Wake on LAN over Internet as Web Service System on Chip

Francisco Maciá Pérez[1], Juan A. Gil Martínez-Abarca[1], Héctor Ramos Morillo[1], Diego Marcos Jorquera[1], Francisco José Mora Gimeno[1], Iren Lorenzo Fonseca[2]

[1] Departamento de Tecnología Informática y Computación
Universidad de Alicante
{pmacia, gil, hramos, dmarcos, fjmora}@dtic.ua.es
http://www.dtic.ua.es

[2] Centro de Estudios de Ingeniería y Sistemas
Instituto Superior Politécnico José Antonio Echevarría.
ilorenzo@ceis.cujae.edu.cu

Resumen. En este artículo presentamos un System on Chip (SoC) diseñado para ejecutar un Web Service (WS) concreto en un Application-Specific Integrated Circuit (ASIC). El sistema se ha diseñado sin procesador y sin software y se ha concebido como un patrón de diseño hardware para un fácil desarrollo de servicios de red que se ofrecen como WS bajo un enfoque de Arquitecturas Orientadas a Servicio. Por tanto, el chip no sólo es capaz de actuar como un Proveedor de Servicios SOAP sino que también es capaz, por si sólo, de registrar el servicio en un Servidor de Registro externo mediante el protocolo estándar de publicación UDDI. Esta propuesta se ha denominado WSoC. El objetivo principal de este WSoC es desarrollar dispositivos de red SOA más viables económicamente hablando y con mantenimiento cero. Para validar la propuesta, se ha implementado un prototipo mediante el uso de la tecnología FPGA. En concreto, se ha seleccionado el servicio de red WoL over Internet, que permite que cualquier cliente WS encienda cualquier dispositivo de red compatible con la tecnología WoL. También se ha desarrollado un escenario completo SOA para testear las funciones del prototipo, validando la propuesta.

1 Introducción

La importancia de las Tecnologías de la Información (TI) en todas las áreas de las actividades humanas en el mundo actual es un hecho indiscutible. Cuanto más simples son esas tecnologías para los usuarios finales más complejos se vuelven los sistemas backend que las soportan. Estas tecnologías proporcionan la base tecnológica en la que la mayoría de los procesos de negocio se sustentan. Es extremadamente importante proporcionar los mecanismos necesarios para garantizar que dichas infraestructuras funcionen siempre de manera correcta.

Aunque los servicios de las TI se vuelven cada vez más complejos, tienden a ser sustentados por pequeños servicios, más o menos estandarizados, que llevan a cabo,

Siguiendo una filosofía similar a la aplicada al término SoC, hemos denominado a esta propuesta: WSoC (Web Service on Chip). El objetivo principal del WSoC es la capacidad de desarrollo de dispositivos de red con menor coste y de mantenimiento cero.

Para validar esta propuesta, se ha desarrollado un prototipo usando un la tecnología FPGA. El servicio de red elegido como núcleo del prototipo WSoC es WoL sobre Internet. Este servicio de red permite la gestión de encendidos remotos de nodos en una LAN, desde cualquier lugar mediante el uso de un cliente WS estándar. La incorporación de un servicio de este tipo a nuestra organización o incluso en el hogar es una simple cuestión de conectar el dispositivo adecuado en nuestra red y ,como mucho, asignarle una configuración inicial para que empiece a funcionar.

Se ha desarrollado un escenario SOA completo para validar las funcionalidades del prototipo: publicación del servicio, descubrimiento y consumo, mostrando la viabilidad de la propuesta.

El artículo está organizado de la siguiente manera; en el apartado 2 se presenta una visión del estado actual de las tecnologías y de las últimas investigaciones; en el apartado 3 se muestra una visión general de la propuesta; en el apartado 4 se describe al arquitectura del chip WSoC; en los apartados 5 y 6 se describe el servicio Wolf para validad el enfoque; el apartado 7 muestra la implementación del prototipo, desarrollado con la tecnología FPGA y se propone un escenario de pruebas para llevar a cabo la validación de las funciones del WSoC; y finalmente, en el apartado 8 se resumen las principales conclusiones.

2 Background

Los primeros estándares abiertos que trataron de abordar los problemas de la gestión de las TIC de una manera global fueron SNMP y CMIP [6], propuestos por el IETF; ambos protocolos orientados principalmente a la supervisión y control de la red. Estos modelos de administración presentaban como principal inconveniente su dependencia de la plataforma.

Basándose en ellos y buscando una integración entre sistemas heterogéneos, surgen dos líneas básicas de trabajo: procurar la integración entre sistemas que utilicen el mismo protocolo de gestión de red, como es el caso de [7] y [8] con el uso de CORBA; o, de carácter más ambicioso, proponer un protocolo de gestión de red independiente de las infraestructuras.

Alguna de las propuestas más extendidas en administración de redesson: CORBA/JIDM, especificación del grupo de trabajo JIDM [34] del OMG [35]; CIM/WBEM, propuesta del DMTF [36] usando técnicas orientadas a objetos CIM e interoperación usando HTTP y XML con WBEM; JMX especificación definida por el JCP [37] que define una serie de API's orientadas a Java para la gestión de red; y WS-Management especificación realizada por varias compañías del sector (SUN, INTEL, MS, AMD) para la integración de sistemas de gestión de servicios y recursos basándose en Servicios Web.

El considerable número de tareas relacionadas con la gestión de la red, así como su muy diversa y compleja naturaleza, hace que el mantenimiento de estos sistemas sea costoss para las organizaciones, tanto en términos de recursos y el tiempo y personal.

El uso de sistemas multiagentes para sistemas de gestión de red informática ofrece una serie de características que favorecen la automatización y la autogestión de los procesos de mantenimiento [9] [10]. La creación de proyectos como AgentLink III, la primera en una acción coordinada sobre la base de los agentes financiados por la 6 ª Comisión Europea Programa Marco [11], es un claro indicador del alto grado de interés en la investigación de agentes software.

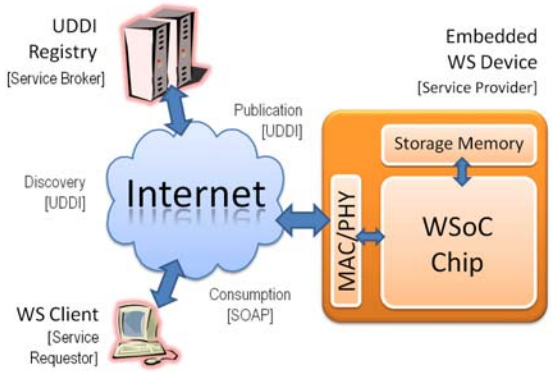

Fig. 1. Escenario SOA basado en tecnologías y estándares WS

Más recientemente, con el desarrollo de tecnologías Web, se han logrado nuevos avances hacia la autogestión, proponiendo modelos autogestionados basados en ontologías como modelos de información y SOA como un modelo operativo, además de proporcionar a los administradores una interfaz ubicua independiente de la plataforma [12].

La diversidad de los actuales modelos de gestión de red demuestra que la necesidad de definir los mecanismos de interactuación entre todos los dominios de gestión involucrados [13]. Esta relación podrá alcanzar un nivel semántico mediante la utilización de ontologías de forma que sea posible trabajar con una vista abstracta de la información de gestión, independiente del modelo usado, y que permitirá a los administradores (personas o software) automatizar las tareas de gestión [14].

En [15], el grupo de operaciones básicas para un Servicio Web realiza una propuesta con el objetivo de estandarizarla dentro de la gestión de redes como contrapartida a la normalización del modelo de información de SNMP bajo XML desarrollado en otros trabajos [16].

Existen diferentes tipos de plataformas embebidas en el mercado que son capaces de soportar un servicio web [5]. Por ejemplo: RabbitCore [2], SHIP [3], o Digi ConnectMe [4]. Estos sistemas, basados generalmente en microprocesadores de 32bits, implementan un servidor Web con un software específico escrito para la pila TCP / IP, todo ello soportado por un sistema operativo embebido. Estas soluciones de bajo coste presentan claras desventajas en términos de rendimiento y responsabilidad, debido a la complejidad del software.

Con el objetivo de ajustar aún más los costes de producción sin sacrificar el rendimiento [17], [18], los sistemas en chip (SoC) también permiten el desarrollo de: servicios de red básicos [19], servicios web [20] y en general las aplicaciones de red [21].

Es precisamente en esta línea en la que se desarrolla nuestra propuesta: el desarrollo de único chip de menor coste y capaz de proporcionar servicios de red autogestionados como Web Services.

3 Descripción de la Propuesta

El principal objetivo de este trabajo es la propuesta de una arquitectura hardware de un chip que, por sí solo, es capaz de proporcionar todo lo necesario para desarrollar dispositivos de red embebidos que ofrecen un servicio de red en forma de un Web service usando un modelo SOA.

3.1 Enfoque SOA

SOA implica algunos elementos adicionales para que el dispositivo proporcione los servicios (ver Fig.1) Por esta razón queremos iniciar la visión global de nuestra propuesta revisando los componentes más importantes:

1. Dispositivo WS Embebido. El Dispositivo WS Embebido ha sido diseñado utilizando el chip WSoC objeto de este trabajo. El dispositivo ejecuta un servicio web y publica su interfaz e información de acceso en el servicio de registro. Cada proveedor debe decidir qué servicios desea exponer, cómo alcanzar el equilibrio entre la seguridad y la fácil disponibilidad, la forma de pago de los servicios, o, si son libres, cómo explotarlos con otro propósito. El proveedor también tiene que decidir en qué categoría de servicios deben estar listados y qué tipo acuerdos comerciales son necesarios para utilizar el servicio. En la terminología de SOA, este dispositivo es conocido como el Proveedor de Servicios.
2. Registro UDDI. Con el fin de garantizar que un cliente y un servidor son verdaderamente independientes, SOA incorpora un Broker Service o Servicio de Registro que contendrá una descripción formal de los diferentes servicios prestados en la red. Esta descripción se hace mediante el Web Service Description Language (WSDL) y contiene información sobre: la dirección de servicio, los parámetros aceptados y la forma de devolver el resultado. En el caso de la tecnología Web, tanto la publicación del servicio como su posterior descubrimiento se hace mediante la especificación "Universal Description Discovery and Integration" (UDDI).
3. Cliente WS. El cliente del servicio Web o service requestor localiza entradas en el service broker utilizando diversas operaciones de búsqueda UDDI, obteniendo la página WSDL en la que el servicio se describe y, a continuación, se enlaza al dispositivo del servicio Web embebido con el fin de invocar uno de sus servicios web mediante el protocolo SOAP.

3.2 Dispositivo WS Embebido

Aunque todos los elementos mencionados son esenciales para el desarrollo de un servicio web totalmente desacoplado bajo el modelo SOA, uno de los principales objetivos de este trabajo es la capacidad de diseño de dispositivos con servicios web embebidos que utilizan el chip WSoC descrito en la sección siguiente.

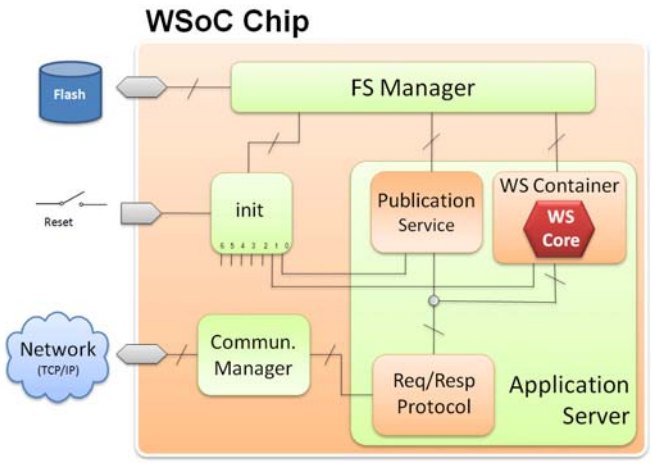

Fig. 2. Arquitectura Web Service on Chip.

Un dispositivo WS embebido consta de una memoria de almacenamiento, una interfaz de red y el chip WSoC (Fig 1). Veámoslo a continuación:

1. Chip WSoC. El WSoC (Web Services on Chip) es el núcleo del dispositivo que proporciona también un marco hardware con todos los elementos funcionales que necesita un servicio web para: registrarse en un registro UDDI y ofrecer sus servicios a un cliente de servicios Web. La funcionalidad, en términos de servicios web de cada WSoC, como se analizará en la siguiente sección, se lleva a cabo en su WS Core.
2. Memoria de Almacenamiento. Una memoria externa de almacenamiento permite almacenar información de trabajo y configuración. La memoria se ha dividido en dos grandes bloques: un espacio para el los archivos de sistema y una zona de usuario. El área del sistema contendrá los archivos del sistema, las hojas WSDL con una descripción de los servicios, archivos para el registro del sistema y otros archivos auxiliares. El área de usuario permite almacenar archivos definidos por el servicio específico implementado en cada chip WSoC.
3. Interfaz de red (MAC/PHY). Un módulo de comunicación con la tarjeta de red (NIC) que proporciona acceso físico a la red TCP / IP.

Por supuesto, este chip WSoC es el componente más importante en este documento, por lo que se describe con mayor detalle en la sección siguiente.

4 Propuesta del Chip WSoC

Una vez expuesta la visión general del sistema, nos podemos centrar en el verdadero objetivo del trabajo: en la arquitectura del chip WSoC, responsable de la provisión de los servicios Web.

Una de las principales características de la propuesta es que no implementa una arquitectura orientada a procesador, SO ni aplicaciones software. Gracias a la potencia de las actuales herramientas de diseño, resulta relativamente fácil plantear soluciones basadas totalmente en bloques arquitectónicos hardware.

Tabla 1. Posibles estados del sistema del Chip WSoC.

Estado	Descripción	Activa Bloque Hardware
0	Estado de Publicación	Service Publication
1	Estado de Servicio	WS Container

A pesar de no existir procesador ni software, el diseño hardware del chip se ha realizado siguiendo una arquitectura similar a la arquitectura software de este tipo de sistemas.

En la fig. 2. se presenta un gráfico con los principales bloques hardware que componen el chip. A continuación analizaremos la funcionalidad de cada uno de estos bloques.

4.1 Init

El modulo init es el primero en tomar el control una vez comienza a actuar el chip. Su función es similar a la que tendría el proceso de igual nombre implementado en un sistema operativo tipo UNIX, siendo el primer proceso que carga y ejecuta el sistema operativo y cuyo identificador de proceso (PID) siempre es 1. Este módulo lee desde la memoria no volátil del dispositivo, a través del modulo de gestión de sistema de archivo, tanto el estado en el que se ha iniciado el chip como el archivo inittab que contiene una descripción de los módulos que el chip debe activar en función del estado en el que se encuentre. Actualmente los estados definidos son dos (ver Tabla I).

En un sistema operativo tipo UNIX en el cual nos hemos basado, el primer proceso (con PID 0), en realidad es un planificador (generalmente el proceso schld) que habilita la multitarea. En nuestro caso, puesto que el actual diseño no contempla esta característica, hemos eliminado dicho proceso de nuestra propuesta.

4.2 FS Manager

El sistema de archivos gestiona el Administrador de I / O del sistema de archivos situado en el almacenamiento de memoria. Permite realizar operaciones de lectura / escritura de archivos en el almacenamiento de memoria. Los archivos pueden ser de distintos tipos (HTML, WSDL, registros, parámetros de usuario, el estado del sistema

o archivos de configuración) en función de la naturaleza del servicio. Para aprovechar mejor el poco espacio de almacenamiento del que se dispone, se plantean versiones optimizadas de cada tipo de archivo. El FS Manager es el responsable de ocultar estos detalles al resto de los módulos.

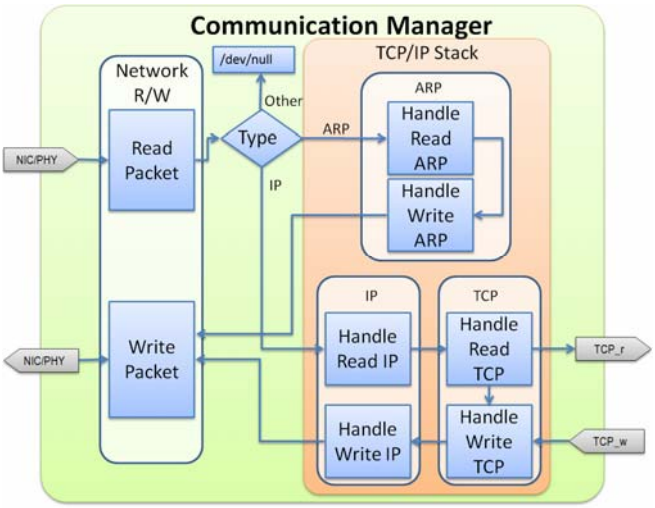

Fig. 3. Diagrama de flujo del bloque Communication Manager.

4.3 Communication Manager

El módulo Gestor de la Comunicación proporciona estándares de comunicación como la pila TCP / IP a través de la MAC / PHY para comunicar la plataforma con otros nodos de la red.

En la Fig. 3 se presenta un gráfico con los principales bloques hardware que componen este módulo junto con el principal flujo de datos entre los mismos. Puesto que no existe procesador ni software explícitos en nuestra propuesta, la conexión física entre los bloques es muy similar al flujo de los datos. Según esto, uno de los puntos de entrada a este bloque se encuentra en Read Packet incluido en el módulo Network R/W. Es el encargado de leer continuamente las tramas de red. En función de su tipo, cada trama es transferida al bloque que debe gestionarla. Si el tipo de trama no puede ser tratada por el chip, ésta se desecha. El bloque ARP gestiona el protocolo de igual nombre, generando, a su vez, las respuestas pertinentes.

El módulo IP gestiona el tráfico IP y, en caso de ser necesario, pasará la trama al bloque Handle Read TCP para continuar con su procesamiento.

Así mismo, el módulo de comunicaciones puede recibir solicitudes a través del bloque Handle Write TCP incluido en el módulo TCP, para enviar información a través de la red de comunicaciones. Este bloque compone un paquete TCP y lo transfiere al módulo IP (bloque Handle Write IP) que acabará completando la información correspondiente a su nivel de protocolo de red, confiándosela

posteriormente al bloque Write Packet que la convertirá en una trama de red apropiada para el tipo de interfaz de red que implemente el dispositivo embebido.

4.4 Application server

Este módulo imita el comportamiento de un servidor de aplicaciones convencional basado en hardware y software (procesador, sistema operativo, librerías, aplicaciones software, etc.). Para ello, siguiendo este modelo, se ha diseñado un bloque para la gestión de los protocolos petición respuesta (básicamente HTTP y SOAP), un bloque que actúa como Servidor Web para la gestión de las solicitudes HTML convencionales, un bloque para la gestión de la auto-publicación del servicio que presta a través del protocolo UDDI y, el bloque más importante según los intereses de este trabajo, un contenedor WS en el que se ubican los servicios que proporcionará el chip, denominado WS Core (Fig. 4).

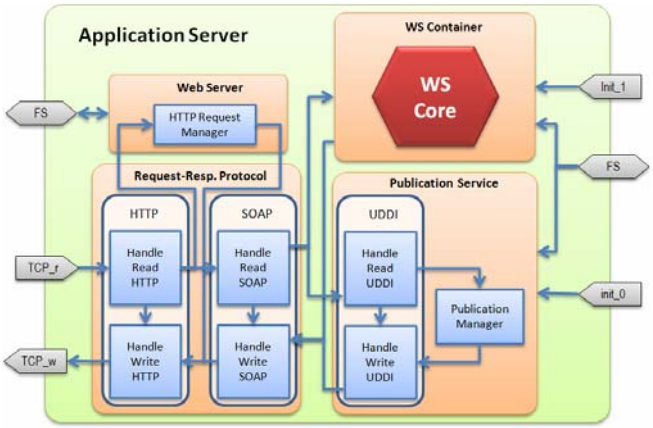

Fig. 4. Diagrama de flujo del Application Server.

Como ya se mencionó al inicio del artículo, el principio de diseño del chip WSoC ha sido que éste sirva como un patrón hardware con el que diseñar diferentes servicios Web embebidos en un chip, de forma que el diseñador pueda centrarse únicamente en los detalles del servicio que desea implementar. Además, toda la arquitectura ha sido concebida imitando una arquitectura software convencional. De esta forma, los desarrolladores de los WS Cores podrán disponer de una interfaz con los recursos que precisen muy similar a la que se encontrarían también en un servidor de aplicaciones software clásico.

4.4.1 Request/Response Protocol

Este bloque contiene las funciones que implementan el protocolo de aplicación del tipo petición-respuesta sobre los que se basa la comunicación de los servicios que se ejecutan en el seno del servidor de aplicaciones. En nuestro caso, los protocolos son:

HTTP, que actuará como protocolo de transporte, y SOAP, que actuará como protocolo de aplicación.

4.4.2 Web Server

El servidor web en nuestro diseño únicamente tiene sentido por motivos de compatibilidad de diseño. Puesto que se desea seguir un diseño hardware que imite la arquitectura software de un servidor de aplicaciones, el cometido del servidor web es el de actuar como front-end del chip. Será el que reciba las solicitudes HTTP, las evaluará y las resolverá directamente, en caso de invocaciones sobre objetos estáticos, o bien las redirigirá al servidor de aplicaciones si las invocaciones son sobre objetos dinámicos.

Otro de los objetivos del servidor Web es poder proporcionar una interfaz de administración del dispositivo y del servicio agradable, fácilmente configurable, basada en estándares y a través de la propia red.

4.4.3 Publication Service

Cuando un dispositivo se conecta por primera vez, el bloque init detecta esta situación leyendo su estado del archivo de inicialización y activando este bloque. El cometido del mismo es registrar automáticamente los servicios que el chip prestará en un Registro UDDI conectado a la red del sistema. Uno de los primeros aspectos que debemos tener en cuenta es cómo representar el servicio que ofrece cada máquina y cómo describir cada una de las actividades y procesos que forman parte de este servicio. Para ello emplearemos una hoja WSDL que describan cada uno de los procesos que es capaz de llevar a cabo la máquina.

La implementación del protocolo se basa en una serie de plantillas WSDL almacenadas en la memoria no volátil. El módulo UDDI Manager es el responsable de recuperar dichas hojas y emplearlas para componer mensajes SOAP con los que comunicarse con el servidor UDDI.

Cada vez que se desee registrar o actualizar los servicios en un servidor UDDI, bastará con configurar el chip en modo 0 (ver Tabla I) y reiniciarlo. Tras un registro satisfactorio, este módulo sitúa el estado del dispositivo en modo 1 y lo reinicia.

4.4.4 WS Container

El contenedor WS tiene como principal función aislar al WS Core de los detalles de la implementación hardware, proporcionándole una interfaz familiar para el diseñador de servicios Web, de forma que con las actuales herramientas de diseño y desarrollo de hardware, su implementación sea muy similar a la programación de un WS estándar.

Como el dispositivo debe ser auto-suficiente, para que un cliente pueda descubrirlo y consumirlo, es necesario que antes de activar el contenedor WS se haya registrado su servicio. Este bloque se activa cuando el bloque init detecta que el estado del dispositivo es 1 (ver Tabla I). En ese momento, tomará el control del chip y proporcionará la interfaz WS a través de la cual los clientes WS podrán poner en contacto con el dispositivo tanto para acceder a los servicios que presta como para su administración.

5 WoLI Web Service

Para dotar a la propuesta de algún tipo de funcionalidad que facilite comprender su funcionamiento y poder desarrollar un prototipo funcional con el que probarla, se propone un servicio de red concreto que ofreceremos como WS. Este servicio se denomina WoLI Service y a lo largo de esta sección lo describiremos en detalle.

El servicio WoLI es un servicio web que permite controlar el arranque de nodos de red mediante el protocolo WoL, utilizando para ello los protocolos estándar de Internet y Arquitecturas Orientadas a Servicios, que hacen el servicio, independiente de la ubicación del administrador o de la plataforma utilizada por el administrador

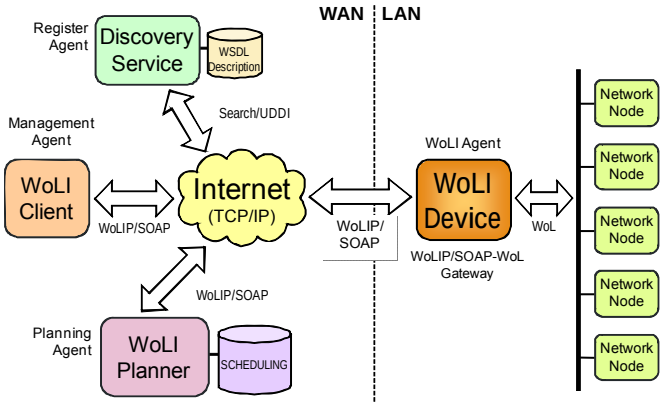

Fig. 5. Escenario de evaluación del dispositivo WoLI.

La principal utilidad de la WoLI servicio es facilitar la administración remota de los nodos de la red a través de una red de área amplia, en general, y en particular de Internet, en el que la simple imposibilidad de poner en marcha, desconectar o reiniciar los nodos del sistema restringe a los administradores la capacidad de actuar a distancia.

La Fig. 5 muestra un diagrama de los principales elementos y actores que participan en el servicio, junto con la relación existente entre ellos. Podemos sintetizar estos como: WoLI agent, WoLI Scheduler, WoLI device y los nodos de la red para que el servicio está destinado. A continuación analizamos cada uno de estos elementos.

El WoLI agent proporciona al usuario, a través de la gestión de los servicios de agente, el acceso al servicio, así como a la programación de tareas, y con el fin de generar WOL instrucciones a través de Internet. Las órdenes se transmiten al planificador o al dispositivo WoLI por medio del protocolo de aplicación WoLIP (Wake on LAN sobre Protocolo de Internet) ha definido para este fin e incrustado en mensajes SOAP. Este agente será generalmente externo al dispositivo.

El WoLI Scheduler normalmente actúa como un panel de control para todos los posibles dispositivos WoLI distribuidos a través de Internet. Este control se lleva a cabo mediante el agente planificador que lleva a cabo, ejecuta y controla todas las

tareas previamente establecidos en el dispositivo WoLI (interruptores en un individuo nodo o un grupo de nodos, verifica su estado, se actualiza el firmware de los dispositivos WoLI e incluso planes la labor de la WoLI dispositivo). Este agente puede residir en un nodo externo fuera de la LAN, por lo general a la ubicación del servicio o proveedor de comunicaciones, o puede estar integrado en un dispositivo WoLI. Desde el punto de vista de un cliente WoLI, el agente planificador se comporta como un servicio Web que puede ser localizado gracias al registro UDDI.

El WoLI Device es la piedra angular del servicio. Se trata de un dispositivo de red embebido diseñado con un chip WSoC que es de pequeño tamaño y está concebido para actuar como un traductor WoLIP-WOL entre la red de área amplia y la red de área local en el que está ubicada.

Tabla 2. Principales comandos soportados por el Chip WSoC

CMD	ARG	FUNCTION
SET	MODE	Informa del modo de operación actual.
	MODE PASSIVE [puerto]	Establece el modo pasivo y opcionalmente el puerto de escucha.
	MODE ACTIVE <ip> [:Puerto]	Establece el modo Activo especificando la dirección IP del servidor y el puerto.
	RUN	Informa del estado actual del servicio WoL
	RUN <STARTS | STOP>	Inicia o Para el servicio
GET	SCHDL	Devuelve las tareas planificadas en el dispositivo
PUT	SCHDL	Añade una tarea o tareas a la planificación
VALIDATE	<user> <pass>	Identificación y Autenticación del Usuario
WAKE	<host>	Enciende un nodo de red mediante el protocolo WoL
PING	<host>	Comprueba si el host especificado está en funcionamiento

En el caso concreto de una instrucción a distancia para arrancar un nodo de red, el agente se compromete a traducir la petición a un datagrama WOL.

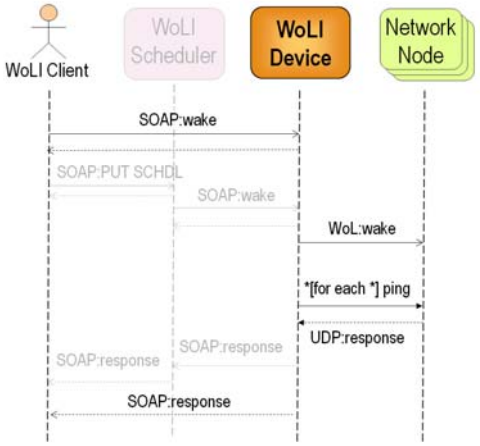

Fig. 6. Ejemplo de diagrama de secuencia del modo pasivo.

El WoLI agent siempre actúa dentro de la LAN y se comporta de forma diferente en función de si actúa en activo o en modo pasivo. En el modo pasivo, el agente espera las solicitudes WoLIP de un agente de gestión, por lo general externo, con el fin de ejecutarlos (Fig. 6). Sin embargo, en modo activo, es el agente que toma la iniciativa y pide un plan de trabajo de un agente planificador (Fig. 7). Su funcionamiento en el modo activo es fundamental, ya que permite que trabajar de forma independiente de las posibles políticas de seguridad aplicadas para la protección de la intranet, ya que, para ello, utilizará las solicitudes HTTP estándar como base para el protocolo WoLIP.

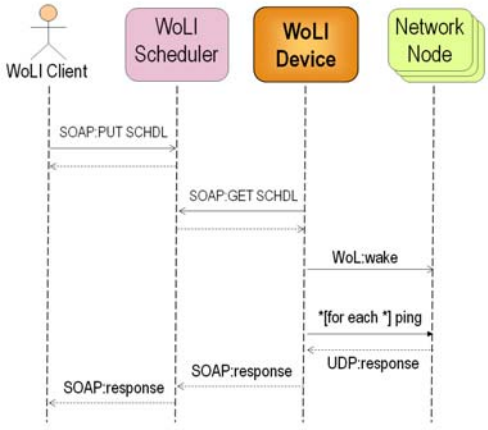

Fig. 7. Ejemplo de diagrama de secuencia del modo activo.

Este agente actúa como un servicio Web, recibiendo instrucciones definidas en el protocolo WoLIP, encapsuladas en mensajes SOAP y se define por medio de páginas WSDL. En nuestra propuesta, este la funcionalidad de este agente es, de hecho, la misma que llevará a cabo el WS Core del WSoC Chip.

Los nodos de la red son el objeto de la administración y comprenden a todos aquellos dispositivos conectados a la red con soporte WOL en sus tarjetas de red. Estamos hablando de PCs, servidores de red, dispositivos de red o cualquier otro dispositivo que cumpla los requisitos establecidos.

El Discovery Service comprende un modelo de servicio de registro UDDI. Es responsable del mantenimiento de las páginas que describe los servicios WoLI en formato WSDL, así como facilitar esta información a los clientes que deseen acceder al servicio.

6 Protocolo WoLIP

El protocolo de servicio WoLIP define una serie de instrucciones utilizadas por los usuarios y por los diversos componentes del sistema con el fin de comunicarse entre sí. Este protocolo está basado en mensajes y es compatible con SOAP, que actúa como un mecanismo de intercambio de información que permite llamadas a procedimientos remotos.

Cada comando del protocolo WoLIP está incrustado en el cuerpo de un mensaje SOAP que contiene el nombre de un procedimiento remoto que implementa la funcionalidad del comando y los argumentos necesarios para su ejecución.

```
SOAP Envelope
    SOAP Msg Header
    SOAPMsg Body
        WoLIP Command
            Action level 1
            Action level 2
Arguments
```

Se distinguen los siguientes elementos:

- Command define las acciones de servicio en los términos de la solicitud. Se corresponde con el nombre del procedimiento remoto que implementa la funcionalidad del comando WoLIP.
- Action level 1 and Action level 2 son parámetros especiales del perfil functional de la solicitud.
- Argument representa la información necesaria para la ejecución de la aplicación.

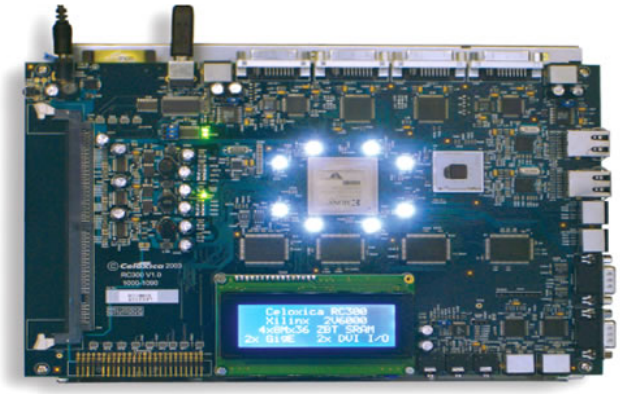

Fig. 8. Placa Celoxica RC203E.

Por cada mensaje de solicitud WoLIP habrá una respuesta correspondiente SOAP, el cuerpo del que se formará la solicitud presentada o de un mensaje de error si hay algún problema en la ejecución.

Las solicitudes que se definen en el protocolo (ver cuadro II) pueden agruparse en tres tipos principales: órdenes de configuración, órdenes básicas y órdenes de control.

La configuración de las variables internas del dispositivo determina su modo de funcionar. Estas variables son gestionadas mediante el comando SET. El servicio básico proporcionado por el dispositivo WoLI se invoca utilizando el comando WAKE. A continuación presentamos un ejemplo de solicitud SOAP para invocar al comando WAKE.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap= \
       "http://schemas.xmlsoap.org/soap/envelope/"
  Soap:encodingStyle=      \
       "http://schemas.xmlsoap.org/soap/">
  <soap:Body>
    <wp:wake xmlns:wp=     \
              "http://www.dtic.ua.es/wolip">
      <host>192.168.1.23</host>
    </wp:wake>
  </soap:Body>
</soap:Envelope>
```

Los comandos GET y PUT, junto con el argumento SCHDL para la programación y la obtención de la programación de un dispositivo, habilitan el proceso de conexión de uno o varios grupos de equipos de forma autónoma.

El servicio permite el control de acceso mediante un usuario y contraseña, siendo gestionado mediante el comando VALIDATE.

Todos los requerimientos del protocolo WoLIP proporcionan una respuesta: OK, si la orden se ejecuta correctamente o ERROR si no es así, salvo GET SCHDL que devuelve la lista de tareas programadas para el dispositivo y el comando PING que asincrónicamente mostrará el estado del nodo de red.

Figura 7 muestra un diagrama de secuencia en el que un cliente WoLI y un Scheduler previamente programado, envían solicitudes de arranque remoto a los dispositivos de red a través de un dispositivo WoLI, que traduce la petición a fin de que los nodos LAN entenderlo. Además el dispositivo WoLI comprueba que el nodo se ha puesto en marcha y lo comunica al solicitante.

7 Implementación y Evaluación

7.1 Implementación del Prototipo

Para el desarrollo del prototipo se ha elegido la tecnología FPGA como una rápida y adecuada herramienta de diseño. Se ha utilizado la placa Celoxica RC203E para la implementación del WoLI Soc. Esta placa tiene una FPGA Xilinx XC2V3000, dos bancos de memoria ZBT 4MB y una memoria Flash SmartMedia para el almacenamiento permanente. La interfaz de red incluida en esta placa es la LAN91C111 Microsystems que incluye un controlador MAC + PHY full-duplex con 16KB de buffer.

El diseño completo del WSoC ocupa 7324 slices de la FPGA (51%), un bloque de RAM (1%) y trabaja a 40 MHz. Tansolo se han usado 10KB de memoria externa SRAM para datos auxiliares.

Se ha usado el lengiuaje de alto nivel de descripción Andel-C para la codificación del sistema, permitiendo un desarrollo rápido e incremental. Se han usado algunas librerías incluidas en el PDK v4.1 (Platform Development Kit) como drivers con los dispositivos externos. Se usó la SmartMedia para almacenar el bitstream de todo el diseño. El diseño no usa ni microprocesador ni software y está basado en la arquitectura vista en el apartado 3.

7.2 Evaluación del Prototipo

Con el fin de evaluar la propuesta, se ha recreado un escenario como el que se indica en la Figura 5. En este escenario se ha incorporado el prototipo implementado sobre la placa RC2003E de Celoxica, un servicio de registro UDDI, un cliente WS para acceder al sistema y una red de PC's compatible con WOL. El único elemento que no se ha incorporado en este escenario es el planificador WoLI debido al hecho de que no se ha implementado en el prototipo el modo activo de este servicio.

7.2.1 Evaluación de la Publicación

Se ha utilizado un servidor Apache jUDDI v0.9rc4 para comprobar el módulo de auto-registro, este servidor soporta UDDI v.2.0. Permite que el servicio de registre de forma estándar. Mediante la conexión del prototipo a la red y la captura del tráfico de la red entre el dispositivo y el servidor, hemos podido verificar la correcta

comunicación. Como se muestra en la Fig.9, el dispositivo busca el servicio en el servidor jUDDI servidor. Si no encuentra el servicio, lo publicará autenticándose previamente para usar las funciones privadas del servidor jUDDI.

Se ha utilizado la herramienta TCPDump en el servidor para monitorizar el tráfico de la red durante la evaluación y la información generada se ha analizado utilizando el "The Wireshark Network Protocol Analizer". Con estas herramientas se ha podido verificar la correcta comunicación entre el dispositivo y el servidor. Las herramientas también nos han permitido capturar el tráfico de la red como el que muestra a continuación. En este fragmento de tráfico, el dispositivo envía un mensaje SOAP con comando save_TModel del protocolo UDDI.

```
POST /juddi/publish HTTP/1.1
Content-length:260
Content-Type: text/xml; charset=utf-8
Host: 172.19.32.39

<?xml version="1.0" encoding="utf-8"?>
     <soapenv:Envelope xmlns:soapenv="http://schemas
     .xmlsoap.org/soap/envelope/">
  <soapenv:Body>
      <save_tModel generic="2.0" xmlns="urn:uddi-
            org:api_v2">
   <authInfo>
authToken:E98101F0-4656-11DC-81F0-FA34E9DBA31E
     </authInfo>
  <tModel tModelKey="">
    <name>wolifpgadtic</name>
    <description>WoLI Service</description>
    <overviewDoc>
      <description>
                        WoLI service by FPGA
                  </description>
      <overviewURL>
                        http://172.019.032.151/index.wsdl
                  </overviewURL>
     </overviewDoc>
    </tModel>
   </save_tModel>
  </soapenv:Body>
</soapenv:Envelope>
```

7.2.2 Evaluación del Servicio

El servicio ha sido probado con éxito utilizando, por un lado, un cliente PHP que utiliza la biblioteca NuSOAP y, por otro lado, un cliente estándar de línea de comandos desarrollado por el grupo Apache conocido como WSIF. Dicho cliente estándar de línea de comandos es capaz de utilizar servicios a través de un invocador escribiendo la función, los parámetros y la dirección donde se almacena la hoja de WSDL, luego genera una llamada al servicio prototipazo y muestra la respuesta obtenida.

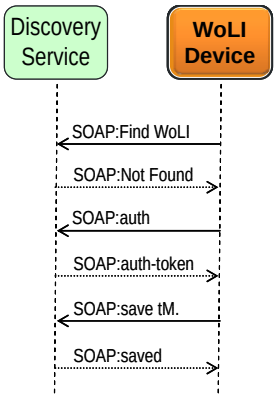

Fig. 9. Diagrama de secuencia del servicio de publicación.

El procedimiento llevado a cabo por los clientes PHP y WSIF después del proceso de auto-registro ha finalizado es como sigue.

- Primeramente, el cliente busca el servicio en el servidor jUDDI para obtener la dirección de la hoja WSDL, que en este caso, está almacenada en la FPGA para testear las capacidades como servidor Web de la FPGA.
- A continuación, el cliente obtiene la hoja WSDL mediante una petición http GET. La FPGA le contesta con la hoja WSDL.
- Una vez que se ha obtenido la descripción WSDL, el cliente construye peticiones SOAP válidas que son enviadas mediante mensajes HTTP POST al servidor de servicios web de la FPGA y este contestará con los mensajes SOAP de acuerdo a los comandos del protocolo WoLIP descrito en la Tabla II.

Con el fin de validar la propuesta, se ha implementado una pequeña selección de comandos del protocolo WoLIP, incluido el comando WAKE del protocolo, y una variación del comando VALIDATE que no requiere una contraseña y que hemos denominado LOGIN <user>. A continuación se muestra una pequeña parte del código fuente de la lógica del servicio. Como puede verse, con el fin de ejecutar correctamente el comando WAKE el host que lo invoca debe haber ejecutado anteriormente el comando LOGIN. La sesión será válida hasta que se invoca el comando LOGOUT.

```
switch(command) {
 case 1: // LOGIN
     if(user==0x64746963 && logedHost==0)
     {
            //Test If Correct user and send Response
            SendTCPData(&IpInfoPtr,&TcpInfoPtr,LOGIDIR,1);
                   logedHost=IpInfoPtr.SourceAddress;
     }
     else
            SendTCPData(&IpInfoPtr,&TcpInfoPtr,LERRDIR,1);
 break;
 case 2: // LOGOUT command
     // Test previous login and send WoLIP response
     if(IpInfoPtr.SourceAddress==logedHost)
```

```
     {
    SendTCPData(&IpInfoPtr,&TcpInfoPtr,LOGODIR,1);
    logedHost=0;
     }
     else
            SendTCPData(&IpInfoPtr,&TcpInfoPtr,LOERDIR,1);
 break;
 case 3: // WAKE command
     if(IpInfoPtr.SourceAddress==logedHost)
     {//Send Wake Packet if previous LOGIN
      SendUDPWol(&IpInfoPtr,TcpInfoPtr.IpsCheck,mac);
      //Send WoLIP Response
      SendTCPData(&IpInfoPtr,&TcpInfoPtr,WAKEDIR,0);
     }
     else
      SendTCPData(&IpInfoPtr,&TcpInfoPtr,WAKERDIR,1);
     break;
 }
```

Al igual que en la evaluación del proceso de registro, se ha utilizado un monitor de red para capturar el tráfico de la red entre el cliente y la FPGA. Como resultado, hemos podido comprobar que la comunicación entre ambos se lleva a cabo de manera correcta, esto se puede ver por medio del mensaje enviado por el cliente que se muestra a continuación.

```
POST /woli.wsdl HTTP/1.0
Host: 172.19.32.151
User-Agent: NuSOAP/0.7.2 (1.94)
Content-Type: text/xml; charset=ISO-8859-1
SOAPAction: "urn:woliwsdl#wake"
Content-Length: 507

<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope
SOAP-ENV:encodingStyle
     =”http://schemas.xmlsoap.org/soap/encoding/”       xmlns:SOAP-
     ENV="http://schemas.xmlsoap.org/soap/envelope/"xml
     ns:xsd="http://www.w3.org/2001/XMLSchema"xmlns:xsi
     ="http://www.w3.org/2001/XMLSchema-  instance"xmlns:SOAP-
     ENC="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:tns="urn:woliwsdl">

<SOAP-ENV:Body>
     <tns:wake xmlns:tns="urn:woliwsdl">
            <host xsi:type="xsd:string">
                   000D88270A36
            </host>
     </tns:wake>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Evidentemente, después de haber enviado la respuesta a este mensaje, se observó que la FPGA enviaba el paquete WoL *magic packet* a través de la red y provocaba el encendido del host especificado.

8 Conclusiones

En este trabajo hemos presentado un diseño SoC de bajo coste y con mantenimiento cero para dispositivos de red que pueden ejecutar servicios Web con estilo SOA.

La principal ventaja de este enfoque es que puede proporcionar servicios que son comunes en el entornos de red (redes industriales de fabricación, redes domésticas, las redes empresariales, etc) sin necesidad de administradores de sistemas especializados, ni para el despliegue inicial y la creación de tareas ni para el posterior mantenimiento.

Este prototipo de WOL sobre Internet basado en FPGA ha sido desarrollado con el fin de probar sus funcionalidades y evaluar la propuesta. El dispositivo es capaz de configurarse y trabajar de manera independiente, o integrarse con otros servicios de las redes existentes, independientemente de si estos se integran con otros dispositivos similares o convencionales como los servicios de red.

Desde la perspectiva del usuario, cada uno de los servicios representa un pequeño dispositivo de red que simplemente requiere conexión a la red de su organización para realizar el servicio requerido.

En estos momentos estamos trabajando con otros WS cores que implementen servicios de red y la integración de todos ellos en un modelo basado en servicios de Web Semántica, por lo que en el futuro no sólo será compatible con los servicios existentes, sino también con nuevos servicios o configuraciones que no fueron considerados en su diseño inicial.

Referencias

1. J. Roy and A. Ramanujan. Understanding web services. IEEE Internet Computing, vol. 3, no. 6, pp. 69–73, Nov.-Dec. 2001.
2. Lech Jóźwiak, Sien-An Ong, "Quality-driven model-based architecture synthesis for real-time embedded SoCs", Journal of Systems Architecture, vol. 54, pp.349-368, March-April 2008.
3. J.J. Rodriguez-Andina, M.J. Moure, M.D. Valdes, "Features, Design Tools, and Application Domains of FPGAs," Trans. on Industrial Electronics, vol. 54, no. 4, pp. 1810-1823, Aug. 2007.
4. V. Gilart, F. Maciá, J.A. Gil and D. Marcos, "Services and networks management through embedded devices and SOA", In Proc. of the 10th IEEE International Enterprise Distributed Object Computing Conference EDOC 2006., Hong Kong, Oct. 2006, pp395-398.
5. J.A. Gil, D. Marcos, F. Maciá and V. Gilart, "Wake on LAN over Internet as Web Service", In Proc. of the 11th IEEE International Conference on Emerging Technologies and Factory Automation ETFA 2006, Prague, September 2006, pp1261-1268.
6. F. Maciá, D. Marcos, and V. Gilart, "Industrial TCP/IP Services Monitoring through Embedded Web Services**"**,Journal on Embedded Systems, Vol. 2008, pp 1-10, 2008.
7. Jeong, M.S., Kim, K.H., Kwon, J.H., Park, J.T. CORBS/CMIP: Gateway Service Scheme for CORBA/TMN Integration." Knom Review, Vol.2, No. 1, pp. 55-62, 1999.

tareas repetitivas y claramente definidas que habitualmente sirven de soporte para las aplicaciones de alto nivel —por ejemplo: servicios de nombre, servicios de configuración de red, servicios de sincronización de hora, servicios de monitoreo de la actividad, servicios de cálculo de rutas o servicios de descubrimiento.

Estos servicios conllevan un cierto número de tareas de administración simples pero repetitivas (configuración, monitoreo, actualización); y requieren de grandes infraestructuras para su implantación que en la práctica, pueden llegar a ser un dolor de cabeza para los administradores de sistemas.

Una propuesta que está ganando credibilidad por simplificar la gestión de dichas infraestructuras y los servicios de las TI es: (1) diseñar Web services (WS), proporcionando una infraestructura débilmente acoplada, basada en varios estándares y protocolos, que permite la fácil integración de diferentes servicios de diferentes proveedores independientemente de la plataforma tecnológica; (2) diseñar servicios y sistemas con un enfoque de mantenimiento nulo; e (3) incorporar dispositivos físicos que se diseñan para proporcionar algunos de los servicios descritos de forma autónoma.

Debido a su complejidad, los WS se implementan principalmente en arquitecturas orientadas a PC. Sin embargo en los últimos diez años ha habido numerosos avances en las tecnologías para el desarrollo de pequeños dispositivos de red, con una más que aceptable capacidad de cómputo, de trabajo autónomo y que proporcionan la posibilidad de incorporarles inteligencia [1],[2], [3].

Nuestra propuesta inicial fue proporcionan servicios de red específicos, con unos requerimientos de gestión y monitorización mínimos para los administradores, basados en modelos auto-configurables compatibles con la tecnología WS, los cuales se embebían en dispositivos de red extremadamente pequeños y económicos [4], [5], [6]. Sin embargo, aunque la tendencia actual es ir disminuyendo el tamaño de dichos dispositivos a la vez que se incrementa su capacidad de cómputo con un precio competitivo conduce a un escenario ideal, en práctica, permanecen algunos problemas significantes: (1) desde un punto de vista económico, la utilización de plataformas embebidas incrementa el coste final del servicio de red; (2) desde un punto de vista técnico, aunque es cierto que con propuestas basadas en dispositivos embebidos y enfoques de Arquitecturas Orientadas a Servicios (SOA), las tareas de administración se han reducido considerablemente, todavía existe una plataforma software y hardware que gestionar; por tanto, todavía existen tareas que no se pueden eliminar completamente.

En este artículo se presenta una arquitectura System on Chip (SoC), diseñada para ejecutar un Web service (WS) concreto en un Circuito Integrado de Aplicación Especifica (ASIC).

El sistema se ha diseñado sin procesador y sin software, aunque se ha imitado la arquitectura software de un servidor de aplicaciones. El SoC se ha concebido como un patrón de diseño para un fácil desarrollo de servicios de red que se ofrecen como WS bajo una arquitectura SOA. Por tanto, el chip no sólo es capaz de actuar como un proveedor de servicios SOAP, sino que también es capaz, por si mismo, de registrar el servicio en un Broker Server mediante el uso del protocolo estándar de de publicación UDDI.

8. Aschemann, G., Mohr, T., Ruppert, M. "Integration of SNMP into a CORBA- and Web-Based Management Environment," in Proc. Kommunikation in Verteilten Systemen, Heidelberg, 1999, pp. 210-221.
9. T.C. Du, E.Y. Li, and A.P. Chang, "Mobile Agents in Distributed Network Management", in Communications at the ACM, vol. 46, no. 7, pp. 127-132, 2003.
10. J. Guo,Y. Liao, and B. Parviz. "An Agent-Based Network management system" in Proceedings of the 2005 Internet and Multimedia Applications, Honolulu (Hawai), Aug. 2005.
11. European co-ordination action for agent-based computing [Online]. Available: http://www.rfc.net.
12. R. Boutaba, J. Xiao, "Network Management: State of the Art" in Proceedings of the 2002 World Computer Congress, Quebec, Aug. 2002, pp. 127-146.
13. J.E. López. V.A. Villagrá, and J.I. Asensio, "Ontologies: Giving Semantics to Network Management Models," IEEE Network, vol. 17, no. 3, pp. 15-21, May-June 2003.
14. J. Peer, "A POP-Based Replanning Agent for Automatic Web Service Composition," in proc. of the 2005 Second European Semantic Web Confenrece, LNCS, pp 47-61, May 2005.
15. J. Sloten, A. Pras, and M. Van Sinderen, "On the standardisation of web service management operations," in proc. of the 2004 X EUNICE Summer School and IFIP WG 6.3 Workshop, Tampere, Finland, June 2004, page 143-150.
16. T. Klie, and F. Straub, "Integrating SNMP agents with XML-based management systems," IEEE Communications Magazine vol. 42 Issue 7, pp. 76-83, July 2004.
17. L.S. Indrusiak, M. Glesner, R. Reis, “On the Evolution of Remote Laboratories for Prototyping Digital Electronic Systems”, ,Trans. on Industrial Electronics, vol. 54, no. 4, pp. 3069-3077, Dic. 2007.
18. R. Marau, P. Leite, P., M. Velasco, P. Marti, L. Almeida, P. Pedreiras, J.M. Fuertes, “Performing Flexible Control on Low-Cost Microcontrollers Using a Minimal Real-Time Kernel”, Trans. on Industrial Informatics, vol. 4, pp. 125-133, 2008
19. O. Laouamri and C. Aktouf, “Remote Testing and diagnosis of System-on Chips Using Network Management Frameworks”, In Proc of Design, Automation & Test in Europe Conference & Exhibition, Nice, France, April 2007, pp. 1-6.
20. S. Cuenca, A. Grediaga, H. Llorens and M. Albero “Performance Evaluation of FPGA-Embedded Web Servers”, IEEE International Conference on Electronics Circuits and Systems, Marrakech, Marocco December, 2007.
21. R. Koch, T. Pionteck, C. Albrecht and E. Maehle, “ An Adaptive System-on-Chip for Network Applications”, In Proc. 20th Parallel and Distributed Processing Symposium, April 2006.