High Performance Physical Environmental Security using Distributed Cooperative Sensor Nodes

Hamid Reza Naji
International Center for Science, High Technology & Environmental Sciences, Iran
Kerman Graduate University of Technology, Kerman, Iran

ABSTRACT

Fusion of cooperative intelligent sensor nodes in a distributed environment can provide a high performance event monitoring system mainly for security issues. Partitioning the hardware design space into entities called agents, which are autonomous units of execution that have the capability of interacting with the environment and each other has been made much more attractive by the recent advances in the capabilities of reconfigurable hardware. In a reconfigurable embedded processing environment one possible benefit is use of multi-agent approach in a common design methodology for both the hardware and software components of the system to do parallel processing with high speed and flexibility. In this paper we will explore the arguments for applying multi-agent techniques to highlight how such techniques can be applied using current-generation hardware description languages in reconfigurable hardware to be suitable for physical environmental security.

Keywords: Security, Reconfigurable Systems, Multi-Agent, Sensor Fusion

1. INTRODUCTION

A system recognizes its external environment by using sensory information. By integrating large amounts of this information, more reliable and multilateral recognition can be achieved. The purpose of sensor fusion is to realize a new sensing architecture by integrating multi-sensor information in a hierarchical and decentralized configuration. This allows the system to provide a higher-level recognition mechanism than traditional systems. In sensor fusion systems, the probabilistic data from multiple sensors is fused together using calibrated error models of the sensors involved, in order to reduce the ambiguity in the sensed data and improve the overall reliability of the system [1],[2],[3].

It should be noted that current technology usually mandates that hardware agents be less complex in structure and more deterministic in operation than those typically implemented in software. In order to facilitate their implementation, as hardware agents within reprogrammable logic, the sets of beliefs, desires, and intentions may be reduced to the minimum set necessary to provide adequate though sometimes sub-optimal understanding and control of the system. In general, the simpler, more regularly structured agents that have highly repetitive operations are good candidates for implementation in reconfigurable hardware [4],[5].

Agents implemented in reconfigurable hardware will most likely be less complex and finer grained than their software agent counterparts and will communicate much more effectively with their neighboring agents because communication is likely to be performed on-chip. Agents may communicate with one another in a tightly synchronized manner using a common global clock, or communication can occur in a less synchronized manner using distributed clocks and a standard handshaking protocol [6],[7].

2. SENSOR FUSION SYSTEMS

Many automated systems have sensors to collect information and conditions of the world around them. They use this information as feedback to control modules. The ability of the system to perform a given task is dependent on the quality of the sensory information which is in turn related to the range and resolution of the sensors. However, sensors themselves are not perfectly reliable. Sensors are subject to noise, calibration errors and non-linearity, and thus sometimes give incorrect results.

Sensor fusion seeks to overcome these drawbacks by integrating or combining information from two or more independent sensor readings. It can be seen that combining readings from several different kinds of sensors can reduce uncertainty and provide significantly more accurate information than the reading of data from a single sensor.

It is presupposed that the intelligent sensor agents can communicate with each other, thus enabling them to fuse the information they acquired with a high speed of information processing and flexible fusion algorithm [8],[9],[10],[11],[12].

3. SECURITY USING COOPERATIVE AGENTS

This section illustrates how cooperative hardware agents can provide a reliable high speed physical environmental security. These hardware agents could be implemented in reconfigurable hardware. Although a conventional embedded processor approach could be used in this application the ideas presented here would be applicable to a system where the greater speed of the hardware agents might be a significant advantage. This system could either be implemented in reconfigurable hardware or in an ASIC. Although an ASIC implementation could result in greater speed it would freeze the design and allow for no further changes.

This sensor fusion application will use homogeneous hardware agents resident in re-configurable hardware. No agents implemented in software will be used. Also, each agent will exhibit the autonomy and sociability that are main characteristics of agents. Consider a grid of cells each of which must be protected from some hazardous event. Consider also a sensor system that is capable of detecting the event either in the cell it is placed in or any of the eight adjacent cells. For this application a n x n grid of cells will be considered (n cells in each row or column of grid). The cells with the sensor units will contain two independent sensor units for fault tolerance and the necessary hardware to implement the agent. Let us also assume that each cell with an agent has the capability to provide a necessary protection for example extinguish a fire in its cell or neighboring cells. Figure1 and 2 show two models of sensor fusion by multi-agent system. The cells with and without agents can be arranged as shown in Figure1 or Figure2, where the black circles indicate cells with agents and the white circles represent cells with no agents.

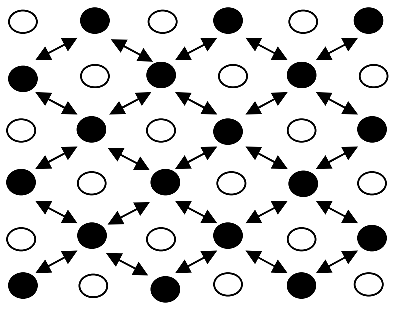
Figure 1 *NxN* grid with $N^2/2$ agents

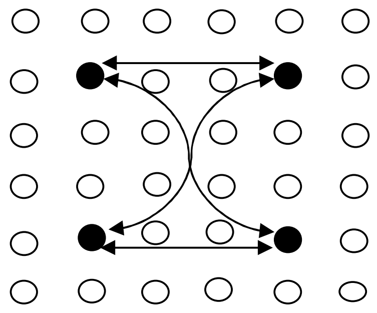
Figure 2 *NxN* grid with $\lfloor N/2 \rfloor + 1$ agents

The double-sided arrows indicate communication paths from agent to agent. Note that any individual agent using only the sensors local in its cell will not have enough information to locate the event. Each agent must communicate with its neighbors to receive more sensor information in order to locate the cell an event has occurred in. Also, an individual agent will have no way to determine if the sensor it is using has developed a fault. Again, by considering sensor data from its neighbors the agent can form a hypothesis as to whether or not its sensor has failed. It can also switch to the redundant sensor and compare the sensor data prior to and after the switch to attempt to confirm or deny this hypothesis. Thus, no agent by itself has sufficient data to implement its mission. Rather the agents must communicate with one another to accumulate sufficient data to make intelligent decisions on the appropriate actions to take. However, each agent retains full autonomy to act on the data received from its own sensor and the data communicated to it by its neighbors to switch its sensor or activate one of its two event reactor such as fire suppression systems. The system does not function as a master slave environment but as a community of peers sharing information so that each agent can build an independent view of the environment.

The communication links among agents are not described in detail for the example. However, it is assumed that each channel is capable of bi-directional communication and that all of the packets transmitted from agent to agent are a fixed size and format. The data transmitted from agent to agent is simply the latest sensor data from the sensor in the transmitting agents cell. All of the sensors are binary sensors. Also, it is assumed that some sort of time-out is implemented so a failed communication can be detected. The data communication path will contain a Boolean flag that will indicate whether or not the last data transmission over that channel was successful. Also, a signal will be generated when new data is available from neighboring agents. An agent will assume that a failed communication indicates that there is no agent in the corresponding position. Even if such an agent exists if the communication link has failed no data will be available from that agent and the end effect will be the same as having no neighbor agent there.

3.1 system's Architecture

The hardware agents for the sample application are implemented as a Beliefs, Desires and Intentions architecture. The beliefs, desires and intentions of each agent are stored as a bit vector of fixed length. A sample beliefs set of the agents for a fire extinguishing system is summarized below as a vector. Each belief is encoded as a single bit that will be one if the agent currently holds the associated belief and zero if the agent does not hold the belief. Remember, these represent beliefs of individual agents and may or may not correspond to objective reality.

A(0..3) - Upper left/ Upper right/ Lower left/ Lower right neighbor does not exist;

A(4) - Sensor 0 is currently used sensor; A(5) - Sensor questionable; A(6) - Sensor 0 has failed; A(7) -Sensor 1 has failed; A(8) -Sensor just switched; A(9) - Previous sensor value was high; B(10) - Left neighbor cell is on fire; B(11) - Right neighbor cell is on fire

The belief bit vector is designed such that a value of all zeroes is the default power-up/reset set of beliefs for an agent. The first four beliefs relate to which neighbors the agent believes to be present. Failure to communicate with a particular neighbor may lead the agent to the belief that the neighbor does not exist. The next seven beliefs encode the agent's beliefs about its redundant sensor system. If a sensor is deemed questionable it may be switched for the redundant sensor. If the sensor data immediately changes it will be considered failed. The initial values indicate that both sensors are operational and sensor 0 is currently being used. The final three beliefs maintain the agents beliefs about where fires exist. The agent is responsible only for extinguishing fires in two cells but is unsure whether the right or left neighbor is its responsibility. Hence, a belief is maintained for each of the three cells concerning the fire status of those cells. The inputs to the agent come from its local sensor and from the communication with the neighboring agents. The inputs to each agent are shown below:

In(0) - Local sensor data (1 indicates fire in cell or adjacent cell);

In(1,3,5,7) - Upper left/ Upper right/ Lower left/ Lower right neighbor sensor data;

In(2,4,6,8) - Upper left/ Upper right/ Lower left/ Lower right neighbor communication error;

The local sensor is a binary sensor that produces a single bit output. The output is one if a fire is detected in the agent's cell or any of the eight adjacent cells. The output is zero if no fire is detected. The input from the other agents each contains two bits of information. The first bit is the latest sensor data bit from that neighbor's sensor. The second bit is a communication error bit. The communication error bit is one if the communication system detected a time-out attempting to communicate with the neighbor. If the error bit is set the sensor data bit is not reliable. In the agent architecture a function exists which maps the current set of beliefs and the inputs to the agent to a new set of beliefs. Since both the inputs and the beliefs for this example are encoded as bit vectors this function can be implemented in combinatorial logic. The set of desires for the agent are also encoded as a bit vector. A sample set of desires for the agent is outlined below.

D(0) - Extinguish fire in current cell ; D(1) - Extinguish fire in left hand neighbor cell;

D(2) - Extinguish fire in right hand neighbor cell ; D(3) - Switch to backup sensor;

The architecture includes a function to map the current state of beliefs to desires. This function is very simple for this example since the desires map directly from the beliefs that a cell is on fire or a belief that a sensor has failed. Finally, the intentions of the architecture must be laid out. A sample set of intentions are listed below:

I(0) -Activate fire suppressant system in agent's cell; I(1) -Activate fire suppressant system in left neighbor cell; I(2) -Activate fire suppressant system in right neighbor cell; I(3) -Switch active sensor;

Note that although each agent may uses plans to activate a fire suppressant system in either the left neighbor cell or the right neighbor cell it will in fact only be connected to one of the two cells. Hence, it is possible for an agent to develop a goal and in fact a plan that it is not capable of bringing about by itself. It may require the cooperation of another agent in the system that is capable of implementing the plan or a portion of it. The function that maps goals to plans is also simple. The goal to extinguish fires or switch sensors maps directly to a plan to do so. The function that implements the plans may have an effect on the beliefs. For example, the function that implements the plan to switch the sensor also updates the belief vector so the agent believes that the sensor has just been switched.

3.2. System's Features

The system is comprised entirely of agents resident in re-configurable hardware and contains no software agents. It should be noted, though that these hardware agents show many of the features associated with traditional software agents. Each hardware agent is autonomous and forms its own decisions on actions to take based upon its beliefs about the environment, its sensor data, and data received from other agents in the system. Furthermore, each of these hardware agents is social and communicates with other agents in the system to achieve a common goal of protecting all of the cells from fire. This is necessary because each agent has incomplete information on the environment and cannot locate a fire without information from other agents. Also, each agent has limited capabilities and is only able to extinguish fires in three cells. This incomplete data and restricted capabilities is another common characteristic of agent-based systems. The decentralized data and control and asynchronous operation of the agents are also typical of agent-based systems.

4. AGENT MODEL

The architecture used for the hardware agents is derived from the well-known Beliefs, Desires, and Intentions architecture that is described extensively in the literature[]. In the Beliefs, Desires, and Intentions architecture each agent maintains a set of Beliefs, a set of Desires, and a set of Intentions. The set of beliefs indicates what the agent currently believes to be true concerning its environment. Note that what an agent believes to be true may or may not be in fact true. The set of desires is a set of outcomes that the agent would like to cause in its environment. Note that the agent may or may not be able to bring its desires about. Bringing its desires about may require action from other agents or may not be achievable at all. Finally, the set of intentions is a set of actions that the agent intends to take to attempt to bring about its desires.

The agent includes a function to map its inputs and current set of beliefs to updated sets of beliefs. Also, the agent has a function to map its current set of beliefs to a set of desires. Finally, the agent has a function that maps its set of desires to the set of intentions that are to be invoked to bring about these desires. It should be noted that current technology usually mandates that hardware agents be less complex in structure and more deterministic in operation than those typically implemented in software. In this case, the sets of beliefs, desires, and intentions may be reduced to the minimum set necessary to provide adequate but sub-optimal understanding and control of the system. In general, the simpler, more regularly structured agents that have highly repetitive operations are good candidates for implementation in reconfigurable hardware [13], [14], [15], [16].

In this paper, it is assumed that a typical reconfigurable hardware architecture will be employed. In such an architecture, the functionality of the reconfigurable hardware will be controlled by placing design information directly into each bank of the configuration memory. In this way, the external environment has the capability to change the hardware's functionality dynamically or during the creation of an application by introducing agents into the appropriate area of configuration memory. In this architecture, the reconfigurable logic is assumed to support partial reconfiguration in that it is assumed that parts of its logic can be changed without affecting other parts. Interaction with the external environment is supported by I/O connections made directly to the reconfigurable logic [17], [18].

Each hardware agent has control ports that include input/output signals for handshaking among different agents and also with the main system, input/output ports for communication with the environment's sensors and actuators and input/output ports for communicating with other agents that may be either hardware agents or software agents.

5. IMPLEMENTATION

Figure 3 shows a part of the HDL code illustrating the specification of the data elements to store the Beliefs, Desires, and Intentions.

```
architecture behavioral of HW_AGENT
is
  -- define signals for Beliefs,
  -- Desires, and Intentions
signal BELIEFS:BIT_VECTOR(0to11);
signal DESIRES:BIT_VECTOR(0to3);
signal INTENTIONS:BIT_VECTOR(0to3);
```

Figure 3. Beliefs, Desires, and Intentions

The beliefs of the agents are binary values that encode whether or not the agent believes that its' neighbor's exist, that its' sensor is functioning, and it or its' neighbor cells have a fire. The inputs to the agent come from its local sensor and from the communication with the neighboring agents (indicated by LOCAL_SENSOR and DATA_AVAILABLE in the sensitivity list of the process in Figure 4). The inputs to each agent include sensor data from each of four neighbor agents. In the agent architecture a VHDL process exists that maps the current set of beliefs and the inputs to the agent to a new set of beliefs. Figure 4 shows a portion of this process illustrating the updating of some of the Beliefs.

The set of desires for the agent include desires to have fires in its own cell or an adjacent cell extinguished. The architecture includes a VHDL process to map the current state of beliefs to desires.

```
-- This process fires when new local
-- sensor data or new data from neighbor
-- agents is available
F : process (LOCAL_SENSOR, DATA_AVAILABLE)
  begin
  --update beliefs of neighboring nodes
  BELIEFS(0) <= NEIGHBOR_DATA(1);
  BELIEFS(1) <= NEIGHBOR_DATA(3);
  BELIEFS(2) <= NEIGHBOR_DATA(5);
  BELIEFS(3) <= NEIGHBOR_DATA(7);
     ⋮
 -- update belief about fire in cells
 -- Update belief about fire in cell
  BELIEFS(11) <= LOCAL_SENSOR and
 (NEIGHBOR_DATA(0) or BELIEFS(0)) and
 (NEIGHBOR_DATA(2) or BELIEFS(1)) and
 (NEIGHBOR_DATA(4) or BELIEFS(2)) and
 (NEIGHBOR_DATA(6) or BELIEFS(3));
     ⋮
  end
```

Figure 4. A process to map sensor data and communication data to new beliefs

This process is very simple for this example since the goals map directly from the beliefs that a cell is on fire or a belief that a sensor has failed. A part of the VHDL process is shown in figure 5.

```
G : process (BELIEFS)
  begin
     if (BELIEFS(11) = '1')
then
          DESIRES(0) <=  '1';
     end if;
   :
   end
```

Figure 5. A process to map beliefs to desires

The intentions of the agent include activating a fire suppressant system in a cell and switching to its redundant sensor. A part of the VHDL process to map desires to intentions is shown in figure 6.

```
-- process to map DESIRES to
INTENTIONS
H : process (DESIRES)
begin
  INTENTIONS(0) <= DESIRES(0);
  INTENTIONS(1) <= DESIRES(1);
  :
end
```

Figure 6. A process to map desires to intentions

5. RESULTS

Tables 1 and 2 show the result of timing and device utilization for agents without filter, and with filter in 8 and 16 bit modes for a similar application. We use filter to remove unwanted signals and noises.

Table 1. Timing results

Agent Type	8 bit	16 bit
Agents without Filter	100 ns	165 ns
Agents with Filter	140 ns	270 ns

Table 2. Device utilization
(Device: Xilinx Virtex II-2v1000fg)

Agent Type	Without Filter		With Filter	
	8-bit	16-bit	8-bit	16-bit
% of used Slices	24%	64%	33%	95%
% of used Flip Flops	2%	4%	3%	4%
% of used LUTs	11%	62%	24%	83%
% of used IOBs	19%	39%	19%	39%
% of used MULT	90%	90%	90%	90%
% of used GCLKs	6%	6%	6%	6%

As the results show the complexity of agents increases with filter. In 8 and 16 bit modes agents are implemented on Xilinx Virtex II-2v1000fg.

6. CONCLUSIONS

In this paper a sample application was provided which used multiple hardware agents that were designed to be implemented in reconfigurable hardware for a sample sensor fusion type application. The hardware agents developed for this application display many of the features associated with more traditional agents implemented in software. Each hardware agent was created to be autonomous and forms its own decisions on actions to take based upon its beliefs about the environment, its sensor data, and data received from other agents in the system. It is believed that the use of hardware agent's may prove useful to a number of application domains, where speed, flexibility, and evolutionary design goals are important issues. Future research should focus upon how to best apply the hardware agent paradigm to more complex forms of hardware agents and on applying such techniques in a unified way to hybrid hardware/software environments.

7. ACKNOWLEDGMENTS

This research is sponsored by International Center for Science, High Technology & Environmental Sciences, Iran.

8. REFERENCES

[1] Roger, A. Dash, R. K. Jennings, “Computational mechanism design for information fusion within sensor networks,” 9th Int. Conf. on Info. Fusion, pp. 1- 7, 2006

[2] Ruben Stranders, N. J., Alex Rogers, “A Decentralized, On-line Coordination Mechanism for Monitoring Spatial Phenomena with Mobile Sensors,” In AAMAS Workshop on Agent Technology for Sensor Networks, pp. 9-15, 2008

[3] Ruben Stranders, N. J., Alex Rogers, “A Decentralized, On-line Coordination Mechanism for Monitoring Spatial Phenomena with Mobile Sensors,” In AAMAS Workshop on Agent Technology for Sensor Networks, pp. 9-15, 2008

[4] Guccione, S.A., “*Reconfigurable computing at Xilinx*, “Proceedings of Euro micro Symposium on Digital Systems Design, pp. 102-107, 2001

[5] Becker, J.; Pionteck, T.; Glesner, M. ,“*Adaptive systems-on-chip: architectures, technologies and applications,*” 14th Symposium on Integrated Circuits and Systems, 2001

[6] Birigit Burmeister, Afsaneh Haddadi,” *Generic, Configurable, Cooperation Protocols for multi Agent systems,*” LNAI 957, 5th european Workshop on modeling Autonomous agents in multi agent Sorld, MAAMAW, 1993

[7] M.A.Hale,and J.I Craig ,“*Preliminary Development of Agent Technologies for a Design Integration Framework* ,” Proceedings of 5 th symposium on multi disciplinary Analysis and Optimization, Panama City, FL,USA, 1994

[8] Rogers, A., Corkill, D., and Jennings, N. R., "Agent technologies for sensor networks,” IEEE Intelligent Systems, Vol. 24, pp. 13-17, 2009

[9] Tham, C.-K. and Renaud, J. ,”Multi-agent systems on sensor networks: A distributed reinforcement learning approach,” ISSNIP, pp. 423-429, 2005

[10] Rogers, A., Osborne, M., Ramchurn, S. D., Roberts, S., and Jennings, N. R. , “Information agents for pervasive sensor networks,” Per Com, IEEE Computer Society, pp. 294-299, 2008

[11] Rogers, A., Osborne, M., Ramchurn, S. D., Roberts, S., and Jennings, N. R. ,”Information agents for pervasive sensor networks,” Per Com, IEEE Computer Society, pp. 294-299, 2008

[12] Ferranti, E. and Trigoni, N. ,"Practical issues in deploying mobile agents to explore a sensor instrumented environment," Technical Report RR-09- 02. Oxford University Computing Laboratory, 2009

[13] Weiss, Gerhard editor, "*Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence*," MIT Press, Cambridge Massachusetts, 1999

[14] Vidal, Jose M., Bohlor Paul A., Huhns, Michael N, "*Inside an Agent*," IEEE Internet Computing, 2001

[15] Kautz H., Selman B., Shah M., "*Combining Social networks and collaborative filtering,*" Communications of the ACM, Vol. 40, pp. 63-65, 1997

[16] Hayzelden, Alex L. G., Bigham, John editors, "*Software Agents for Future Communications Systems*, Springer-Verlag, 1999

[17] Hamid Reza Naji , John Weir , B. Earl Wells, "Applying the multi-agent paradigm to reconfigurable hardware: a sensor fusion example," Second international workshop on Intelligent systems design and application, Atlanta, pp.207-212, 2002

[18] Mohamed Khalgui, Olfa Mosbahi, Zhiwu Li, Hans- Michael Hanisch, "Reconfigurable Multi-Agent Embedded Control Systems: From Modelling to Implementation," IEEE Transactions on Computers, 2010.

AUTHORS PROFILE

HamidReza Naji is an assistant professor in the International Center for Science and High Technology & Environmental Sciences and Dean of the College of Electrical and Computer Engineering in Graduate University of Technology, Kerman, Iran. His research interests include embedded, reconfigurable, and multi-agent systems, networks, and security. Naji has a PhD in computer engineering from the University of Alabama in Huntsville, USA. He is a professional member of the IEEE. Contact him at hamidnaji@ieee.org