Embedded Reconfigurable Processing for \( \mu \)UAV Applications

Part I - Onboard Processing

Chris Papachristou
Case Western Reserve University
Cleveland, Ohio 44106
cap2@case.edu
Outline

• On board Information processing
• Main Technologies
  – Digital Signal Processors - DSP
  – Reconfigurable Computing, FPGAs
  – Embedded processing
  – Self Reconfigurable processing
  – Evolvable Hardware
Onboard Processing Requirements

- Computational Performance
  - sufficient to accomplish complex imaging algorithms
- Low power and Low Energy
  - management of circuitry, architecture
- Minimal physical characteristics
  - packaging, weight
- Communication Performance
  - antennas, digital soft radio, protocols
- Storage
Key Component Technology

- Digital Signal Processors, DSPs
- Reconfigurable processors, FPGAs
- Embedded processors
Digital Signal Processors (DSPs)
What are DSPs?

- Embedded microprocessors that are designed to handle digital signal processing applications in a very cost effective manner
- Current market leaders:
  - TI, Motorola, Lucent
- Market well over - $ 50 Billions
Nature of DSPs

- DSPs utilize special hardware to meet performance, power, and price points
- Sacrifice orthogonality and ease-of-use to meet goals
- Assume hand-assembly or libraries used for core algorithms
- Compiler mostly used for control and glue logic
DSP principle

• Converting a continuously changing waveform (analog) into a series of discrete levels (digital)
DSP principle

- The waveform is sliced into equal segments and the amplitude is measured in the middle of each segment.
- The measurements make up the digital representation of the waveform.
ADC and DAC

- **ADC**: analog conversion to digital
- **DAC**: digital conversion to analog
- Both operations are approximate as the waveforms do not completely match - filtering needed to smooth them out
DSP algorithms

- Basically various filtering type of algorithms
- FIR: finite impulse response
- IIR: infinite impulse
- Bandpass filter
- AR: autoregressive
FIR filter

- Most widely used filter
- series of delays, multipliers, and adders
- frequency response output fine-tuned to filter's length

\[ y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \]
DSP - Architecture Characteristics

- DSP architecture is designed to solve one problem well
  - Digital filters (FIR, IIR) and FFTs
  - In Real-Time
- Architecture features added to speed up this problem
  - **MAC**: multiply & accumulator, speedup FIR tap
  - **Circular buffer**: speedup shifting FIR delay registers
  - RISC based: single clock per instruction
  - Harvard Architecture: separate instruction & data
  - Word oriented
DSP characteristics

• Disadvantages: not a general purpose computer
  – slow character processing
  – No multi-user operating system support
  – No virtual memory, no translate lookside tables
  – No memory page protection (Read, Write, Execute)
FIR filter architecture

- Example

\[ x \rightarrow D \rightarrow D \rightarrow \cdots \rightarrow D \rightarrow \cdots \rightarrow D \rightarrow + \rightarrow + \rightarrow \cdots \rightarrow + \rightarrow + \rightarrow \cdots \rightarrow + \rightarrow + \rightarrow y \]

- \( x \) = input samples
- \( y \) = output samples
- \( h \) = filter coefficients
- \( D \) = unit time delay

**Filter characteristics**
- governed by number of taps,
- selection of filter coefficients.

\[ \Sigma x \cdot h \]
FIR on typical processor

- simple assembly FIR routine

```assembly
loop:
    mov     *r0,x0
    mov     *r1,y0
    mpy     x0,y0,a
    add     a,b
    mov     y0,*r2
    inc     r0
    inc     r1
    inc     r2
    dec     ctr
    tst     ctr
    jnz     loop
```

Problems:
- Memory bandwidth bottleneck
- Control code and addressing overhead
- Possibly slow multiply

(Computes one tap per loop iteration)
Early DSP architecture

- simple datapath and memory structure
**Von Neuman architecture**
- Area efficient but requires higher bus bandwidth because instructions and data must compete for memory.

**Harvard architecture** was coined to describe machines with separate memories.
- Speed efficient: Increased parallelism.
FIR filter on conventional DSP

Use of dot product

Do dotprod UNTIL CE;
dotprod:

\[
MR = MR + MX0 \times MY0 \quad (SS),
\]
\[
MX0 = DM(I0,M0),
\]
\[
MY0 = PM(I4,M4);
\]
Baseline DSPs

• Common attributes
  - Arithmetic: 16 or 24-bit or even 40-bit fixed point (fractional), or 32-bit arithmetic operations
  - Instructions: 16-, 24- or 32-bit instructions
  - Issue: one instruction per cycle, single-issue
  - complex, compound instruction encoding, many operations
  - highly constrained, non-orthogonal architecture
  - dedicated addressing hardware
  - specialized addressing modes
Baseline DSP

• attributes (cont)
  - on-chip memory architecture
  - dedicated hardware for loops and other execution control
  - on-chip peripherals and I/O interfaces
  - low cost, low power, low memory usage
Increasing Parallelism

• Boosting performance beyond faster clock speeds requires the processor to do more work per cycle

• Two ways to increase the processors' parallelism:
  – Increase the number of *operations* that can be performed in every cycle
  – Increase the number of *instructions* that can be issued and executed in every cycle

• this leads to pipelining and parallelism
More Operations per instruction

• How to increase the number of operations performed in each instruction?
  – Add execution units (multiplier, adder, i.e. add hardware)
    • enhance the instruction set to take advantage of extra hardware
    • Possibly, increase the instruction word length (width)
    • Use wider buses to keep the processor fed with more data
  – Add SIMD capabilities -  \textit{data parallelism}
Architectures for DSPs

• Enhanced conventional DSPs
  – Lucent DSP16xxx, ADSP 2116x

• VLIW (Very Long Instruction Word) DSPs
  – TI TMS320C6xxx, Siemens Carmel, Philips Trimedia

• Superscalar DSP
  – ZSP ZSP164xx

• Hybrid processors
  – PowerPC with Altivec Hardware, TriCore
Example

DSP16000 Data Path
Enhanced conventional DSPs

- More parallelism via:
  - Multi-operation data path
    - e.g., $2^{nd}$ multiplier, adder
    - SIMD capabilities
  - Highly specialized hardware in core
    - e.g., application oriented datapath operations (crypto)
  - Co-processors
    - Viterbi decoder, FIR filtering, mpeg7, etc
**SIMD – single instruction multiple data**

- Split words into smaller chunks for parallel operations
- Some SIMD processors support multiple data widths, such as 16-bit, 8-bit,...)
  - For example, Lucent DSP16xxx, ADI ADSP 211x
Challenges to SIMD

• Algorithms, data organization must be amenable to data parallelism
  – Programmers must be creative, pursuing alternatives
  – Reorganization penalties can be significant
  – SIMD most effective on algorithms that process large blocks of data
More Instructions per clock

- How to increase the number of instructions issued and executed in every clock cycle?
  - Use VLIW techniques
    - static scheduling
  - Use Superscalar techniques
    - dynamic scheduling
Superscalar vs VLIW: scheduling

Memory

| INS 1 | INS 2 | INS 3 | ⋮ | INS n |

Instruction scheduling, dispatch

| INS 1 | INS 2 | INS 3 | INS 4 | INS 5 | INS 6 |

Execution Units

| ALU | MAC | BMU | ⋮ |

| INS 1 | INS 2 | INS 3 | INS 5 |
VLIW concept

On-Chip Program Memory

2 Independent Data Paths

Dispatch Unit

L1 S1 M1 D1
Register File A

L2 S2 M2 D2
Register File B

On-Chip Data Memory

32x8=256 bits
VLIW Application

- FIR filter loop

```assembly
LOOP:
    ADD .L1 A0,A3,A0
    ADD .L2 B1,B7,B1
    MPYHL .M1X A2,B2,A3
    MPYHL .M2X A2,B2,B7
    LDW .D2 *B4++,B2
    LDW .D1 *A7--,A2
[B0] ADD .S2 -1,B0,B0
[B0] B .S1 LOOP
; LOOP ends here
```
Evaluation

• Advantages
  - performance
  - regular structure
  - easier to program – depending on tools

• Disadvantages
  - difficult tools -- compilers/schedulers
  - deep pipeline latencies
  - code size explosion
  - higher power consumption
Superscalar DSPs

- Characteristic
  - hardware support for instruction control
  - 2-4 instruction issue per cycle
  - lots of parallelism

- Example FIR filter

```
LOOP: LDDU   R4, R14, 2
      LDDU   R3, R15, 2
      MAC2.A R4, R8
      AGN0   LOOP
```

- All four instructions exec in parallel
Evaluation

• Advantages
  - performance
  - easier tools -- compilers
  - smaller code size

• Disadvantages
  - dynamic behavior complicates software development
  - execution time unpredictability
  - high energy consumption
Hybrids

• Typical approach: Embedded DSP and microcontrollers
  – heterogeneous multi-core systems including
    • Regular processor cores
    • DSP co-processors
    • Advanced cryptoprocessors

• Design methods
  – tweaking a GPP with DSP support, or
  – tweaking a DSP with some microcontrol support, or
  – entire new design from scratch
Example: TI OMAP Hybrid Processor
Reconfigurable Processing
Reconfigurable Processing

- Ability of a device to change its internal structure, functionality, and behavior, either on command, or autonomously.

- Two methods for execution of algorithms:
  - hardwired technology: high performance
  - software-programmed microprocessors: high flexibility

- A third approach: Reconfigurable computing
  - intended to fill the gap between hard and soft, achieving potentially much higher performance than software, while maintaining a higher level of flexibility than hardware
Reconfigurability Classes

- Static Configuration: performed while device is off line.

- Dynamic Configuration: device is on-line, "on the fly".

- Self Reconfiguration: performed autonomously by device.

- Evolution type: Self Reconfiguration with adaptation such as replication and growth, "bio-inspired".
Reconfigurable Logic

• Currently implemented by FPGAs

• **Static reconfiguration** is achieved by downloading into the FPGA chip a new configuration while the FPGA is off-line

• obvious disadvantages in configuration time
Traditional FPGAs

- Logic cell
- I/O cell
- Programmable interconnections
- Programmable functions
- Configuration
Static Configuration

- Graphic editor
- VHDL
- Schematic
- Partition
- Placement
- Routing
- Configuration bit-string
Dynamic Configuration

- It is achieved by inserting new FPGA functionality on the fly, i.e. while the chip is active
- Certain areas of the device can be reconfigured while others remain unaffected
- In practice, partial configuration is used to achieve run-time **dynamic reconfiguration**
  - Xilinx Virtex families
  - Altera FPGAs
  - Atmel, etc.
Partial Configuration styles

- **Module based**: distinct portions of the design (modules) that can be reconfigured separately (Bus Macros)
  - independent modules
  - communicating modules
- **Difference based**: making small design changes in local areas e.g. LUTs, block RAMs, but not routing
Virtex II Architecture

- CLBs, Block RAMs, Config Columns
Module-based Partial Configuration
Bus macros

- Connecting reconfigurable and fixed modules in partial configuration maps
Major Constraints

- Size and position of a module cannot be changed.
- Modules can communicate only with neighbors.
- No global signals are allowed except clocks.
- I/O blocks exclusively accessed by adjacent modules.
Difference-based Partial Configuration

• Small changes on the FPGA configuration
• Manually done, usually via an FPGA Editor
• What can be modified?
  – LUTs equations
  – BRAM contents and BRAM write modes
  – I/O standards and pull-ups or pull-downs on external pins
  – Flipflop initialization and reset values,
• What cannot be modified?
  – Routing, very dangerous: internal contentions
Self Reconfiguration

• A second way for dynamic reconfiguration:
• The chip modifies its own configuration based on peripheral or internal signals
• This may occur
  – under command, or
  – autonomously
• This idea leads to the concept of Self Reconfiguration
Why Self Reconfiguration?

- Ability to operate autonomously in remote, challenging and hostile environments
- Perform on-board processing and communication
- Variability of function and operation modes
- Quick response to changing ambiance
Potential Self Reconfiguration Apps

- Space exploration probes
- Military & commercial satellites
- UAVs and μ UAVs
- Deep underwater rescue
- Nuclear or chemical plants
- Autonomous robotics
Key Issues of Self Reconfigurable architectures

- Autonomy
- Real time response
- Low power and energy consumption
- Reliability
About UAVs: Rationale

• Increasing need for flexible embedded processing on board a variety of aerial vehicles especially μUAVs.

• To perform their mission, μUAVs need unconventional on board processing capabilities:
  – Performing multitude of computationally intensive functions
  – Operating autonomously, adapting from one input to another
  – Meeting low power and reliability requirements
Rationale (cont)

- Reconfigurable processors based of FPGAs have two traits: *flexibility* and *parallel processing*.
- However, FPGAs lack autonomous adaptation capability while suffering from power consumption.
- Clearly, mini aerial vehicles, e.g. satellite sensors and μUAVs, need autonomous, adaptable and dynamically reconfigurable processors, beyond conventional FPGAs.
Sensor Web Scenario

Communication Tradeoffs
Bandwidth = Buffer/Latency, Data Rate, Protocol, Error Bit Rate
Self Reconfiguration Approach

- Novel autonomous, adaptable and self reconfigurable system has been proposed consisting of 2 basic units:
  - Adaptation software manager
  - Dynamic reconfigurable hardware fabric
- The approach is based on the twofold concept: adaptation of the application software coupled with dynamic reconfiguration of the hardware.
Approach (cont)

• Architecture: reconfigurable at four Layers:
  - Layer 4: the Adaptation Manager.
  - Layer 3: the Real-Time Operating System RTOS.
  - Layer 2: the Embedded Processors and Memory.
  - Layer 1: the Reconfigurable Hardware Fabric.
Reconfiguration Strategy

• Occurs at several levels:
  – Selection of application modules by the Adaptation Manager.
  – Mapping of modules into the hardware fabric or the embedded processors, depending on performance requirements.
  – Configuration of the hardware fabric and the embedded processor to meet performance and data delivery requirements.

• The reconfigurable hardware is essential for mapping of communications algorithms such as:
  – IR filtering,
  – multichannel CDMA,
  – complex encoding,
  – advanced imaging.
Adaptation manager

- The adaptation manager captures real time sensor inputs and interacts with the Function Libraries.
- The Libraries store pre-built configurations for application functions
- The manager decides on which configuration to be fed into the hardware fabric.
- The manager also involves a software learning process to adapt configuration decisions.
Self Adaptation - Dynamic Configuration

Self Adaptation

Function
Libraries

Dynamic Configuration

Adaptation Manager

RTOS

Reconfigurable Hardware

I/O Sensors
Reconfigurable Fabric

• The reconfigurable fabric consists of a number of processing tiles each having capability of dynamic reconfiguration.

• Tiles are equipped with regularly structured functional units capable of operation level parallelism.

• Tiles can be hierarchically assembled at several levels using dynamically interconnected switch-buffer matrices.
  – Distributed buffer memory

• Configuration can be achieved within a tile, and along several interconnected tiles.

• This approach provides good scalability, growth and fault tolerance.
Reconfigurable Fabrics and Tiles
Reconfigurable Tiles
Core Switch Matrix
Self Reconfigurable and Evolvable Systems

• There is an overlap between the two concepts
• Self reconfiguration operates in real time
• Evolvable reconfiguration implies self-growth and replication of the reconfigurable hardware at slower pace.
• Evolvable hardware use bio-inspired approaches and may need technologies not based on CMOS.
Evolvable Hardware

- Evolvable Hardware, EHW, is capable of on-line adaptation.

- EHW can change its architecture and behavior dynamically and autonomously, either through software or by directly morphing the hardware.

- At present, EHW use evolutionary algorithms or genetic algorithms as their main adaptive mechanism. However, other techniques are possible such as Neural Networks.
“Design” goal: survival
Evolution in nature has lead to species highly adapted to their environment: adaptation ensured survival.

The most fit individuals survive becoming parents; children inherit parents characteristics, with some variations, and may perform better, increasing the level of adaptation.

Design goal: meet system specifications
Same evolutionary principles can be applied to machines.

Potential designs compete; the best ones are slightly modified to search for even more suitable solutions.

Accelerated evolution, ~seconds for electronics
Evolvable Hardware Classes

- **Extrinsic EHW**: simulates evolution by software and only loads the best configuration to hardware in each generation.
- **Intrinsic EWH**: simulates evolution directly in hardware.
- Most evolution approaches are extrinsic or off-line types
Evolutionary design and adaptation of circuits

**Evolutionary Algorithm**
- Search on a population of chromosomes
  - select the best designs from a population
  - reproduce with variation
  - iterate till goal is reached.

**Chromosomes**
- 1011001101
- 0111010110
- 1101101101

**Conversion to a circuit description**
- Models of circuits
- Control bitstrings

**Simulators** (e.g., SPICE)
- Extrinsic

**Circuit response**
- Intrinsic evolution

**Target response**
- Evaluate responses, assess fitness

**Reconfigurable hardware**

Potential electronic designs/implementations compete; the best ones are slightly modified to search for even more suitable solutions.
Evolutionary design: extrinsic - intrinsic

Path from chromosome to behavior data file

- Parameters
- Model
- Simulator
- Data file
- Configuration
- Reconfigurable HW
- Stimulus
- HW evaluator testing equipment
- Data file

extrinsic

intrinsic
Genetic Evolutionary Operations

- Selection
- Crossover
- Mutation
- Use an Objective or Fitness function
Principles of Evolution

- Coding solutions as chromosomes.
- Operating on code, not solutions.
- A string is a candidate solution.
Evolutionary implementation

- Current approaches to EHW implementation:
  - use powerful compute engines to run GAs for evolution
  - use reconfigurable HW or FPGAs to load evolved HW
- Requires:
  - fast evaluation
  - low cost for failure
- Future: everything should be seamlessly integrated in HW
Where is Hardware Evolution?

- Many learning algorithms are inspired by nature
  - GA/GP
  - ANNs
  - Immune Systems
  - Ant Colony Optimisation

- Nature also inspires hardware designers
  - Spiking ANNs
  - Fault tolerance
  - Design Optimisation
Some examples: evolving an FPGA design

- A circuit can be evolved using a GA
- The chromosome is the bitstream
- Each individual is evaluated in 2 steps:
  1. Configure FPGA with bitstream/chromosome
  2. Test configured FPGA by applying all possible input combinations, using output for fitness
Is it practical?

- For most practical real world problems, human designers plus tools still outperform evolution
- However,
- Hardware evolution does have some niche applications
Adaptive Systems

- Evolution + Reconfigurable Hardware = Real-time Adaptation
- Can adapt autonomously to changes in environment
- Useful when real-time manual control not possible
  - E.g. spacecraft systems (sensor processing)
- Non-critical systems are more suitable
  - E.g. data compression systems
  - plant power management
  - ATM cell scheduling
Traditional vs. Evolutionary Search

- Traditional design decomposes from the top down into known sub-problems
- Applies constraints to ensure design behaves like known sub-problems
- Evolution works from the bottom up
- Evolution uses fitness to guide performance
- Not directed by prior knowledge
- Oblivious to complexities of the interactions within the circuit
Innovative Circuits

- Circuits that could not be found using traditional design abstractions are innovative
- Solution may have high performance
- May use less gates than traditional designs
- Analysis shows internal non-digital behaviour
- Examples: evolvable multiplier, adder
Traditional vs Evolvable Multiplier

Traditional = 26 gates

Evolved = 21 gates + 7 MUXes
Application Examples of custom EHW

Japan Higuchi EHW-chips
Industrial, specific

JPL’98 FPTA-0
Research, general

JPL’2001 FPTA-2
Integrated 64 cells (each 44 programmable transistors)

Germany (Heidelberg)
Array of 16x16 programmable transistor cells

UK Edinburgh Palmo

Boards MUX-based
UK Sussex (Evolvable motherboard)

Brazil -PAMA
Evolved Antennas

DEvAn Reconfigurable antenna based on EvAn’s grid antenna
- Same layout except that its perimeter is closed with switches
- 48 switches vs EvAn’s 30
- ~1/5 scale of EvAn antenna
Inspiration
NN seek biological inspiration for
  • computational elements,
  • architecture
  • mechanisms
for certain problems where biology does well (and attempts beyond)

EHW seeks biological inspiration for methodology leading to designs (1,2) appropriate to situations/application
  1. of various types of HW
  2. freeing from biological constraints

Building block
  • NN: Simplified/distorted models of biological neuron
  • EHW: Domain oriented reconfigurable cell

Mechanisms
EHW vs. Self Reconfigurable, again

- Key issue: **real time** efficiency
- Self reconfigurable hardware requires fast responses whereas Evolvable HW is still slow paced
- Combining the two is important for future applications
Layered approach to EHW/Self Reconfigurable

• Key feature: interaction and coordination of two basic entities
  - evolvable adaptation software
  - dynamic reconfigurable hardware

• We could have used an evolution-based approach to design both units, i.e. the manager and the fabric. However,

• An Evolvable Hardware fabric would ultimately require unconventional hardware, not yet available.

• The Evolvable Manager uses a software approach based on Neural Net learning technique to evaluate and perform adaptation of application functions
Self Adaptation - Dynamic Configuration

Self Adaptation

Function
Libraries

Adaptation Manager

RTOS

Dynamic Configuration

Reconfigurable Hardware

I/O Sensors
Evolvable Adaptation Model

• Evolvable hardware model consists of two interacting components
  – dynamic reconfigurable hardware and
  – a neural network
• The idea is to achieve evolution in the hardware by evolving configuration candidates via the neural network and testing them for fitness.
Evolvable Platform

Neural Network

Reconfigurable Hardware

Evolvable Hardware
Evolution Modes

- **Operation mode**: Neural Net (NN) generates configuration code.
- **Training mode**: NN incrementally evolves configurations by training itself on input stimuli as well as configuration data that are recurrently applied after being improved by genetic operations.
- Other evolution modes e.g. self-diagnosis and self repair are also feasible.
Training

- During training, candidate configurations are selected from a population via genetic operations.
- Training continues until a candidate passes a fitness test depending on responses from the reconfigurable fabric.
- Training may start on command or autonomously, in new environment, new functions or upgrading for better performance.
- A major aspect of this scheme is to design a robust training mechanism for configuration evolution of the dynamic reconfigurable fabric.
Evolvable Hardware Training

Input

Neural Network

preliminary Configuration

Fitness test

Final Configuration

Configuration data

Genetic Algorithm
Evolution Training
Summary and the Future

• Self reconfigurable and evolvable systems have the potential to be an important future technology especially for avionics and space infrastructure.

• EHW based on bio-inspired paradigm using GAs and software simulation off-line to evolve and discover hardware.

• This is fine in slow growth and self paced evolution but not in real time.

• In the future, there is need to integrate seamlessly Evolvable software with Neural Network techniques into dynamic reconfigurable hardware platforms.
Embedded Reconfigurable Processing for µUAV Applications

Part II (a) - Information Flow

Chris Papachristou
Case Western Reserve University
Cleveland, Ohio 44106
cap2@case.edu
Outline

• Motivation
• System Concept and Methodology
• Architecture
• Inquiry Processing and Query Processing
• Training
• Conclusions
Motivation: Autonomous UAVs

- UAV scenarios
  - Civilian and Military applications
  - Threat assessment, rescue & recovery, reconnaissance

- UAV real time info flow
  - Queries and Inquiries to UAVs
  - Sensory signal processing
  - Feature processing
  - UAV Response feedback
  - Networking UAV
Autonomous UAV Scheme

- **Master UAV**
- **Slave UAV**
- **Central Station**
- **Target Area**

Connections:
- Broadcast
- Query/Response
- Sensory data

Flow:
- Master UAV communicates with the Central Station and Slave UAVs.
- Slave UAVs exchange sensory data among themselves and with the Master UAV.
- The Central Station interacts with the Master UAV.
Autonomous UAV Real Time Requirements

- Queries and Inquiries to UAVs
- Autonomous & Hierarchical Cognitive Learning
- Image processing
- Master/Slave UAV organization
- Networking formations and UAV collaboration
- Mission strategies
System Concept (cont)

- The system architecture and methodology manages the real-time information flow between CC and the Master UAV.

- A key property is the adaptation and learning capability of the Master in order to respond intelligently to CC queries.
System Architecture

- A central command control (CC) communications center interacts in real time with a Adaptable Mobile Agent (AMA) on the Master UAV.
  - The Master collects sensory information.
  - CC evaluates feedback, accepting or modifying it.
  - CC resends query and updates its database.
System Architecture Information Flow

- CC prepares queries for the Master
- Master processes queries
  - knowledge base
  - sensory data
- CC evaluates feedback, accepting or modifying it
- CC resends inquiry and updates its database
System Architecture information flow

- System Architecture processes the information flow for an inquiry session between CC and the Master.
- Queries operations may involve:
  - Processing & analyzing “existing” knowledge
  - “Augmenting” the knowledge base
System Architecture information flow (cont)

- Adaptive Query Learning:
  - Due to rejections or otherwise unsatisfaction of CC's evaluation of result
- Incremental Sensory Adaption & Learning:
  - Preloaded Master with sensory knowledge (i.e. High altitude video)
  - Independently of its current mission, the Master is updating its sensory
Inquires

• A key element of the CC is an inquiry processor which transforms user inquiries into formal queries for the AMA.

• An inquiry consists of a number of phrases that resemble a restricted natural language specifically

• Consists of an <action>, <qualifiers> and a single <object>.
Inquires: Feature Qualifiers

- For example, “find preferred in Ohio landing area”
  
  where
  
  <action> is “find”;
  
  <qualifiers> are “preferred” & “in Ohio” and <object> is “area”.

- Feature Qualifiers are characterized by a particular trait which exhibits a fuzzy description such as “good” or “preferred”.

- For example, “preferred” has a method describing the human meaning into low level terrain sensory features, e.g. “clearance”, “roughness”, etc.
Prolog

- Declarative Language
  - Declarative Clause Grammars, DCG
  - Straight forward mapping to parallel hardware technologies

- Al-based goal searching and Pattern Matching
  - Image processing and object recognition
  - Optical technologies

- Formulate logical database queries
  - Natural-like language processing
  - Biophotonics technologies
  - Associative memories
Queries

- A formal query is a symbolic expression which can be described by Prolog's Declarative Clause Grammar (DCG).
- For example, the CC Inquiry
  - find preferred in Ohio landing_area
- becomes transformed the AMA query fro the Master UAV:
  - find ( clearance > 7 and roughness > 8) and in Ohio landing_area
- Feature qualifiers require offline and online learning by using a combination of supervised learning (i.e. Neural Nets) and fuzzy system descriptions.
Feature Training Strategies

• Hierarchical training modules
  – Sensory features on distinct sensor modules on Master UAV
  – Preference training module

• Two levels of Learning
  – Adaptive feature learning
  – Preference-based learning

• Inquiry: find preferred in Ohio landing_area
• Query: find ( clearance > 7 and roughness > 8)
Master UAV architecture

- Adaptive recognizer of patterns and images through sensors
  - For Features & Preferences
- Reconfigurable processor
- On board Library of training functions
- I/O interfaces and sensors
Hierarchical training modules
- Training sensory features on distinct sensor modules on UAVs
- Preference training module
Training Strategy

(a) Master Training Strategy

(b) Central Station Training Strategy
Training strategies

• Master:
  – Filtering for each parameter data type
  – Parameter sensory training based on query preference
  – Trained data collected into one feature vector

• Central station (CC):
  – Similar to Master but operating on simulated data
  – Software filtering
  – Feature vector data at the Central derived by simulations using the training knowledge from the Master.
Query processing: Master

sensory trainer is stored in the local knowledge base
query execution process real-time data from the local sensors.
Query processing: Slave

Slave UAVs organized into fleets to collect multiple data
Collaborative trainer: resolving inconsistencies from multiple UAV data.
Key issues in Master (AMA)

• Associative memories and processors is an enabler information technology

• All Master subsystems will benefit from large associative memories

• Command and Control, CC
  – Inquiry system – Prolog engine
  – Preference Learning – Neural Network Classifier

• Adaptable Mobile Agents, AMA
  – Query learning – Supervised Learning, Fuzzy Logic
  – Cognitive Image Recognizer – Unsupervised Learning
  – Sensory Learning – Supervised Learning
Key Attributes in UAV Scheme

- All UAV subsystems will benefit from large biophotonic associative memories, specifically
  - Command and Control
    - Query system – Prolog engine
    - Preference Learning – Neural Network Classifier
  - Master UAV
    - Cognitive Image Recognizer
    - Sensory Learning
  - Slave UAVs
    - Advanced imaging – wavelets
    - Collaborative learning
- Associative memories and processors are enablers for information technology on UAVs
Example: terrain landing
# Feature properties

<table>
<thead>
<tr>
<th>Features</th>
<th>Min Range</th>
<th>Max Range</th>
<th>Sensory Data</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obstacle clearance</td>
<td>1</td>
<td>9</td>
<td>Learning</td>
<td>Yes</td>
</tr>
<tr>
<td>Roughness</td>
<td>1</td>
<td>9</td>
<td>Learning</td>
<td>Yes</td>
</tr>
<tr>
<td>Ground hardness</td>
<td>1</td>
<td>9</td>
<td>Learning</td>
<td>Yes</td>
</tr>
<tr>
<td>Inclination</td>
<td>$0^0$</td>
<td>$30^0$</td>
<td>Reading</td>
<td>No</td>
</tr>
<tr>
<td>Wind speed</td>
<td>0 mil/hr</td>
<td>54 mil/hr</td>
<td>Reading</td>
<td>No</td>
</tr>
<tr>
<td>Temperature</td>
<td>$0^0$ F</td>
<td>$150^0$ F</td>
<td>Reading</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
<th>Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obstacle clearance</td>
<td>limited</td>
</tr>
<tr>
<td></td>
<td>accessible</td>
</tr>
<tr>
<td></td>
<td>clear</td>
</tr>
<tr>
<td>Roughness</td>
<td>fine</td>
</tr>
<tr>
<td></td>
<td>coarse</td>
</tr>
<tr>
<td></td>
<td>rough</td>
</tr>
<tr>
<td>Ground hardness</td>
<td>hard</td>
</tr>
<tr>
<td></td>
<td>firm</td>
</tr>
<tr>
<td></td>
<td>soft</td>
</tr>
<tr>
<td>Inclination</td>
<td>leveled</td>
</tr>
<tr>
<td></td>
<td>tilted</td>
</tr>
<tr>
<td>Wind speed</td>
<td>still</td>
</tr>
<tr>
<td></td>
<td>low</td>
</tr>
<tr>
<td></td>
<td>high</td>
</tr>
<tr>
<td>Temperature</td>
<td>low</td>
</tr>
<tr>
<td></td>
<td>high</td>
</tr>
</tbody>
</table>
## Cluster Characterization

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>accessible</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>limited</td>
<td>limited</td>
<td>accessible</td>
<td>accessible</td>
</tr>
<tr>
<td>2</td>
<td>clear</td>
<td>limited</td>
<td>x</td>
<td>limited</td>
<td>x</td>
<td>limited</td>
<td>clear</td>
<td>accessible</td>
</tr>
<tr>
<td>3</td>
<td>limited</td>
<td>clear</td>
<td>clear</td>
<td>clear</td>
<td>clear</td>
<td>accessible</td>
<td>limited</td>
<td>accessible</td>
</tr>
<tr>
<td>4</td>
<td>limited</td>
<td>limited</td>
<td>accessible</td>
<td>clear</td>
<td>clear</td>
<td>accessible</td>
<td>limited</td>
<td>clear</td>
</tr>
<tr>
<td>5</td>
<td>limited</td>
<td>limited</td>
<td>limited</td>
<td>clear</td>
<td>clear</td>
<td>limited</td>
<td>accessible</td>
<td>clear</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>good</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>poor</td>
<td>ok</td>
<td>ok</td>
</tr>
<tr>
<td>2</td>
<td>good</td>
<td>good</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>poor</td>
<td>ok</td>
<td>poor</td>
</tr>
<tr>
<td>3</td>
<td>poor</td>
<td>good</td>
<td>good</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>poor</td>
<td>ok</td>
</tr>
<tr>
<td>4</td>
<td>poor</td>
<td>poor</td>
<td>poor</td>
<td>good</td>
<td>good</td>
<td>ok</td>
<td>poor</td>
<td>ok</td>
</tr>
<tr>
<td>5</td>
<td>poor</td>
<td>poor</td>
<td>poor</td>
<td>good</td>
<td>good</td>
<td>poor</td>
<td>ok</td>
<td>ok</td>
</tr>
</tbody>
</table>
## Query examples and answers

<table>
<thead>
<tr>
<th>Possible queries</th>
<th>Local Station answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Find all possible sites for landing.</td>
<td>All grids which are not ‘x’.</td>
</tr>
<tr>
<td>2. Find all good sites for landing.</td>
<td>(1,1)(1,2)(2,2)(2,3)(3,3)(4,4)(5,4)(4,5)(5,5)</td>
</tr>
<tr>
<td>3. Find the best site for landing.</td>
<td>(3,3)</td>
</tr>
<tr>
<td>4. Find the closest site for landing from a given location</td>
<td>The system needs to know the current location of the vehicle, altitude, and other information for flight landing plan</td>
</tr>
<tr>
<td>5. Find the best site for landing within a given length of time</td>
<td>The system needs to know the current location of the vehicle, altitude, and other information for flight landing plan</td>
</tr>
</tbody>
</table>
Conclusions

• System Architecture and methodology to manage the massive information flow and inquires between CC and AMA in realtime.

• Bio-inspired learning techniques are needed for query and inquiry processing for feature qualifiers and preferences.

• Bio-inspired learning are needed for pattern recognition of sensory information.

• Prolog, optical and biophotonic technologies are needed for processing realtime massive information flow.
Embedded Reconfigurable Processing for µUAV Applications

Part II (b) – Cognitive Processing Approach

Chris Papachristou
Case Western Reserve University
Cleveland, Ohio 44106
cap2@case.edu
Outline

- Mission scenario
- Heterogeneous sensor nets
- Characteristics of platforms
- Information management
- Our Approach
  - On board UAV architecture
  - Adaptive processing
  - Evolutionary learning and training
  - Associative cognition
Three coordinating sensory networks: ground sensors, UAVs, trooper sensors
**Expeditionary Operation**

- Three distributed sensory networks
  - ground sensors
  - UAV sensors
  - trooper sensors
- The trooper sensory is normally operating in passive mode, i.e. avoiding transmissions while receiving data from UAVs and ground sensors.
- UAV sensors coordinate with the ground sensors to track information about the target. This information is transmitted to troopers.
- The ground sensory consists of redundant heterogeneous sensors that are dispersed en masse to monitor targets.
Ground Sensors

- Sensors are heterogeneous, redundant and disposable.
- They are self-organized by their monitoring threat identity types:
  - Motion
  - Sound
  - Imaging (infrared)
  - Proximity, location
  - Chemical, bio, radiation
- Ground sensors communicate point-to-point with other sensors
- Ground sensors normally operate in passive monitoring mode. They are activated by the UAVs for transmission.
UAV Sensors

- UAV sensors are equipped with long-range communication devices. They respond to ground sensory, troopers and other UAVs.
- UAVs may be organized in hierarchical network formations, i.e. master UAVs and lower flying mini UAVs.
- Possible intelligent information and threat discovery by UAVs
  - Threat identity
  - Threat coordination
  - Overall threat assessment
  - Threat pattern tracking
- Information bridge between troopers, ground sensors and distant command station.
Trooper Sensors

- Trooper sensors are carried on soldiers to retrieve information normally from the UAVs, occasionally the ground sensors and in emergency the command center.

- Characteristics
  - Support information retrieval and interpretation
  - Support coordination among trooper sensors
  - Passive sensory: mostly receiving
  - Threat avoiding and/or safe threat practice for safety of troopers
Data Gathering Principles

• Troopers gather data from their own sensors and from nearby sensory assets, i.e. UAVs and ground sensors. Sensory data is relayed all the way from assets located close to the target.

• Gathering of sensory data is determined by transmission rate, transmission range, quantity, quality, energy and real time constraints.

• There is priority of selecting sensory data types (for example, audio vs. visual) based on mission objectives, threat level, etc.

• Ground sensors transmit raw data with small data rates

• UAVs can transmit processed data that may have been analyzed by the UAV systems or at the Command Station.
Sensor Suite

• Depends on mission requirements

• **expeditionary missions** to discover hidden hostiles under cover and slow moving targets
  – UAV sensors: visual, audio, infrared
  – Ground sensors: motion, chemical, possibly sonar

• **reconnaissance missions** to passively gathering data
  – UAV sensors: long-range visual, infrared, radar
  – Ground sensors: audio, possibly visual

• **surveillance missions** to monitor behavior of people, objects, or processes in large region
  – UAV sensors: visual, infrared
  – Ground sensors: motion, audio, possibly visual
Platform characteristics improvement

- **UAVs**: onboard processing and communication capabilities. Adaptive hardware and software to the mission objectives. Associative processing to enable real-time identification and recognition.

- **Ground sensors**: Minimal computation and communication. Very low energy consumption, possibly energy scavenging.

- **Troopers**: Low power processing and passive communication capability.
Validation and experimentation

- There is need to collect real time data in simulated scenarios that closely relate to real scenarios.
- All key actors in an expeditionary scenario, the troopers, the UAVs and the ground sensors, should interplay to collect real data.
- Data capture capability for post mission analysis.
Real time information management

- **Troopers**: management of sensory data received by their own sensors and ground sensors. This involves *prioritization* of sensories and *weighting* their responses.
  - For example, a motion signal from a ground sensor may reinforce an infrared image from the trooper sensor to decide about a target.

- **UAVs**: More sophisticated data management and analysis in real time.
  - imaging
  - sensory pattern recognition
  - adaptive pattern training

- **Ground sensory**
  - Low level data recording
  - Raw data transmitted with low data rates
Sensory inter-operation

• A key issue of sensory inter-operation is *how* and *where* to evaluate sensory data in real time.

• We propose that the initial evaluation be done by the UAVs. Patterns of diverse sensory data could be weighted to provide a real time response.

• Sensory patterns could be recognized by adaptation and (off-line) training based on data from previous missions. (More details to follow).

• All unresolved information would be sent to command center for expert decisions with time delay penalty
Approach: UAV Centric

UAV Scheme to Assist Trooper Operations
Hierarchical System Concept

• Operation level 1: Local evaluation
  - UAVs collect sensory information from their sensors and the ground sensors concerning the target
  - UAVs process and evaluate the data in real time to locate and identify the target and threat level
  - UAVs transmit a response to the troopers and/or command center

• Operation level 2: Remote analysis
  - For deeper analysis, UAVs communicate with the central command in real time. A feedback received is transmitted to the troopers.

• A key property is the adaptation and learning capability of the UAVs based on associative processing to provide responses in real time.
**UAV on-board architecture**

- Onboard associative storage of training functions
- Adaptive recognizer of patterns and images through sensors
- Reconfigurable processor
- I/O interfaces and sensors
- Associative memory technology can be applied to
  - Sensory Processing
  - Adaptive Recognizer
Key Attributes of UAV-Centric Scheme

- All UAV subsystems as well as command center will benefit from large associative memories, specifically
  - **UAVs:** efficient real time response
    - Cognitive imaging
    - Sensory learning
    - Collaborative training
  - **Command:** computation intensive
    - Preference learning – Neural Classifiers
    - Query system
- Associative memories and reconfigurable processors are enablers for sensory and cognitive technology on UAVs.
Expeditionary Application
On-board UAV Operations
Our approach builds on several on-board function modules to assist the troopers and central command by preprocessing information, correlate corrected data, and preliminary association with existing knowledge.

These three tasks undergo evolutionary learning processes during online operation, or in offline training sessions.
Offline UAV training process

- Hierarchical training modules
  - Training sensory features on distinct sensor modules in the UAV
  - Preferential training module
Online/Offline Evolutionary Learning

• Evolutionary neural process learning employs a neural network training and mapping approach between input and output representations via an evolutionary learning algorithm.

• Evolutionary learning supports online training during realtime application and offline training during traditional training sessions.

• Evolutionary learning augments UAV functions as follows:
  - Adaptive data processing
  - Multi sensory information management
  - Associative recognition
Our learning and training scheme employs a three phase associative memory process:

- Sensory data transformations
- Interfusion of sensory transforms
- Associations and approximate recall

All three phases use evolutionary neural network processing to perform their internals.
Adaptive Sensory Data Pre-processing

- Heterogeneous sensors pre-process raw data using distinct adaptive modules on the UAV
- High level features of information are extracted from raw sensory data, i.e. Fourier, wavelet
- Other information can be preferential rather than traditional math-based. This is achieved with evolutionary preferential neural process learning.
Phase 1 - Sensory preprocessing

a) pre-processing filters,

b) high level feature extractors,

Feature vector $f_k$ represents data of Sensor-$k$
Multi Sensory Information Fusion

- Extracted high level information from sensors are merged into a single vector stream.
- Merging uses statistical parameter, e.g. mean, weighted average, etc.
- Merging can be *biased* with existing knowledge rather than standard unbiased statistical parameters. This is achieved with evolutionary bias neural process learning.
Phase 2 - Interfusion

Inputs
Feature vectors

Output
Signature vector representing sensory data

\[ g_k = \mathcal{N}(b_k; w_k) \]

\[ b_k = (f_{1k} \ldots f_{sk}) \]
Phase 3: Associative Recognition

- Associative recognizers measure statistical similarity or difference between merged information and stored associative memory data.
- Similarity measures determine the association matching levels between the merged information and stored data.
- Often in practice, similarity measures in real association are not mathematically well defined. This is achieved with evolutionary undefined neural process learning.
Evolutionary Training Scheme

- key advantage
  - neural net structural evolution
  - training data enrichment
- The training scheme consists of three nested processes
  - a) coefficient training process (inner loop),
  - b) structural evolution process (middle loop) and
  - c) training data enrichment process (outer loop),
Training

- The coefficient training process (inner loop) is a variation of the back propagation algorithm using adaptive learning rates to accelerate convergence.
- The structural evolution process (middle loop) evolves the net structure to improve accuracy.
- The training data enrichment process (outer loop) explores and enhances the input-output training data space efficiently. This helps to further improve the output accuracy and convergence.
Data Enrichment Process

- **Properties**
  - Addresses the multi-label and imbalanced data problem.
  - Manipulates the imbalanced data by sub-sampling into more balanced data.
  - Iteratively updates the sub-sampled data through the training.
  - Improves the generalization performance of the training net.

- **Differences**
  - Multi-label & imbalanced data problem addressed by ONE net.
  - Unique data selection technique to improve classifier training.
  - Avoids sparsely data distribution assumption.
  - No prior knowledge required.
Enrichment Initialization

- Enrichment Initialization re-samples the available imbalanced training data to create a subset of more balanced training data for first time neural net classifier training.

- Steps
  - Select a number of initial clusters, g
  - Cluster training data into g clusters
  - Select equal number of data from each cluster as active training set
Enrichment Initialization

Total of 30 selected points
Enrichment Update

- Enrichment Update incrementally adds and removes training data to/from the active training set at the end of an enrichment training iteration.

- Steps
  - Train neural net with the active training set.
  - Separate the active training set into 2 groups with respect to their errors.
  - Add neighbors for high-group and remove half of low-group.
Enrichment Termination

- Enrichment Termination controls the enrichment process to iterate for a pre-set number of enrichment training iterations.

- Steps
  - Check if the enrichment process repeats for a pre-set number of iterations.
  - If NO, repeat Step 2. Otherwise, stop.
Experimentation: Robot arm controller

Left: sampled data with inputs (Angle, Gain) and expected output (Internal State, i.e. rotate left, rotate right, idle).

Right: same sampled data with three distinguished expected states, illustrating multi-label imbalanced data problem.
Metrics

- **True-Positive**: classification result is confirmed as correct known label.
- **False-Positive**: classification result is miss-classified as different known labels.
- **Control Accuracy** (Ac): the percentage of accepted classified state in the sampled data.

\[ Ac = \frac{\text{accepted}}{\text{all}} \times 100\% \]

- **Control Error** (Ee): the Euclidean distance between estimated control state and the sampled state in training data.

\[ Ee = \sqrt{\sum_{j} (t_j - o_j)^2} \]
Advantages

- Data Enrichment supports neural network classifier in both linearly separable and non-linearly separable cases.
- Data Enrichment improves the neural network training yielding better trained network than conventional techniques.
- Data Enrichment reduces the time required to train a neural network for multi-label and imbalanced data.
Some Results

- Training repeats for 4 updates
- Step1: Enrichment Initialization
- Step2: Enrichment Update
- Step3: Enrichment Termination

<table>
<thead>
<tr>
<th>Iterations</th>
<th>Accuracy</th>
<th>Energy Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27.00%</td>
<td>2.0759</td>
</tr>
<tr>
<td>2</td>
<td>87.00%</td>
<td>0.4353</td>
</tr>
<tr>
<td>3</td>
<td>89.00%</td>
<td>0.3958</td>
</tr>
<tr>
<td>4</td>
<td>98.00%</td>
<td>0.3104</td>
</tr>
</tbody>
</table>

**TABLE I**
Non-Linear Data Example
Status

- Theoretical model of the three-phase sensory cognitive processing scheme for training and operational mode
- Evolutionary Training of the cognitive processor including self-modification of the NN structure and augmentation of additional training data
  - simulation for NN coefficient weights training
  - simulation for the NN structural modification
  - simulation of data enrichment process
- Needs:
  - Controlled scenario for testing and verification of the cognitive processing scheme with realistic information
  - We will move the simulation to FPGA prototype
Assumptions on Existing Technologies

- We assume the following technologies already exist or are in process of development
  - Communication and transmission technologies portable for short- and long-range
  - Power and energy efficient devices for all sensory units
  - Troopers equipped with sensor technologies
  - Networking technologies for effective sensor communication and processing
  - small, reliable UAVs
Applications to UAV Missions

• Consider a reconnaissance mission
• Sensory transform phase maybe embedded in low flying mini UAVs to collect and process diverse sensory data
• Another UAV, possibly a master at higher altitude, may perform data interfusion
• Depending on the application mission, the master UAV may also perform data associations and matching in real time. An associative memory should be included in the master UAV
• However, if associations sizes are large, the master UAV may contact the central command which can perform intensive associations for matching