Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica

Expansión de las capacidades de depuración de la Plataforma de Validación de Producto del Procesador Itanium® 2

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Rodney Cubero Barker

Cartago, Noviembre del 2007

INSTITUTO TECNOLOGICO DE COSTA RICA

ESCUELA DE INGENIERIA ELECTRONICA

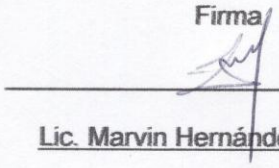
PROYECTO DE GRADUACIÓN

TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Firma



Lic. Marvin Hernández Cisneros

Profesor lector

Firma

MSc. Carlos Badilla Corrales

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

Cartago, 19 de noviembre 2007

Declaro que el presente informe ha sido realizado por mi persona, utilizando y aplicando literatura referente al tema, así como la información que haya suministrado la empresa para la que se realizara el proyecto, y aplicando e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad por el contenido de este informe.

Cartago 19 de noviembre 2007

Rodney Cubero Barker
Cédula: 701310076

Dedication:

To my beloved mother Mirtha Barker
and all the Barker family
for always being there for me,
I know as you know this was not easy,
but finally We got there
and this is just the beginning of greater things.

Agradecimientos

Este proyecto se realizo gracias a la confianza por parte del Grupo de Desarrollo SDM –CR de la Empresa Intel Costa Rica.

Un profundo agradecimiento al Ingeniero Luís Rojas por motivarme a incursionar de fondo en el campo de la arquitectura de procesadores de alto rendimiento así como su apoyo y confianza a lo largo de este arduo proyecto.

Al Prof. Carlos Badilla simplemente excelente persona, siempre velando por mí en todos los aspectos. No solo me enseño de electrónica si no también a que debo ser mejor y que todo tiene su tiempo.

A la Prof. Arys Carrasquilla por sus múltiples consejos y regaños siempre con el fin de motivarme a terminar lo que inicie. Gracias por toda su ayuda, espero continuar seguir en contacto para futuros proyectos.

Al Ingeniero Carlos Chacon y Subramanian Anandakugan por su confianza y motivación para finalizar el proyecto con éxito y seguir adelante como un profesional integro y capaz de lograr lo que me proponga. Además por enseñarme lo importante que es el aspecto de manejo de proyectos.

A mis compañeros del TEC, en especial a Alfredo Romero, que la verdad han estado conmigo en las buenas y en las malas. Ha sido la mejor época de mi vida.

Mi más profundo agradecimiento a la familia Garita Chinchilla, en especial a Karina por siempre estar ahí apoyándome en los momentos difíciles y disfrutando los momentos alegres.

Resumen

Este proyecto se realizó para el “Grupo de Desarrollo y Manufactura de Servidores en Componentes Intel de Costa Rica”, y se ha enfocado en el procesador Itanium® 2 de nueva generación con características innovadoras y capacidades extendidas que lo hacen competir en el segmento de mercado de las supercomputadoras.

El proyecto se planteó como una investigación para determinar la factibilidad de poder controlar y observar este procesador por medios no convencionales. Se expone aquí la posibilidad de crear procedimientos funcionales a partir de los resultados de la investigación.

Requirió el uso del puerto de pruebas descrito por el estándar IEEE.1149.1a, que define tanto los parámetros como las regulaciones que la especifican.

Este es esfuerzo innovador para determinar las posibilidades de ampliar las facultades de diagnóstico durante el Proceso de Validación de Producto, con este proyecto se refuerza la capacidad de realizar el análisis de las fallas que se presenta en el procesador a nivel de arquitectura, proveyendo así un mejor nivel de entendimiento que permita al desarrollador plantear mejoras al diseño de este en menor tiempo.

Palabras clave: Itanium, IA64, Arquitectura, Ensamblador, Tesis, JTAG, TAP, Halt, Go, EPIC PPV Prueba.

Abstract

This project was performed within the Server Development and Manufacturing Group at Intel Costa Rica. It's focused on Itanium 2 processor family which comes with innovative features and extended capacities which enables it to compete with the Hi Performance Computing market segment.

The project was conceived as a research, in order to determine the feasibility of performing by nonconventional means the control and observation of the processor. The project also exposes the possibility of creating functional procedures depending on the results of the research.

The task requires the use of the test port described by the IEEE.1149.1a standard, which defines the parameters and the regulations that specifies it.

The innovative effort foreseeing in this project is to expand the debugging capabilities for Product Platform Validation, providing a better understanding to the developer of the root causes of the failure at an architectural level, driving improvements to the processor design in a timely fashion.

Key words: Itanium, IA64, Architecture, Assembler, Thesis, JTAG, TAP, Halt, Go, EPIC, PPV, Test.

ÍNDICE GENERAL

Dedication: 4
Agradecimientos 5
Este proyecto se realizo gracias a la confianza por parte del Grupo de Desarrollo SDM –CR de la Empresa Intel Costa Rica. 5
Resumen 6
Abstract 7
ÍNDICE GENERAL 8
ÍNDICE DE FIGURAS 10
ÍNDICE DE TABLAS 11
CAPÍTULO 1 INTRODUCCIÓN 12
1.1 Problema existente e importancia de su solución 14
1.2 Solución seleccionada 16
CAPÍTULO 2 METAS Y OBJETIVOS 19
2.1 Meta. 20
2.2 Objetivo general. 20
2.3 Objetivos específicos. 20
CAPÍTULO 3 MARCO TEÓRICO 23
3.1 Descripción del sistema o proceso por mejorar. 24
3.2 Antecedentes Bibliográficos. 27
3.3 Descripción de los principales principios electrónicos relacionados con la solución del problema. 28
3.3.1 Itanium® 2 & IA64 [1, 2] 28
3.3.2 IEEE 1149.1a-1993, JTAG/TAP [3]. 31
3.3.2.a Alcance. 31
3.3.2.b Perspectiva General. 32
3.3.2.c Uso del estándar en productos ensamblados. 33
3.3.2.d Boundary-Scan 34
3.3.2.e TAP. 36
CAPÍTULO 4 PROCEDIMIENTO METODOLÓGICO 40
4.1 Reconocimiento y definición del problema. 41
4.2 Obtención y análisis de la información. 41
4.3 Evolución de las alternativas y síntesis de una solución. 42
4.4 Implementación de la solución. 42
4.5 Reevaluación y rediseño. 43
CAPÍTULO 5 DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN 44
5.1 Análisis de soluciones y selección final. 45
5.1.1 Interfaz de PPV [4]. 47
5.2 Descripción del hardware. 48
5.2.1 Módulo de PPV. 48
5.2.2 TIU. 49
5.2.3 Centrix. 50
5.2.4 ITP [7]. 51
5.3 Descripción del software. 53

5.3.1 Systest. 53
5.3.2 Diseño de los algoritmos. 55
5.3.3 Estructura de datos y transferencia. 56
5.3.4 HALT. 59
5.3.5 GO. 63
CAPÍTULO 6 ANÁLISIS DE RESULTADOS 66
6.1 Resultados 67
6.1.1 Resultado Código vs. ITP 67
6.1.2 Ejecución de secuencia Halt-Go 72
6.2 Análisis 73
CAPÍTULO 7 CONCLUSIONES Y RECOMENDACIONES 75
7.1 Conclusiones. 76
7.2 Recomendaciones 77
CAPÍTULO 8 BIBLIOGRAFÍA 78
APÉNDICES 80
A.1 Glosario, abreviaturas y simbología 81
A.2 Información sobre la empresa. 83
A.2.1 Descripción de la empresa. 83
A.2.2 Descripción del departamento. 85
A.3 Recursos necesarios para el desarrollo del proyecto. 86
A.4 Recursos disponibles en la empresa. 87
A.5 Hoja de información del proyecto. 88
ANEXO 90
Anexo A.1 Portada de referencias bibliográficas. 91

ÍNDICE DE FIGURAS

Figura 1.1 Diagrama de nivel del proyecto. 15
Figura 1.2 Diagrama que describe la solución seleccionada. 17
Figura 3.1 Posición de PPV en el flujo de ensamble y prueba de Intel. 24
Figura 3.2 Diagrama de flujo simplificado del proceso de PPV y análisis. 26
Figura 3.3 Formato de instrucción en la arquitectura de Itanium. 30
Figura 3.4 Estructura del grupo de instrucciones por ejecutar en paralelo. 30
Figura 3.5 La líneas en rojo simulan la inyección de instrucciones y datos a través del TAP usando el protocolo del JTAG. 33
Figura 3.6 Una celda de un registro perteneciente al Boundary-Scan. 34
Figura 3.7 Ejemplo de un Chip con Boundary-Scan. 35
Figura 3.8 Diagrama de estados para el controlador del TAP. 38
Figura 5.1 Diagrama de alto nivel de la solución seleccionada 46
Figura 5.2 Diagrama de bloques de la interfaz de PPV. 47
Figura 5.3 Fotografía frontal de un TIU de Itanium. 48
Figura 5.4 Diagrama de bloques de los componentes básicos de una TIU. 49
Figura 5.5 Fotografía frontal de 2 TIU de Itanium. 50
Figura 5.6 Fotografía de una tarjeta Centrix. 50
Figura 5.7 Diagrama de una tarjeta Centrix. 51
Figura 5.8 Fotografía del ITP 52
Figura 5.9 Ambiente gráfico de usuario del ITP 52
Figura 5.10 Interfaz de usuario de Systest. 53
Figura 5.11 Interfaz de programación de Systest. 54
Figura 5.12 Diagrama de flujo de alto nivel del algoritmo desarrollado. 55
Figura 5.13 Ejemplo de Load y su equivalente hexadecimal. 57
Figura 5.14 Formato del arreglo que contiene la instrucción a ejecutar. 58
Figura 5.15 Función Jtag_Dscan. 58
Figura 5.16 Estructura de datos para LoadReg. 59
Figura 5.17 Diagrama de Bloques de Halt. 60
Figura 5.18 Instrucciones que ejemplifican los modos de direccionamiento. 61
Figura 5.19 Diagrama de bloques de Go. 63
Figura 5.19 Código para restaurar los registros generales. 64

ÍNDICE DE TABLAS

Tabla 1 Valores de obtenidos con ITP vs. Halt/Go en Windows 69
Tabla 2 Valores de obtenidos con ITP vs. Halt/Go en Linux 70
Tabla 3 Valores de obtenidos con ITP vs. Halt/Go en EFI 71

CAPÍTULO 1
INTRODUCCIÓN

El proyecto plantea un proceso innovador para la recolección de datos mediante el uso del estándar IEEE.1149.1a conocido como JTAG, el cual se presenta como un medio no convencional de adquisición de datos referentes al estado de la arquitectura a nivel de la Plataforma de Validación de Producto.

Se buscó (mediante el uso de lenguaje ensamblador en combinación con Visual Basic Script) hacer uso del JTAG y de la arquitectura del procesador para obtener la información necesaria para poder detener la ejecución de éste en forma determinativa y, en consecuencia, poder restaurar su estado de ejecución.

La inyección de instrucciones mediante el JTAG directamente a la unidad de ejecución es uno de los más importantes conceptos electrónicos aplicados en este proyecto.

La finalidad del proyecto se orientó hacia la innovación de la manera en que se utiliza el estándar para fines como el diagnóstico de fallas, de donde se deriva la posibilidad de desarrollar nuevas formas de control y uso del procesador usando los recursos disponibles en la Plataforma de Validación de Producto. En la actualidad Intel utiliza una herramienta de diagnostico llamada ITP (In-Target Probe), la cual consiste en un equipo y software que de forma independiente y fuera de la línea de manufactura permite al ingeniero determinar las posibles causas de falla del procesador utilizando el JTAG.

1.1 Problema existente e importancia de su solución

Dentro del flujo de pruebas al cual es sometido un procesador durante el proceso de manufactura por parte Intel Costa Rica –flujo utilizado para garantizar la calidad de sus productos- se encuentra la Plataforma de Validación de Producto, PPV* (Product Platform Validation, por sus siglas en inglés). En esta etapa se evalúa el correcto funcionamiento del procesador emulando el ambiente de usuario final.

Como ya se mencionó, la PPV emula el ambiente de usuario final: por ejemplo ejecución de sistemas operativos, aplicaciones dentro de estos y procedimientos para evaluar el rendimiento de un procesador tales como algoritmos numéricos o manipulaciones masivas de datos. Dichas pruebas son ejecutadas bajo condiciones específicas de operación (voltaje, frecuencia y temperatura de carcasa entre otras).

Los procesadores manufacturados por Intel poseen un puerto compatible con el estándar IEEE.1149.1a, comúnmente conocido como el "Grupo de Cooperación para el Acceso a Pruebas", JTAG (Joint Test Access Group, por sus siglas en inglés). Este estándar fija las especificaciones para la infraestructura de diagnóstico presente en sistemas digitales de alta complejidad, lo cual permite el acceso a la arquitectura del procesador de manera indirecta. Esto quiere decir que se permite observar y controlar el procesador por medios no convencionales con el fin de diagnosticar fallas internas.

El problema planteado fue determinar si existe la posibilidad de establecer una metodología y una herramienta que permitiera observar y controlar el procesador utilizando el JTAG durante la ejecución de la pruebas en PPV, utilizando los recursos disponibles.

* Ver Glosario

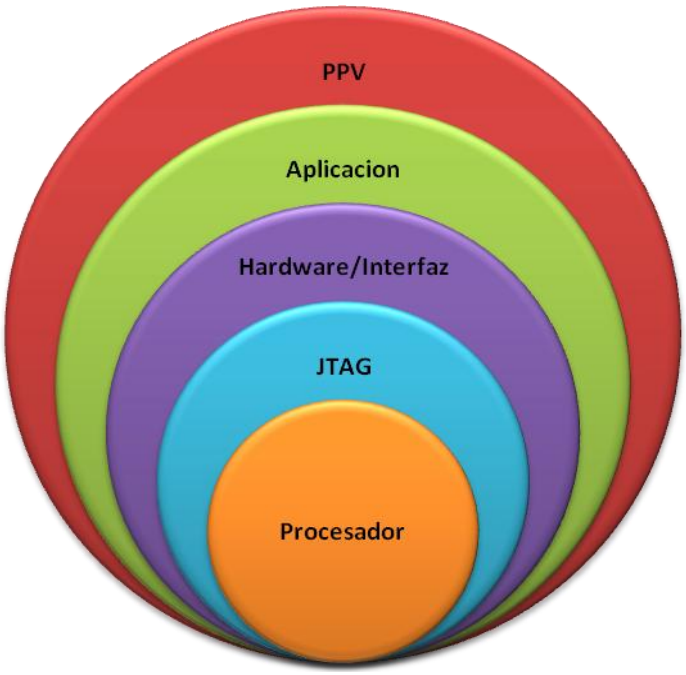


Figura 1.1 Diagrama de nivel del proyecto.

En la figura 1.1 se muestra el diagrama de nivel de la relación entre las partes involucradas en el proyecto.

La importancia del proyecto respaldó su justificación en la constante mejora de los procesos productivos, ya que si se logra establecer la posibilidad de controlar y observar el procesador se podrán idear e implementar distintas mejoras al proceso de pruebas en PPV, lo que implica una reducción en diversos indicadores de manufactura (como por ejemplo las *"horas hombre"* que se necesitan para el diagnóstico de fallas utilizando ITP).

Lista de beneficios derivados de la resolución del problema:

1) Aprovechamiento de recursos del flujo de pruebas para el diagnóstico.
2) Reducción del tiempo para la determinación del origen y causa de la falla.
3) Reducción del *"tiempo hombre"* necesario para el diagnóstico de fallas.
4) Reducción en el costo del desarrollo de un producto.
5) Posibilitar el lanzamiento anticipado del producto.
6) Idear nuevas pruebas para los procesadores.

1.2 Solución seleccionada

El estudio de la arquitectura* y sus modos de operación permitió establecer los parámetros mínimos requeridos para lograr satisfactoriamente la consecución de los objetivos.

La solución seleccionada fue utilizar el ambiente de programación de PPV como plataforma para la investigación y el desarrollo de los algoritmos necesarios que cumplieran con los requisitos planteados para el proyecto.

Mediante la interacción entre software y hardware se logró acceder a los elementos de arquitectura necesarios para el cumplimiento de los objetivos.

La selección de esta solución ha planteado un cambio en algunos de los requerimientos del proyecto, debido a los resultados obtenidos y a limitaciones del sistema (plataforma).

Como se muestra en el diagrama de bloques, Systest es el ambiente de trabajo utilizado. Éste incorpora las funciones de Halt & Go, las cuales tienen subrutinas que gobiernan la tarjeta llamada Centrix, que a su vez contiene los dispositivos necesarios para el manejo del JTAG, permitiendo acceder a la arquitectura del procesador. El "Puerto de Acceso a Pruebas", TAP* (Test Access Port, por sus siglas en inglés), es el nombre común que se le da al puerto físico que define el estándar del JTAG

* Ver Glosario
* A partir de aquí TAP y JTAG se usará indistintamente.

Figura 1.2 Diagrama que describe la solución seleccionada.

El proyecto presentaba los siguientes requerimientos:

1. Investigar la posibilidad de crear una metodología que permitiera la ejecución de procedimientos a través del TAP:

    a. Halt: procedimiento que se realiza para llevar al procesador de un estado de ejecución a un estado de parada, almacenando el momento de arquitectura presente, con el fin de poder restablecerla.

    b. Go: procedimiento que lleva al procesador de un estado de parada a uno de ejecución, restableciendo su momento de arquitectura.

c. Puntos de parada: procedimiento que detiene la ejecución de del procesador y guarda su estado de arquitectura en un punto específico del programa.

d. Ejecución paso a paso: Procedimiento que combina los procedimientos de Halt y Go y permite el avance de un programa mediante la ejecución interrumpida de una instrucción a la vez.

2. Desarrollar algoritmos y metodologías que definieran los procedimientos anteriores.

3. Desarrollar una herramienta que permitiera la utilización de los procedimientos mencionados a través del TAP.

4. Desarrollar un intérprete de comandos que actúe como elemento de control para la herramienta e interfaz con el usuario.

CAPÍTULO 2
METAS Y OBJETIVOS

2.1 Meta.

Contar con una herramienta que permita la observación y control del procesador a través del “Puerto de Acceso a Pruebas” (TAP).

2.2 Objetivo general.

Investigar la posibilidad de establecer una metodología que permita el control y observación del procesador por medio del “Puerto de Acceso a Pruebas” (TAP).

2.3 Objetivos específicos.

1. Determinar si existe la posibilidad de inyectar instrucciones al Pipe-Line* del procesador a través del TAP.

2. Establecer el formato necesario de la información a enviar.

3. Especificar las limitantes y capacidades del JTAG.

4. Especificar las limitantes y capacidades del procesador.

5. Determinar los parámetros y señales eléctricas que necesita la interfaz para permitir el acceso y control del TAP.

6. Determinar la secuencia y sincronización de las señales de TDI* y TDO* en el hardware del TAP.

7. Establecer una metodología (y los algoritmos necesarios) que permitan realizar los procedimientos “Halt”, “Go”, “Punto de Parada” y “Ejecución paso a paso”, en el procesador a través del TAP.

* Ver Glosario

8. Halt: investigar y crear un procedimiento que, mediante el TAP así como en el procesador, permita detener la ejecución y la captura de su estado de arquitectura con el fin de poder restablecerla.

9. Go: investigar y desarrollar los pasos e instrucciones necesarias tanto para el TAP como en el procesador, que permitan restablecer su estado de arquitectura en el momento de la interrupción.

10. Punto de parada: investigar y generar un procedimiento que permita establecer, en un programa o secuencia de instrucciones, una interrupción (capturando su estado de arquitectura).

11. Generación de un documento que contenga la información relevante de la investigación, así como de los procedimientos y rutinas necesarias para llevar a cabo la observación y el control del procesador a través de JTAG. Entre los temas de mayor interés están:
    a. La metodología de acceso y control a través de la infraestructura de diagnóstico.
    b. Halt.
    c. Go.
    d. Punto de parada.
    e. Ejecución paso a paso.
    f. El intérprete de comandos.

12. Creación de un manual de usuario que permita, a personas interesadas y con conocimiento del proceso de depuración, utilizarlo con facilidad.

13. Realizar las iteraciones necesarias para establecer la validez de la metodología.

14. Establecer la experimentación necesaria para verificar la funcionalidad de la herramienta.

15. Implementar ajustes necesarios en el flujo de pruebas de PPV para acoplar la herramienta.

16. Situar la herramienta dentro del flujo de pruebas.

17. Formalizar experimentos que verifiquen el impacto de la herramienta en el flujo de pruebas.

CAPÍTULO 3
MARCO TEÓRICO

3.1 Descripción del sistema o proceso por mejorar.

El proceso en el cual se realizó el proyecto se encuentra dentro del flujo de ensamble y prueba que realiza Intel Costa Rica. En la figura 3.1 se muestra de forma resumida este flujo. PPV se encuentra en las etapas finales y para algunos es parte del control de calidad del producto debido a su capacidad para emular el ambiente de usuario final del cliente.

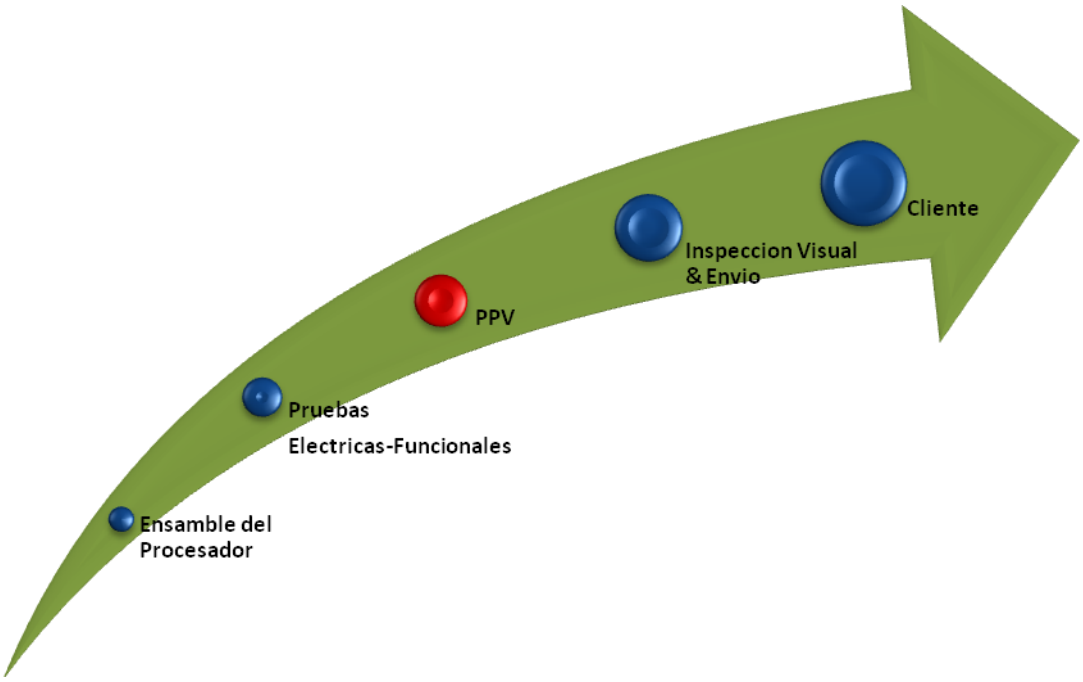


Figura 3.1 Posición de PPV en el flujo de ensamble y prueba de Intel.

Dados ciertos factores externos a nivel de pruebas y equipo es posible que una unidad funcionalmente buena falle en el conjunto de pruebas aplicadas en PPV. Es por esta razón que a nivel de manufactura una unidad puede ser probada un máximo de tres veces (para así minimizar la pérdida de unidades debido a la marginalidad del equipo). Cada intento es llamado sumario. Si una unidad falla en los tres sumarios es rechazada del flujo de producción para posteriormente pasar al departamento de Ingeniería de Desarrollo de Producto, en donde se realizará el diagnóstico de la causa funcional por la cual la unidad no logró ejecutar el conjunto de pruebas aplicado en PPV.

El proceso actual para el diagnóstico de las fallas es realizado por el *Ingeniero de desarrollo*, quien utiliza un hardware especializado para esta tarea llamado ITP (In-Target Probe, por sus siglas en inglés). Es un proceso repetitivo en el que se emplea gran cantidad de tiempo y esfuerzo para determinar la falla de una gran cantidad de unidades.

Luego de la determinación de la falla se procede a crear un reporte, compuesto básicamente, por un análisis de los datos obtenidos por los registros de monitoreo que posee el procesador, el cual es enviado a todos los interesados, incluyendo al equipo de diseño.

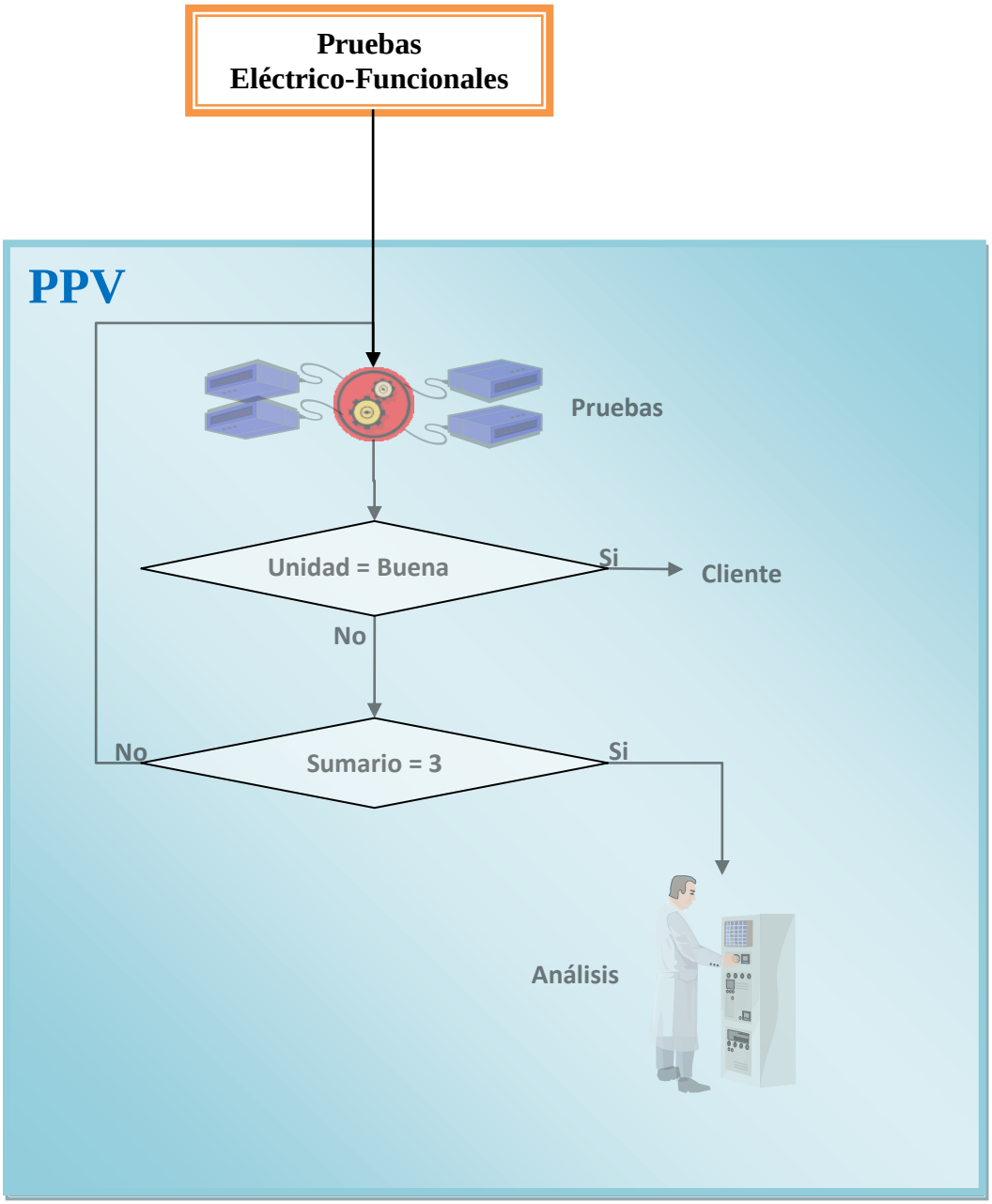


Figura 3.2 Diagrama de flujo simplificado del proceso de PPV y análisis.

En PPV se proyectó la utilización de los resultados del proyecto para mejorar el diagnóstico de fallas funcionales y el entendimiento del funcionamiento del procesador, lo cual ayudaría a mejorar el tipo de pruebas que se le realizan actualmente.

3.2 Antecedentes Bibliográficos.

El proyecto se desenvolvió en un ambiente de desarrollo por lo cual las referencias bibliográficas son limitadas y catalogadas como secreto empresarial, o simplemente no tienen referencia alguna.

Los antecedentes al proyecto fueron incursiones por parte del asesor del proyecto el cual de una manera superficial logró obtener resultados cuasi funcionales en lo que respecta a los objetivos del proyecto.

Las referencias de algo similar en Internet son prácticamente inexistentes.

3.3 Descripción de los principales principios electrónicos relacionados con la solución del problema.

Entre los principales principios electrónicos relacionados con el proyecto se encuentra la arquitectura del procesador.

3.3.1 Itanium® 2 & IA64 [1, 2]

En 1999 Intel anunció el lanzamiento de Itanium® 2, marca registrada para su línea de procesadores de 64 bits.

Ese fue el inicio de la familia Itanium® 2, en la cual la nueva generación de procesadores Itanium® 2 incluye características desarrolladas con el fin de competir con procesadores de alto rendimiento, ingresando en la competencia en el segmento de mercado de las súper computadoras.

La arquitectura de Itanium difiere dramáticamente de las arquitecturas x86 y X86-64, usadas en otros procesadores Intel. La arquitectura se basa en un paradigma computacional llamado “Instrucciones Computacionales Explícitamente Paralelas” ó EPIC (Explicit Parallelism Instruction Computing, por sus siglas en inglés).

La nueva generación de Itanium® 2 posee dos núcleos de procesamiento dual, es decir cuatro unidades lógicas de ejecución y un total de memoria Caché de 26.5 Megabytes.

Uno de los mayores beneficios de la arquitectura de 64 bits es la cantidad de memoria que puede ser direccionada: 16 Terabytes, 4 billones de veces más que con la arquitectura de 32 bits. Sin restar mérito al el impacto del incremento del direccionamiento, el avance más significativo se presenta con EPIC.
EPIC es un paradigma computacional que inició su investigación en la década de los noventa. La meta de EPIC es incrementar la habilidad del procesador

para ejecutar instrucciones en forma paralela, mediante el uso del compilador en vez de circuitos complejos, para determinar y mejorar las oportunidades de ejecuciones en paralelo.

Es una combinación innovadora de especulación, predicción y paralelismo explícito, lo que representa un avance en la tecnología de procesadores, por su capacidad para solventar las limitaciones de rendimiento que se encuentran en las tecnologías CISC* y RISC*. Las dos últimas han utilizado diferentes técnicas para tratar de procesar más de una instrucción a la vez donde fuera posible, y el grado de paralelismo en el código es determinado durante la ejecución por parte del procesador que trata de analizar y reordenar las instrucciones. Esta metodología implica una pérdida de tiempo, ya que reordena las instrucciones en vez de ejecutarlas. EPIC rompe la naturaleza secuencial de los procesadores convencionales, porque permite al software comunicar explícitamente al procesador cuándo una operación puede ser realizada en forma paralela.

Esta característica permite al procesador tomar un conjunto de instrucciones y ejecutarlas simultáneamente. En el caso de Itanium® 2 se ejecutan 6 instrucciones por ciclo de reloj. El aumento en el rendimiento es logrado mediante la reducción de saltos y predicciones erradas de estos mientras se disminuyen los efectos de la latencia entre el procesador y la memoria.

El conjunto de instrucciones representa otro gran avance en la arquitectura de los microprocesadores. Al utilizar la tecnología de EPIC se permite entregar una gran cantidad de recursos con escalabilidad inherente, lo cual no era posible en arquitecturas anteriores.

* Ver Glosario

En la arquitectura de Itanium las instrucciones están agrupadas en 3 dentro de un manojo de 128 bits, cada instrucción es de 41 bits, con 5 bits usados como plantilla. La plantilla especifica qué tipo de unidad de ejecución procesa la instrucción. En las figuras 3.3 y 3.4 se representa el formato y un grupo de instrucciones que se ejecutan en forma paralela.

```
"{.m|mi|  adds r4 = 0x6f0, r0 ; mov r1 = gsr[r4]; nop.i 0x0 }"
```

Figura 3.3 Formato de instrucción en la arquitectura de Itanium.

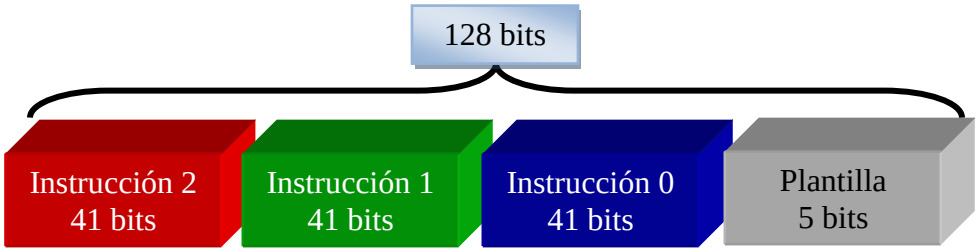


Figura 3.4 Estructura del grupo de instrucciones por ejecutar en paralelo.

Itanium® 2 tiene gran cantidad de recursos computacionales que incluyen:

- 128 registros de uso general.
- 128 registros de punto flotante.
- 6 ALU's de uso general, 2 unidades de enteros y 1 unidad de desplazamiento.
- 4 unidades de datos para Caché.
- 6 unidades de multimedia.
- 4 unidades de punto flotante.
- 3 unidades de saltos.
- Tecnología Multi-hilo, permite tener 2 procesadores lógicos por cada núcleo físico. En este caso son 2 núcleos físicos que generan 4 procesadores lógicos.

Esas son algunas de las características que presenta el procesador Itanium® 2.

3.3.2 IEEE 1149.1a-1993, JTAG/TAP [3].

Entre las características que se definen en el estándar, se busca proveer de una solución al problema de realizar pruebas a circuitos impresos ensamblados y otros productos basados en circuitos digitales de alta densidad y complejidad. También se busca aportar los medios para acceder y controlar infraestructuras de diagnóstico y pruebas construidos dentro de los circuitos integrados digitales.

El estándar nació en 1985 con el nombre de JETAG, formado por miembros exclusivamente europeos. Luego de la adhesión de países norteamericanos en 1986, cambió su nombre al actual JTAG. Su aprobación inicial se dio en febrero de 1990, con lo que se inicia una serie de esfuerzos entre los desarrolladores y usuarios para la creación de un suplemento con el propósito de corrección, clarificación y mejora del estándar, lo cual culminó con el estándar IEEE std 1149.1a-1993, aprobado en Junio de 1993.

3.3.2.a Alcance.

El estándar define la lógica de pruebas que puede ser incluida e integrada al circuito para proveer metodologías estandarizadas a:

- Pruebas a las interconexiones entre circuitos una vez que hayan sido ensamblados en un circuito impreso u otro tipo de sustrato.
- Auto pruebas de un circuito integrado.
- Observación ó modificación de la actividad del circuito durante la operación normal del componente.

La lógica de Pruebas consiste en registros llamados Boundary-Scan y otros bloques constitutivos accedidos a través del TAP.

3.3.2.b Perspectiva General.

La circuitería definida por estándar permite que las instrucciones de prueba y los datos asociados a éstas sean alimentadas a los componentes y, subsecuentemente, permite que los resultados de las ejecuciones de tales instrucciones puedan ser leídos externamente. Toda la información (instrucciones, datos de pruebas y resultados de las pruebas), es comunicada en formato serial.

La secuencia de operación será controlada por un bus maestro. El control se logra mediante la aplicación de una señal a la entrada del TMS y al TCK.

El primer paso es cargar en forma serie, dentro del componente, el código de la instrucción por ejecutar en forma hexadecimal. Una vez cargada la instrucción el circuito de pruebas seleccionado es configurado para actuar.

En algunos casos es necesario cargar datos en el circuito de pruebas seleccionado para obtener una respuesta. Estos datos son cargados en serie de forma análoga a las instrucciones. Cabe destacar que el movimiento del dato de pruebas no tiene efecto alguno en la instrucción presente en la circuitería de pruebas. Un ejemplo sería cargar la instrucción de suma y sus correspondientes argumentos (los cuales representan los datos) para así obtener una respuesta.

Luego de la ejecución de la instrucción de prueba el resultado puede ser examinado mediante el desplazamiento de los datos fuera del componente por medio del bus maestro.

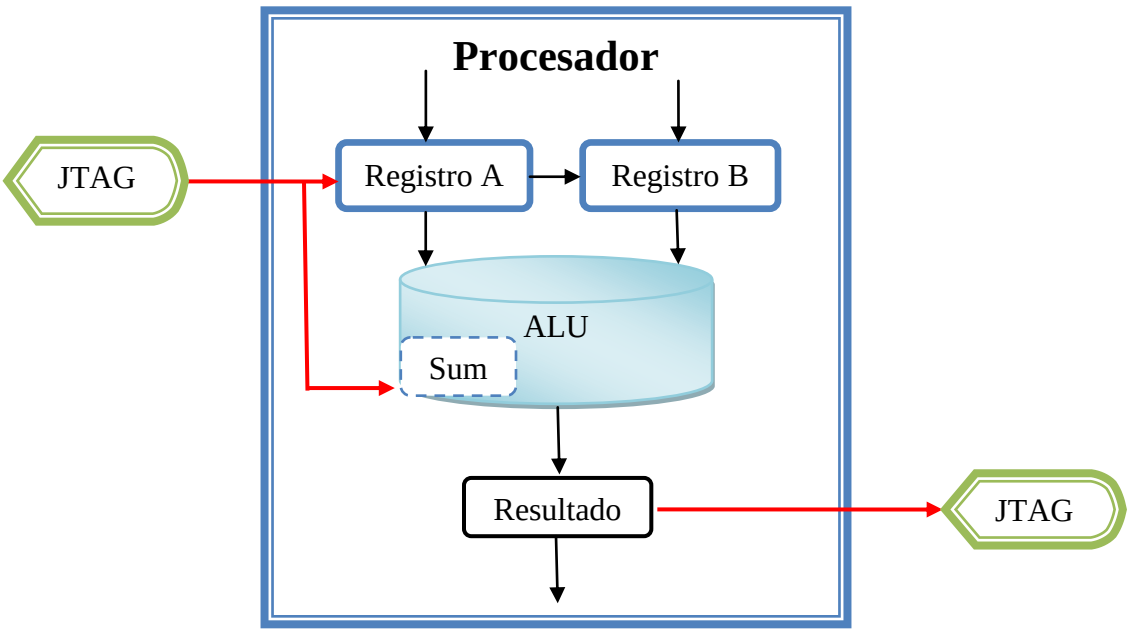


Figura 3.5 La líneas en rojo simulan la inyección de instrucciones y datos a través del TAP usando el protocolo del JTAG.

Varias operaciones de pruebas pueden ser ejecutadas de manera similar a la descrita anteriormente mediante la ejecución de una serie de instrucciones especiales.

3.3.2.c Uso del estándar en productos ensamblados.

Para las pruebas de un producto conformado por una colección de distintos componentes, el problema se divide en tres metas:

1. Comprobar que cada componente lleva a cabo su función.
2. Confirmar que los componentes están interconectados de manera correcta.
3. Confirmar que los componentes interactúan de manera correcta y que el producto logra su cometido.

3.3.2.d Boundary-Scan

Es una técnica que involucra la inclusión de registros de desplazamiento adyacente a cada pin del componente para que cada señal en la frontera del componente pueda ser controlada y observada utilizando los principios de Pruebas.

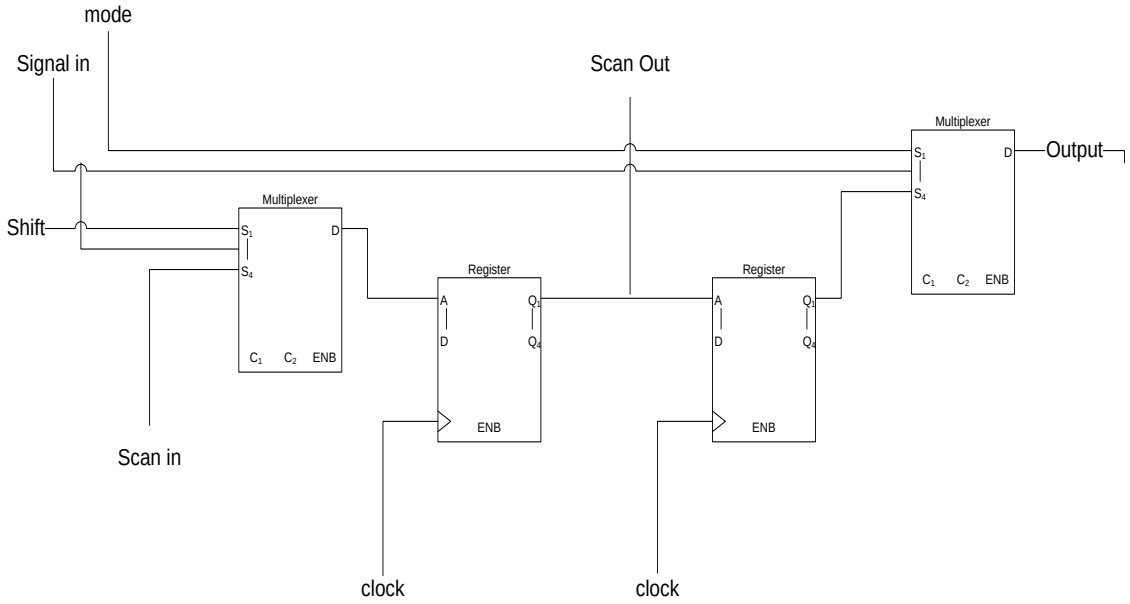


Figura 3.6 Una celda de un registro perteneciente al Boundary-Scan.

En la figura 3.6 se muestra un ejemplo de implementación de una celda del registro de Boundary-scan que puede ser usada como conexión de entrada o salida a un circuito integrado. Dependiendo de una señal de control aplicada al multiplexor, el dato puede ser leído o escrito.

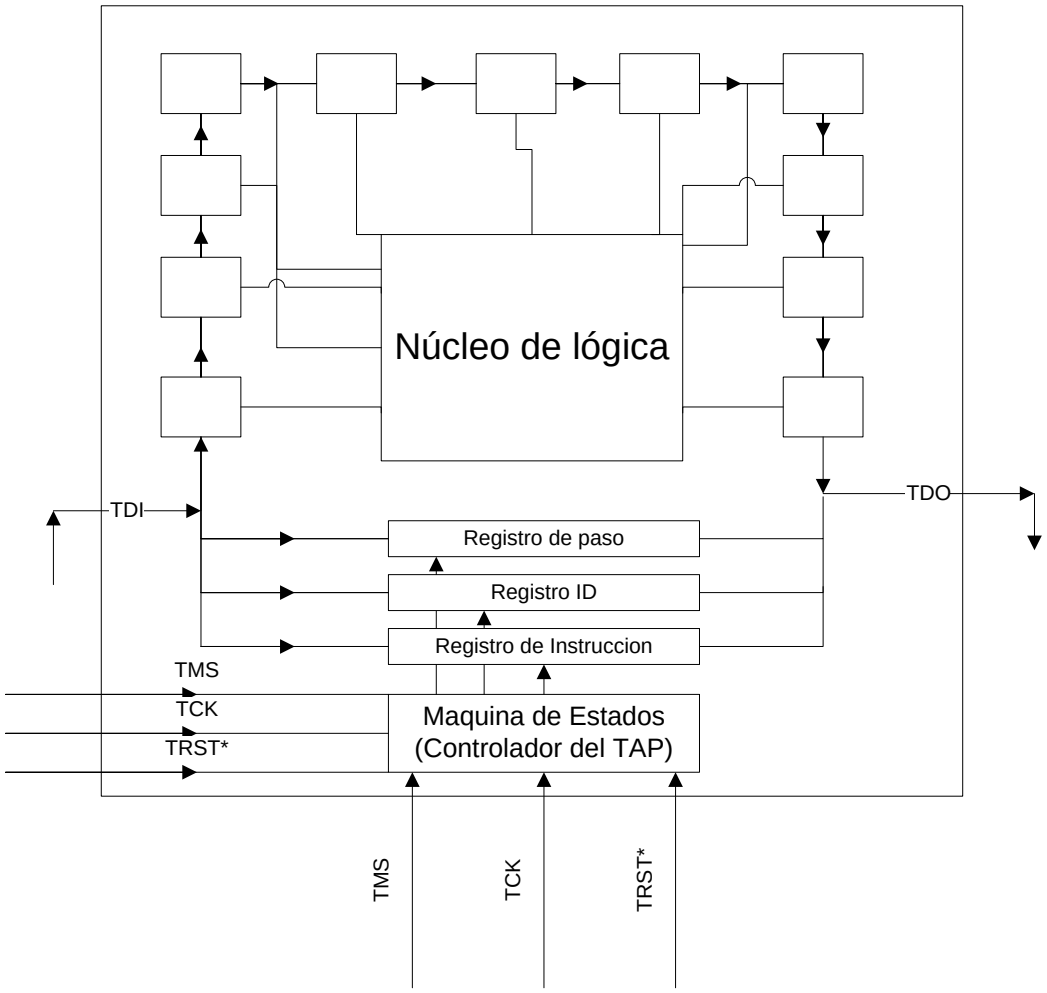


Figura 3.7 Ejemplo de un Chip con Boundary-Scan.

En la figura 3.7 se muestra un chip con una serie de registros que forman el Boundary-Scan en el perímetro.

3.3.2.e TAP.

El TAP es un puerto de propósito general, que provee acceso a muchas funciones de pruebas construidas dentro de un componente electrónico, incluyendo la lógica de prueba definida por el estándar. El TAP está compuesto por un mínimo de tres entradas y una salida, para cumplir con los requerimientos del estándar. Una entrada opcional puede ser implementada para proveer una inicialización asincrónica de la lógica de pruebas.

Las conexiones del TAP son:

- ✓ TCK (Test Clock Input): esta entrada aporta la temporización para la lógica de pruebas. Es un reloj independiente del reloj general del sistema, con lo que se garantiza independencia de los circuitos en caso de falla. Esta independencia permite que los datos de las pruebas puedan ser leídos sin cambiar el estado ni afectar la ejecución del sistema.

- ✓ TMS (Test Mode Select): la señal recibida en TMS es decodificada por el controlador de TAP para gobernar las operaciones de prueba. Los valores de la señal presentes en TMS son muestreados por la lógica de pruebas en el flanco creciente de TCK.

- ✓ TDI (Test Data Input): En este puerto se reciben las instrucciones y datos de prueba. Provee “movimiento serie” a los datos de prueba a través del circuito. Los valores presentes en TDI son muestreados en el flanco creciente de TCK.

- ✓ TDO (Test Data Output): “Salida serie” para las instrucciones y datos de la lógica de pruebas. A diferencia del TDI, esta salida es muestreada en el flanco decreciente del reloj TCK.

- ✓ TRST (Test Reset Input): Entrada opcional que provee una inicialización asincrónica del controlador del TAP.

El TAP es gobernado a través de un controlador, el cual es, básicamente, una máquina de estados finita que se presenta en el siguiente diagrama de estados.

Todas las transiciones de la máquina de estados del TAP deben ocurrir basadas en el valor de TMS al momento del flanco creciente de TCK.

Las acciones de la lógica de prueba (instrucciones, datos, etc.), deben ocurrir ya sea en el flanco creciente o decreciente de TCK en cada estado del controlador.

Algunas de las instrucciones que formaron parte del proyecto son las siguientes:

- ✓ Run-Test/Idle: estado del controlador en medio de operaciones de lectura ya sea de una instrucción o datos.
- ✓ Shift-DR: en este estado del controlador el registro de datos conectado en medio de TDI y TDO es cargado con los datos de los resultados de las pruebas realizadas. Este valor es desplazado a TDO.
- ✓ Pause-DR: permite mantener los datos previamente cargados.
- ✓ Shift-IR: en este estado la instrucción es desplazada a TDO.
- ✓ Pause-IR: permite mantener las instrucciones previamente cargadas.

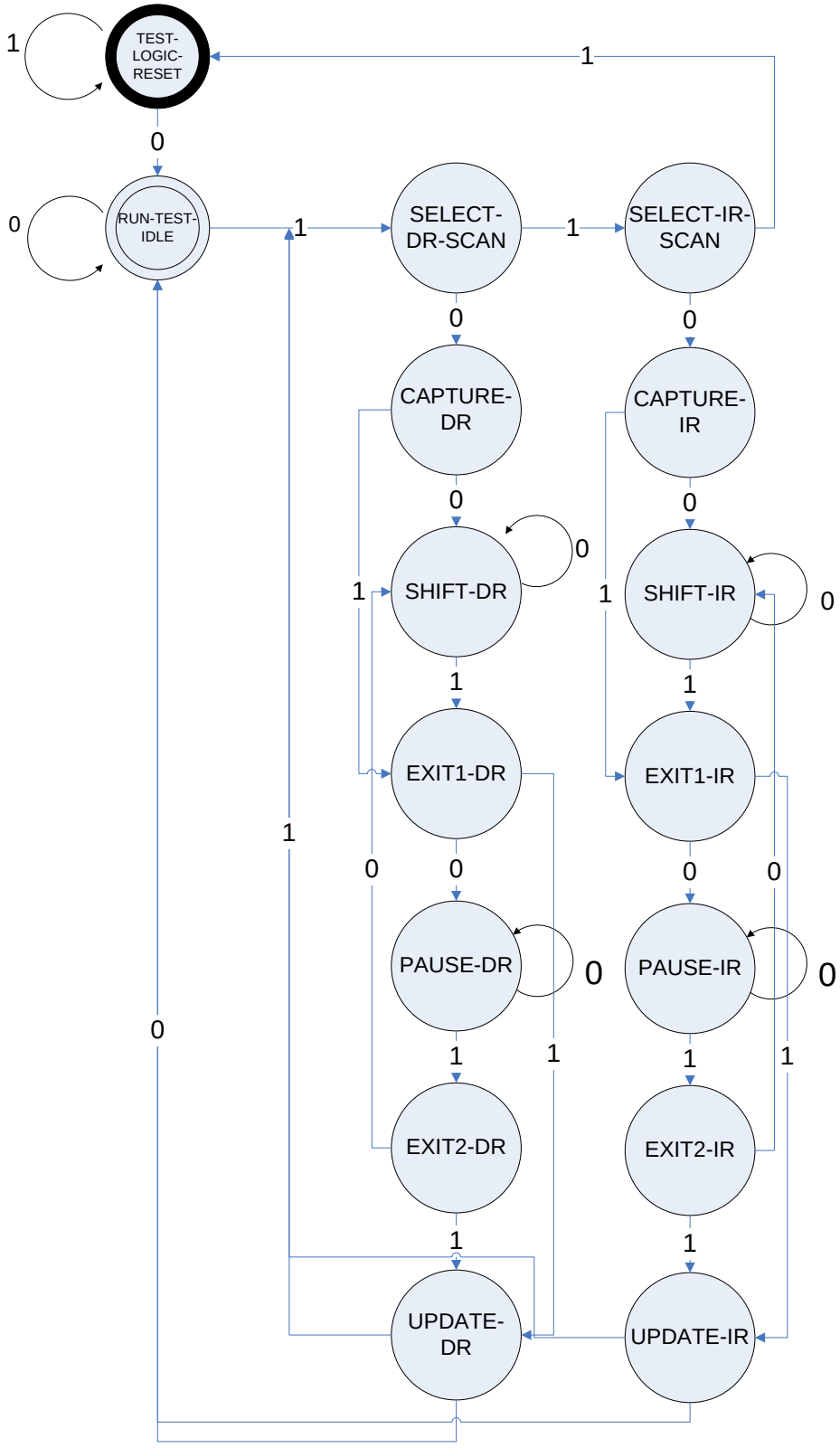


Figura 3.8 Diagrama de estados para el controlador del TAP.

Para efectos del proyecto se trabajó directamente con este estándar y los componentes descritos anteriormente. Además, se contó con diversos componentes de diagnóstico y pruebas que incluye la arquitectura de Itanium® 2, y permiten un diagnóstico de su funcionamiento.

Con lo anterior se ilustra parte de los elementos y principios electrónicos utilizados.

En resumen se destaca:

1. La utilización de instrucciones en ensamblador.

2. Utilización de componentes electrónicos, como interfaces y controladores.

3. Todos los conceptos de direccionamiento y acceso a unidades de memoria interna del procesador.

4. El proceso de investigación de arquitecturas auxiliares.

5. Estándares de la IEEE.

6. Desarrollo de algoritmos.

CAPÍTULO 4
PROCEDIMIENTO METODOLÓGICO

4.1 Reconocimiento y definición del problema.

1. Realización de entrevistas con expertos con el fin de entender el problema y su contexto (15 días).
2. Determinación de las funciones que se pueden realizar por medio del TAP (15 días).
3. Comprensión y acercamiento al proceso de validación del producto, PPV (15 días).
4. Definición del resultado requerido por la empresa (15 días).

4.2 Obtención y análisis de la información.

1. Obtener, desde la base de datos, las especificaciones técnicas del estándar de la IEEE.(5 días)
2. Obtener, desde la base de datos, las especificaciones del procesador Itanium®. (5 días).
3. Obtener, desde la base de datos, la guía de usuario del lenguaje ensamblador de 64 bits.(5 días)
4. Obtener las especificaciones del equipo utilizado en PPV y del software.(15 días)
5. Estudiar el estándar de la IEEE con el fin de determinar sus capacidades y limitantes. (15 días)
6. Investigar el funcionamiento de los componentes de hardware y software en PPV. (15 días)
7. Estudiar la arquitectura de Itanium® (15 días)
8. Estudiar el lenguaje ensamblador de 64 bits. (15 días)

4.3 Evolución de las alternativas y síntesis de una solución.

1. Experimentación con las funciones del JTAG para establecer limitantes y capacidades (15 días).
2. Experimentación con las instrucciones de ensamblador, con el fin de familiarizarse con ellas (15 días).
3. Definición de hipótesis sobre las posibles soluciones (15 días).
4. Documentación las experiencias y descarte de hipótesis (15 días).

4.4 Implementación de la solución.

La implementación de los resultados en una posible herramienta incluyó:

1. Recolectar información de las limitantes y capacidades del JTAG (15 días).
2. Simplificar, mediante un diagrama de flujo, la funcionalidad del JTAG (5 días).
3. Determinar las capacidades y limitantes de la arquitectura del procesador (15 días).
4. Simplificar, mediante un diagrama de flujo, la funcionalidad del procesador (5 días).
5. Documentar los datos obtenidos sobre el JTAG y el procesador (15 días).
6. Recolectar información sobre las instrucciones necesarias para crear los procedimientos requeridos (15 días).
7. Crear metodologías y procedimientos que permitieran el control del procesador (15 días).
8. Establecer algoritmos que describieran la metodología (15 días).
9. Codificar los algoritmos en un lenguaje de programación de alto nivel (15 días).
10. Crear una herramienta que incluyera todos los algoritmos necesarios (20 días)
11. Documentar el código y los algoritmos (5 días).

12. Documentar los efectos de la utilización de la herramienta (15 días).
13. Presentar un reporte del trabajo realizado, en el que estuvieran claramente expuestos los datos obtenidos tanto en la parte de limitantes como en la de capacidades (15 días).
14. Establecer conclusiones simples y claras del trabajo realizado (5 días).

4.5 Reevaluación y rediseño.

Fueron sugeridas las siguientes acciones que permitieran la reevaluación y rediseño de la herramienta y la investigación realizada

1. Planteamiento de nuevas hipótesis (5 días).
2. Sugerencia de nuevas metodologías (5 días).
3. Determinación de los perfiles de funcionamiento de la herramienta y sugerencia de mejoras (5 días).
4. Solicitud de una reunión con expertos para confrontar ideas (5 días).

CAPÍTULO 5
DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN

5.1 Análisis de soluciones y selección final.

Durante el proyecto se plantearon dos soluciones, de las cuales se seleccionó una para proceder a implementarla. Dicha solución presentó modificaciones al método inicial propuesto.

Ambas soluciones tenían como común denominador el uso del estándar JTAG y las instrucciones de ensamblador necesarias para cumplir con los objetivos.

La primera opción consistía en crear una biblioteca con los distintos algoritmos que cumplieran con los requisitos. Las primeras ideas apuntaban a que la biblioteca estuviera dentro de la memoria del sistema, en este caso el disco duro de la TIU. Se descartó esta opción pues si se lograba que el sistema entrara en modo interrumpido “Halt” no podría ejecutarse ninguna instrucción a través del Bus de datos ni del JTAG, ya que el sistema estaría completamente detenido y no se podrían realizan los accesos al disco duro. Por ende la biblioteca no lograría ejecutarse. Este análisis concluyó que la solución debía darse por medio de un sistema independiente al sistema objetivo.

Se determinó entonces el uso del ambiente de desarrollo de Systest y el hardware de la Centrix como plataforma para lograr los objetivos.

Systest tiene enclavadas funciones que gobiernan los diferentes dispositivos de la Centrix y que permiten manipular el JTAG. Se tomaron como referencia los logros del desarrollo anterior, el cual explota estas capacidades. La ventaja que presentó el uso de esta solución es que el sistema es completamente independiente del sistema a atacar, por lo que la captura de datos y la inyección de las instrucciones estarían sujetas al hardware del sistema. En la figura 5.1 se muestra una representación de la solución escogida.

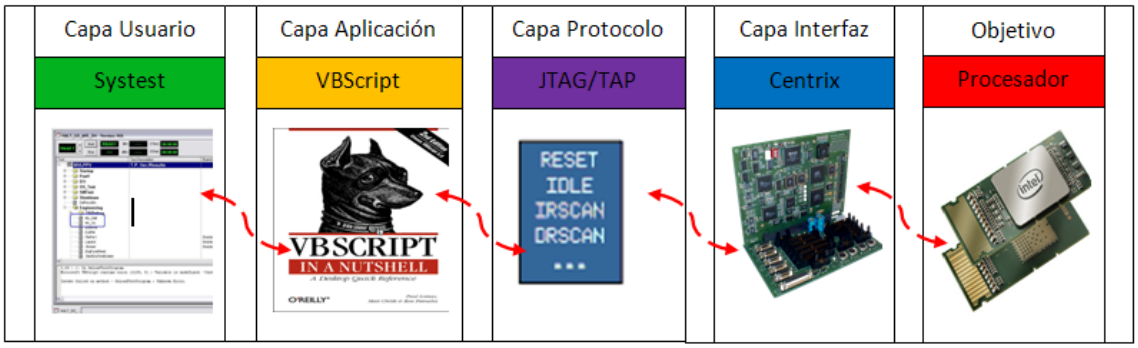


Figura 5.1 Diagrama de alto nivel de la solución seleccionada

Se realizó un estudio de la arquitectura y la metodología utilizadas por parte de la herramienta ITP como base para generar los algoritmos, el código y los datos a usar.

El análisis de la metodología de “Halt-Go” indicó que es necesario resguardar los datos contenidos en ciertos registros durante la ejecución de “Halt” con el fin de restaurarlos en la ejecución de “Go” para así devolver el sistema a su estado original antes de la interrupción. Además de resguardar los datos contenidos en esos registros, también se determina la configuración de registros de operación, los cuales llevan al procesador a un modo de operación que permite inyectar instrucciones al “Pipe-Line”.

Esta solución presentó todas las prácticas necesarias para la consecución de los objetivos planteados en el proyecto.

5.1.1 Interfaz de PPV [4].

La interfaz necesaria para el sistema de pruebas en PPV se especifica de la siguiente manera:

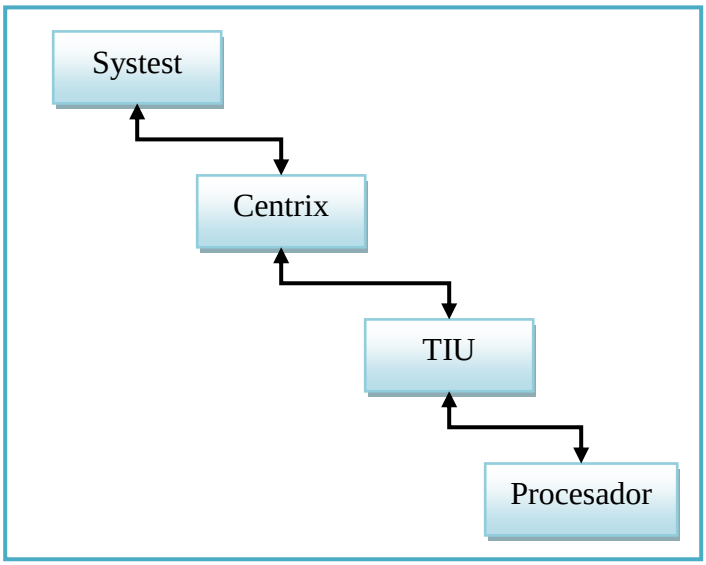


Figura 5.2 Diagrama de bloques de la interfaz de PPV.

Se observan los distintos niveles de trabajo, en donde para el usuario los niveles intermedios son casi invisibles para lo que se refiere a este proyecto. Systest es la herramienta con la cual se interactúa directamente, mientras que la Centrix y la TIU son los medios por los cuales se accede al procesador.

5.2 Descripción del hardware.

La descripción del hardware se enfoca en los elementos más importantes para el desarrollo del proyecto. No se describirán elementos como tarjeta madre y demás ya que no son de relevancia.

5.2.1 Módulo de PPV.

El modulo de PPV para Itanium consta de un equipo llamado Racal, que está compuesto por una PC con Systest instalado y varias fuentes que alimentan a 4 TIU. En la figura se observa un Modulo de PPV típico.

Figura 5.3 Fotografía frontal de un TIU de Itanium.

5.2.2 TIU.

La TIU es el equipo requerido en PPV para realizar las pruebas al procesador. TIU se define como Unidad de Interfaz de Prueba (Test Interface Unit, por sus siglas en inglés). La TIU es, básicamente, una computadora con gran variedad de componentes extra que permiten simular diferentes condiciones de operación del procesador. En la figura 5.4 se muestra un diagrama de bloques básico de lo que es una TIU.

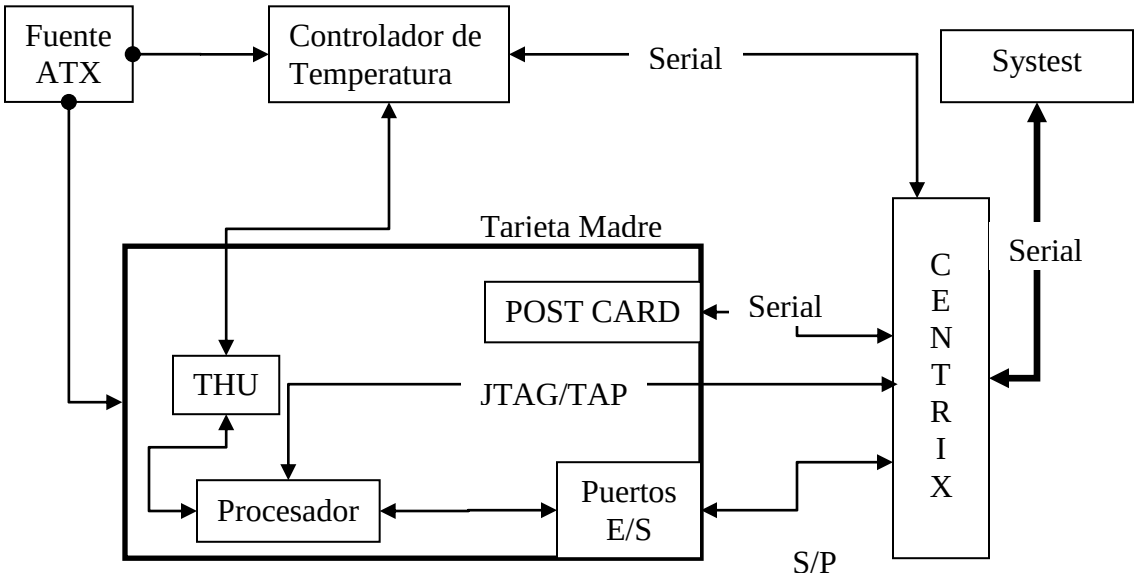


Figura 5.4 Diagrama de bloques de los componentes básicos de una TIU.

Como se observa en le diagrama todos los componentes están conectados ya sea directa o indirectamente con la Centrix, también se resalta la conexión directa del TAP con la Centrix y la del bus de datos en forma separada. Con este diagrama se explica más claramente la interfaz necesaria para la realización del proyecto (en donde también se plantea la manipulación de la Centrix como puente entre la herramienta y el TAP).

Figura 5.5 Fotografía frontal de 2 TIU de Itanium.

5.2.3 Centrix.

La Centrix es la interfaz entre Systest y los componentes que integran a la TIU. Con la Centrix se realiza el control y la comunicación con los distintos componentes, así como con el procesador.

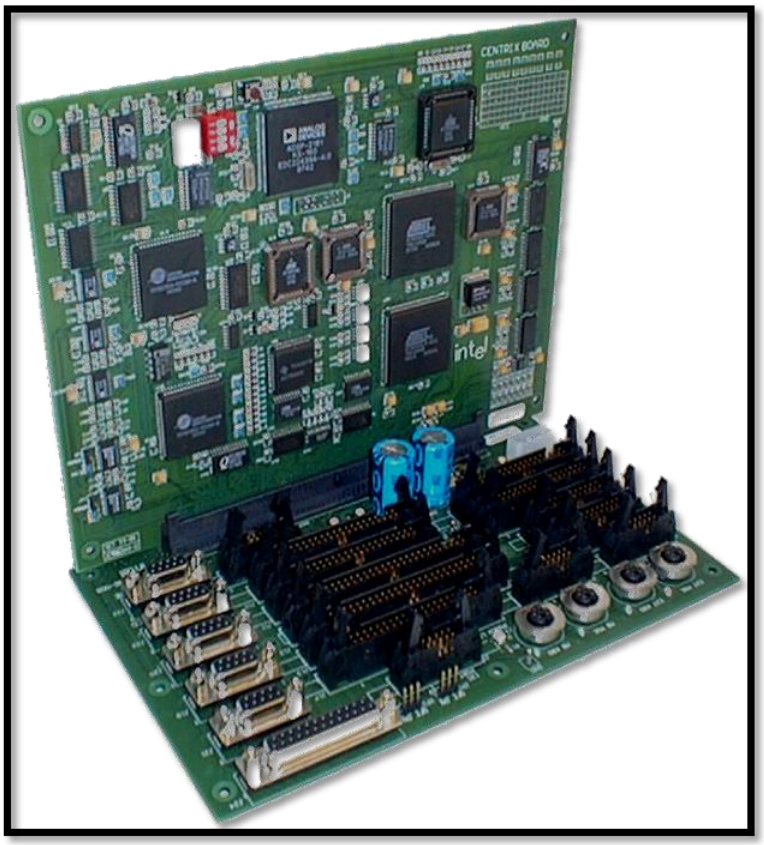


Figura 5.6 Fotografía de una tarjeta Centrix.

La Centrix es una tarjeta con gran variedad de circuitos procesadores de señales digitales, así como circuitos de entrada y salida digital. Su objetivo primordial es proveer a Systest de control y comunicación sobre el TIU.

Esta tarjeta aporta el hardware necesario para la comunicación con el puerto de pruebas TAP. A través del control de la Centrix se pretende realizar el control del TAP y a su vez del procesador, como se muestra en la figura 5.7.

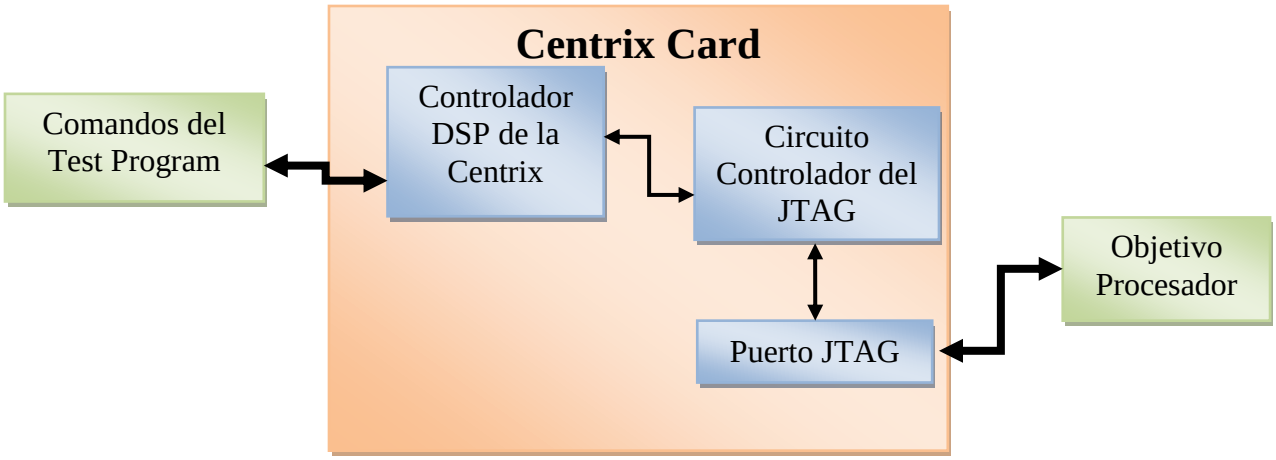


Figura 5.7 Diagrama de una tarjeta Centrix.

5.2.4 ITP [7].

ITP es una herramienta de depuración creada por Intel, que consiste en hardware y software que permiten observar y controlar la ejecución tanto de un programa como del procesador, permitiendo identificar la ubicación y causa del error para así corregirlo.

Esta herramienta se conecta directamente al TAP del sistema y es utilizada como parámetro de medición con respecto a los resultados obtenidos (con ella se realizaron los ensayos de los objetivos a alcanzar en este proyecto).

En las figuras 5.8 y 5.9 se muestra un ejemplo de conexión del ITP a un sistema así como su interfaz de usuario (en donde se observan varias de las funciones).

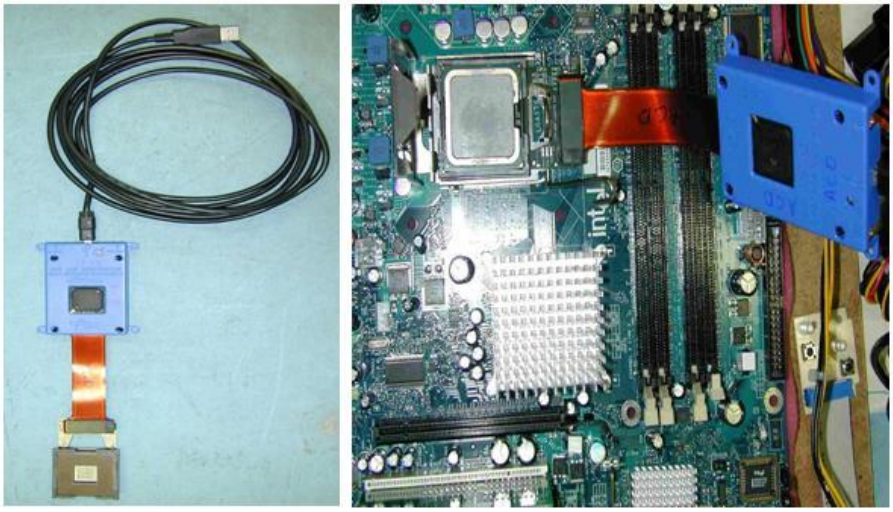


Figura 5.8 Fotografía del ITP

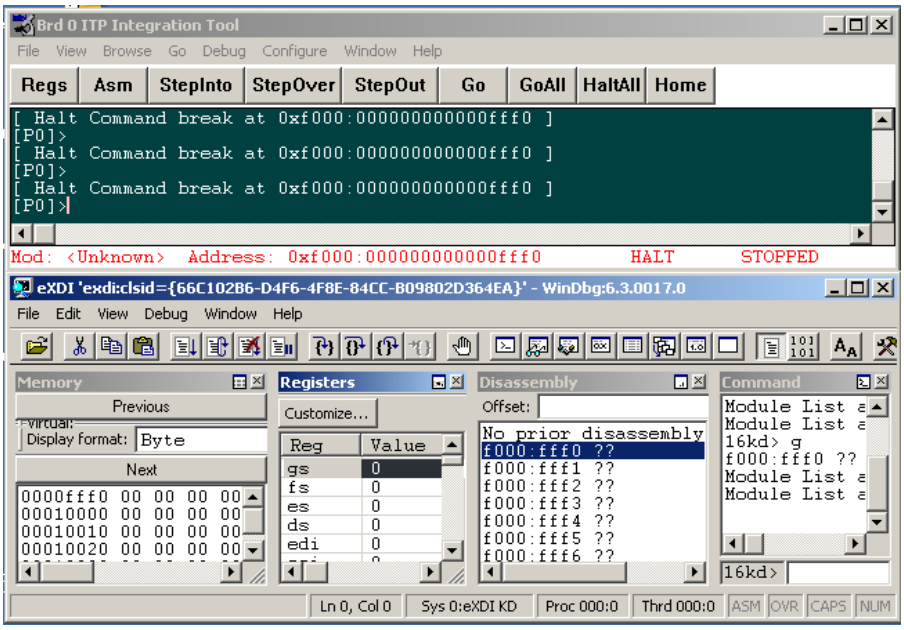


Figura 5.9 Ambiente gráfico de usuario del ITP

5.3 Descripción del software.

5.3.1 Systest.

Es la interfaz de usuario que integra un conjunto de módulos, herramientas, menús y otros elementos que permiten crear, probar y refinar los Programas de Pruebas en un mismo lugar. Systest da al desarrollador la facilidad de controlar y programar diferentes pruebas y condiciones que se le aplican al procesador en PPV. El lenguaje nativo de Systest es VBScript, que por su facilidad de uso es el preferido para llevar a cabo las tareas en PPV (aunque esta contrasta con las limitaciones inherentes si se compara con lenguajes compilados como por ejemplo C++).

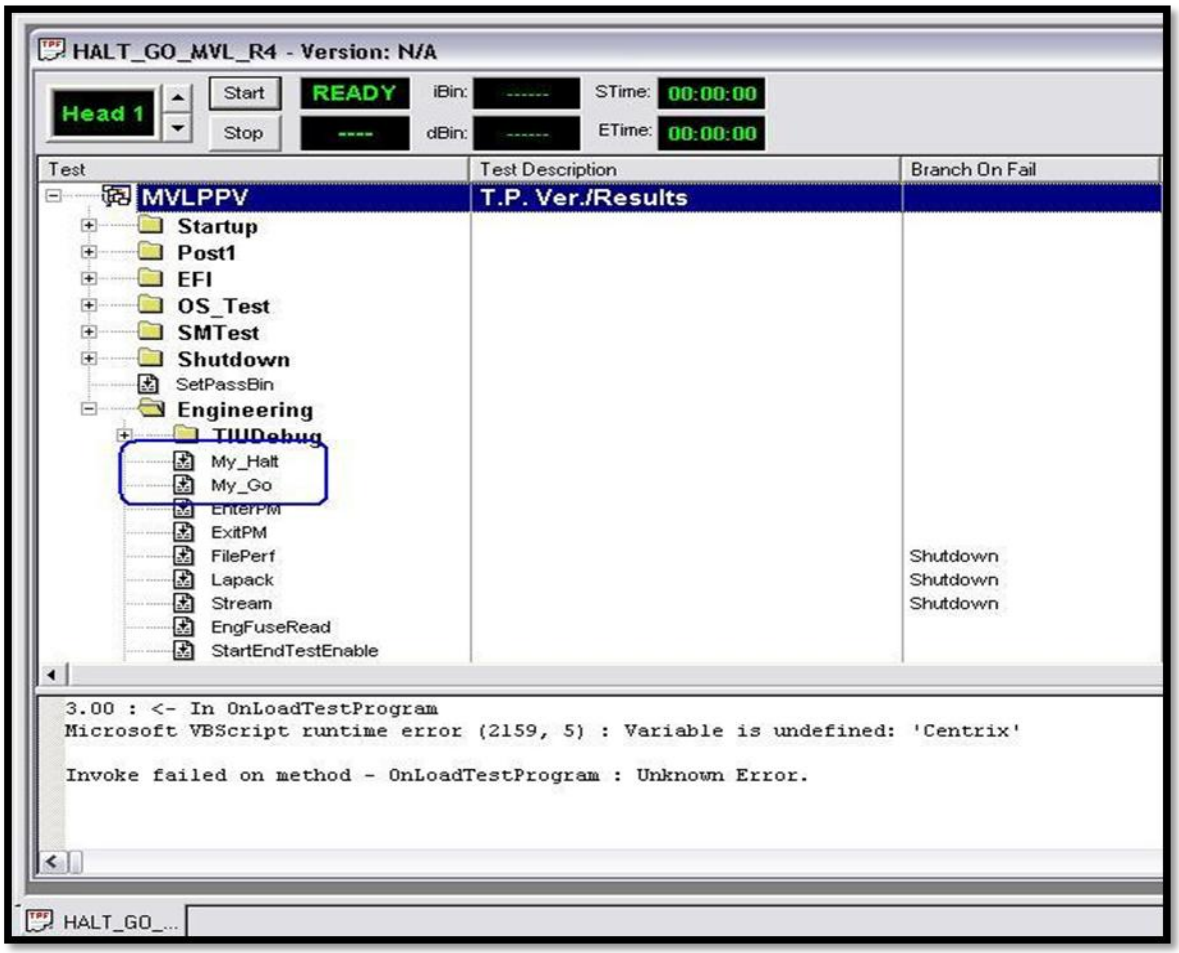


Figura 5.10 Interfaz de usuario de Systest.

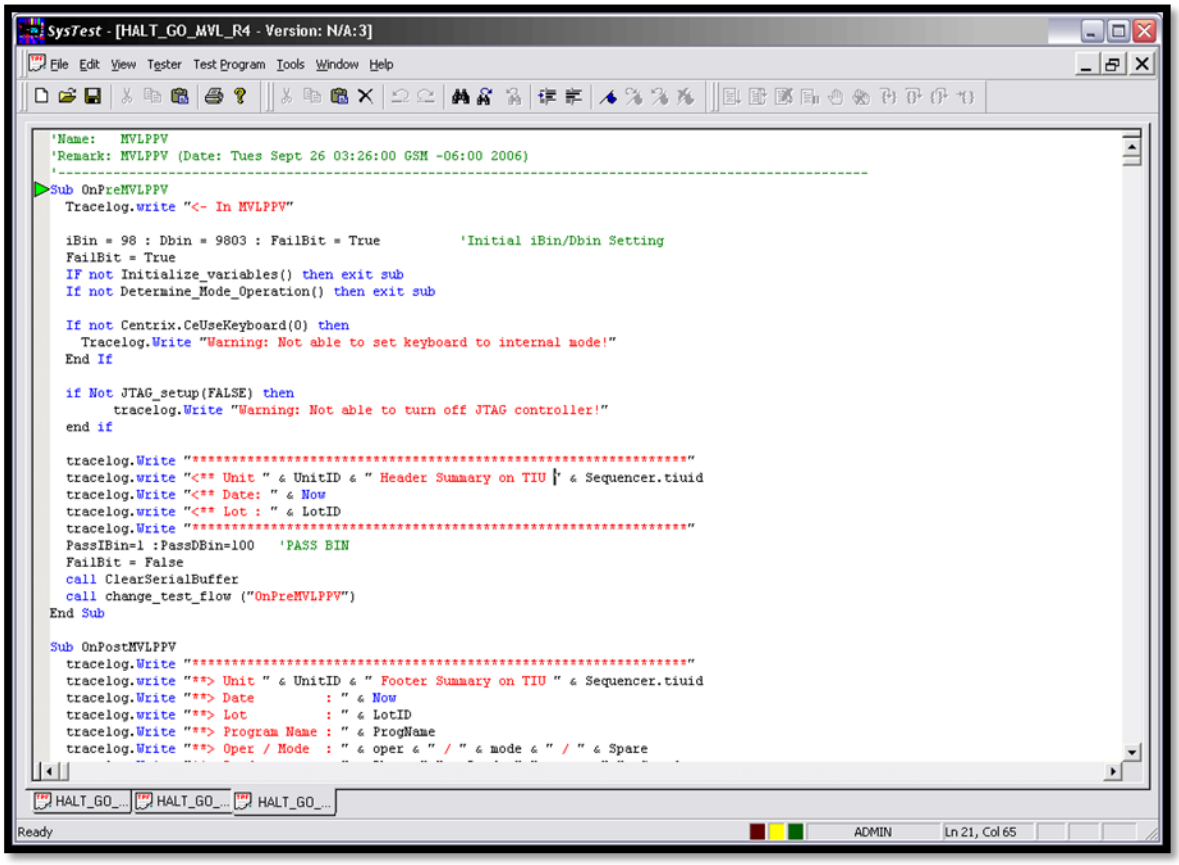


Figura 5.11 Interfaz de programación de Systest.

5.3.2 Diseño de los algoritmos.

El diseño del algoritmo que realiza las acciones necesarias para la consecución de los objetivos se realizó mediante el estudio de la arquitectura del procesador y del estándar de la IEEE.

El algoritmo consta de 6 partes fundamentales que describen en la figura 5.12, en donde HALT y GO son los pilares en el cual se sostiene el proyecto.

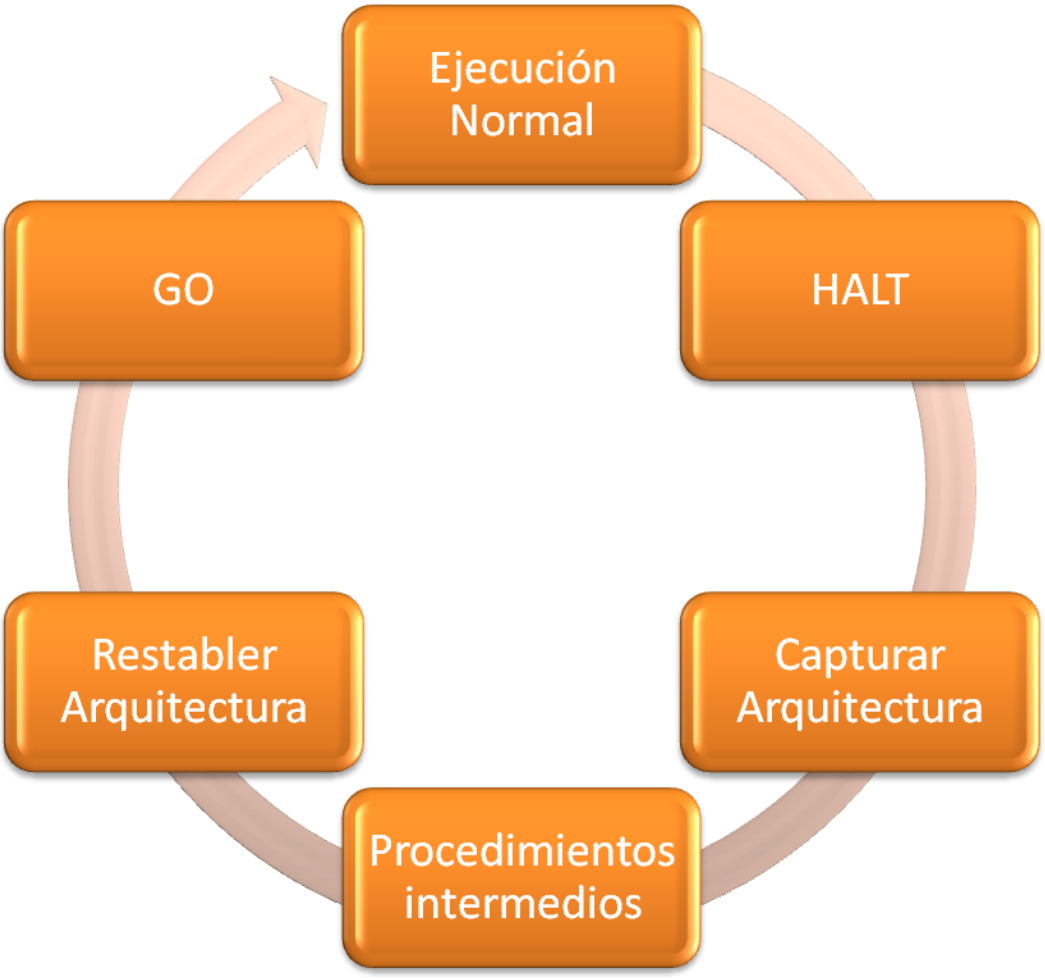


Figura 5.12 Diagrama de flujo de alto nivel del algoritmo desarrollado.

Ejecución Normal: representa la operación en condiciones normales de un procesador mientras ejecuta una aplicación.

HALT: representa el momento en donde se establecen las condiciones que detienen la operación normal del procesador.

Capturar arquitectura: representa la etapa del código -dentro de Halt- en donde se almacenan los valores de registros de configuración y de trabajo para luego ser restaurados.

Procedimientos intermedios: representan las diferentes actividades futuras que podrán ser desarrolladas utilizando el TAP, como la ejecución de instrucciones especificas a alguna unidad funcional.

Restablecer la arquitectura: representa la etapa del código en donde se restauran los valores de los registros -antes almacenados- que por distintas operaciones han sido modificadas.

GO: representa la etapa en la cual se libera el procesador -con su estado de arquitectura en el momento en que se solicito la detención - (HALT).

5.3.3 Estructura de datos y transferencia.

Antes de entrar en detalles de las funciones, se debe establecer la estructura y formato en los que las instrucciones y los datos son inyectados usando el TAP.

El procesador no compila las instrucciones necesarias, por lo que una pre-compilación es requerida.

Las instrucciones son estructuras de 128 bits que sumadas a los bits de configuración representan un total de 144 bits que son inyectados al sistema. El mnemónico “Load” indica las transferencias de instrucciones.

Los datos están estructurados por 64 bits de datos más bits que definen el registro de origen o destino y las operaciones de lectura y escritura entre otros. En este caso el mnemónico “LoadReg” indica una transferencia de datos.

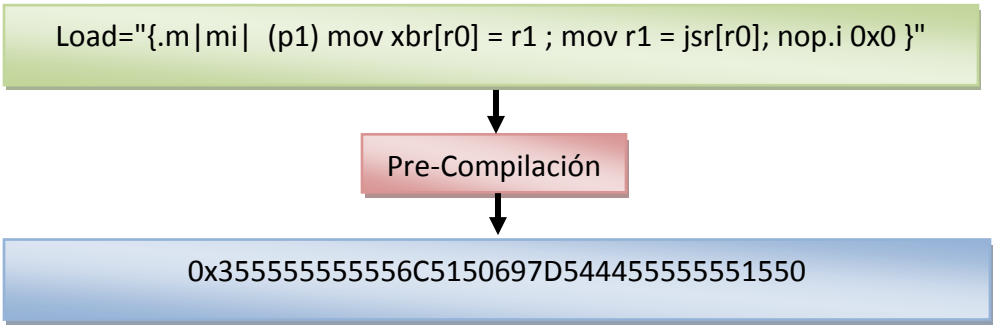


Figura 5.13 Ejemplo de Load y su equivalente hexadecimal.

Una vez realizada la pre-compilación se procede a acomodar la forma de introducir los datos. La estructura de datos se define sobre un arreglo llamado DataToSend, como se observa en el recuadro. Al ser una comunicación serial el LSB es el primer dato por enviarse.

```
DataToSend(0)=&h50
DataToSend(1)=&h15
DataToSend(2)=&h55
DataToSend(3)=&h55
DataToSend(4)=&h55
DataToSend(5)=&h45
DataToSend(6)=&h44
DataToSend(7)=&hD5
DataToSend(8)=&h97
DataToSend(9)=&h06
DataToSend(10)=&h15
DataToSend(11)=&hC5
DataToSend(12)=&h56
DataToSend(13)=&h55
DataToSend(14)=&h55
DataToSend(15)=&h55
DataToSend(16)=&h55
DataToSend(17)=&h35
```

Figura 5.14 Formato del arreglo que contiene la instrucción a ejecutar.

De esta forma se introducen instrucciones al TAP usando una función llamada "JtagScan" (presente en Systest) cuya sintaxis es la siguiente:

```
jtag_Dscan("State", OUT, IN, size)
```

Figura 5.15 Función Jtag_Dscan.

En esta, el "State" precisa el estado en el que se encuentra la máquina de estados del JTAG; "OUT" es definido por una variable que contiene los datos a ser enviados; "IN" es la variable que almacena los valores a recibir; y "size" determina el tamaño de las variables en bits.

Los datos son introducidos de manera distinta, utilizando dos funciones llamadas “ReadReg” y “WriteReg” escritas para una aplicación previa a este proyecto; estas funciones fueron modificadas para poder satisfacer necesidades específicas de este proyecto. Ambas funciones requieren de ciertos parámetros para su correcto funcionamiento. Entre ellos está el valor del registro a leer ó escribir, el núcleo físico en el cual reside el registro, el hilo de ejecución lógico y, finalmente, el valor a escribir. El tamaño de la trama de datos es de 80 bits compuestos por el siguiente formato:

```
dato[0]       : bit tonto
dato[2:1]     : leer/escribir → leer = '10' , escribir = '01'
dato[3]       : Núcleo a accesar → Núcleo 0 = '0' , Núcleo 1 = '1'
dato[14:4]    : Registro a accesar (Hex)
dato[78:15] : dato
dato[79]      : bit tonto
```

Figura 5.16 Estructura de datos para LoadReg.

Con la estructura de datos definida se procede con las funciones que hacen uso de ella para el proyecto.

5.3.4 HALT.

Halt, es el proceso en el cual se ejecutan, a través de TAP, las instrucciones necesarias para detener la operación normal del procesador. Aquí se trabaja directamente con la unidad de ejecución que permite movilizar datos, a través de la arquitectura del procesador, a un medio legible a través del TAP y almacenado por el código de VBScript en Systest.

Entre las técnicas utilizadas para la manipulación de datos se utilizan los modos de direccionamiento de registros y direccionamiento indirecto mediante registros. Como se menciona en el marco teórico las operaciones van en un conjunto formado por 3 instrucciones.

La descripción del algoritmo que fue utilizado para desarrollar Halt se muestra en el siguiente diagrama de flujo:

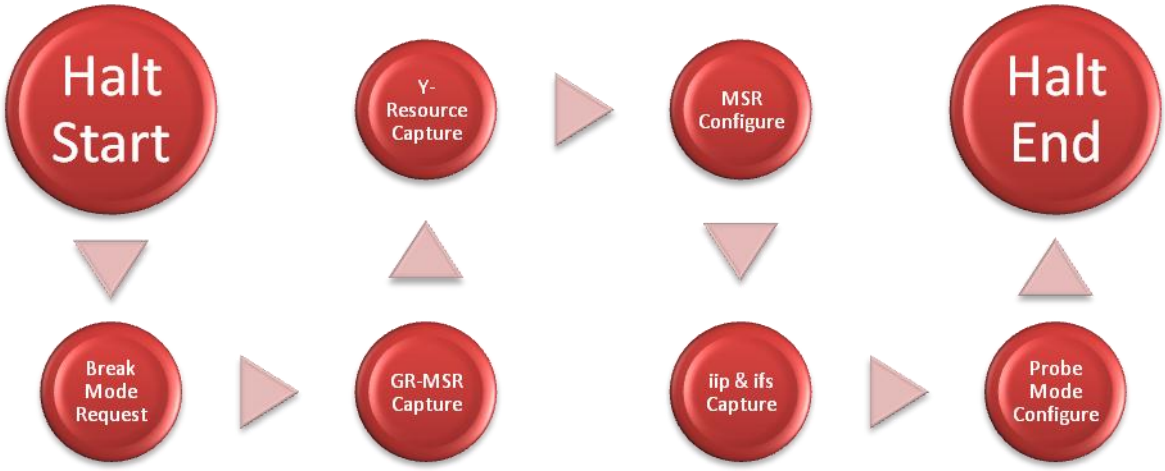


Figura 5.17 Diagrama de Bloques de Halt.

El algoritmo define 6 bloques importantes para la consecución de Halt, los cuales se ejecutan secuencialmente. Cada bloque consta de un grupo de instrucciones y desplazamiento de datos.

Break Mode Request: este bloque comprende la ejecución del código necesario para detener al procesador, y la configuración del registro de modo de prueba.

GR-MSR Capture: consiste en la ejecución de las instrucciones necesarias para desplazar los valores de los registros de uso general y los registros que monitorean el estatus del procesador. En el siguiente recuadro se observa cómo se logra capturar los valores de los registros generales y uno de los registros de estatus del procesador. Cabe aclarar que algunos valores y nombres han sido cambiados para proteger la información confidencial de la empresa:

```
load = "{.m|mi|  (p1) mov gsr[r0] = r0 ; nop.m 0x0; tnat.z p1, p0 = r1 }"
load = "{.m|mi|  (p1) mov xbr[r0] = r1 ; mov r1 = gsr[r0]; nop.i 0x0 }"
load = "{.m|mi|  (p1) xor r1 = 0x2, r1 ; nop.m 0x0; tnat.z p1, p0 = r3 }"
load = "{.m|mi|  (p1) xor r1 = 0x08, r1 ; (p1) mov fbr[r0] = r3; nop.i 0x0 }"
load = "{.mi|i|  nop.m 0x0; tnat.z p1, p0 = r2 ; (p1) xor r1 = 0x4, r1 }"
load = "{.m|mi|  xor r1 = 0x0f, r1 ; mov gsr[r0] = r1; nop.i 0x0 }"
load = "{.m|mi|  xor r3 = 0x43, r0 ; (p1) mov gsr[r3] = r2; nop.i 0x0 }"
loadReg(0x200)//R store R1
loadReg(0x320)//R store R2
loadReg(0x500)//R store R3
```

```
load = "{.m|mi|  adds r2 = 0x737, r0 ; mov r1 = gsr[r2]; nop.i 0x0 }"
load = "{.mii|  mov gsr[r3] = r1; nop.i 0x0; nop.i 0x0 }"
loadReg(noop)//R 0x737
```

Figura 5.18 Instrucciones que ejemplifican los modos de direccionamiento.

En la figura 5.18 se observa las instrucciones necesarias para mover el valor de los registros generales R1, R2 y R3 a registro legibles a través del TAP. Se recalca el uso de los modos de "direccionamiento de registros" y "direccionamiento indirecto mediante registros". En el recuadro posterior se muestra el uso del direccionamiento indirecto para poder leer el registro de status del procesador.

Y-Resources Capture: en este bloque se realiza la captura de los registros compartidos entre los núcleos físicos y lógicos. Estos registros deben almacenarse con el fin de retornar el sistema a su estado antes de la solicitud de Halt. Si esto no se ejecuta así, se podría provocar un evento catastrófico que llevaría al mal funcionamiento del procesador, y requeriría del reinicio de la maquina.

MSR Configure: en esta parte del código se configuran ciertos registros necesarios para ejecutar instrucciones durante el modo de prueba, así como el manejo de las interrupciones a través del TAP, y el control de la potencia consumida por el procesador.

iip & ifs Capture: aquí se almacenan los valores de las direcciones en las cuales el procesador fue detenido con el fin de retornar a este punto y continuar con la ejecución normal.

Probe Mode Configuration: este bloque se encarga de las configuraciones finales antes de tener totalmente el estado de Halt. Aquí el registro de configuración de modo de prueba es dispuesto de forma que reporta el estado de Halt en el cual se encuentra el procesador.

Con la consecución de estos bloques se completan los pasos necesarios para llevar a un procesador de una ejecución normal a un estado de Halt, también llamado “modo de prueba”, con las variables necesarias para revertir el proceso.

5.3.5 GO.

Go se define como el conjunto de operaciones formadas por instrucciones que restauran los valores de los registros previamente capturados en Halt, con el fin de restablecer el procesador a su ejecución normal -en el momento que se solicito el Halt-. Todo esto se realiza a través del TAP utilizando las mismas funciones de “Load”, “ReadReg” y “WriteReg”. En la mayoría de los casos, si el valor restablecido es el equivocado o esta corrupto, se forzaría al procesador a un error destructivo generándose un reinicio automático ó la ejecución de una aplicación de manejo de errores, como por ejemplo las conocidas pantallas azules en el caso de Windows.
En el diagrama se muestran los antónimos de las funciones de Halt.

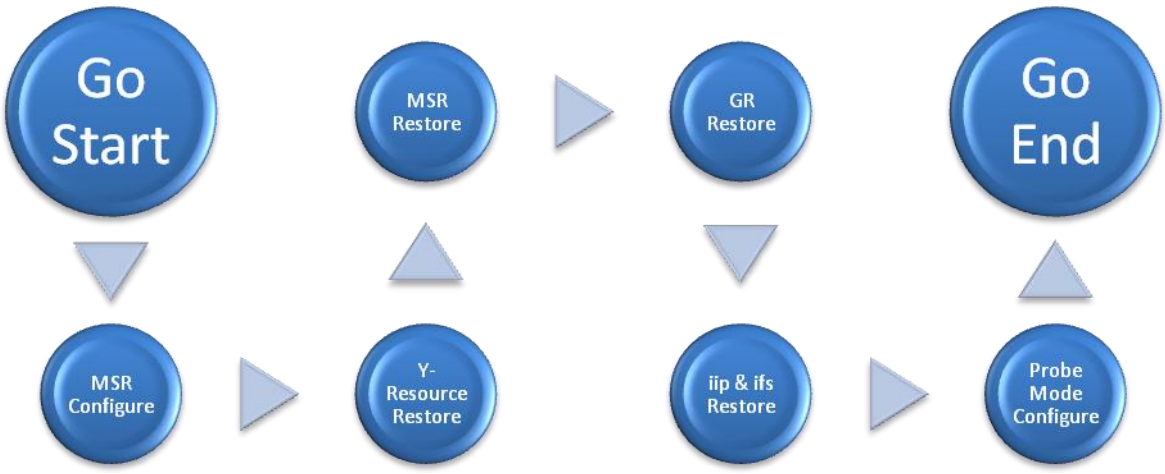


Figura 5.19 Diagrama de bloques de Go.

El diagrama consiste en 6 bloques de ejecución complementarios a los de Halt. Se observa que el orden de realización no es el mismo.

MSR Configure: en este conjunto de instrucciones se configuran registros que permiten la habilitación de la función de retorno de interrupción. Esta función permite especificar cuál conjunto de instrucciones es reiniciado, ignorando cualquier instrucción anterior.

Y-Resources Restore: Este bloque contiene instrucciones que permiten restaurar los valores de los recursos compartidos, los que a su vez permiten al sistema operativo recuperar el control de la aplicación original ante una solicitud de modo de prueba.

MSR Restore: restaura los valores de los registros de estado del procesador; entre los registros que se restauran están los que permiten activar los hilos de ejecución.

GR Restore: conjunto de instrucciones que devuelve a su valor original los registros de trabajo R1, R2 y R3, con el fin continuar con la ejecución normal de la aplicación en curso cuando se solicitó el Halt. En el recuadro se muestra cómo es que se realiza esta función:

```
loadReg(0x150) = 0x000000007fc6e000
loadReg(0x101) = 0x000000000000038a
loadReg(0x402) = 0x0000000000000000
loadReg(0x448) = 0x10a0808800c4a353
//this part of the code restore the values of r1,r2 & r3.
loadpir = "{.mib|  nop.m 0x0; nop.i 0x0; cover }"
loadpir = "{.m|mi|  mov r1 = gsr[r3] ; nop.m 0x0; mov pr = r1, -0x2 }"//pr
loadpir = "{.m|mi|  mov r1 = jbr[r0] ; mov r2 = jbr[r0]; nop.i 0x0 }"//r1=0x150; r2=0x101
loadpir = "{.mii|  mov r3 = gsr[r0]; nop.i 0x0; nop.i 0x0 }"//r3=0x402
```

Figura 5.19 Código para restaurar los registros generales.

Se observa la carga de los valores antes capturados por Halt en registros de trabajo, los cuales a su vez fueron escritos en los registros correspondientes en forma indirecta. Como se menciona en el apartado de estructura de datos, estas instrucciones son pre-compiladas e inyectadas en forma hexadecimal.

iip & ifs Restore: antepenúltimo bloque, que restaura la dirección al puntero de instrucciones con la finalidad de dejar el procesador apuntando a la instrucción inmediata al momento de la solicitud de “Break Mode”.

Probe Mode Configure: este bloque configura una serie de registros entre los cuales está el que determina cuándo el procesador esta en modo de prueba. El código libera ciertos bits que le devuelven el control al sistema operativo; éste continúa su ejecución normal sin percatarse de la interrupción sufrida.

Con la consecución de los bloques de Go se logra ejecutar las instrucciones necesarias para retornar el procesador a su estado de ejecución normal.

CAPÍTULO 6
ANÁLISIS DE RESULTADOS

6.1 Resultados

Este proyecto de investigación contó con dos parámetros para medir los resultados.

El primero se basó en la equivalencia de los valores obtenidos mediante ITP, contra los valores capturados mediante el código.

El segundo parámetro de medición fue lograr ejecutar la secuencia Halt-Go de tal forma que no causase ningún mal funcionamiento del procesador.

6.1.1 Resultado Código vs. ITP

La metodología de medición utilizada consistió en llevar al procesador a un punto replicable de ejecución, por ejemplo el inicio de EFI o Windows, con lo que se garantiza que los valores leídos tanto por ITP como por el código son equivalentes.

Como se mencionó anteriormente, la estructura de los datos es hexadecimal; por tanto, la comparación es directa. Los resultados obtenidos mediante ITP y los obtenidos con el código se muestran para 3 distintos sistemas operativos: Windows, Linux y EFI.

Entre los resultados obtenidos se observan valores equivalentes y discrepantes. Los resultados equivalentes se debe a que algunos registros de configuración básica y de operación se mantienen estáticos a lo largo de la ejecución normal del procesador en un sistema operativo dado; pero no todos los valores equivalentes se deben a registros estáticos, sino también al estado ó momento de ejecución del sistema operativo en el cual se realiza la lectura de los registros, se recalca los valores obtenidos en los registros generales R1, R2 y R3, que en un punto de ejecución determinada presentan un mismo valor, el

punto de ejecución seleccionado en el caso de los tres sistemas operativos es el momento inactivo del sistema operativo luego de haber iniciado.

Los resultados discrepantes provienen de registros dinámicos los cuales varían constantemente ó cada cierto tiempo, ese es el caso de los registros punteros de instrucciones (ip), el procesador al estar en constante ejecución de instrucciones ya sea por parte del sistema operativo ó alguna aplicación residente, modifica los valores de estos registros por lo que la discrepancia de valores entre los obtenidos con ITP y el código son explicables. Asimismo los valores entre sistemas operativos son diferentes en la mayoría de los casos debido a su naturaleza.

Tabla 1 Valores de obtenidos con ITP vs. Halt/Go en Windows

Register Name	ITP @ Windows	Halt / Go @ Windows
MSR_DBG_CFG DBG Configuration	0x4100000000000800	0x4100000000000800
MSR_ISD_VIR_MODE Configures Instruction Dispersal	0x0000000000000000	0x0000000000000000
MSR_IPC_SCRATCH General Purpose Scratch Register	0x00001010086a6018	0x0000000000000001
MSR_REN_DEBUG REN Block Configuration Register	0x0000000000000000	0x0000000000000000
DBR0	0x0000000000000000	0x0000000000000000
R1	0xE000000084486000	0xE000000084486000
R2	0xE0000000842E3B98	0xE0000000842E3B98
R3	0xE0000000842E3BE0	0xE0000000842E3BE0
MSR_DCC_CR1 DCC Configuration Register 1	0x000000008001010	0x0000000080010100
cr.isr	0x00000A1400000000	0x00000A1400000000
MSR_XPN_YIP YIP REGISTER	0xE0000165C8458090	0xE0000165C8458090
YPSR MSR_DCS_YPSR Y Resources Copy of PSR	0x0000000000000000	0x0000000000000000
MSR_REN_YFS YFS Register	0x0000000000000000	0x0000000000000000
MSR_XPN_DFT XPN Configuration Register	0x0BE78367020C01FF	0x0BE78367020C01FF
MSR_XPN_DBG Configures and Controls Probe Mode	0x0000000000000000	0x0000000000000000
cr.ipsr	0x00001210086A2018	0x00001210086A2018
cr.iip	0xE0000000E0B3CF40	0xE000000084098B20
cr.ifs	0x8000000000000308	0x8000000000000308
shift out the pr	0x5555555555553917	0x5555555555553917
MSR_DBG_BTSADDR BTS Address Reg.	0x0001000000000000	0x0001000000000000
MSR_BCD_MC_XR1 PAL Scratch Register	0x0000000000000001	0x0000000000000001
MSR_EBL_CR_CTL0 EBL Control Register 0	0x0000000000000007	0x0000000000000007
cr.ipsr	0x00001210086A2018	0x00001210086A2018
MSR_BCD_MC_XR4 PAL Scratch Register	0x0100000007000800	0x0100000007000800
SR_DBG_BTSADDR BTS Address Reg.	0x0001000000000000	0x0001000000000000
MPE_LCKS_CONFIG Lockstep configuration register	0x0000000000000000	0x0000000000000000
ar.cflg	0x0000060100000111	0x0000060100000111
ar_eflag	0x0000000000000202	0x0000000000000202
MSR_BCD_MC_XR0 PAL Scratch Register (threaded)	0x88000000003FFFCA	0x88000000003FFFCA

Tabla 2 Valores de obtenidos con ITP vs. Halt/Go en Linux

Register Name	ITP @ Windows	Halt / Go @ Linux
MSR_DBG_CFG DBG Configuration	0x4100000000000800	0x4100000000000800
MSR_ISD_VIR_MODE Configures Instruction Dispersal	0x0000000000000000	0x0000000000000000
MSR_IPC_SCRATCH General Purpose Scratch Register	0x0000000000000000	0x0000000000000000
MSR_REN_DEBUG REN Block Configuration Register	0x0000000000000000	0x0000000000000000
DBR0	0x0000000000000000	0x0000000000000000
R1	0x6000000000ADA228	0x6000000000ADA228
R2	0xC000000000000916	0xC000000000000916
R3	0x6000000000CEF260	0x6000000000CEF260
MSR_DCC_CR1 DCC Configuration Register 1	0x0000000080010100	0x0000000080010100
cr.isr	0x00000A0400000000	0x00000A0400000000
MSR_XPN_YIP YIP REGISTER	0xA000000000177C20	0xA000000000177C20
YPSR MSR_DCS_YPSR Y Resources Copy of PSR	0x0000000000000000	0x0000000000000000
MSR_REN_YFS YFS Register	0x0000000000000000	0x0000000000000000
MSR_XPN_DFT XPN Configuration Register	0x0BE78367020C01FF	0x0BE78367020C01FF
MSR_XPN_DBG Configures and Controls Probe Mode	0x0000000000000000	0x0000000000000000
cr.ipsr	0x0000101B085A6010	0x0000101B085A6010
cr.iip	0x6000000000EFF200	0x6000000000ABB700
cr.ifs	0x8000000000000003	0x8000000000000003
shift out the pr	0x10A0808805A5AA6B	0x10A0808805A5AA6B
MSR_DBG_BTSADDR BTS Address Reg.	0x0001000000000000	0x0001000000000000
MSR_BCD_MC_XR1 PAL Scratch Register	0x0000000000000001	0x0000000000000001
MSR_EBL_CR_CTL0 EBL Control Register 0	0x0000000000000007	0x0000000000000007
cr.ipsr	0x0000101B085A6010	0x0000101B085A6010
MSR_BCD_MC_XR4 PAL Scratch Register	0x0100000001000800	0x0100000001000800
SR_DBG_BTSADDR BTS Address Reg.	0x0001000000000000	0x0001000000000000
MPE_LCKS_CONFIG Lockstep configuration register	0x0000000000000000	0x0000000000000000
ar.cflg	0x0000060080000011	0x0000060080000011
ar_eflag	0x0000000000000092	0x0000000000000092
MSR_BCD_MC_XR0 PAL Scratch Register (threaded)	0x88000000003FFFCA	0x88000000003FFFCA

Tabla 3 Valores de obtenidos con ITP vs. Halt/Go en EFI

Register Name	ITP @ Windows	Halt / Go @ EFI
MSR_DBG_CFG DBG Configuration	0x4100000000000800	0x4100000000000800
MSR_ISD_VIR_MODE Configures Instruction Dispersal	0x0000000000000000	0x0000000000000000
MSR_IPC_SCRATCH General Purpose Scratch Register	0x0000000000000000	0x0000000000000000
MSR_REN_DEBUG REN Block Configuration Register	0x0000000000000000	0x0000000000000000
DBR0	0x0000000000000000	0x0000000000000000
R1	0x000000007FC6C000	0x000000007FC6C000
R2	0x0000000000000000	0x0000000000000000
R3	0x0000000000000001	0x0000000000000001
MSR_DCC_CR1 DCC Configuration Register 1	0x0000000080010100	0x0000000080010100
cr.isr	0x00000A0400000000	0x00000A0400000000
MSR_XPN_YIP YIP REGISTER	0x000000007FE13780	0x000000007FE13780
YPSR MSR_DCS_YPSR Y Resources Copy of PSR	0x0000000000000000	0x0000000000000000
MSR_REN_YFS YFS Register	0x0000000000000000	0x0000000000000000
MSR_XPN_DFT XPN Configuration Register	0x0BE78367020C01FF	0x0BE78367020C01FF
MSR_XPN_DBG Configures and Controls Probe Mode	0x0000000000000000	0x0000000000000000
cr.ipsr	0x0001000000002018	0x0001000000002018
cr.iip	0x000000007EA2C2C0	0x000000007FA2C5A0
cr.ifs	0x8000000000000695	0x8000000000000695
shift out the pr	0x10A0808800C46453	0x10A0808800C46453
MSR_DBG_BTSADDR BTS Address Reg.	0x0001000000000000	0x0001000000000000
MSR_BCD_MC_XR1 PAL Scratch Register	0x0000000000000001	0x0000000000000001
MSR_EBL_CR_CTL0 EBL Control Register 0	0x0000000000000007	0x0000000000000007
cr.ipsr	0x0001000000002018	0x0001000000002018
MSR_BCD_MC_XR4 PAL Scratch Register	0x0100000003000800	0x0100000003000800
SR_DBG_BTSADDR BTS Address Reg.	0x0001000000000000	0x0001000000000000
MPE_LCKS_CONFIG Lockstep configuration register	0x0000000000000000	0x0000000000000000
ar.cflg	0x0000000000000112	0x0000000000000112
ar_eflag	0x0000000000000093	0x0000000000000093
MSR_BCD_MC_XR0 PAL Scratch Register (threaded)	0x88000000003FFFCA	0x88000000003FFFCA

6.1.2 Ejecución de secuencia Halt-Go

Para este parámetro se tenía como resultado la ejecución exitosa de la secuencia Halt-Go en los 3 sistemas operativos. Entre las pruebas realizadas para corroborar el buen funcionamiento de esta secuencia se incluyeron:

1. Protector de pantalla: esta aplicación es sensible a cambios de estado en el procesador, por lo que sirve de parámetro para determinar si al final de la secuencia el estado de arquitectura retorna de forma intacta o imperceptible al sistema operativo.
2. Programa de prueba de memoria elástica 3D de diferencia finita: este programa ejecuta operaciones en arreglos de punto flotante en PPV. Se utiliza para determinar tanto fallas en las unidades de Punto Flotante como errores en la memoria Caché del sistema. Permite determinar si el código corrompe algún estado que concierne a la memoria del sistema.
3. Juegos de video: se comprueba que al ejecutar la secuencia se logra continuar sin ningún incidente el desarrollo del juego de video.
4. P1: conjunto de aplicaciones que se utiliza en PPV. Consiste en una ejecución masiva de distintas aplicaciones al mismo tiempo; entre las funciones realizadas en las aplicaciones se encuentran copia de datos masivos y aplicaciones graficas simultáneas.

Como resultado de estas pruebas, más algunas otras que no son mencionadas, se obtuvo una ejecución exitosa de la secuencia de Halt-Go, imperceptible para el funcionamiento normal del procesador. A su vez la realización consecutiva de la secuencia de Halt-Go a lo largo de la ejecución de algún programa en el procesador demostró la estabilidad del código así como del algoritmo. Se logró ejecutar 51 ciclos, en intervalos de 15 minutos, obteniendo resultados satisfactorios. En el Anexo A.6 se incluyen un extracto de esta iteración la cual se realiza en Linux.

6.2 Análisis

El análisis de los resultados obtenidos se realizó con base en los objetivos.

Como objetivo general se investigó la posibilidad de establecer una metodología que permitiera la observación del procesador por medio del Puerto de acceso a pruebas ó TAP. Dados los resultados obtenidos, se considera alcanzado este objetivo, pues se logró determinar la metodología y probar su efectividad controlando y observando el procesador por medio del uso de TAP.

Entre los alcances de este objetivo se encuentra la posibilidad de generar una nueva metodología de prueba en PPV complementaria a la actual, lo que permitiría capturar el estado de arquitectura del procesador durante la falla de éste con el fin de determinar en dónde ocurre el error y qué lo provoca. Esto con la intención de aportar al equipo de diseño la información necesaria para mejorar la validación y diseño del producto.

Entre las limitantes encontradas durante la investigación, estuvo la velocidad de transmisión limitada a la frecuencia del TAP (2 MHz), que hizo más lenta la ejecución de una gran cantidad de instrucciones. No se logró determinar limitantes directamente relacionadas al procesador.

Dada la naturaleza de la solución escogida los objetivos de hardware no se lograron, pues para lograr la ejecución utilizando las funciones presentes en Systest no es necesario determinar las señales eléctricas involucradas ni la sincronización de las señales del TAP.

En cuanto a los objetivos de software, se logró establecer los algoritmos necesarios para ejecutar de manera exitosa los procedimientos de Halt y Go, los cuales son pilares para la consecución de los procedimientos de Punto de Parada y Ejecución paso a paso.

Se logran realizar variantes en los procedimientos conocidos permitiendo realizar funciones que no estaban disponibles. El código por su naturaleza no es lo mas óptimo en lo que se refiere a tiempo de ejecución ya que al ser un lenguaje interpretado, con lo que cada instrucción se traduce una a una en un tiempo de ejecución a un lenguaje intermedio o lenguaje maquina siendo típicamente 10 veces más lento que los programas compilados.

El proyecto consiguió cumplir con los objetivos de investigar e implementar los procedimientos de Halt y Go ejecutando instrucciones a través del TAP de forma exitosa. No se logró concluir con las demás funciones como Punto de Parada ni Ejecución Paso a Paso.

La implementación presentaba objetivos que se lograron, pues como se describe en la solución seleccionada, se alcanzó una integración directa del código y la funcionalidad al programa de pruebas en PPV.

Entre las finalidades de la implementación, las iteraciones realizadas en el proyecto demostraron la estabilidad del código y de los algoritmos. Con una ejecución de la secuencia Halt-Go en intervalos de 15 minutos en el sistema operativo Linux, se lograron 51 ejecuciones consecutivas sin afectar la operación normal del procesador ni del sistema operativo. Esto demuestra la estabilidad del código y de la plataforma.

Entre las experimentaciones mencionadas en los objetivos de implementación, se encuentra la ejecución de distintos ambientes con aplicaciones que demandan de un uso extensivo del procesador. Al utilizar Systest como plataforma de realización del proyecto, se logró, desde el inicio, la integración al programa de pruebas de PPV.

Quedan, como parte de las recomendaciones, la consecución de los distintos objetivos de Software e Implementación que no se alcanzaron en este proyecto.

CAPÍTULO 7
CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones.

1. Se logró controlar y observar el procesador Itanium 2 a través del TAP, sin ITP.

2. Se logró ejecutar la secuencia Halt-Go sin afectar la estabilidad del sistema basado en Itanium 2.

3. Se determinó la secuencia de instrucciones necesaria para la captura y restauración de la arquitectura del procesador Itanium 2.

4. La velocidad de ejecución estuvo limitada a 2 MHz

5. Ejecuciones seguidas (1 tras otra) de Halt-Go no implicaron distorsión alguna en la ejecución normal del procesador.

6. Las instrucciones requieren de una pre-compilación antes de ser inyectadas al Pipe-Line.

7. Se pudo identificar los recursos de arquitectura necesarios para la consecución de Halt-Go.

8. Los valores de capturados fueron congruentes con los obtenidos con ITP.

9. Se determinó qué valores corruptos ingresados en Go generan eventos destructivos en el sistema.

7.2 Recomendaciones

1. Investigar la posibilidad de reducir la cantidad de instrucciones necesarias. Basado en los resultados obtenidos se observan valores estáticos, realizando la experimentación necesaria puede definirse si es necesaria la captura y re-escritura de los mismos. Con esto se lograría un ahorro en el tiempo de ejecución.

2. Indagar sobre la capacidad de aumento en la velocidad de transmisión de datos a través del TAP.

3. Determinar cómo detectar cuándo una ejecución del procesador se cumple, con la finalidad de establecer la metodología para la Ejecución Paso a Paso.

4. Establecer la metodología para múltiples procesadores.

CAPÍTULO 8
BIBLIOGRAFÍA

[1] Intel Corporation. “Itanium® 2 Architecture Software Development Manual”. Volume 1: Application Architecture. Revision 2.1. October 2002.

[2] Hewlett Packard Corporation EPIC: Architecture for Instruction-Level Parallel Processors. February 2000. HPL-1999-111.

[3] The Institute of Electrical and electronics Engineers, Inc. “IEEE Standard Test Access Port and Boundary-Scan Architecture.” United States of America. July 2001. IEEE Std 1149.1-2001. PDF: ISBN 0-7381-2945-3 SS94949

[4] *“Home - ITO - Intel Test Operation”*. Creado el 7/4/2006. Recuperado el 25 de junio del 2006, de http://tmeweb.intel.com/sites/atced/default.aspx

[5] Halvorson, M. “Learn Visual Basic 6.0 Now”. Microsoft Press. 1999

[6] Chávez, Ana C. “Normas para redactar citas bibliográficas”. Instituto Tecnológico de Costa Rica Vicerrectoría de Vida Estudiantil y Servicios Académicos. Cartago, Costa Rica. 1995.

[7] Intel Corporation. “ITP Debug Port”. Design Guide. Revision 2.1. February 2004.

APÉNDICES

A.1 Glosario, abreviaturas y simbología

- ✓ **Arquitectura:** se refiere a toda una estructura y a los detalles necesarios para que sea funcional, es decir, cubre sistemas informáticos, microprocesadores, circuitos y programas del sistema
- ✓ **CISC:** Complex instruction Set Computer. Conjunto de instrucciones que usualmente necesitan más de un ciclo de reloj para completar su ejecución.
- ✓ **CPU:** Unidad Central de Procesamiento. Dispositivo lógico programable que ejecuta instrucciones lógicas y matemáticas.
- ✓ **EPIC:** Explicit Parallelism Instruction Computing, "Instrucciones Computacionales Explícitamente Paralelas", nueva filosofía de ejecución de instrucciones que busca un mejor aprovechamiento de las características de paralelismo presentes en procesador.
- ✓ **Hardware Breakpoint, Punto de parada:** en un programa, marcador que le indica al procesador que detenga la ejecución actual.
- ✓ **Interfaz:** La interfaz es el medio que permite la interacción entre dos ó más elementos.
- ✓ **JTAG:** Joint Test Access Group: Grupo que se encarga de establecer tanto los parámetros como las regulaciones de la infraestructura de pruebas.
- ✓ **MCA:** Machine Check Architecture.
- ✓ **MSR:** Machine Status Register.
- ✓ **PDE:** Product Development Engineer.
- ✓ **Pipe-Line:** Un arreglo serie de registros o procesadores que realiza parte de las tareas y que pasa su resultado al siguiente registro (procesador) hasta completar la tarea.
- ✓ **PPV:** Plataforma de Validación de Producto, proceso que valida el producto antes de que éste sea enviado al cliente.

- ✓ **Pro activo:** referido a actuar antes de que una situación se convierta en una confrontación o crisis. Persona pro activa es la que incentiva el mejoramiento.
- ✓ **RISC:** Reduced Instruction Set Computer. Una computadora que requiere de menos instrucciones para ejecutar una aplicación.
- ✓ **SDMG:** Server Development and Manufacturing Group. Grupo de Desarrollo y Manufactura de Servidores
- ✓ **TAP:** Test Access Port. Puerto de acceso a la infraestructura de pruebas.
- ✓ **TDI:** Test data Input. Puerto de entrada del TAP.
- ✓ **TDO:** Test Data Output. Puerto de salida del TAP.

A.2 Información sobre la empresa.

A.2.1 Descripción de la empresa.

La corporación Intel® así como su filial Componentes Intel® de Costa Rica S.A., tienen como misión trabajar de la mejor manera en beneficio de sus clientes, empleados y accionistas, y es el principal suplidor de plataformas computacionales para la economía mundial de Internet.

La empresa presenta cinco valores principales: orientación a resultados, toma de riesgos, un excelente lugar para trabajar, calidad, orientación al cliente y disciplina. Con estos valores, Intel persigue alcanzar objetivos que se traducen en ser la principal plataforma computacional de Internet, mejorar sus capacidades de producción y obtener un mayor segmento en el mercado de los servidores.

Fundada en 1968, se consolidó como líder mundial en la fabricación y desarrollo de microprocesadores. Actualmente la empresa cuenta con más de 100000 empleados alrededor del mundo.

Su cartera de clientes está conformada por empresas de servicios de Internet, fabricantes de equipo computacional y usuarios de sistemas de cómputo.

Componentes Intel® de Costa Rica está localizado en la Ribera de Belén, Heredia; cuenta con más de 2000 empleados y tiene como principal función el ensamble y prueba de microprocesadores. Con más de 8 años de experiencia en el país, ha logrado ser líder en su campo y tiene a su cargo la manufactura del 95% de los procesadores para servidores, pieza fundamental dentro de la corporación.

Cabe destacar que la llegada de Intel a Costa Rica ha elevado la calidad en materia de seguridad laboral e higiene ambiental, estableciendo parámetros desde los que las demás empresas miden sus operaciones en esas áreas.

A.2.2 Descripción del departamento.

El departamento en el que se realizó el proyecto es el SDM-CR PDE, el cual presenta la visión de:

“Ser un equipo de Ingenieros de Desarrollo de Producto preferido y técnicamente competente que entregue las mejores soluciones en su clase para los productos de servidores en Costa Rica, proveyendo un ambiente de crecimiento técnico y de alianzas a sus miembros como también manteniendo el valor de tener un excelente lugar para trabajar”.

El departamento tiene poco más de 1 año y medio de existir y compite con departamentos con más de dos décadas de antigüedad. Tiene como función formar parte de las tareas de desarrollo y depuración de productos como Itanium® 2 y Xeon®.

Con el fin de cumplir con su visión, el departamento promueve las acciones innovadoras de carácter técnico y de ingeniería para así lograr ser un grupo de preferencia al realizarse cualquier consulta referente al producto en que se esté involucrado.

El grupo está conformado, en su mayoría, por ingenieros en electrónica y eléctricos.

A.3 Recursos necesarios para el desarrollo del proyecto.

Materiales.

Recurso	Cantidad	Costo total
Manuales técnicos	Varios	
Material de oficina	Varios	$50
Licencia Visual Studio	1	$850

Equipos.

Recurso	Cantidad	Costo total
Computadora portátil IBM	1	$3000
Equipo de pruebas	1	$1.500.000

Servicios no personales.

Recurso	Costo	% Tiempo y disponibilidad en el proyecto
Impresión	$20	100%
Internet		100%

Servicios personales.

Recurso	Cantidad	Costo total
Estudiante	1	$1200

Infraestructura.

Recurso	Cantidad	Costo total
Cubículo con escritorio	1	
Laboratorio	1	

A.4 Recursos disponibles en la empresa.

Materiales.

Recurso	Costo	% Tiempo y disponibilidad en el proyecto
Manuales técnicos		100%
Material de oficina	$50	100%
Licencia Windows	$300	100%
Licencia Microsoft Office	$560	100%

Equipos.

Recurso	Costo	% Tiempo y disponibilidad en el proyecto
Computadora	$3000	100%
Equipo de Pruebas	$500.000	100%

Servicios no personales.

Recurso	Costo	% Tiempo y disponibilidad en el proyecto
Impresión	$20	100%
Internet		100%

Infraestructura.

Recurso	Costo	% Tiempo y disponibilidad en el proyecto
Aposento con escritorio		100%
Laboratorio		100%

A.5 Hoja de información del proyecto.

Información del estudiante:

Nombre: Rodney Cubero Barker

Cédula: 701310076 **Carné I.T.C.R.:** 9722379

Dirección de su residencia en época lectiva: Cartago, 300 m. oeste, 100 m. norte, 25 m. oeste del I.T.C.R.

Dirección de su residencia en época no lectiva: Cartago, 300 m. oeste, 100 m. norte, 25 m. oeste del I.T.C.R.

Teléfono en época lectiva: 334-8212 **Teléfono época no lectiva:** 334-8212

E-mail: rodneycbarker@yahoo.com **Fax: ---------**

Información del proyecto:

Nombre del Proyecto: *Expansión de las capacidades de depuración de la Plataforma de Validación de Producto del procesador Itanium® 2*

Área del Proyecto: Sistemas Digitales y Microprocesadores.

Información de la empresa:

Nombre: Componentes Intel S.A.

Zona: Heredia, San Antonio de Belén

Dirección: 600 m. Oeste de la Firestone, calle 129

Teléfono: 296-6000 **Fax:** 298-7323

Actividad Principal: Ensamblaje y prueba de componentes electrónicos.

Información del asesor en la empresa:

Nombre: Luis Diego Rojas

Puesto que ocupa: Ingeniero de Desarrollo

Departamento: *Server Development and Manufacturing Group.*

Profesión: Ingeniero Eléctrico **Grado académico:** Bachiller

Teléfono: 256-8096

E-mail: luis.d.rojas.munoz@intel.com

ANEXO

Anexo A.1 Portada de referencias bibliográficas.

Intel® Itanium® 2 Processor

Specification Update

February 2007

Notice: The Intel® Itanium® 2 processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.

Document Number: 251141-049

IEEE Std 1149.1-2001
(Revision of IEEE Std 1149.1-1990)

IEEE Standard Test Access Port and Boundary-Scan Architecture

Sponsor

Test Technology Standards Committee
of the
IEEE Computer Society

Approved 14 June 2001

IEEE-SA Standards Board

Abstract: Circuitry that may be built into an integrated circuit to assist in the test, maintenance, and support of assembled printed circuit boards is defined. The circuitry includes a standard interface through which instructions and test data are communicated. A set of test features is defined, including a boundary-scan register, such that the component is able to respond to a minimum set of instructions designed to assist with testing of assembled printed circuit boards. Also, a language is defined that allows rigorous description of the component-specific aspects of such testability features.

Keywords: boundary scan, boundary-scan architecture, Boundary-Scan Description Language, boundary-scan register, BSDL, circuit boards, circuitry, integrated circuit, printed circuit boards, TAP, test, test access port, VHDL, VHSIC Hardware Description Language

---

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Print: ISBN 0-7381-2944-5 SH94949
PDF: ISBN 0-7381-2945-3 SS94949

HEWLETT PACKARD

EPIC: An Architecture for Instruction-Level Parallel Processors

Michael S. Schlansker, B. Ramakrishna Rau
Compiler and Architecture Research
HP Laboratories Palo Alto
HPL-1999-111
February, 2000

E-mail:{schlansk, rau}@hpl.hp.com

EPIC architecture, VLIW architecture, instruction-level parallelism, MultiOp, non-unit assumed latencies, NUAL, rotating register files, unbundled branches, control speculation, speculative opcodes, exception tag, predicated execution, fully-resolved predicates, wired-OR and wired-AND compare opcodes, prioritized loads and stores, data speculation, cache specifiers, precise interrupts, NUAL-freeze and NUAL-drain semantics, delay buffers, replay buffers, EQ and LEQ semantics, latency stalling, MultiOp-P and MultiOp-S semantics, dynamic translation, MultiTemplate and VariOp instruction formats, history of EPIC

Over the past two and a half decades, the computer industry has grown accustomed to, and has come to take for granted, the spectacular rate of increase of microprocessor performance, all of this without requiring a fundamental rewriting of the program in a parallel form, without using a different algorithm or language, and often without even recompiling the program. The continuation of this trend is the topic of discussion of this report. For the time being at least, instruction-level parallel processing has established itself as the only viable approach for achieving the goal of providing continuously increasing performance without having to fundamentally re-write applications. In this report, we introduce the Explicitly Parallel Instruction Computing (EPIC) style of architecture which was developed, starting eleven years ago, to enable higher levels of instruction-level parallelism without unacceptable hardware complexity. We explain the philosophy underlying EPIC as well as the challenges faced as a result of adopting this philosophy. We also describe and discuss the set of architectural features that together characterize the EPIC style of architecture, and which can be selectively included in some specific instance of an EPIC instruction set architecture.