# Context-Sensitive Object Request Broker for Ubiquitous Computing Environments

Stephen S. Yau and Fariaz Karim
Computer Science and Engineering Department
Arizona State University
Tempe, AZ 85287, USA
{yau, karim}@asu.edu

## Abstract

Context-awareness is an important property of applications in ubiquitous computing and communication (ubicomp) environments. Context-sensitive applications use various data from the surrounding environments and the devices to adapt their behavior. Using context as a criterion to transparently establish communication links among devices requires effective system support. A context-sensitive middleware is effective to address these emerging applications since such a middleware provides both development and runtime supports to context-sensitive applications in addition to providing the interoperability in heterogeneous ubicomp environments. This implies that the object request broker (ORB) of the middleware must provide a mechanism to provide context-sensitive communication among objects. In this paper, the principles and software-hardware co-design aspect of a context-sensitive ORB is presented to facilitate context-sensitive communication among application objects in ubicomp environments.

**Keywords:** Ubiquitous computing environment, reconfigurable context-sensitive middleware, object request broker, mobile ad hoc networks, FPGA, Bluetooth, software-hardware co-design.

## 1. Introduction

One of the important aspects in future distributed computing is the environments consisting of numerous wearable, handheld, and embedded devices. Rapid growth in inexpensive, short range, and low-power wireless communication hardware and network standards are now enabling the construction of ubiquitous computing and communication (ubicomp) [1] environments. Ubicomp environments enhance the computing and communication capabilities of human by integrating embedded computers with various environments and daily lives, thus making computing and communication essentially autonomous and transparent to the users. Computing nodes in ubicomp environments form short range and low power ad hoc networks, whose topologies are usually dynamic due to arbitrary node mobility. Typical applications in ubicomp environments are context-sensitive [2,3], which means applications using various data about the surrounding environment to control their interactions with other devices. Since various contexts depend on external and runtime conditions, context-sensitive communication among devices occurs transparently and dynamically. A middleware to provide the necessary support for such type of communication is appropriate due to interoperability, application development support, and runtime services that are basic functions of a middleware framework.

For ubicomp environments, a middleware architecture should itself be context-sensitive to manage the communication among objects in a transparent fashion. The Object Request Broker (ORB) of a middleware plays an important role, because an ORB is responsible for managing communication channels with remote objects, marshalling data to/from the other devices, and distributing information about the object services to other devices. This implies that the ORB of a context-sensitive middleware must be designed to enable context-sensitive object activation, context-sensitive service discovery, and remote object connection. In addition, reconfigurability, high-efficiency, and small footprint of an ORB are also important factors for designing efficient middleware for embedded devices since ubicomp environments consist of high-density of resource-constrained embedded devices (implying large volume of connection management operations among various devices). Hence, proper design and context-sensitive operation of an ORB is an important issue in the design of ubicomp middleware.

These issues related to ubicomp middleware, and especially context-sensitive ORB have not been fully investigated. Among the related work, TSPACES [4] by IBM focus on a Java-based communication middleware based on the concept of tuple-spaces. The ORB-equivalent component of TSPACES provides asynchronous messaging-based communication facilities without any explicit support for context-awareness. BlueDrekar [5], which is a Bluetooth-based middleware, provides a protocol layer and a set of APIs to enable communication

Proceedings of the Eighth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'01)
1071-0485/01 $17.00 © 2001 IEEE

between two Bluetooth-equipped devices. The LIME approach [6] adopts a coordination perspective based on a tuple space model. The programming model supported by LIME views mobility as transparent changes in the content of the tuple space. LIME also supports context-awareness in a limited sense by adding location-aided tuple update and reaction operations. In [7], QoS specification and adaptation issues of ubicomp middleware are studied. However, none of these approaches use a context-sensitive ORB to provide context-triggered communication among different devices. In this paper, we will present R-ORB, which is a software-hardware co-designed ORB to provide context-sensitive communication in our RCSM [2,3,8], a middleware for ubicomp environments.

## 2. Our Approach

Our approach to designing R-ORB is part of our ongoing research on RCSM (Reconfigurable Context-Sensitive Middleware) [2,3,8], which addresses context-sensitivity for devices in ubicomp environments and mobile ad hoc networks [9]. RCSM is based on a software-hardware co-design approach to construct both reconfigurable and high-performance ubicomp middleware for embedded devices. Its Context-Aware Interface Definition Language (CA-IDL) [2,3] allows application developers to specify context-aware object interfaces by defining different context variables as first class objects. RCSM allows a software-based implementation of application objects while providing hardware-based core middleware services, such as the R-ORB.

Our approach to designing R-ORB can be summarized as follows:

1. Design a context-sensitive communication establishment procedure for ubicomp environments, and identify the goals for the design of R-ORB.
2. Design a context-sensitive service distribution and discovery and object activation mechanism for R-ORB.
3. Apply a software-hardware co-design approach to allocate the components of R-ORB in software and reconfigurable FPGAs.

We will elaborate these steps in Sections 3 to 5. In Section 6, we will discuss the implementation of R-ORB and preliminary performance. We will also augment Section 3 by an illustrative example. Our current design of R-ORB is based on Bluetooth L2CAP protocol [10], and hence we will use Bluetooth networking terminology as needed.

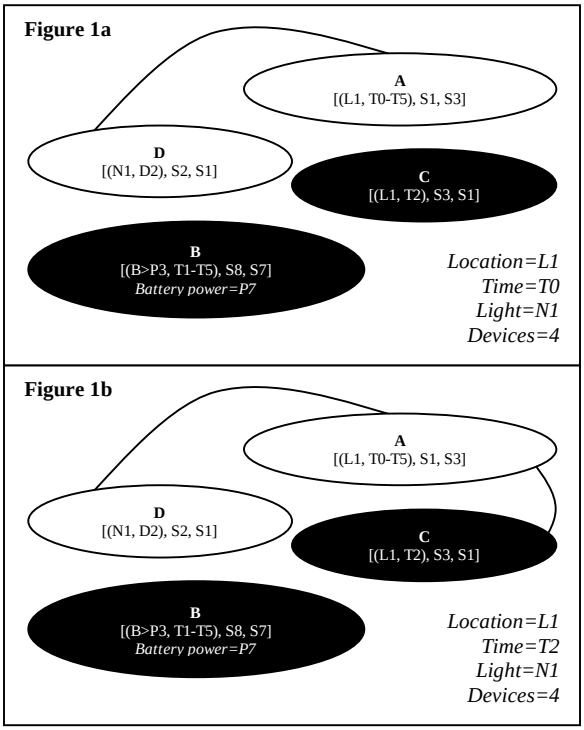


Figure 1: Ad hoc network of nodes A, B, C, and D, forming context-sensitive communication links based on current context.

## 3. Context-Sensitive Communication

R-ORB provides a mechanism to provide context - sensitive communication among the applications in ubicomp environments. As mentioned in [2], this type of communication is initiated based on specific environmental and device data, known as contexts [2,11].

In this section we will present the procedure to establish context-sensitive communication (CSC) and illustrate this procedure through two scenarios in a small ad hoc network of four devices. The following procedure is for a device to establish context-sensitive communication with a remote device:

1. Periodically collect data from the device and its onboard sensors.
2. Analyze the data to determine the suitability of object activation according to the context-aware interface specification [3, 8].
3. If the suitable condition for an object $O_i$ exists, generate a *context-match* event, which is the necessary condition for establishing a context-sensitive connection link with the remote device.

Proceedings of the Eighth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'01)
1071-0485/01 $17.00 © 2001 IEEE

4. Perform inquiries to discover other devices, and exchange service attributes of $O_i$ with remote devices to determine the suitability of exchanging information with objects resident in remote devices.
5. If a suitable matching is found in Step 4, generate an *id-match* event, which is the sufficient condition for establishing a context-sensitive link.
6. Establish a communication channel with the remote devices to exchange information and activate object $O_i$.

To illustrate this procedure, consider the example in Figure 1, which has four devices A, B, C, and D forming an ad hoc network using radio transmitters, such as Bluetooth or IEEE 802.11. A white background indicates that the corresponding node is currently participating in a context-sensitive link. Below, we describe the services and the context-sensing capabilities of each device:

**Device A:** It has the capability to detect the current location and current time. It exchanges information through incoming service S1 and outgoing service S3 whenever it is in location L1 between times T1 and T5, assuming T5 > T1. However, even if A is in L1 at time T2, thus satisfying the necessary condition, it may not be able to establish a context-sensitive link with other devices if no other device in the same ad hoc network provides an outgoing service S1 and/or incoming service S3, in order to satisfy the sufficient condition.
**Device B**: It is activated based on a set of different contextual conditions – the battery power in B must be greater than P3 between the times T1 and T5. When B is activated, it exchanges information through incoming service S3 and outgoing service S5.
**Device C:** It is ready to communicate when it is in location L1 during time T2. Its incoming and outgoing services are S3 and S1 respectively.
**Device D:** It activates incoming service S2 and outgoing service S1 whenever the brightness of light in the surrounding environment is N1 and number of devices in the ad hoc network is 2 (i.e. D2).

**Scenario 1:** As shown in Figure 1a, at time T0, all devices are in location L1, where the brightness is N1. We see that Devices A and D establish a context-sensitive link. Device C does not participate since its desired activation time T2 is currently not true. In case of Device C, the necessary condition is not satisfied. On the other hand, Device D's necessary condition is satisfied, but none of its neighbors (i.e. A, B, and C) have incoming/outgoing services of S8 and S7. So, D's sufficient condition is not satisfied.

**Scenario 2:** As shown in Figure 1b, at time T2, all devices are still in Location L1. Now, C's contextual conditions are satisfied, and hence it establishes a communication channel with A by matching their respective services. Device B remains isolated, since no other device with the required service has joined the ad hoc network.

Note that, the services are implemented as context-aware objects, which specify the desired context values and their relationships using our timed-regular expressions [3], incorporated into CA-IDL.

We can identify the properties for R-ORB as follows: first, devices do not exchange information or activate objects unless the desired context of the application is matched with the current context of the device and the environment. Second, devices through context-sensitive communication links do not need to agree on a common context. Rather, each device satisfies the necessary and sufficient conditions (i.e. *context-match*) based on its own specification using CA-IDL.

R-ORB utilizes the context-sensitivity of the application objects to perform the following operations:

## 4. Service Distribution and Object Activation

Devices in ubicomp environments dynamically form ad hoc networks, which are often short lived. As such, it is impossible for devices to know the services of the neighbor devices a priori. Service distribution and discovery must address multiple issues, which are addressed in Section 4.1. In Section 4.2, we discuss the actual procedure for checking the desired context of a context-aware object and activating the object when the specified condition becomes true.

### 4.1 Service Information Distribution and Discovery

Service information distribution and discovery in ubicomp environments should address multiple issues, such as what information to disseminate, how to disseminate the service information to other devices in an ad hoc network, and finally how often to perform such action. We will address these issues in R-ORB in this section. The following procedure is used in R-ORB distribute and discover services:

**Initialization phase:** Devices form ad hoc networks using underlying network protocols. In Bluetooth, it implies formation of *piconets* and/or *scatternets* [10] to enable nearby nodes to communicate with each other. Also, R-ORB statically stores the service

attributes and interface information of each context-aware object [8] based on the CA-IDL specification as follows:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $S_1$ | $S_2$ | … | $S_n$ | $O_i$ | $N_i$ | $M_1$ | $P_{1a}$ |
| $P_{1b}$ | $P_{1c}$ | … | $P_{1n}$ | $M_2$ | $P_{2a}$ | … | $P_{2n}$ |

Here, $S_1$, $S_2$, etc. are the service attributes of the corresponding context-aware object $O_i$. $N_i$ indicates the number of methods in $O_i$. The method id and the parameters are encoded and represented as $M_i$ and $P_{ia}$, $P_{ib}$, … $P_{in}$ respectively.

**Service information distribution phase:** A service distribution is initiated on demand based on two conditions – i) an object's contextual condition becomes true, which was not true previously, and ii) an R-ORB detects changes in its neighbors due to arrival of new nodes.

**Service discovery phase:** A service discovery procedure is initiated if a context-aware object has the suitable context for activation, but no peer object is yet found for establishing a link between two objects.

R-ORB only distributes the information about the context-aware objects that currently satisfy the necessary condition, which is presented in Section 3.

### 4.2 Context-Aware Object Activation

After each service discovery, the following procedure is used to establish a connection between two remote objects:

i) Assume after the service discovery process, the R-ORBs of two devices detect two objects suitable for communicating with each other.

ii) The R-ORBs in each device determines the suitability of object connection. To accomplish this, we have defined the context-sensitivity of a method in [7]. R-ORB checks if the following condition holds:

$$\left(\lambda^{t}_{p} = 0\right) + \left(CS^{q}_{m} < 1\right) = True \quad , \qquad (1)$$

where $CS^{q}_{m}$ represents the context-sensitivity of the method $m$ to be invoked in object $q$; $\lambda^{t}_{p}$ represents the rate of change of context $p$ at time $t$.

If (1) is satisfied, R-ORB determines the suitability of the connection between two remote objects as follows:

$$R / \lambda^{t}_{p} \geq b_t / b_c, \qquad (2)$$

where $b_t$ represents the total bandwidth required to exchange the object data; $b_c$ represents the current bandwidth of the link; R is defined as follows:

$$\begin{aligned} R &= \left|V^{r}_{p} - V^{c}_{p}\right| \; if \; 0 < \lambda^{t-t^{\alpha}}_{p} \leq \lambda^{t}_{p}, \\ &= \left|V^{l}_{p} - V^{c}_{p}\right| \; if \; 0 < \lambda^{t}_{p} < \lambda^{t-t^{\alpha}}_{p}, \end{aligned} \qquad (3)$$

where $V^{r}_{p}$ and $V^{l}_{p}$ represent the allowed boundary values of the context $p$; $V^{c}_{p}$ represents the current value of context $p$, and $\lambda^{t-t^{\alpha}}_{p}$ represents the rate of context changes in the previous sampling cycle.

## 5. Software-Hardware Co-Design of R-ORB:

### 5.1 Justification for using FPGAs in R-ORB

Unlike other middleware design, R-ORB is based on a software-hardware co-design approach to achieving both reconfigurability and performance desired in ubicomp environments. We use Field Programmable Gate Arrays (FPGA) to design and implement the hardware part of the design. FPGAs are a type of reconfigurable hardware components that can be reprogrammed after fabrication to achieve flexibility and customizability. FPGAs can currently have up to 500,000 gates, and their capacity will continue to increase. Due to the flexibility of reconfiguration, these units have been traditionally used for application-specific customization in communications, military, and intelligence applications that often reside in different embedded devices. Moreover, it is shown that reconfigurable processors are useful in computations that are regular [11]. Examples include signal and protocol processing, encryption and compression operations. This property lead us to choose FPGAs for R-ORB implementation since most ORB-specific operations, such as data marshalling, context-processing connection management, and R-GIOP protocol processing are also regular and require high performance to increase the context-sensitivity [8] of the overall RCSM services. In addition, reconfigurability is also useful in R-ORB, since it is expected that different configurations of R-ORB may be necessary to fit the sensor configuration and constraints of embedded devices.

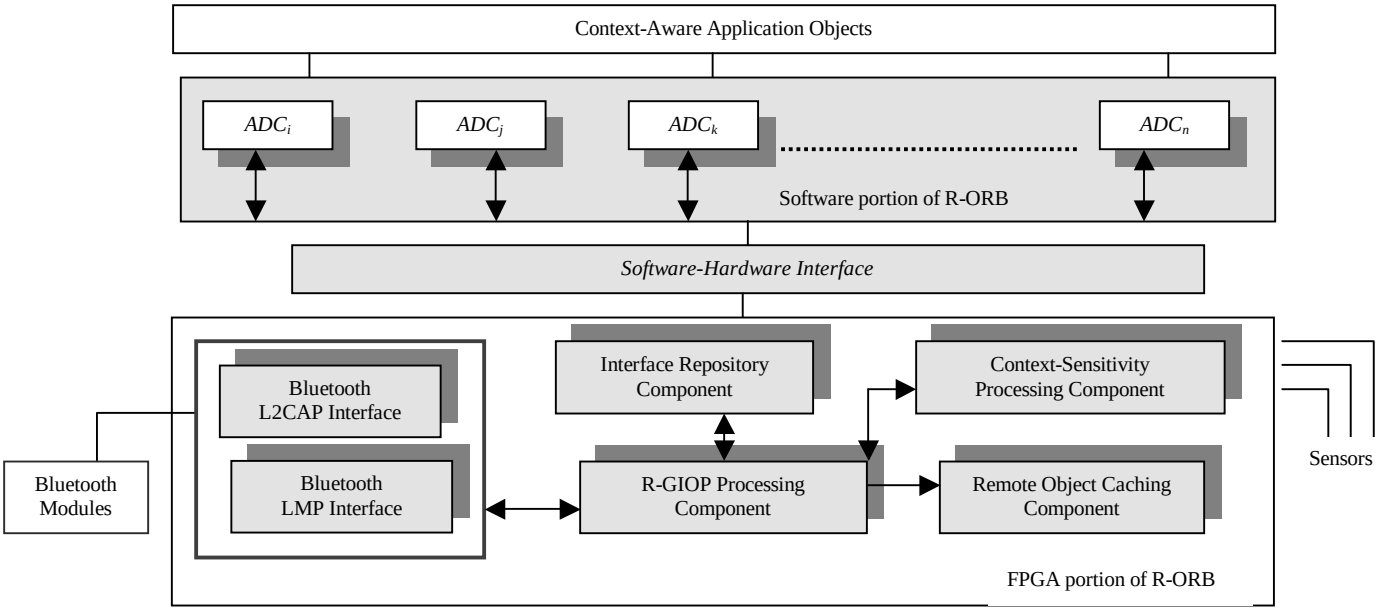


Figure 2: Software-Hardware Co-Design of R-ORB.

**5.2 R-ORB Components**

In this section, we will give an overview of the components in our R-ORB design. The configuration of these components is shown in Figure 2. The arrows in the figure show the directions of data flow among the components. The figure also shows the Bluetooth L2CAP and LMP interfaces that interact with R-ORB.

- R-GIOP processing component: This component is responsible for establishing communication links with the remote objects. It uses our R-GIOP inter-ORB protocol, which is a symmetric protocol (as oppose to CORBA GIOP, which is an asymmetric protocol), and it uses *context-match* and *identity-match* events [3] to initiate object connection. As shown in Figure 2, the R-GIOP component is also responsible for interacting with the modules of Bluetooth radio transmitter. R-GIOP also performs the operations described in Section 3 to establish connection with remote objects.

- Context-sensitivity processing component: This component is responsible for interacting with the sensor hardware and storing both the values of different contexts and their rates of change over a period of time. Periodically, it supplies the context data to the application objects.

- Interface repository component: This component stores the interface and related service attributes of the application objects. In addition, the specified context values of each method are stored.

- Remote object caching component: This component performs as a cache to hold the information about the objects that are currently in contact with the host device. The entries in the cache are updated as other devices lose connection due to mobility.

- Adaptable Object Container (ADC): ADC is responsible for object activation, method invocation, and detecting appropriate contexts. In addition, it is responsible for adapting the context-sensitivity of an object to prioritize the execution of critical methods. In R-ORB architecture, there is a 1-1 correspondence between an ADC and a context-aware object.

The software-hardware co-design approach is based on the trade-off between the costs of making changes in software vs. the performance of hardware components. The current R-ORB configuration is shown in Table 1, where *S/W* and *H/W* represent software and hardware implementation of the

| R-ORB Components | S/W | H/W |
|---|---|---|
| Interface repository component | √ | |
| Adaptable Object Container | √ | |
| Object caching component | | √ |
| Data-marshalling component | | √ |
| Context processing component | | √ |
| R-GIOP processing component | | √ |

Table 1: Software-Hardware partitioning of R-ORB components in our current design.

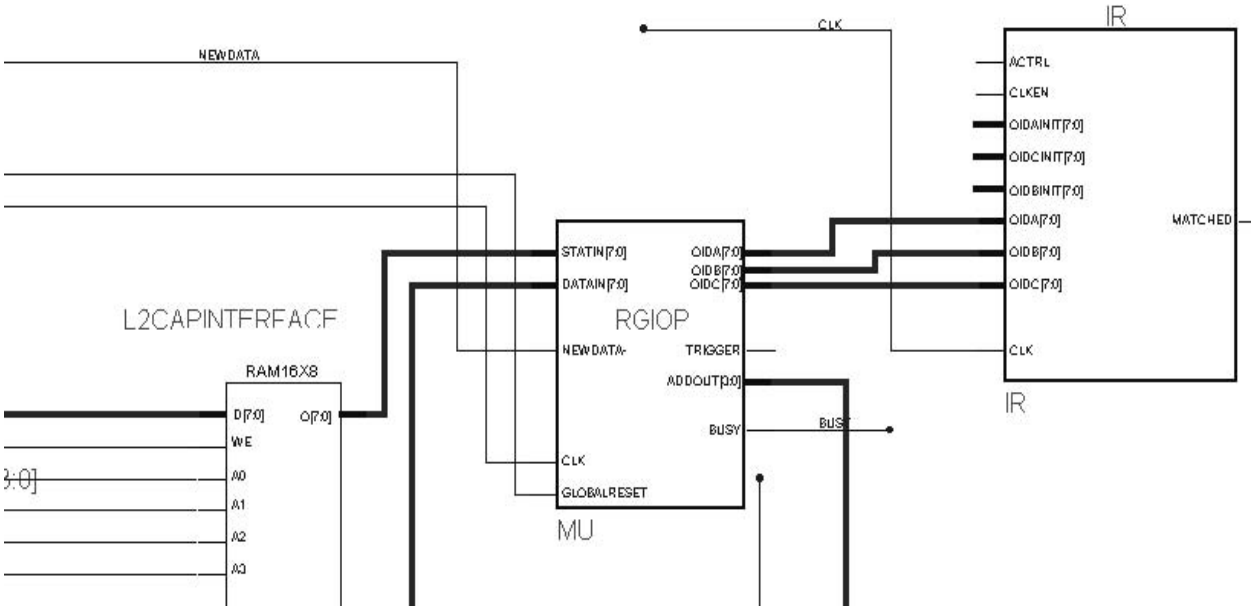


Figure 3: FPGA design of R-ORB showing interface among Bluetooth L2CAP, R-GIOP unit, and IR component.

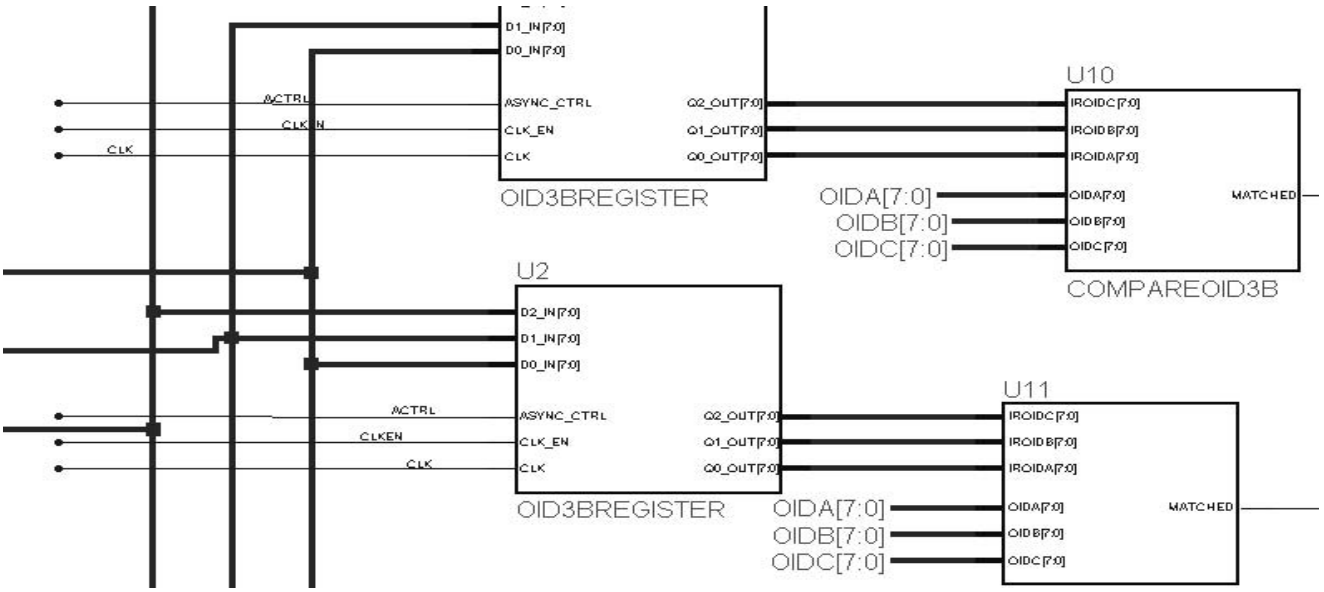


Figure 4: FPGA design of IR component using dedicated registers arrays for efficient object id matching.

components respectively. We derive this configuration by applying the following procedure:

1. Let $L$ be the expected number of *modification* in the application software after the deployment. A *modification* is defined as an activity that causes the interfaces of context-aware objects to be changed and the service attributes to be updated. Let T be the number of such *modifications.*

2. Identify the components of R-ORB that need to be updated *every time* following a *modification* from Step 1. Let these components be $C_i$, $C_j$, $C_k$, etc.

3. Let $C_s^i$ be the cost of updating each component $C_i$, from Step 2, if implemented in software. Let $C_h^i$ be the cost if $C_i$ is implemented in reconfigurable hardware. If (L > 0) and ($C_s^i > C_h^i$), then component $C_i$ is implemented in reconfigurable hardware. Otherwise, $C_i$ is implemented in software.

4. The remaining components, which are not selected in Step b), are implemented in hardware.

## 6. R-ORB Implementation

The software portion of the RCSM includes *ADC* components [8] that are responsible for detecting the appropriate contexts for a specific application and activating/deactivating the application objects.

The hardware part of the R-ORB architecture is divided into several hardware-processing blocks and one memory block. For clarity, Figures 3 and 4 illustrate the relationship among the components (i.e. blocks) at the hardware logic block level. All of these blocks are implemented in a Xilinx 20,000 gate FPGA, which uses a 3.3-volt power supply. The design was initially simulated for functional correctness using Xilinx Foundation 2.1 tool. The maximum achievable clock frequency is almost 250 MHz, which is expected to increase if pipelined architecture is used. In each clock cycle, the current implementation of R-ORB can process almost 1 byte of data processed between 10 and 20 nanoseconds. This implies that the maximum throughput is almost 100 megabytes/second, which far exceeds the current capacity (approx. 730 kilobytes/second) of Bluetooth radio specification [9]. However, the flexible nature of FPGAs allows us to scale down the clock frequency of R-ORB to synchronize with the Bluetooth transmitter. The higher clock frequency of R-ORB can be used to address current (e.g. IEEE 802.11) and future high-speed wireless communication technology.

## 7. Discussion

In this paper, we have presented a context-sensitive R-ORB, which is the principal component of RCSM to establish context-triggered communication among devices in ubicomp environments. The R-GIOP protocol communicates with remote R-ORBs by periodically sensing the context. R-GIOP further checks the benefit of new links by performing a simple analysis based on the rates of changes in context and the available bandwidth of the channel. Currently, we use a Xilinx XC4010XL FPGA to implement the hardware components whose functionality remain unchanged across different applications and devices. The R-GIOP part of R-ORB is designed to communicate with the Bluetooth L2CAP [10] layer. The software parts of R-ORB are responsible for periodically evaluating the current contexts to determine the suitability of method invocations.

Future research in R-ORB includes evaluating the context-triggered communication involving multiple devices forming ephemeral groups. We also plan to investigate the benefits of different modes of R-ORB operation in response to different context. We will also perform further experimentation in both *piconets* and *scatternets* by integrating R-ORB with Bluetooth radio transmitters, and study the correlation, if any, between context-sensitive communication and multihop routing in Bluetooth networks.

## References


[1] M. Weiser, "Some Computer Science Issues in Ubiquitous Computing, *Comm ACM*, Vol. 36, No. 7, pp. 72-84, July 1993.

[2] S. S. Yau and F. Karim, "Reconfigurable Context-Sensitive Middleware for ADS Applications in Mobile Ad Hoc Network Environments", *Proc. 5th IEEE Int'l Symp. Autonomous Decentralized Systems (ISADS 2001)*, Dallas, USA, pp. 319-326, March 2001.

[3] S. S. Yau and F. Karim, "Context-Sensitive Middleware for Real-Time Software in Ubiquitous Computing Environments", *Proc. 4th IEEE Int'l Symp. Object Oriented Real-Time Distributed Computing (ISORC 2001)*, Magdeburg, Germany, pp. 163-170, May 2001.

[4] IBM Research, TSpaces Project, http://www.almaden.ibm.com/cs/TSpaces/.

[5] The BlueDrekar Project, http://www.research.ibm.com/BlueDrekar/.

[6] A. Murphy, G. Picco, and G.-C. Roman, "LIME: A Middleware for Physical and Logical Mobility", *Proc. 21st Int'l Conf. Distributed Computing Systems (ICDCS 2001)*, Phoenix, USA, April 2001, http://www.cs.rochester.edu/u/www/u/ murphy/.

[7] K. Nahrstedt, D. Xu, D. Wichadakul, B. Li, "QoS-Aware Middleware for Ubiquitous Computing", to be published *IEEE Communications*, 2001, http://cairo.cs.uiuc.edu/ papers.html.

[8] S. S. Yau and F. Karim, "Context-Sensitive Distributed Software Development for Ubiquitous Computing Environments", to be published *Proc. 25th IEEE Int'l Computer Software and Applications Conference (COMPSAC 2001)*, Chicago, USA, October 2001.

[9] IETF, Mobile Ad Hoc Networking (MANET) Charter, http://www.ietf.org/html.charters/manet-charter.html.

[10] Bluetooth, "Specification of the Bluetooth System", Core, version 1.0 B, December 1999, http://www.bluetooth.com/.

[11] A. K. Dey, "Understanding and Using Context", to be published *Personal and Ubiquitous Computing*, 5(1), 2001, http://www.cc.gatech.edu/fce/publications.html.

[12] A. DeHon, "The Density Advantage Configurable Computing, *Computer*, Vol. 33, No. 4, April 2000, pp. 41-49.