An Effective Network Processor Design Framework - Using Multi-Objective Evolutionary Algorithms and Object Oriented Techniques to Optimise the Intel IXP1200 Network Processor

Liam Noonan, Dept E&CE University of Limerick

Colin Flanagan, Dept E&CE University of Limerick

UNIVERSITY of LIMERICK
OLLSCOIL LUIMNIGH

intel

TIPPERARY INSTITUTE

Introduction

- In this presentation we present a framework for design space exploration of a network processor, that incorporates parameterisation, power and cost analysis.
- This method utilises multi-objective evolutionary algorithms and object oriented analysis and design.
- Using this approach an engineer specifies certain hard and soft performance requirements for a multi-processor system, and allows it to be generated automatically by competitive evolution/optimisation, thus obviating the need for detailed design.
- To make the proposal concrete, we use the Intel IXP1200 network processor as a baseline complex system design and show how various improvements can be make to this architecture by evolutionary/competitive design.
- Various approaches to multi-objective optimisation (Darwin, Lamarck Baldwin, etc.) are compared and contrasted in their ability to generate architectures meeting various constraints.

The need for speed

- Network processors are high performance programmable processors that are optimised to process packets.
- In order to provide high rates of packet processing, NPs have typically adopted a strategy of distributed processing and parallelism.
- This programming model on paper is very effective, however in practice it can be difficult to implement and debug.
- This is due to its inherent complexity; designing a network processor can involve the optimisation of many component devices and subsystems.
- The first generation of network processors utilised a great number of different approaches, which made assessment of the technologies a complex issue.

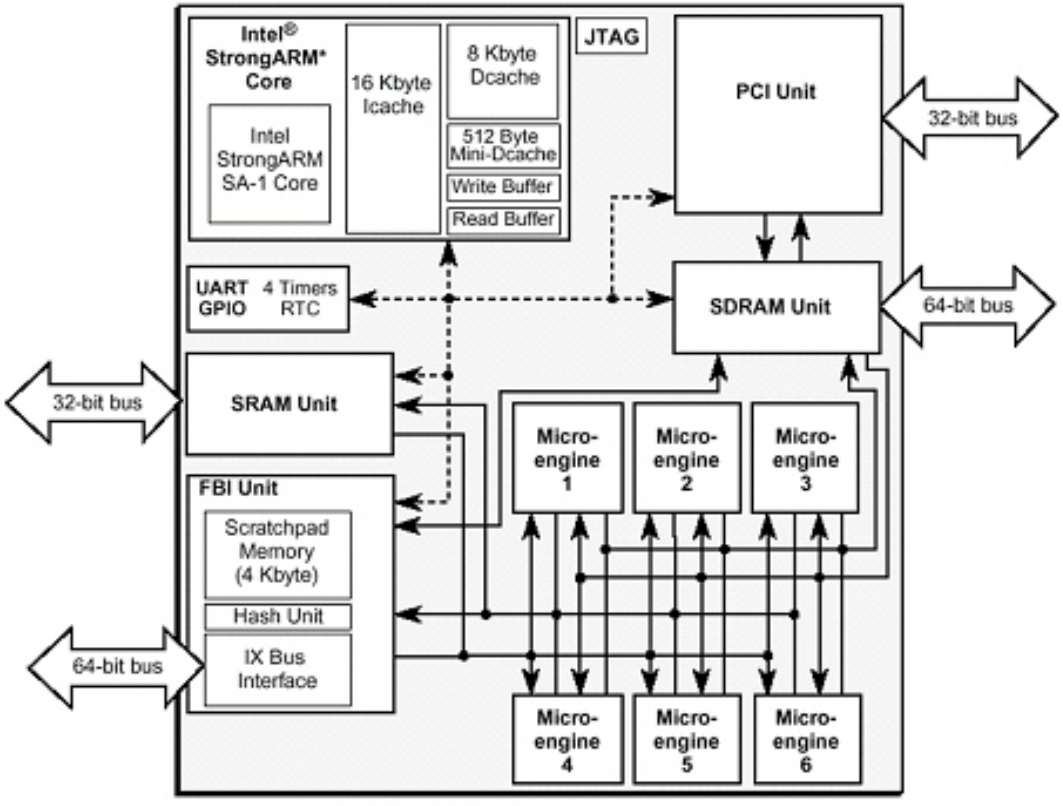
The challenge

- The challenge facing industry is to develop innovative products of increasing complexity within shorter design times. Decisions made at an early stage can have later repercussions.
- These early decisions can affect the over all performance of the system.
- Are these architectures a good fit to the real problem?
- How do we discover this and improve the design?
- There is a need for an architectural exploration tools that can assess the performance of various designs.
- This tool should utilise proven software modelling and development techniques.

The framework

- Utilises a flexible object oriented model of a NP. The model is constructe using UML and implemented using the object oriented modeling languag POOSL
- User-supplied parameters allow the designer to describe the manufacturing process and microprocessor characteristics in great detail
- Various evolutionary algorithms are employed to generate configurations from randomly initialised search spaces, or to attempt to improve user-supplied configurations
- Multi-objective fitness criteria based on empirical calculations and analysi (e.g., chip area, power consumption, cost per die) are utilised
- The result is a configuration description of the multi-core microprocessor that provides details on the bus widths, speed of processing units and chip area.
- We believe that the combination of features offered in this framework will provide significant assistance to system architects and designers.

A complex network processor



Intel IXP1200 Network Processor

Notes:

— 32-bit Data Bus

--- 32-bit ARM System Bus

Language Review

- The design exploration tool should allow the engineer th facility to quickly and easily modify the architecture of th network processor and assess the implications of thi modification.
- If the tool has been appropriately designed, it should b possible to target other architecture families.
- With this in mind, we conducted a review of the languag alternatives that were available to us.
  - ROOM
  - SDL
  - Esterel
  - **SystemC**
  - C++
  - POOSL

The decision

- A design exploration tool should utilise a language/methodology that offers concurrency, communication, data types, timers and probabilistic functionality.
- There is a requirement for asynchronous processes, conditional message passing and sharing of data objects between concurrent activities.
- The language should offer an intuitive interface with low-level trace and debug tools, as well as the facility of object and code reuse.
- Furthermore the language should be an OO language as this facilitates the independent modelling of specific components and code reuse.
- Language should support UML 2.0
- We chose POOSL as it met the above criteria.

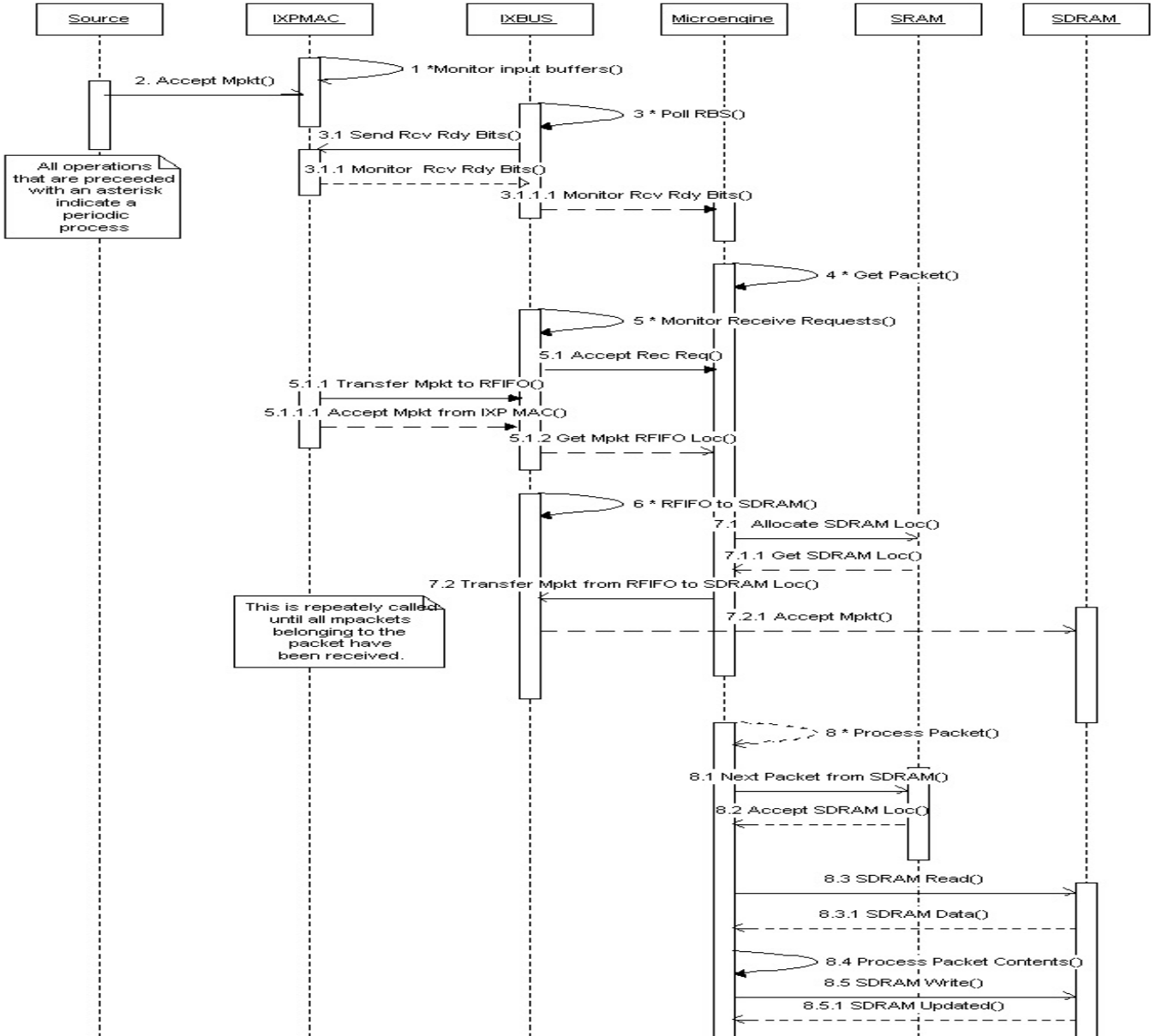
POOSL

- Based upon smalltalk
- Utilises Mathematically defined semantics
- Consists of a process part and data part
- System level model consists of:
  - Process, cluster and data objects

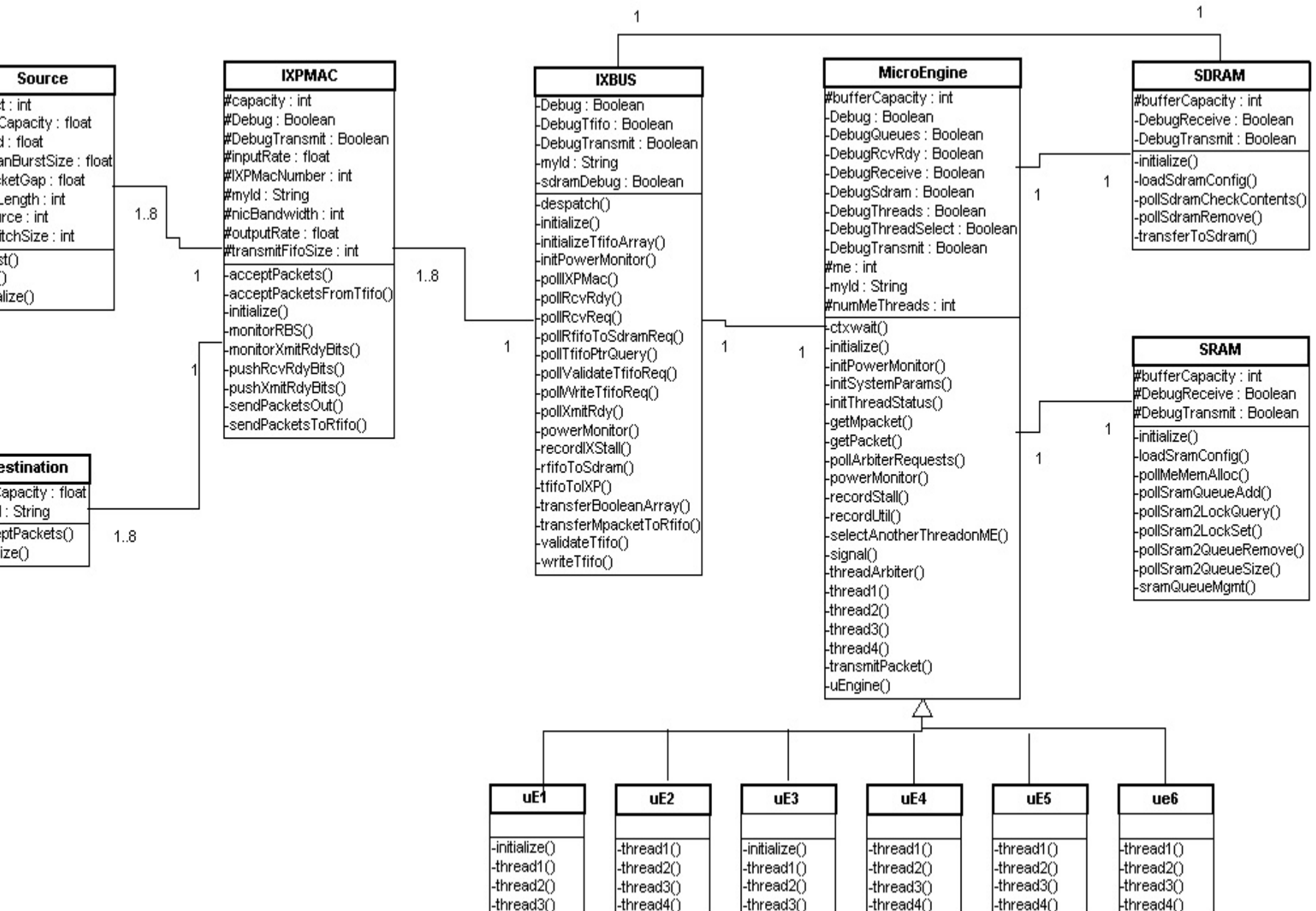
Analysing the Problem

- Before construction of the model was attempted a flow diagram was constructed so as to provide an overview of the system.
- This allowed assessment of the various components that were required as well as the sequence in which they occurred.
- This approach also highlighted the various parallel asynchronous processes that were required as well as the process classes in which they would reside.

Flow/High level Seq diagram



High level implementation class diag



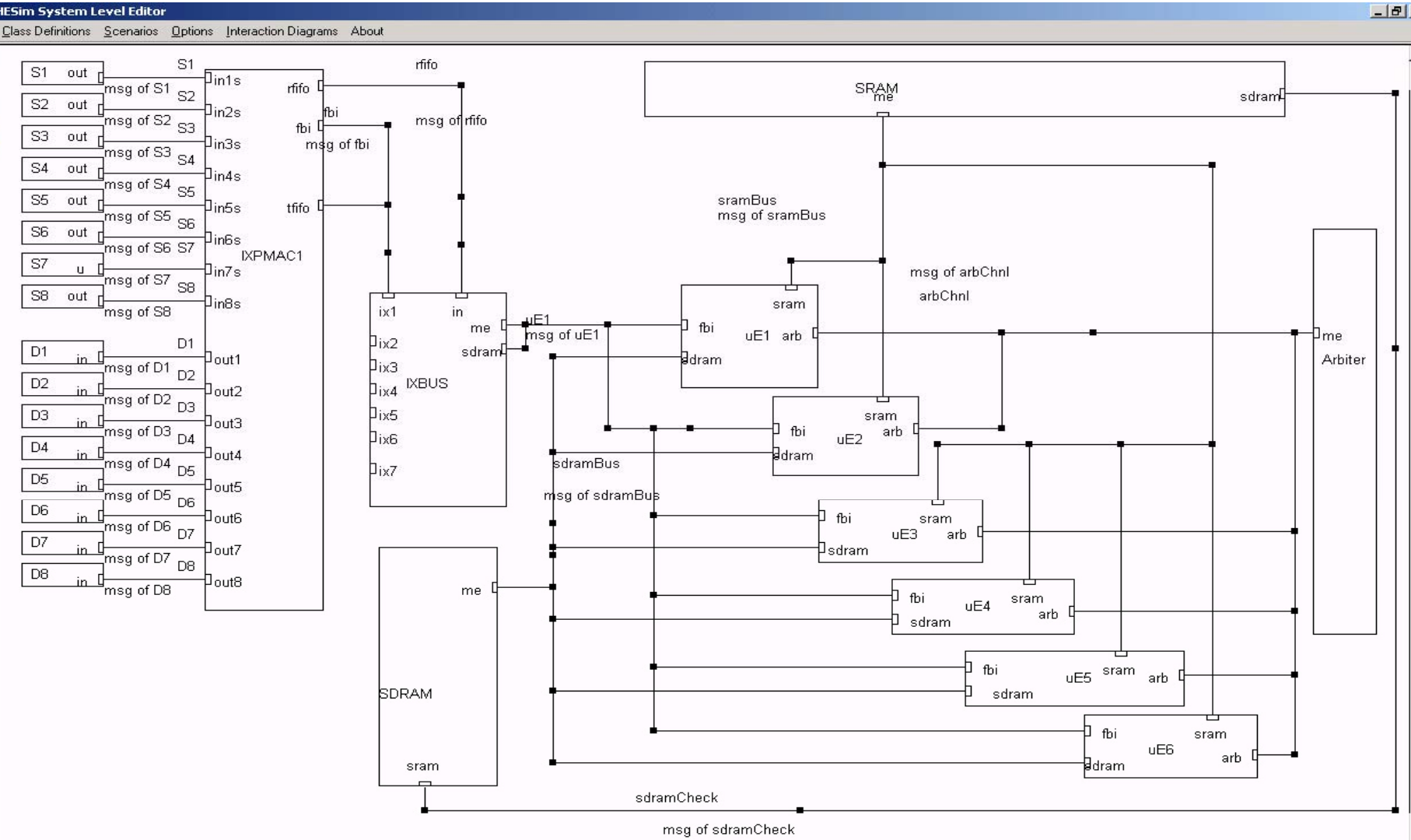
Modelling the NP - processes

- Translating the network processor model into POOSL syntax was achieved through the identification of process and data classes.
- The process classes in general reflected the individual hardware components of the NP (network processor). A superclass and polymorphism was utilised so as to implement generic Microengines that implemented specific function calls in the threads.
- SDRAM, SRAM etc were modelled as separate classes

Modelling the NP – data & SHESim

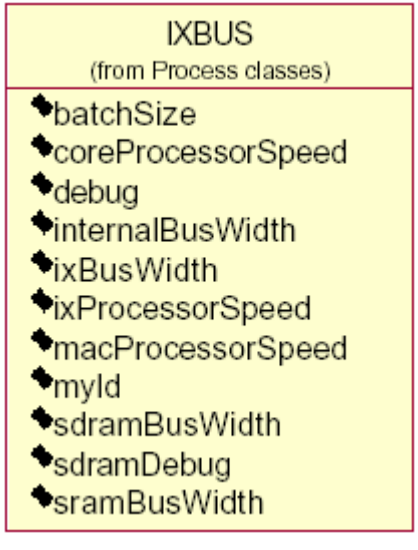
- The data classes that were used to represent packets, Mpackets, storage etc were able to make full use of and exploit polymorphism and inheritance.
- The methods from the base class were available to all classe that inherited from them and this aided in the rapid prototyping of the NP.
- SHESim is a graphical tool intended for incremental specification and modification of POOSL models that can be validated by interactive simulation
- The process and data layers of a model are denoted in textual form, whereas the architecture layer is specified graphically.
- The numerous inspection possibilities and the user-friendly graphical interface make SHESim very suitable for revealing model inadequacies

SHESim Environment



Model Parameters

- The ability to change the width of data buses and the size of data storage provided a level of simulation that was not previously available for the IXP1200.
- Debugging was also added so as to examine in detail all aspects of the mpacket flow through the various components.



Comparison of Intel and POOSL

ME	IX	Intel In	POOSL In	% Diff		Intel Out	POOSL Out	% Diff
177	66	504	515	*2.18*		504	500	*-0.79*
177	80	526	544	*3.42*		525	525	*0*
177	104	524	579	*10.49*		523	567	*8.41*
200	66	504	519	*2.97*		504	506	*0.39*
200	80	526	548	*4.18*		525	534	*1.71*
200	104	524	581	*10.87*		524	570	*8.77*
232	66	553	520	*-5.96*		552	512	*-7.24*
232	80	608	548	*-9.86*		607	535	*-11.86*
232	104	647	583	*-9.89*		646	571	*-11.6*

- The Intel developer workbench was configured to run at wire speed usin 64 byte packets so as to gauge the ability of the IXP1200 to perform under heavy packet loads.
- The assessment of accuracy of the model was based upon on how close the measurements for receiving and sending at the IX interface corresponded to the Intel workbench results

Determining the NP core config

- The designers develop a parameterised POOSL model of the proposed application in terms of packet processing actions.
- The parameters that can be modified include system speed various components as well as bus widths, this allows the designers to assess th change in performance characteristics due to a change in model parameters.
- The designers decide on the criteria to be optimised. Cost, throughput etc.
- The genetic algorithms determine the parameter values for the POOSL model.
- The POOSL model is then executed and the outputs from the model are utilised in fitness functions. Population sets are created and evolutionary processes are applied
- The configuration that produces the best fitness function is identified as the optimal configuration

GA encoding

- GA's encode using chromosomes.
- The chromosomes are either single bits, or blocks of adjacen bits, that encode a particular element of candidate solution e.g., an integer value.
- The target NP is encoded using 7 chromosomes of type integer, which are listed in the table below.

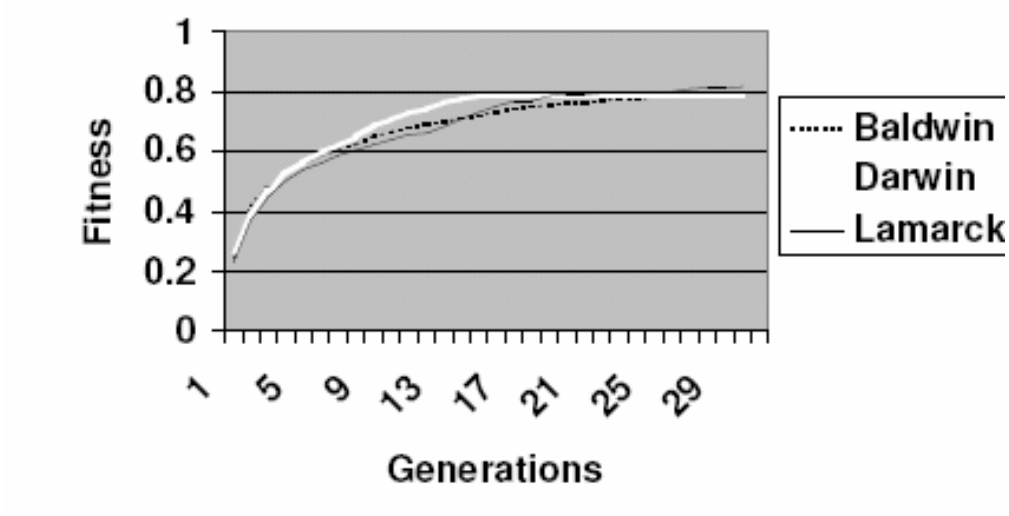
Chromosome	Description
0	IX Bus Width
1	Internal Bus Width
2	SDRAM Bus Width
3	SRAM Bus Width
4	MAC Processor Speed
5	IX Processor Speed
6	ME Processor Speed

Comparison of Intel to ga generated configs

	Intel	Darwin	Lamarck	Baldwin
Cost Per Die	70.6	17.98	18.68	21.75
Power (Watts)	2.48	1.50	1.23	1.31
Throughput	95	95	98	95
Over all Fitness	0.51	0.79	0.82	0.80
ME Mhz	232	137.9	143.3	136.2
IX Mhz	104	132.64	110.2	111.2
Mac Mhz	104	132.64	319.3	232.6
Internal Bus Width	32	8	8	16
IX Bus Wdth	64	88	24	40
SDRAM Bus Wdth	64	8	8	8
SRAM Bus Wdth	32	8	8	8
Die Size (mm)	126	83.37	83.31	92.64

Tabu was applied to the above configurations.
The generated configurations were better, but usually except for the Intel configuration the percentage gains in fitness were negligible.

Generation Analysis



- What can be observed from this chart is the fact that significant improvements are made within the first 16 generations of the process, after that point the gains reduce dramatically in value.
- By the time generation 22 is processed the improvement in the fitness is negligible. The latter end of this process recorded a very small value for the standard deviation for the mean fitness of a generation

Proposed Configs

- What is interesting to note about the above results are that the proposed configurations from the various approaches are not identical.
- This implies that the algorithms have utilised the random search space in a broad but effective manner.
- The decision as to which configuration should be utilised rests with the designer.
- This approach offers the design team a choice of architectures for them to consider and further analyse.
- The three approaches have however, highlighted a common design theme. Cost and power savings can be achieved by increasing the clock speeds of the IX and MAC units and reducing the microengine clock speeds and various bus widths.
- This requires an “island clocking" policy that allows for components to ru at separate clock speeds, as well as handshaking logic to allow message passing between clock islands.
- The evolutionary approaches identified the real bottleneck in the system i.e., the IX and Mac units. It is more apparent when the complete OO sequence diagram for sending and receiving a packet is analysed.

Island Versus Common clocking

	IX and MAC island clocking	IX and MAC common clocking
ME Mhz	190	194
IX Mhz	178	203
Mac Mhz	431	203
Int Bus	8	8
IX Bus	16	24
SDRAM Bus	8	16
SRAM Bus	8	8
Cost Per Die	24.8	25.3
Throughput	95	95
Power (Watts)	1.75	1.81

- Above config is for a 44 signature detecting NDS
- There are some notable differences in the two architectures with respect to bus widths.
- The difference in cost (2%) and power (3.4%) is marginal however.
- This small extra cost would be well worth considering as it simplifies the overall design of the NP

Conclusions

- The possibilities for utilising a design exploration framework that employs multi-objective evolutionary algorithms with a parameterised Object Oriented model are encouraging.
- The parameter-driven nature of the POOSL model allows one to alter significantly the behaviour of the IXP1200 and quickly assess the impact of this configuration change.
- This feature has been exploited successfully by the multi-objective evolutionary algorithms.
- Improvements to the architecture from a cost and power consumption point of view have been achieved.
- Comparisons across a broad range of scenarios for a proposed architecture have also been assessed.
- The approach offers the design team a choice of performance-evaluated architectural alternatives for them to consider, along with a rapid and objective way to identify potential performance bottlenecks.
- This helps to turn computer architecture from an art to a quantifiable science.