by someone externally (<receive>), generating the response of an input/output operation (<reply>), waiting for some time (<wait>), copying data from one place to another (<assign>), indicating that something went wrong (<throw>), terminating the entire service instance (<terminate>), or doing nothing (<empty>).

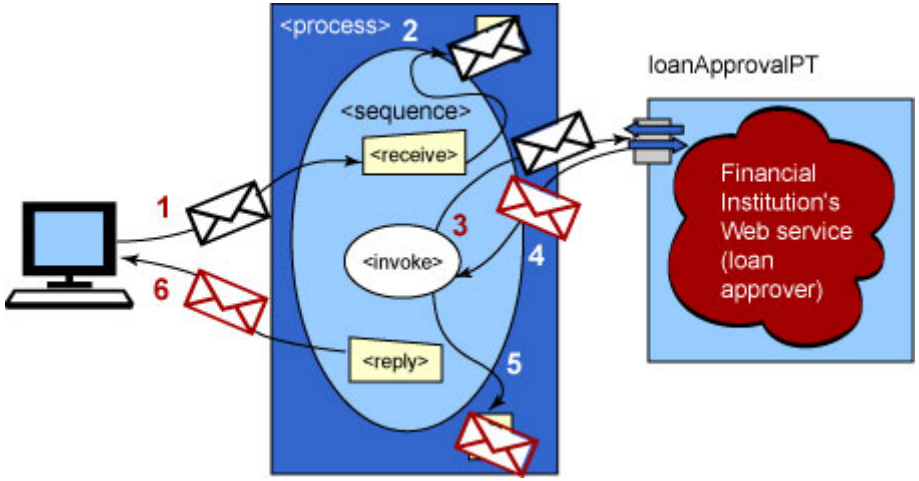
```
process ::= service | scope |
interaction | receive
| reply |
asynchronous/synchronous  invole |
sequence | flow | while | pick
```

These primitive activities can combined into more complex algorithms using any of the structure activities provided in the language. These are the ability to define an ordered sequence of steps (**<sequence>**), the ability to have branching using the now common "case-statement" approach (**<switch>**), the ability to define a loop (**<while>**), the ability to execute one of several alternative paths (**<pick>**), and finally the ability to indicate that a collection of steps should be executed in parallel (**<flow>**). Within activities executing in parallel, one can indicate execution order constraints by using the *links*. BPEL4WS allows you to recursively combine the structured activities to express arbitrarily complex algorithms that represent the implementation of the service.



プロセスの生成

```
<process name="loanApprovalProcess"
targetNamespace="http://acme.com/simpleloanprocessi
ng"
mlns="http://schemas.xmlsoap.org/ws/2002/07/busines
s-process/"
xmlns:lns="http://loans.org/wsdl/loan-approval"
xmlns:loandef="http://tempuri.org/services/loandefi
nitions"
xmlns:apns="http://tempuri.org/services/loanapprove
r">
```



http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol2/

```
<sequence>
<receive name="receive1"
    partner="customer"
    portType="apns:loanApprovalPT"
    operation="approve"
    container="request"
    createInstance="yes">
</receive>
```

```
<invoke name="invokeapprover"
        partner="approver"
portType="apns:loanApprovalPT"
        operation="approve"
        inputContainer="request"
outputContainer="approvalInfo">
</invoke>
```

```
<reply name="reply"
    partner="customer"
    portType="apns:loanApprovalPT"
    operation="approve"
    container="approvalInfo">
</reply>
</sequence>
</process>
```

High-Integrity Component-Based Engineering for Enterprise Systems-(3)

DSSE will develop ways of combining behavioural specification techniques with XML and of relating the following:

- the abstract business process models describing business services;
- the specifications of the architectural components;the specifications of the operational characteristics of these business services (their expected qualities of service);
- and the specifications of the target IT systems upon which the business services are to be deployed.

DSSE will also undertake a number of case studies to validate their proposals.
The initial investigation will deliver:

- Proposals for enhancements to XML and the XML tool including support for consistent checking between the three aspects of the specifications outlined above.
- Techniques for relating high-level business process models with architectural components along with supporting case studies.
- A plan for a more comprehensive programme of collaborative research.

DSSE Group

The DSSE Group, led by Prof Peter Henderson, is a research group of the Department of Electronics and Computer Science at the University of Southampton. Members of the group have considerable expertise in advanced software engineering approaches such as Business Process Modelling and Formal Methods. The Group has been involved in the application of these methods and in the development of tools to support their use. The Group attracts UK Government and European funding and has several on-going industrial collaborations in the areas of Business Process Modelling and Formal Methods.

FPGA-based networking systems for high data-rate and reliable in-vehicle communications

Publisher EDA Consortium San Jose, CA, USA

ABSTRACT

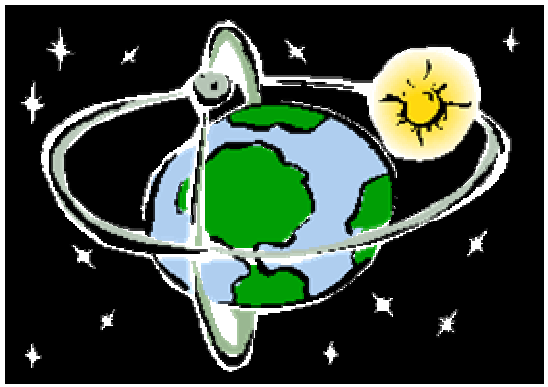
The amount of electronic systems introduced in vehicles is continuously increasing: X-by-wire, complex electronic control systems and above all future applications such as automotive vision and safety warnings require in-car reliable communication backbones with the capability to handle large amount of data at high speeds. To cope with this issue and driven by the experience of aerospace systems, the Space Wire standard, recently proposed by the European Space Agency (ESA), can be introduced in the automotive field. The SpaceWire is a serial data link standard which provides safety and redundancy and guarantees to handle data-rates up to hundreds of Mbps. This paper presents the design of configurable SpaceWire router and interface hardware macrocells, the first in state of the art compliant with the newest standard extensions, Protocol Identification (PID) and Remote Memory Access Protocol (RMAP). The macrocells have been integrated and tested on antifuse technology in the framework of an ESA project. The achieved performances of a router with 8 links, 130 Mbps data-rate, 1.5 W power cost, meet the requirements of future automotive electronic systems. The proposed networking solution simplifies the connectivity, reducing also the relevant volume and mass budgets, provides network safety and redundancy and guarantees to handle very high bandwidth data flows not covered by current standards as CAN or FlexRay.

http://portal.acm.org:80/citation.cfm?id=1266470

PtolemyIIプロジェクト

UCバークレー校Edward Lee教授はPtolemyIIプロジェクトを推進。
自動車、航空機メーカがスポンサーとなっている。

http://ptolemy.eecs.berkeley.edu/~eal/
http://ptolemy.eecs.berkeley.edu/ptolemyII/index.htm



CSPモデルを採用

http://ptolemy.eecs.berkeley.edu/papers/99/HMAD/html/csp.html

理由は、マルチスレッドの問題点を回避するため。

http://ptolemy.eecs.berkeley.edu/publications/papers/06/problemwithThreads/

AVACS

http://www.avacs.org/

AVACS

Automatic Verification and Analysis of Complex Systems

AVACSは自動車のコンソーシアムにも大きな影響を与えている。OZ/**CSP**/DCの手法を採用しMOBYと呼ばれる自動検証ツールを開発している。無線による列車自動制御、自動車の自動巡航制御(car platoon)などがある。

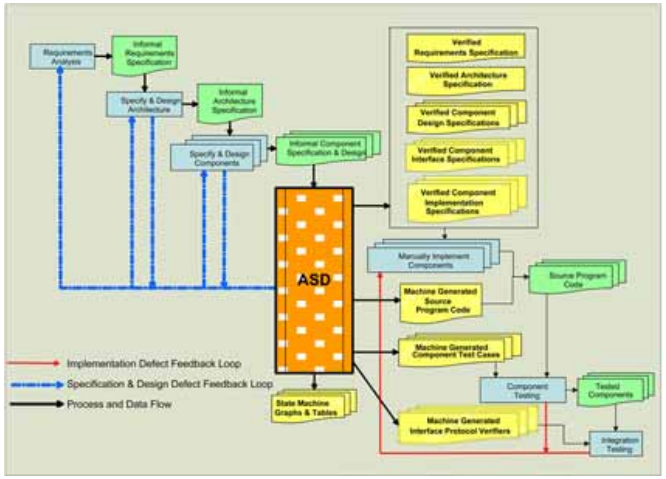
Verum

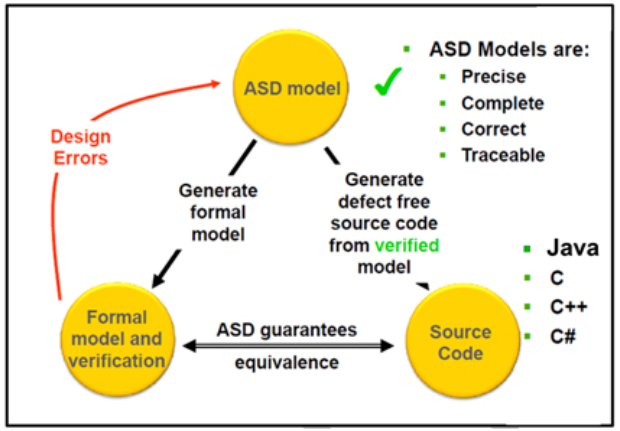
verum®
Tools for building mathematically verified software

http://www.verum.com/

ASD

Verum Software Technologies BV (Waalre, Netherlands) has received a European patent for its Analytical Software Design (ASD) technologies. The patent (# EP1749264) recognizes a method of applying the formal methods technique called communicating sequential processes (CSP) algebras to develop defect free software architectures and software codes which are event driven and exhibit lots of concurrency.





http://www.electronics-eetimes.com/en/verum-gains-patent-for-formal-methods-based-tool.html?cmp_id=7&news_id=222901332&vID=220

CSPモデル検証の後にソースコード(Java/C/C++/C#)を生成する。

http://www.healthcare.philips.com/main/

http://www.apptech.philips.com/

http://www.press.ce.philips.com/

http://www.ccm.nl/

http://www.nanda-tech.com/

http://www.asm.com/

http://www.fei.com/

http://www.nxp.com/

http://ajar.ukmn.com/

http://www.sioux.nl/

http://www.verum.com/markets/customers.html

Computing without Computers

Handel-C

- 1990年頃にスタートしたOxford大学のComputing Labの Hardware Compilation Research Groupが中心としたプロジェクトが基礎となっている。

Harp1 

Harp2 

- Handel-CはHW/SWの協調設計(Co-Design)を目指す。
- CSPモデルによる設計の正確さが証明されている。
- Handel-Cは「Compiling occam to FPGA」とも表現される。
- seq, par, alt, pri, timer、チャネルなどのコンストラクターはC言語の拡張として扱われる。
- Verilog, VHDLのようなHDLの制約は無く、Cのアプリケーションをハードウェア化することができる。

Handel-C

Handel-CはOxford大学のIan Pageによって開発され、C言語の様な仕様をもっています。セマンティックスはCSP/occamのモデルであります。occamのコンストラクタ(SEQ,PAR,ALT,WHILE,?,!, etc)はでXilinxのFPGAのセルマッピングされます。応用分野は幅広い用途で採用されています。

自動車

生体認証

通信

コンシューマ・
エレクトロニクス

防衛・航空宇宙

エンタープライズ及びHPC

画像処理

工業製品

セキュリティ

http://www.celoxica.co.jp/corporate/endmarkets/default.asp

欧米での適用事例まとめ

- 設計手法の比較
  - ROI上での優位性比較　－　米国防衛産業　航空機メーカー　(従来手法に対して1/8のコストと1/5の期間短縮)
  - 設計生産性上での比較　－　欧州通信系顧客　(SS7プロトコール：VHDL.vs.Handel-C　½の開発期間、性能優位) , 欧州モバイル系顧客　(IP V6プロトコール: Handel-C：1/6の開発期間、同一性能 .vs. Verilog .vs. VHDL), 欧州防衛系顧客 (画像処理：Handel-C：設計の柔軟性を大幅改善 .vs. VHDL)、米国半導体系顧客 (JPEG2000: Handel-C：開発期間1/3、性能優位 .vs. VHDL (JPEG2000エキスパートによる作業)、米国航空産業系顧客 (信号処理：Processor .vs. FPAG ：CPUに対して23倍の高性能を実現、ソフトウエア技術者がハード設計を実行)
  - アルゴリズム高性能化における比較　(Processor .vs. Hardware)　(欧州、米国など)　表添付
  - City of London (モンテカルロアルゴリズム：FPGA. .vs. GPU .vs. Cell .vs. 高性能プロセッサ間の性能評価　)

- 欧州事例紹介
  - フィリップス　HDTV　各プラットフォーム上での色のにじみ差異を自動補正　－　フィルターシミュレーションが48時間からリアルタイム処理で可能になった。評価期間が12ヶ月から3ヶ月に短縮。ソフトウエア技術者がハードを設計。
  - BAEシステム　センサーフュージョン　－　画像、レーダー、ジャイロスコープの入力情報のパラレルリアルタイム処理 (航空機システムの制御コンセプトを自動車に適用)

既に300以上の適用事例がある

防衛＆航空･宇宙

Lockheed NASA
Hubble
Telescope
Celoxica Software

Silicon Graphics
Blade
Server
Celoxica Software

高速演算処理 (HPC)

Intel
Networking Products
Celoxica Software

コミュニケーション

Marconi UK
Teleconferencing
Celoxica Software

454 Corp
Genome
Computing
Celoxica Software and Hardware

生物工学

- *Encryption/Decryption*
- *Pattern Matching*
- *Genome Sequencing*

Hienergy Europe
Radiation Detectors
Celoxica Software and Hardware

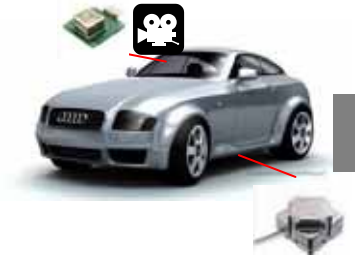
セキュリティ

Sharp UK R&D
3D Imaging
Celoxica Software and Hardware

コンシュウマー

- *Imaging*
- *JPEG, MPEG*
- *Video-Audio*

BAE SYSTEMS
Sensor Fusion
Celoxica software and Hardware



オートモーティブ

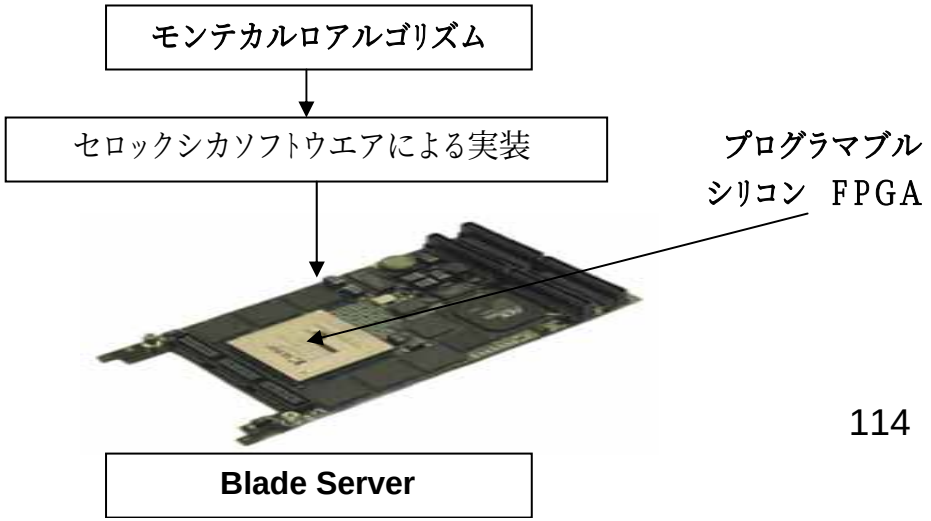
モンテカルロアルゴリズムでの比較

*City of London*での金融計算の性能評価事例

$$P(S,T) = Ke^{-rT}N(-d_2) - SN(-d_1).$$

	FPGA		GPU	CELL		PC
FP	Double	Single	Single	Double	Single	Double
CLK	67MHz	61MHz	400MHz	3.2GHz	3.2GHz	2.5GHz
Speed Ratio	15	41	32	5	29	1

200000万のモンテカルロの
シミュレーションケースを
73秒で終了



(*Processor .vs. Hardware*コプロセッサ)

各アプリケーション	ハードコプロセッサ	ソフトのみ (1とする)
Hough & inverse Hough processing **ハフマン処理**	2 seconds of processing time @20Mhz 370x	12 minutes processing time Pentium 4 - 3Ghz
AES 1MB data proc/cryptography rate Encryption Decryption **暗号**	424 ms/19.7 MB/s 424 ms/19.7 MB/s 13x	5,558 ms / 1.51 MB/s 5,562 ms / 1.51 MB/s
Smith-Waterman ssearch34 from FASTA **遺伝子**	100 sec FPGA processing 64x	6461 sec processing time Opteron
Multi-dimensional hypercube search **多次元ハイパーキューブサーチ**	1.06 Sec FPGA@140Mhz Virtex II 113x	119.5 Sec Opteron - 2.2 Ghz
Callable Monte-Carlo Analysis 64,000 paths **モンテカルロ**	10 sec of Processing @200 Mhz FPGA system 10x	100 sec processing time Opteron - 2.4 Ghz
BJM Financial Analysis 5 million paths **金融計算**	242 sec of Processing @61 Mhz FPGA system 26x	6300 sec processing time Pentium 4 – 1.5 Ghz

ハードウエア･コプロセッサの可能性
ハードと効果的な融合化によりアルゴリズムの高性能化を実現

超高精細ディスプレイによる アンビエント情報環境

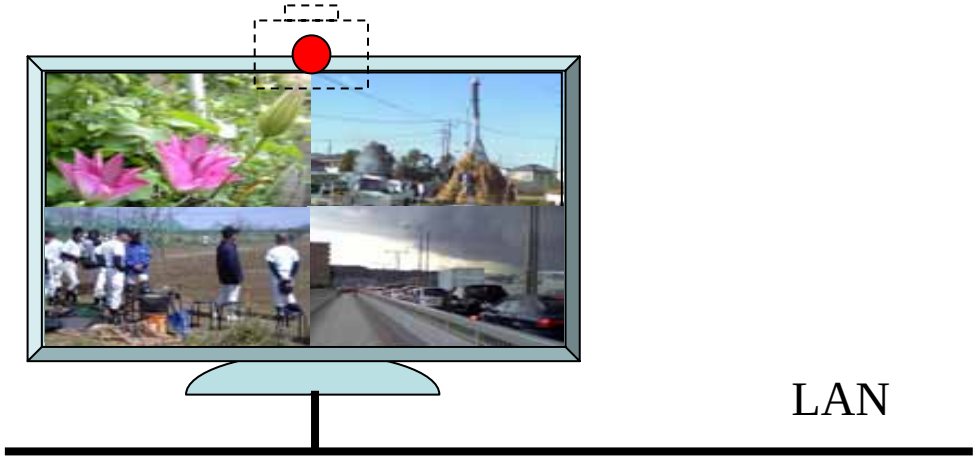
URCF Ultra-Realistic Communications Forum　超臨場感コミュニケーション産学官フォーラム

http://www.scat.or.jp/urcf/　普及促進部会アプリケーション分科会開発環境班

環境の中のディスプレイから人間にアクセスしてきて、アドバイスや提案をしてくれます。

ターゲット解像度
4Kx2K, 8Kx4K

*EDSF2008*展示会でのデモ



- ディスプレイ上にTVカメラを設置。
- TVカメラが人の動作を認識し、サービスを提供する。
- 周辺機器、ネットワークなどとインターラクティブに反応する。
- 煩雑な従来の押しボタンスイッチから開放される。
- Handel-Cを使えば、ビデオ信号処理をFPGA数個で解決！！
- 消費電力は高々数十ワット！！

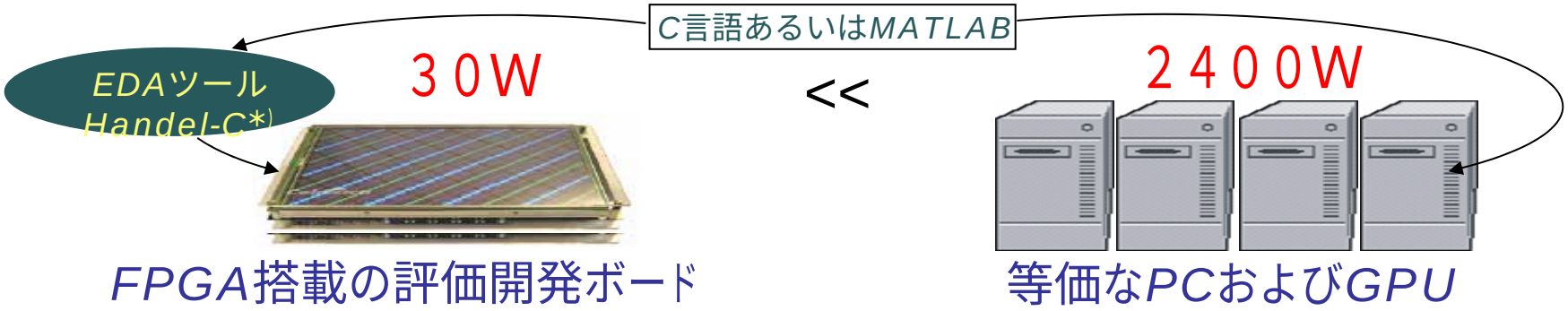
株式会社エイコー
FPGA搭載開発ボードの省労力と省電力

4Kディスプレイ＋4台のカメラによるリアルタイム画像処理

目的：8Kディスプレイのリアルタイム画像処理も提供可能な開発環境のデモ。

省労力：C言語あるいはMATLABからFPGAに並列処理を含むプログラム実装が 出来るので、VDHLなどのハードウェア記述言語(HDL)のスキルが不要。

省電力：FPGAをサポートした高性能評価開発ボードの消費電力は30W。



*） Handel-Cは、FPGAの設計作業を自動化するためのElectronic Design Automation ツールです。

EICOH The Best Solution for Success.

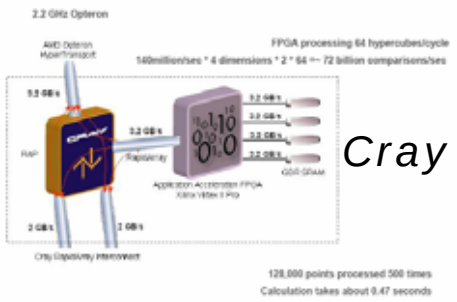
代理販売：株式会社エイコー
Webソリューション事業部
TEL：03-5470-9531 FAX：03-5472-5203
http://www.eicoh.com

Agility
http://www.agilityds.com

Computing without Computers

Wintel アーキテクチャーの限界を破る！！

- *CSP/occam* モデルによる*Handel-C* プログラミングは、更に性能が向上します。
- *FPGA* の低いクロック動作で超高速演算が実現できます。
  - 性能は、何と*Pentiumプロセッサの20-200倍！！*
- *SerDes* による*P2P* の超高速通信リンクが実現できます。
- 超並列計算*(Massively Parallel)* が実現できます。
- 各種ネットワーク･アーキテクチャを実装する事ができます。
  - *Grid, Torus, Ring, Pipe, Systolic Array, Hypercube, Butterfly* などです。
- *SoC/NoC* の導入が簡単です。
- 形式手法とリンクする事が可能です。



Cray XD1

*EDSF2008*での展示会パネル

NSL

NSL（Next Synthesis Language）とは

RTLより一歩先を行く新世代のハードウェア記述言語です。
優秀なハードウェア設計者の皆様のユーザーエクスペリエンスをいかに向上させるかを目標に、自らがハードウェア設計者であるアーキテクトが議論を重ねて言語設計を行いました。もちろん、LSIの大規模化に伴い求められる生産性向上も目的のひとつです。

NSLのルーツは、CSP（Communicating Sequential Processes, 1978, C.A.Hoare）です。同理念をノイマン型コンピュータに採用した代表言語には、Transputerで有名なOCCAM（1983, D.May）があり、ハードウェアに採用した例としてはSFL/PARTHENON（1985, NTT）があります。われわれは、CSPをハードウェアに最適適用するにあたり、過去の事例を詳細に検討し、ハードウェアアーキテクトの視点から、細粒度の並列処理と順序処理の絶妙なバランスを実現した新しい言語体系を確立したものと自負しています。

NSLの論理合成エンジンの開発は、2002年、IP ARCH, Inc.（米国ハワイ州）でスタートし、継続的に熟成を進めてきました。マイクロプロセッサはもちろんのこと、PCIバスコントローラ、SDRAMコントローラ、USBコントローラなど、タイミングクリティカルな設計適用事例も多く、その合成エンジンの信頼性は確たるものがあります。
合成エンジンの上に乗る言語プロセッサは、新世代言語NSLとして2009年末より新規に設計しなおし、オーバートーン株式会社から提供することになりました。

NSLは、初めてこの言語を利用する皆様の過去の資産と経験を大切にします。過去に蓄積したVerilog HDLやVHDLの設計資産は、NSLを使った新世代の設計にそのまま統合できます。さらに、検証のためSystemCとNSLを統合することもできます。また、NSLの表面上の表記は、過去のRTLを利用してきた皆様の直感を阻害しないよう、Verilog HDLにもC言語にも似たものにしています。NSLと巷にあふれる動作合成言語たちとの大きな違いは、ハードウェア生成をクロックサイクル単位で100％コントロールできる点にあります。そのため、NSLで記述すれば、インターロックを含む高度なパイプライン処理を行う高機能マイクロプロセッサなどでも簡単に実現できます。ハードウェアを知り尽くした、優秀なエンジニアである、あなたのための言語、それがNSLです。

Safety and Critical Systems

- ProCoS II project
  - http://archive.comlab.ox.ac.uk/procos/procos2.html

  Oxford大学、Denmark工科大学、Oldenburg大学等が中心となって行う。
- SAFEMOS project
  - http://formalmethods.wikia.com/wiki/SAFEMOS

  Inmos Ltd, SRI, Oxford大学, Cambridge大学の間で行われた。

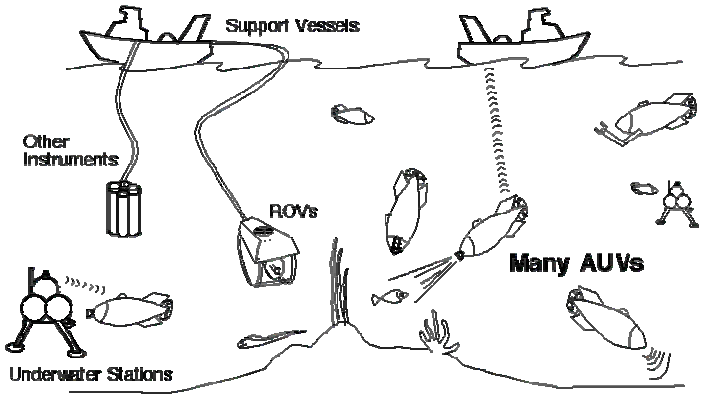
証明されたシステム構築のために、CSP/occam/Transputerモデルと Duration Calculusが採用されている。証明にはその他Z, HOLも使用されている。

Fault Tolerant

- **CSP Channels for CAN-Bus Connected Embedded Control Systems**
  - **http://doc.utwente.nl/43827/**
- クライアント/サーバシステムにおける信頼性向上の一手法
  - **http://www.unisys.co.jp/tec_info/tr52/5202.htm**
- MAFTIA
  - **http://www.maftia.org/**

Autonomous System

- NASA autonomous ground control system
  - http://isd.gsfc.nasa.gov/Papers/DOC/aireview1.pdf
- Twin-Burger (海中ロボット)
  - http://underwater.iis.u-tokyo.ac.jp/robot/tb/tb-intro-e.html



UK Grand Challenges for Computing Research

http://www.nesc.ac.uk/esi/events/Grand_Challenges/

このプロジェクトの推移は経済産業省、産業総合研究所もウォッチしている。

GC1	In Vivo–in Silico (iViS)
GC2/4	Global Ubiquitous Computing
GC3	Memories for Life: managing information over a human lifetime
GC5	The Architecture of Brain and Mind
GC6	Dependable Systems Evolution
GC7	Journeys in Non-Classical Computation
GC8	Learning for Life
GC9	Bringing the Past to Life for the Citizen

GCにおいてCSP/π計算モデルとoccam-π/JCSP/C++CSP/Handel-C等のプログラミング言語が主な開発ツールになっている。

UK Grand Challenges for Computing Research

http://www.ukcrc.org.uk/about/member.cfm

メンバーの多くは形式手法、CSP/π-calculus等の大家である。著名な名前をピックアップすると：ー

Domain Theory →	Samson Abramsky	University of Oxford
	Muffy Calder	University of Glasgow
VDM →	Cliff Jones	University of Newcastle
CSP →	Mark B Josephs	London South Bank University
Robot	Michel Brady	University of Oxford
Real-Time(occam/Ada)	Alan Burns	University of York
Ambient-calculus →	Luca Carrelli	Microsoft Research, Cambridge, UK
CSP →	Tony Hoare	Microsoft Research, Cambridge, UK
LTSA	Jeff Kramer	Imperial College London
CSP →	Marta Kwiatowska	Oxford University Computing Laboratory
LTSA	Jeffrey Magee	Imperial College London
Transputer考案者 →	David May	University of Bristol
Pi-calculus →	Robin Milner	University of Cambridge
CSP →	Bill Roscoe	Oxford University Computing Laboratory
	Steve Schneider	University of Surrey
CSP/Z/Circus	Jim C P Woodcock	University of York

CoSMoSプロジェクト

Complex Systems Modelling and Simulation infrastructure

Project Aims

Our project will build capacity in generic modelling tools and simulation techniques for complex systems, to support the modelling, analysis and prediction of complex systems, and to help design and validate complex systems. Drawing on our state-of-the-art expertise in many aspects of computer systems engineering, we will develop CoSMoS, a modelling and simulation process and infrastructure specifically designed to allow complex systems to be explored, analysed, and designed within a uniform framework.
CoSMoS will comprise: **a modelling and analysis process**, based on computational concepts such as class, state, behaviour, and communication, and on complex system emergent properties, expressed in part as rich argumentation, modelling, analysis, and refactoring Pattern Languages; **a massively parallel and distributed simulation environment**, **based on CSP** and system modelling technologies that encompass a wide range of process granularities, targeted to the specific properties of complex systems: vast numbers of (relatively) simple agents interacting and communicating in parallel, in an (often stigmergic) environment. The development of CoSMoS will be **case study driven**, to ensure that it contains the necessary generic components, and so that several iterations of the method and toolset, of increasing functionality and applicability, can be produced and validated. The detailed project aims are:

・to design **pattern languages** for: abstract computational representations suitable for modelling complex systems; analyses of their collective and emergent properties; refactorings, both of composed models, and for targetting simulations; argument structures, to reason about validity
・systems, as instantiatable code frameworks in **occam-π, Handel-C, and JCSP**, targeting multiple processors
・to bring these together in an **integrated process**, that guides the tasks of probing a complex system in order to build suitable abstract models (modelling pattern language), mapping a model to the simulation framework (refactoring pattern language), instantiating the framework to produce a simulation, arguing the validity of the simulation against the original system (argumentation pattern language), and using the simulation in a predictive manner (analysis pattern language)
・to model and simulate a range of complex system case studies, both for driving the initial development and for performing the eventual validation of the entire CoSMoS process (modelling, mapping, instantiating, validating, predicting).

FM2009

16th International Symposium on Formal Methods
Eindhoven, the Netherlands, October 30 - November 7, 2009

Events
This page provides a short description of the events of Formal Methods Week, and links to further information.
FM2009 …
FMICS …
PDMC …
REFINE …
TESTCOM/FATES …
FACS …
CPA Two-and-a-half-day workshop on Communication Process Architectures, organized by WoTUG, a forum set up to support those applying the CSP model of parallel processing. The conference theme is concurrency, and models of concurrency - at all levels of granularity, and as applicable to both software and hardware. CPA aims to stimulate ideas and discussions relevant to the engineering of concurrent systems.

CSP/occam国際会議

http://www.win.tue.nl:80/fm2009/index.html

大学での教育・研究状況

CSP/FDR2/occam-pi/JCSP/C++CSP/Handel-Cを使った並列処理の研究、教育は:

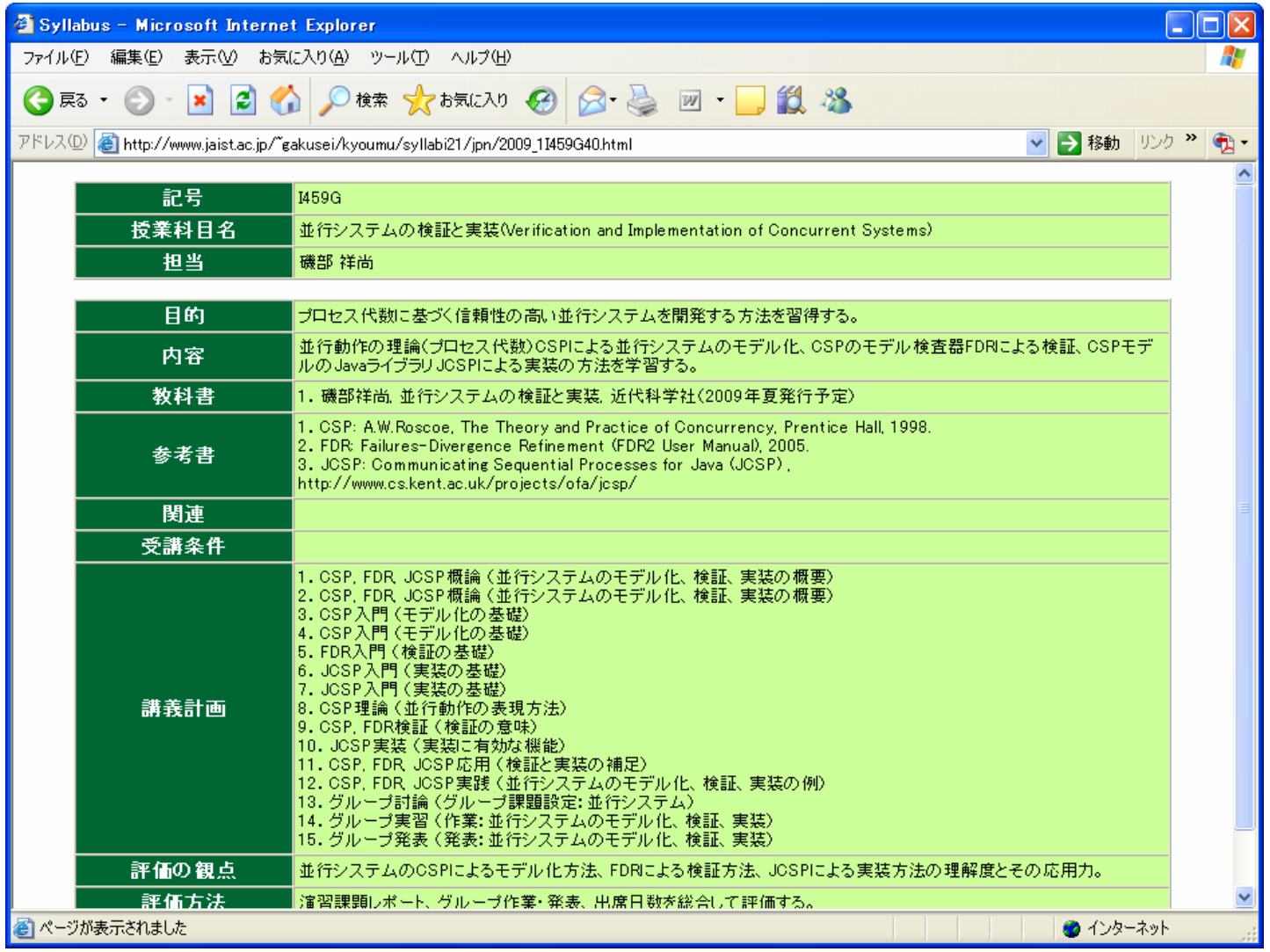
英国（Kent, Bristol, Oxford, Keele, Surry, Birmingham, Manchester, York, Loughborough University, Southampton, London, Imperial, South Bank, Essex, Plymouth, Bradford, MacMaster、Newcasltle upon Tyne、、、等非常に多くある。基本的にはUKの国家政策である。）、
米国(MIT, Bell A&T Lab, CMU, UCB, UCSD, PSU, RIT, USU, Colgate大学, Colby College, NYU, Syracuse, Colorado, Texus,等)、
ドイツ(Paderborn, Oldenburg, Bremen, Berlin, Passau, Dresden, Ulm等)、
フランス(INRIA,IMAG)、デンマーク(Copenhagen)
オランダ(Twente, Utrecht, Leiden)、
イタリア(Bologna, Pisa)、
ベルギー(Limburgs)、
スペイン(Malaga)、
シンガポール(NUS)、
オーストラリア(Queensland)、
インド(IIT)、
カナダ(Calgary)など多くの大学で教育･研究がなされています。

日本では(東大?)、首都大学東京、国立情報学研究所、産業総合研究所、法政大、慶応大学、北海道職業能力開発大学校、神奈川工大などの先生方は熱心である。Handel-Cは東京大学、東工大、大阪大学、奈良先端科学大学院大学、北陸先端科学大学院大学、名古屋大学、九州大学、静岡大学、金沢大学などで使われている。

Top SE 講座

国立情報学研究所においてTop SE教育が行われており、その中で「並行システムのモデル化と検証」の講義の中で産業総合研究所の磯辺氏によってCSP、FDR2、JCSPの講義が実施されている。

http://www.topse.jp/courses.html



北陸先端科学技術大学院大学

http://www.jaist.ac.jp/~gakusei/kyoumu/syllabi21/jpn/2009_1I459G40.html

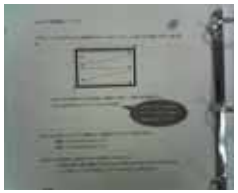
実践的な教育システムへ取り組みとアカデミックでの積極採用

- 日本セロックシカでの対応
  - 月単位の定期2日基本コース （座学、ワークショップで教育機材による実践指導）



基本コースの案内とテキスト、教育ボード　　日本語実践マニュアルと内容

  - 顧客の要望にもとずいたオンサイトトレーニングの提供
  - アプリケーショントレーニング （個々のアプリケーションに対応した実践教育）




組み込みアプリケーションに
対応した教育内容

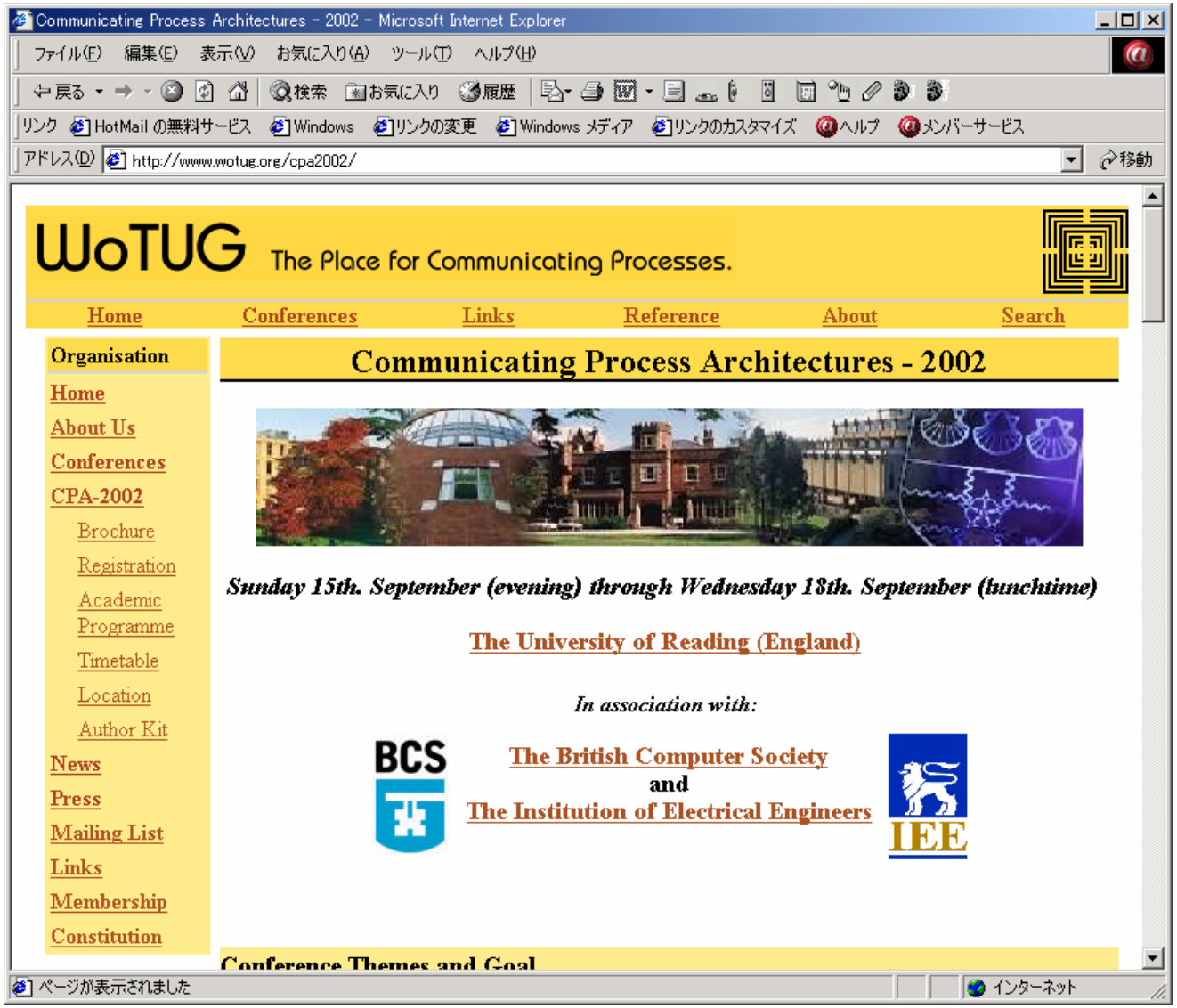
- 海外での対応事例
  - Oxford大学でのCSP/Handel-Cの教育講座
  - Essex大学でのHandel C教育講座
    - ハードウエア・ソフトウエア協調設計に対してHandel-Cをベースに教育コースを開設
  - Imperialカレッジでの Handel C Forumの立ち上げ
    - http://www.doc.ic.ac.uk/~akf/handel-c/cgi-bin/forum.cgi
  - シンガポールのNTU/NUS大学での積極的なHandel-C教育の推進

東京大学VDECでのトレーニング例

Essex大学のハードソフトの
協調設計教育講座の案内

Imperial大学のWeb page

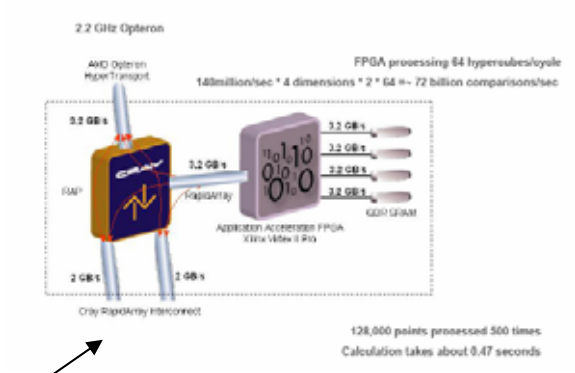
World occam and Transputer User Group (http://www.wotug.org/)



1984年からCSP/occam/JCSP/C++CSPによる並列処理の理論、形式手法、応用に関して幅広く発表されている。よき仲間が集まるところ。

量産品となっているもの

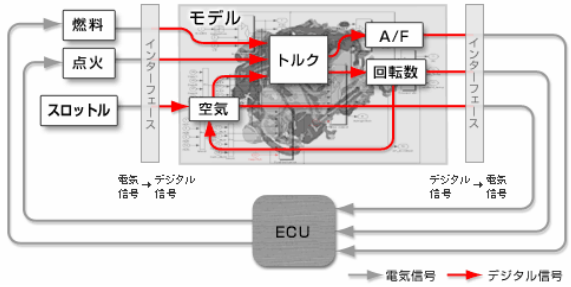
- STMicroelectronics社のSoC
  - 32bit MPU, ST20+STB/GPS/DTV/DSC/DVD/MPEG/他
  - STBは今でもNECを抜いて第一位になっている。
  - 累積、数千万個以上のチップが販売されたと推測される。
- 4Links社
  - SpaceWireアナライザ、PCIボード、ASIC化も検討中。
  - NASA、ESA、等世界中の宇宙システム機器の大規模ネットワーク制御



- Agility社
  - FPGA設計ツール、評価ボード(http://www.celoxica.co.jp)
  - (TVで宣伝している) 大手メーカのビデオ機器での高速画像処理チップ
  - 東芝MePデジタル･メディア向けSoC開発キット
  - CRAY XD1 演算ブレード
  - SGIのBlade Server
  - モータの”hardware in loop”シミュレーション (富士通テン)

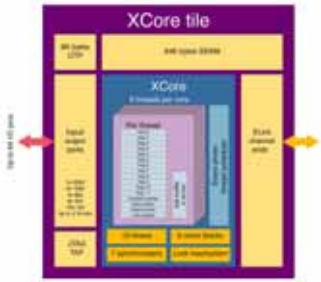
TOSHIBA

MeP Media embedded Processor



- ARM社のチップ
  - 携帯電話メーカの大半が採用している
- セイコーエプソンのプリンター
- Philips社の製品

- XMOS社の開発キットの発売開始(2008)
  - http://www.eetimes.jp/contents/200712/29105_1_20071210205510.cfm



- CSP.NET版CSPモデル対応のJava, C++はデンマークのAxon7社から発売されている。
  - http://www.axon7.com/

CSP研究会

CSP研究会は4ヶ月に1回開催しています。
モデル検証、アプリケーション事例を広く知らしめるために、
発表資料を以下のサイトに公開しています。

http://www.csp-consortium.org/