

*RIC User's Manual*

*Sun Microelectronics*

## *RIC User's Manual*

©1997 Sun Microsystems, Inc. All rights reserved.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT ANY EXPRESS REPRESENTATIONS OR WARRANTIES. IN ADDITION, SUN MICROSYSTEMS, INC. DISCLAIMS ALL IMPLIED REPRESENTATIONS AND WARRANTIES, INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

This document contains proprietary information of Sun Microsystems, Inc. or under license from third parties. No part of this document may be reproduced in any form or by any means or transferred to any third party without the prior written consent of Sun Microsystems, Inc.

Sun, Sun Microsystems and the Sun Logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The information contained in this document is not designed or intended for use in on-line control of aircraft, air traffic, aircraft navigation or aircraft communications; or in the design, construction, operation or maintenance of any nuclear facility. Sun disclaims any express or implied warranty of fitness for such uses.

*Sun Microelectronics*

# Contents



<b>1. Overview.....</b>	<b>1</b>
1.1 Introduction.....	1
1.2 Definition of Terms .....	2
1.3 Partition Overview .....	3
1.4 RIC Chip Overview.....	4
<b>2. Pin Descriptions .....</b>	<b>7</b>
2.1 I/O Driver Specifications .....	7
2.2 Resets Interface Signals.....	8
2.3 Interrupt Concentrator Interface Signals .....	9
2.4 Scan Interface Signals.....	10
2.5 Clock Controller Interface Signals .....	11
2.6 PAL Block Signals .....	12
2.7 Miscellaneous Signals .....	13
2.8 RIC Pin Count .....	13
<b>3. RIC Functional Description .....</b>	<b>15</b>
3.1 RIC Overview .....	15
3.2 Detailed Internal Block Diagram.....	16
3.3 Block Overviews.....	17
<b>4. Reset Block .....</b>	<b>19</b>
4.1 Overview .....	19
4.2 Signal Descriptions.....	22
4.3 Reset Logic Functional Description .....	23

<b>5. Interrupt Concentrator .....</b>	<b>29</b>
5.1 Overview.....	29
5.2 Signal Descriptions.....	31
5.3 Interrupt Concentrator Functional Description .....	32
<b>6. Scan Controller .....</b>	<b>47</b>
6.1 Overview.....	47
6.2 Signal Descriptions.....	49
6.3 Scan Controller Functional Description .....	50
6.4 Scan Controller Timing Diagrams.....	58
6.5 Scan Block Bugs and Solutions .....	58
<b>7. Clock Controller .....</b>	<b>61</b>
7.1 Overview.....	61
7.2 Clock Controller Function Descriptions.....	63
<b>8. PAL Block.....</b>	<b>65</b>
8.1 Overview.....	65
8.2 PAL Block Function Descriptions .....	68

## List of Figures



RIC Chip System Block Diagram.....	3
RIC Chip Logical Block .....	6
RIC Chip Block Diagram .....	16
RIC Chip Block .....	20
Reset Block .....	21
RIC_Reset .....	22
POR - Detailed Block .....	23
POR Timing Diagram .....	24
Button_POR - Detailed Block .....	25
Button_POR Timing .....	25
Button_XIR - Detailed Block .....	26
Button_XIR Timing .....	27
Interrupt Concentrator System Block Diagram .....	29
Interrupt Concentrator Block .....	30
Ric Intcon .....	31
Encoder Block .....	34
State Diagram of the First Stage encode .....	36
Encoder 1 Timing .....	39
Dispatcher Block .....	40
Dispatcher State Machine .....	42
Dispatcher Timing .....	45
Scan Controller System Block .....	47
Scan Controller Block Diagram .....	48
Scan_Control .....	49
JTAG+ Interface Detailed Block.....	52
JTAG+ FSM State Machine .....	53
JTAG+ Interface Timing Diagram .....	54

JTAG Internal Scan Ring Block .....	55
JTAG State Machine .....	57
JTAG Scan Ring Timing Diagram .....	58
Clock Controller System Block .....	61
RIC_Clogen .....	62
PAL Block .....	66
RIC_PAL.....	66

## List of Tables



Driver Descriptions .....	7
Resets Interface Signals .....	8
Interrupt Concentrator Interface Signals .....	9
Scan Interface Signals .....	10
Clock Controller Signals .....	11
Ebus and PAL Signals .....	12
Miscellaneous Signals .....	13
RIC Pin Count .....	13
RIC Gate Count estimates .....	18
Input/Output Signals of Reset Logic .....	22
Interrupt Signals .....	32
en1_fsm State Description .....	35
State Table description .....	41
Scan Controller Signals .....	50
JTAG FSM State .....	53
JTAG+ Scan Address Register .....	54
JTAG Scan Ring Data/Instruction Register .....	56
Jtag state machine .....	56
Clock Controller signal descriptions .....	62
System clock select .....	63
PAL Block signal descriptions .....	67
PAL Logic Address Map .....	68





## 1.1 Introduction

The RIC chip (STP2210QFP) supports the system resets, system interrupts, system scan, system clock control as well as other functions for the UltraSPARC system boards.

Features:

- Supports resets from power supply, reset buttons, and scan
- Delivers system power on reset, button power-on reset, and the external button resets to the system controller (SC)
- Concentrates all the interrupts and sends interrupt numbers to the system I/O (U2S)
- Directs SCAN inputs and outputs through the scan chains
- Provides scan power on reset, and scan external interrupt resets through an internal scan chain
- Determines the system speed from the CPU speed inputs
- Provides decoding for SC, Lab Console address and data registers, Frequency Margining chip and PROM write address space

## ***1.2 Definition of Terms***

### ***1.2.1 Frequently Used Terms***

- RIC - Reset, Interrupt, Scan, and Clock Control
- U2S - I/O Controller
- SC - System Controller
- POR - Power on Reset
- XIR - External interrupt Reset
- UPA - Uniform Port Architecture

### ***1.2.2 Conventions***

- Most significant bit is the bit with the highest bit number. Least significant bit is the bit with the lowest bit number
- Byte = 8 bits, Halfword = 16 bit, Word = 32 bit, Doubleword = 64 bit
- Addresses are byte address
- In a byte stream if byte n is located at address a, byte n+1 is located at address a+1. The same is true for the other data types
- Bytes, halfwords, words and doubleword datums are located at byte, halfword, word and doubleword addresses respectively. A halfword address is evenly divisible by 2, a word address is evenly divisible by 4 and a doubleword address is evenly divisible by 8
- In a halfword, most significant byte is at the lower address, In a word, most significant byte or halfword is at the lowest address. In a doubleword, the most significant byte, halfword or doubleword is at the lowest address
- Assertion = True = 1 and De-assertion = Negative = False = 0
- All descriptions and state diagrams are logical, They are completely independent of the electrical value unless explicitly called out as “high” or “low”
- Negative true signals are suffixed with an “\_”

### 1.3 Partition Overview

The RIC chip interfaces with the power on circuits, interrupts sources, scan logics, and the system clock chip.

#### 1.3.1 Partition Block Diagram

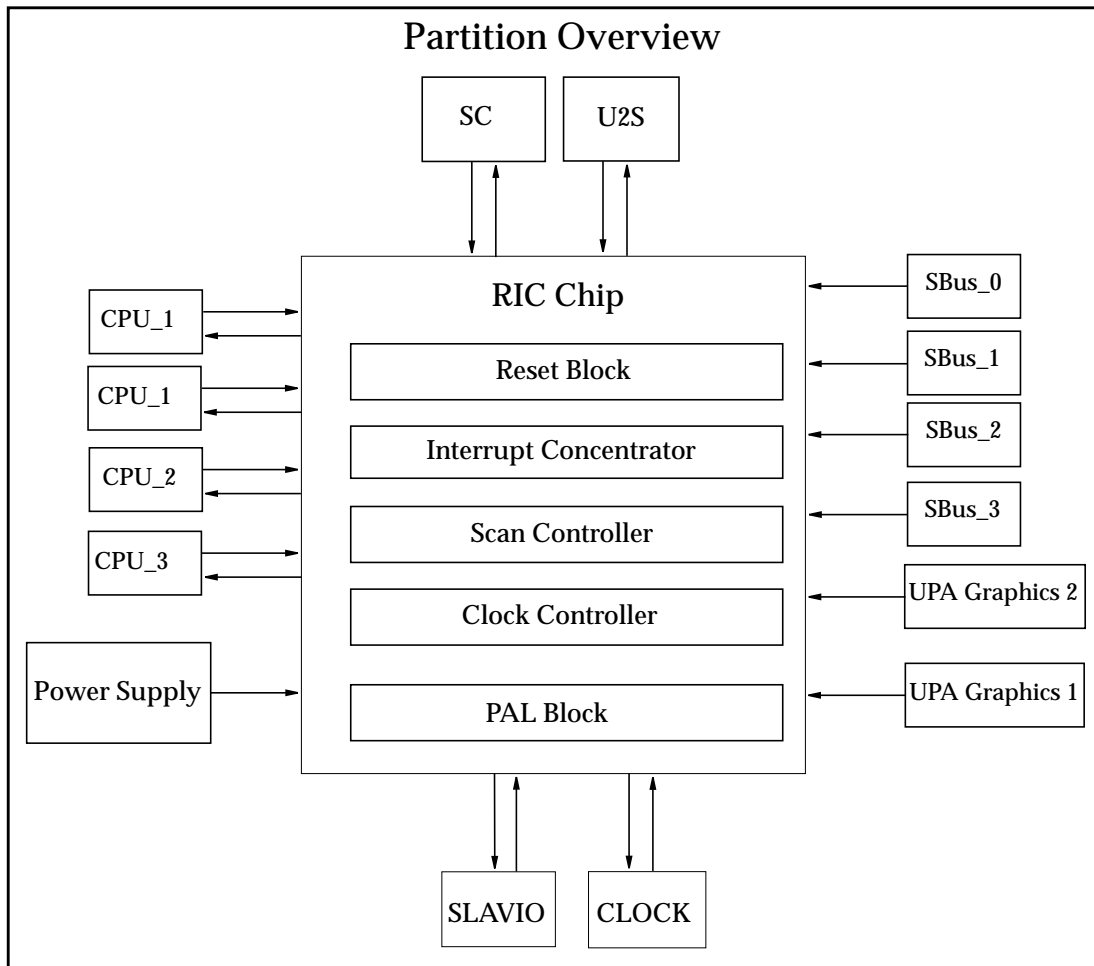


Figure 1-1 RIC Chip System Block Diagram

## **1.4 RIC Chip Overview**

The RIC Chip includes five major functional modules. The five modules are Resets, Interrupt, Scan, Clock Control and the PAL block.

The five modules are completely independent of each other and can be implemented in separate chips.

### **1.4.1 Resets**

The reset logic generates an asynchronous power on reset signal to the system controller (SC) from the power ok signal from the power supply. It also generates the p\_button\_ reset to SC when a signal is received from a power on reset button or scan\_por\_ signal from scan logic. A x\_button\_ reset is generated to SC if the XIR\_ reset button is pressed or a scan\_xir\_ signal is received from the scan logic.

### **1.4.2 Interrupt**

This interrupt module collects all the interrupts from SBus devices, EBus devices, and UPA expansion devices. It will send the interrupt number (binary encoded) to U2S on a round robin basis. The best case latency from detecting an interrupt and sending it to U2S is five Sbus<sup>[1]</sup> clocks.

### **1.4.3 Scan**

The scan module directs the flow of the serial scan chains at the system level. It receives the scan chain identification from the service controller and directs the corresponding scan input to the scan output. It also implements a separate scan chain to initiate the XIR and POR resets to system controller (SC).

---

1. If the RIC chip is used in a PCI based system, all references to SBus can be replaced with PCI bus. Equally replace all references to 25MHz Sbus clock with 33MHz PCI clock. When the RIC chip is used in a PCI based system, the RIC chip meets the necessary timing requirements for the 33MHz PCI bus. Note that this manual also has relevance for PCI based system designs.

#### ***1.4.4 Clock Control***

The clock control module receives speed inputs from all the CPU modules. Based on the slowest CPU speed it determines the system frequency and sends clock select signals to select the system frequency.

#### ***1.4.5 PAL Block***

The PAL Block module implements decode logic for the system controller (SC) address and data register, Lab Console (LC) address and data register, support for the frequency margining chip and a decode for the PROM write address space. Also, the PAL block provides an XOR gate.

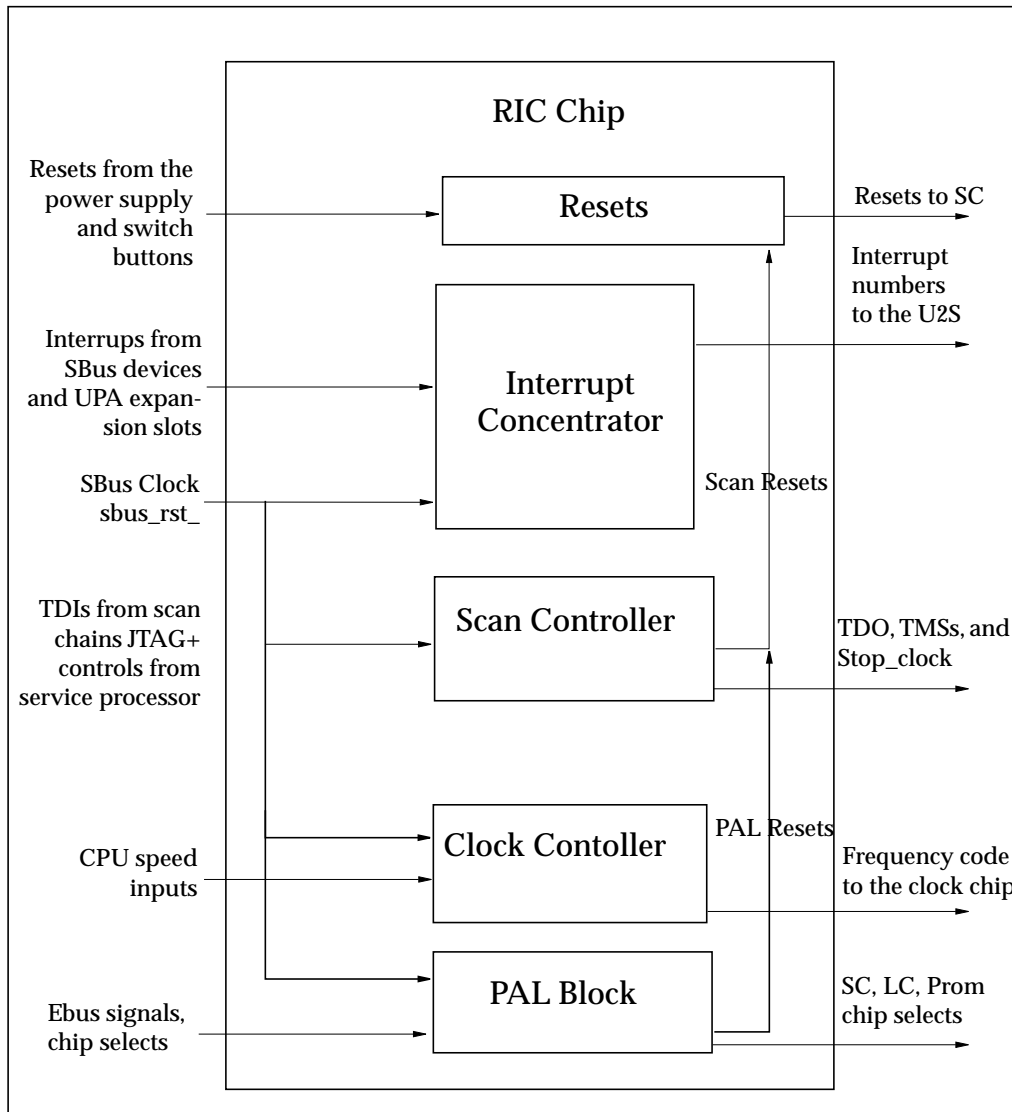


Figure 1-2 RIC Chip Logical Block

This chapter describes the pinout for the RIC chip.

## 2.1 I/O Driver Specifications

The following drivers are used in the RIC:

Table 2-1 Driver Descriptions

Driver Name	Buffer Type I,O,IO	Strength	Notes
PINC10T	Input	1.44 mW	BIN10T with 100K pull-up
PINY10T	Input	1.44 mW	BIN10T with 50K pull-down
BIN10T	Input	1.44 mW	Noninverting, TTL input buffer
BIN04T	input	6.17 mW	Noninverting, TTL input buffer
PIMC06F_INV	Input		Inverting, 5 Volt tolerant PIMC06F input driver
BOM4T	Output	4mA	Noninverting, MOS, 3state output
BOT6T	Output	6mA	Noninverting, TTL, 3 state output
BOT10T	Oupput	10 mA	Noninverting, TTL, 3state output
BON6T	Output	6 mA	Noninverting TTL output
BOM6F_TRD	Output	6 mA	5 Volt tolerant BOM6F output driver
BON6F_TRD	Ouput	6 mA	5 Volt tolerant BON6F output driver
BON8F_TRD	Ouput	8 mA	5 Volt tolerant BON8F output driver
BON10F_TRD	Ouput	12 mA	5 Volt tolerant BON10F output driver
BON12F_TRD	Ouput	12 mA	5 Volt tolerant BON12F output driver

## ***2.2 Resets Interface Signals***

These signals connect from the RIC chip to the SC, power supply, and switch buttons.

*Table 2-2*      Resets Interface Signals

Signal Name	Internal Signal Name	Pin Count	I/O	Driver	Description
POWER_OK_L	ipower_ok_	1	I	PINY10T	Signal asserted during power on
BUTTON_POR_L	ibutt_por_	1	I	PINC10T	Signal asserted by push button switch
BUTTON_XIR_L	ibutt_xir_	1	I	PINC10T	Signal asserted by push button switch
SYS_POR_L	osys_por_	1	O	BOM6F_TRD	Signal to SC during pow on reset
P_BUTTON_RESET_L	op_button_	1	O	BOM6F_TRD	Signal to SC asserted during push button power reset
X_BUTTON_RESET_L	ox_button_	1	O	BOM6F_TRD	Signal to SC asserted during push button XIR.



### 2.3 Interrupt Concentrator Interface Signals

These signals connect from the RIC chip to the interrupting devices and U2S.

Table 2-3 Interrupt Concentrator Interface Signals

Signal Name	Internal Signal Name	Pin Count	I/O	Driver	Description
SB_IRQ0_L[7:1]	isbus0	7	I	PINC10T	interrupts from sbus slot 0
SB_IRQ1_L[7:1]	isbus1	7	I	PINC10T	interrupts from sbus slot 1
SB_IRQ2_L[7:1]	isbus2	7	I	PINC10T	interrupts from sbus slot 2
SB_IRQ3_L[7:1]	isbus3	7	I	PINC10T	interrupts from sbus slot 3
SCSI_INT_L	iscsi_int	1	I	PINC10T	interrupt from scsi
PP_INT_L	ipar_int	1	I	PINC10T	interrupt from parallel port
GRA1_INT_L	igra_int1	1	I	PINC10T	interrupt from on board graphics
AUDIO_INT_L	iaudio_int	1	I	PINC10T	interrupt from audio device
PFAIL_INT_L	ipfail_int	1	I	PINC10T	interrupt from power supply
GRA2_INT_L	igra_int2	1	I	PINC10T	interrupt from graphics module
KEY_INT	key_int	1	I	PINY10T	interrupt from key board/ mouse/serial ports
FLOP_INT	iflop_int	1	I	PINY10T	interrupt from a floppy device
SPARE_INT_L	ispare_int	1	I	PINC10T	interrupt from a spare device
ETH_INT_L	ieth_int	1	I	PINC10T	interrupt from ethernet
SKEY_INT_L	iskey_int	1	I	PINC10T	Key board int for future use
SMOU_INT_L	ismou_int	1	I	PINC10T	Mouse Int for future use
SSER_INT_L	isser_int	1	I	PINC10T	Serial Int for future use
INT_NUM[5:0]	oint_num	6	O	BOT6T	interrupt number to U2S
SLAVIO_RST_L	oslavio_rst_	1	O	BOM4T	sbus reset to slavio

## 2.4 Scan Interface Signals

These signals connect from the RIC chip to the Scan service processor, and scan rings on the system board.

Table 2-4 Scan Interface Signals

Signal Name	Internal Signal Name	Pin Count	I/O	Driver	Description
SP_TAS	is_tas_	1	I	BIN10T	Test Address Strobe from Service Processor
SP_TDI	is_tdi	1	I	BIN10T	Test data out for Scan from Service Processor
SP_TCLK	is_tclk	1	I	BIN04T	Test Clock for Scan from Service processor
SP_TMS	is_tms	1	I	PINC10T	Test Mode Select for Scan from Service processor
SP_REST_L	is_rest_	1	I	PINC10T	Test Reset for Scan from Service Processor
SP_MP_L	is_mp_	1	I	PINC10T	Service processor present
TDI[8:1]	itdi	8	I	PIMC06F_INV	Test Data from ASICS on UUT
TCLK	tclk	1	I	BIN04T	Scan Input Clk
SP_TDO	os_tdo	1	O	BON6T	Test data out to Scan processor
TDO_A	otdo	1	O	BON6F_TRD	Test data out to all ASICS on board
TDO_B	otdo_o	1	O	BON6F_TRD	Test data out to asics(same as tdo)
TCLK_1	is_tclk	1	O	BON12F_TRD	Test Clock Out to all ASICS
TCLK_2	is_tclk	1	O	BON12F_TRD	Test Clock out(same as tclk)
TCLK_3	is_tclk	1	O	BON12F_TRD	Test Clock out(same as tclk)
TCLK_4	is_tclk	1	O	BON12F_TRD	Test Clock out(same as tclk)
TCLK_5	is_tclk	1	O	BON12F_TRD	Test Clock out(same as tclk)
TCLK_6	is_tclk	1	O	BON12F_TRD	Test Clock out(same as tclk)
TRST_L	otrst_	1	O	BON12F_TRD	Reset for spitfire clock logic
TMS[8:1]	otms	8	O	BON8F_TRD	Test Mode select for ASICS
TMS1_B	otms1_o	1	O	BON6T	Test mode select (same as tms1)
TRST_5V_L		1	O	BON10F_TRD	5V Reset for tap controller

## 2.5 Clock Controller Interface Signals

These signals connect from the RIC chip to the CPU modules and the clock generation chip.

Table 2-5 Clock Controller Signals

Signal Name	Internal Signal Name	Pin Count	I/O	Driver	Description
CPU_SP4[2:0]	icpu_sp1	3	I	PIMC06F_INV	Speed Inputs from CPUs
CPU_SP1[2:0]	icpu_sp2	3	I	PIMC06F_INV	Speed Inputs from CPUs
CPU_SP2[2:0]	icpu_sp3	3	I	PIMC06F_INV	Speed Inputs from CPUs
CPU_SP3[2:0]	icpu_sp4	3	I	PIMC06F_INV	Speed Inputs from CPUs
CLK_SEL[2:0]	oclk_sel	3	O	BOM6F_TRD	Selection Code to the Clock chip

## 2.6 PAL Block Signals

These are the Ebus interface signals to the internal logics.

*Table 2-6* Ebus and PAL Signals

Signal Name	Internal Signal Name	Pin Count	I/O	Driver	Description
EB_ADR_0	ieb_adr[0]	1	I	BIN10T	Ebus Address 0 input
EB_ADR_1	ieb_adr[1]	1	I	BIN10T	Ebus Address 0 input
EB_ADR_13	ieb_adr[2]	1	I	BIN10T	Ebus Address 0 input
EB_ADR_14	ieb_adr[3]	1	I	BIN10T	Ebus Address 0 input
EB_ADR_19	ieb_adr[4]	1	I	BIN10T	Ebus Address 0 input
EB_WR_L	ieb_wr_	1	I	PINC10T	Ebus Write bit
EB_CS_L	ieb_cs_	1	I	PINC10T	Ebus Chip Select
EB_RDY_L	oeb_rdy_	1	I	BOT10T	Ebus Ready line
PR_CSIN_L	ipr_csin_	1	I	PINC10T	Prom Chip Select In
PR_CSOUT_L	opr_csout_	1	O	BOT6T	Prom Chip Select Out
S_LOAD	os_load	1	O	BOM6F_TRD	Freq Margining chip load
S_CLOCK	os_clock	1	O	BOM6F_TRD	Freq Margining chip clock
SC_CS_L	osc_cs_	1	O	BOT6T	SC Chip Select
LC_BS_L	olc_bs_	1	O	BOT6T	LC Chip Select
XOR_IN1	ixor_in1	1	I	PINC10T	Xor Input 1
XOR_IN2	ixor_in2	1	I	PINC10T	Xor Input 2
XOROUT	oxorout	1	O	BOT6T	Xor output

## 2.7 Miscellaneous Signals

These are the clock input, reset, and other signals to the internal logics.

Table 2-7 Miscellaneous Signals

Signal Name	Internal Signal Name	Pin Count	I/O	Driver	Description
SBUS_CLK	isbus_clk	1	I	BIN04T	25 Mhz input clock
SBUS_RST_L	isbus_rst_	1	I	PINC10T	SBus Reset from U2S
ENET_CLK	ienet_clk	1	I	BIN10T	10 Mhz clk for reset counter

## 2.8 RIC Pin Count

Table 2-8 RIC Pin Count

Subblock	In	Out	Total
Reset	3	3	6
Interrupt	41	7	48
Scan	15	20	35
Clock Control	12	3	15
PAL Block	11	6	17
Misc	3	0	3
Subtotal	85	39	124
Vcc/VDD/Gnd			36
Total			160



### *3.1 RIC Overview*

This chapter contains the functional description of the RIC chip at the top level. Overall features of the chip functional description of each major sub block are discussed.

RIC chip contains five major modules. The five modules are:

- Reset
- Interrupt Concentrator
- Scan Control
- Clock Control
- PAL block

There are no major data paths between any of these blocks.

## 3.2 Detailed Internal Block Diagram

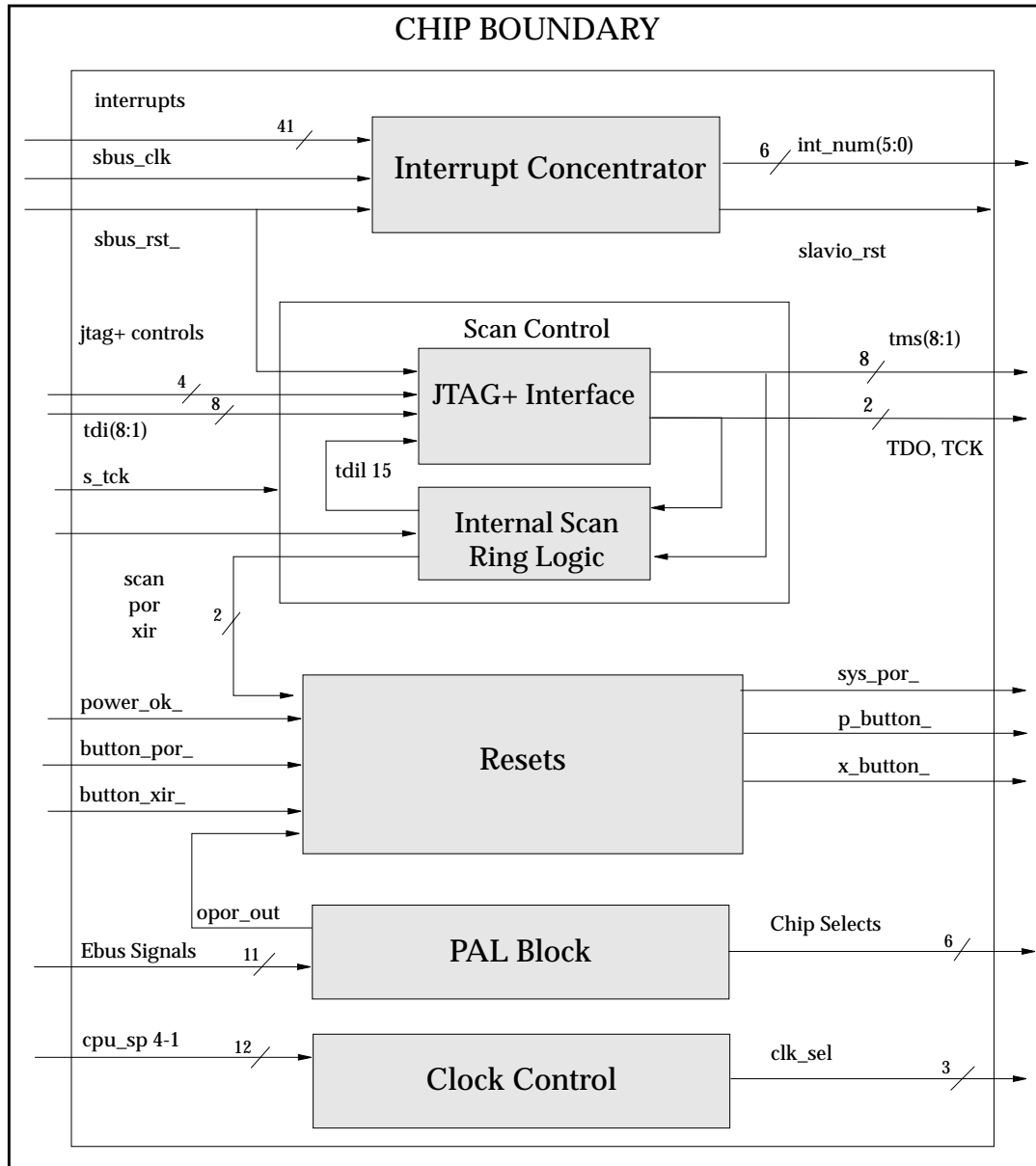


Figure 3-1 RIC Chip Block Diagram



### 3.3 Block Overviews

The following chapters describe each functional block in detail. These chapters are organized as follows.

#### 3.3.1 Resets

This block is responsible for generating resets to the System Controller. It generates three different resets: Power On Reset, Button POR and Button XIR. The assertion and de-assertion of all the resets are asynchronous to the system clock. This block can receive resets from external power detect logic on AC power on, POR and XIR resets from external switch button and from the internal Scan Controller block.

#### 3.3.2 Interrupt Concentrator

The function of the Interrupt Concentrator block is to detect any active interrupt coming from any of the sbus slots, SLAVIO or UPA expansion ports, generate the interrupt source number and deliver it to the U2S. This block does not communicate with any other blocks in the chip.

#### 3.3.3 Scan Controller

The function of the Scan Controller is to act as an interface between a remote service processor and the Unit under test (UUT). It supports the JTAG+ Bus protocol. It controls the scan rings for all the ASICs and the processor Modules. It can also generate POR and XIR resets.

The Scan Controller has two subblocks. JTAG+ Interface block and Internal Scan Ring block. The JTAG+ interface block interfaces to the external service processor to receive/send scan data and control signals. It also interfaces to the system board and controls all the scan chains. It can receive up to 8 different scan chains and selects one of the chain to pass it along to the service processor. TDI15 scan ring chain is from the Internal scan ring block. The internal scan ring uses TMS 15 and TDO signals from the JTAG+ interface block to set up its data.

#### 3.3.4 Clock Control

The function of the Clock Control block is to select the clock speed based on the input from processor modules. The clock select signals will be used by the clock chip to generate the system clock.

### **3.3.5 PAL Block**

The function of the PAL Block is to provide decode logic for the SC and Lab Console address and data registers, Freq Margining chip support and PROM write address decode. Additionally it also provides an XOR gate.

### **3.3.6 RIC Gate Counts**

*Table 3-1*      RIC Gate Count estimates

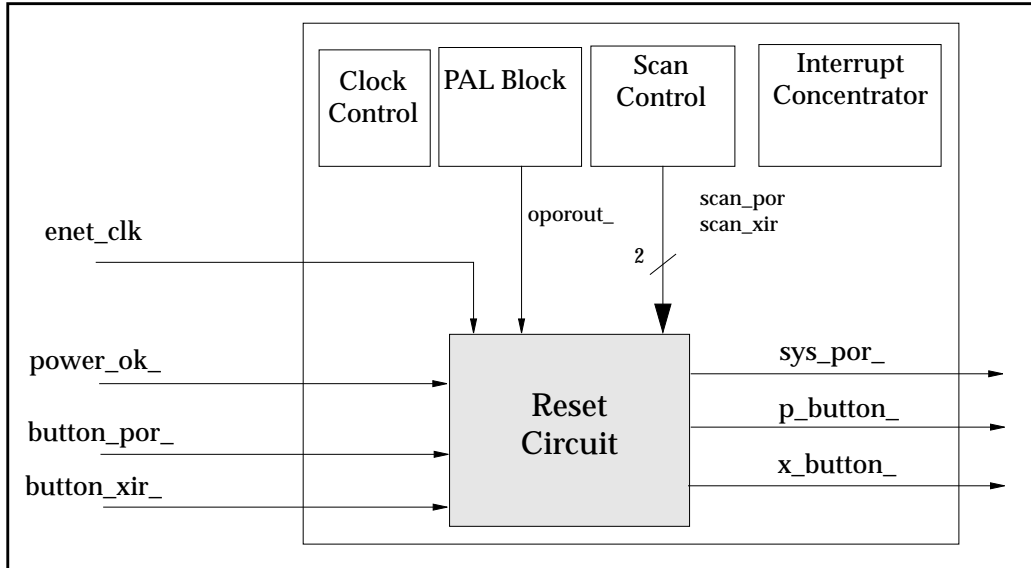
Subblock	Gate Count
Reset	600
Interrupt	1800
Scan	500
Clock Control	50
PAL	250
Total	3200

### *4.1 Overview*

The reset logic on the UltraSPARC system boards is divided among the RIC chip, system controller (SC), U2S and discrete components. The RIC chip generates three different reset signals to the system controller (SC) chip. Based on these resets, the SC will generate resets to the rest of the system. RIC uses six different sources (POR, button POR, PAL Block POR, SCAN\_POR, button XIR and SCAN\_XIR) to generate the three resets. Each of these three resets are handled differently by SC.

#### *4.1.1 Reset Logic*

The reset logic is divided into three different blocks: POR, Button POR, Button XIR. The POR block generates a reset (SYS\_POR\_L) on AC power on. The Button POR block generates a reset (P\_BUTTON\_RESET\_L) when it gets a signal from the POR press button switch on the lab console board, POR signal from the PAL block or SCAN\_POR command from the JTAG Scan controller. The Button XIR block generates a reset (X\_BUTTON\_RESET\_L) when it gets a signal from the XIR press button switch from the lab console board or SCAN\_XIR command from the JTAG Scan controller.



*Figure 4-1* RIC Chip Block

### 4.1.2 Reset Logic Detailed Block Diagram

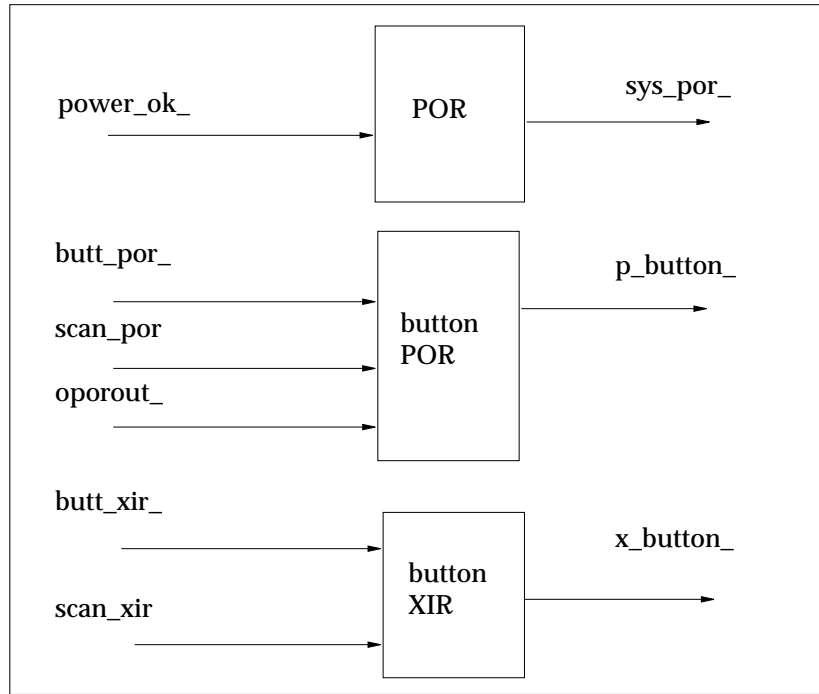


Figure 4-2 Reset Block

### 4.1.3 Reset Logic Gate Count Estimates

The current gate equivalent for the reset block is approximately 600 gate equivalents.

## 4.2 Signal Descriptions

The reset block diagram looks as follows:

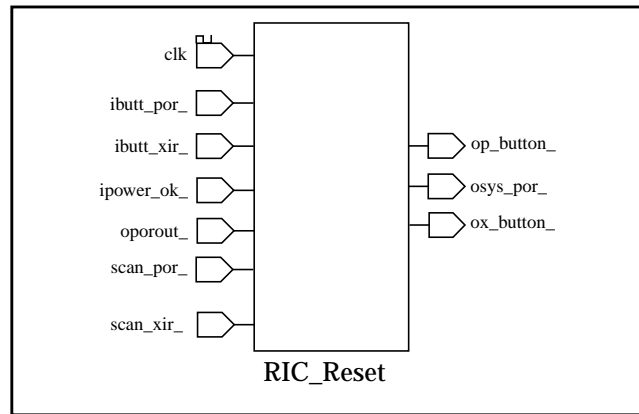


Figure 4-3 RIC\_Reset

Table 4-1 Input/Output Signals of Reset Logic

Signal Name	Signals	I/O	Description
ipower_ok_	1	I	Signal asserted during power on
ibutt_por_	1	I	Signal asserted by push button switch
scan_por	1	I	Signal asserted by Scan Controller
oporout_	1	I	Signal asserted by PAL block
ibutt_xir_	1	I	Signal asserted by push button switch
scan_xir	1	I	Signal asserted by Scan controller
ienet_clk	1	I	10 mhz clock
osys_por_	1	O	Signal to SC during power on reset
op_button_	1	O	Signal to SC asserted during push button power reset
ox_button_	1	O	Signal to SC asserted during push button XIR.

### 4.3 Reset Logic Functional Description

The following sections give a detailed description of all the sub blocks associated with reset generation logic.

#### 4.3.1 POR Description

##### 4.3.1.1 Overview

The POR logic is responsible for generating power on reset signal and keeping it low for a minimum of 2ms.

##### 4.3.1.2 POR Block Diagram

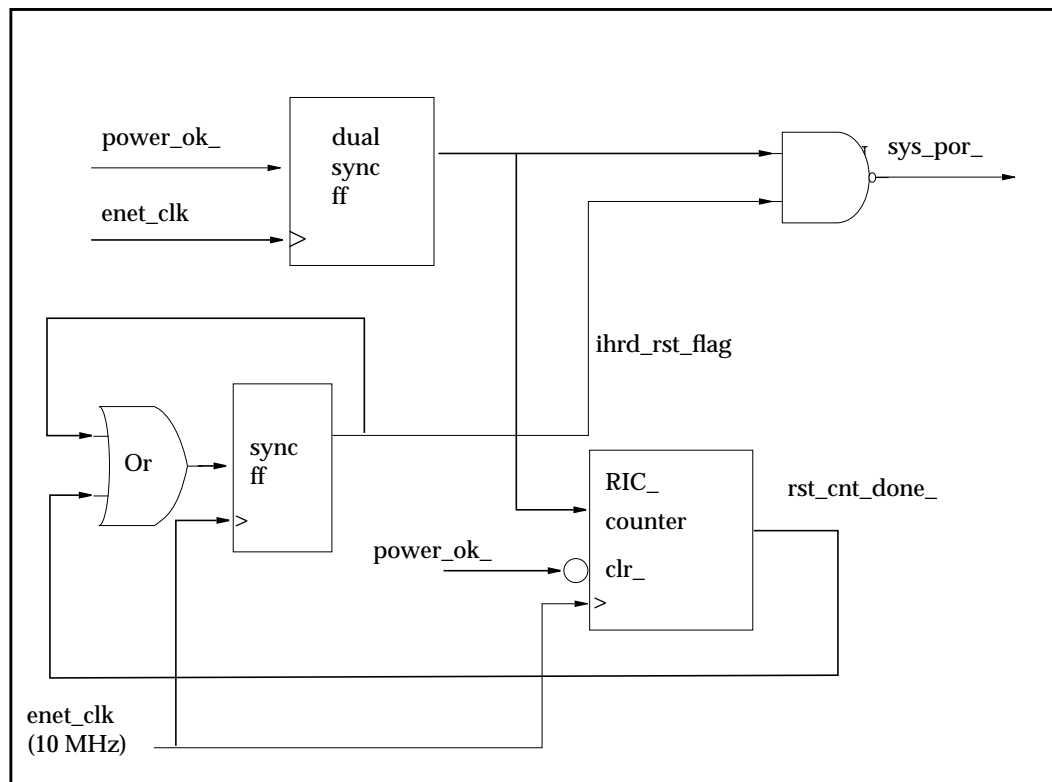


Figure 4-4 POR - Detailed Block

### 4.3.1.3 POR State Descriptions

The sys\_por\_ will be asserted asynchronously when it detects the power\_ok\_ signal, from the power supply. Once the power\_ok\_ gets de-asserted the counter starts counting . When the counter reaches full count, the rst\_cnt\_done will de-assert the ihrd\_rst\_flag, which in turn will de-assert sys\_por\_. It will be asynchronous to the SC chip which runs on the system clock frequency.

### 4.3.1.4 POR Timing Diagram

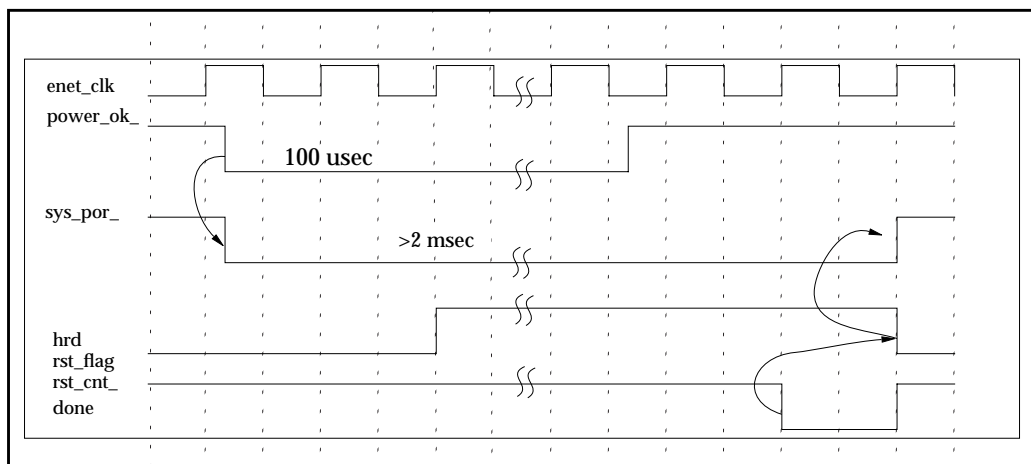


Figure 4-5 POR Timing Diagram

## 4.3.2 Button\_POR description

### 4.3.2.1 Overview

The Button\_POR block is responsible for generating a reset when it gets a signal from button por reset switch on the lab console board, a porout signal from the PAL block or a scan\_por command from Scan controller.



### 4.3.2.2 Button\_POR Block Diagram

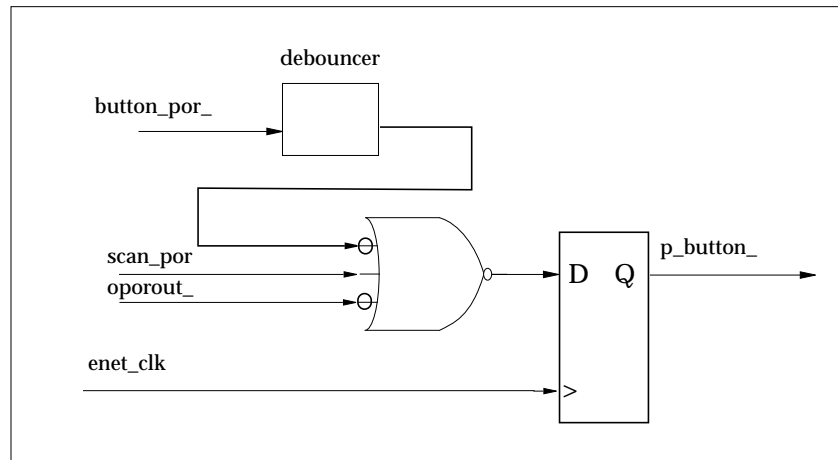


Figure 4-6 Button\_POR - Detailed Block

### 4.3.2.3 Button\_POR State Descriptions

The `button_por_` signal coming from the switch will go through a debouncer circuit and combinatorial logic before it goes into a flip flop. So when `scan_por` or `button_por_` are asserted, the flip flop will assert the `p_button_` for at least 1 clock cycle.

### 4.3.2.4 Button\_POR Timing Diagram

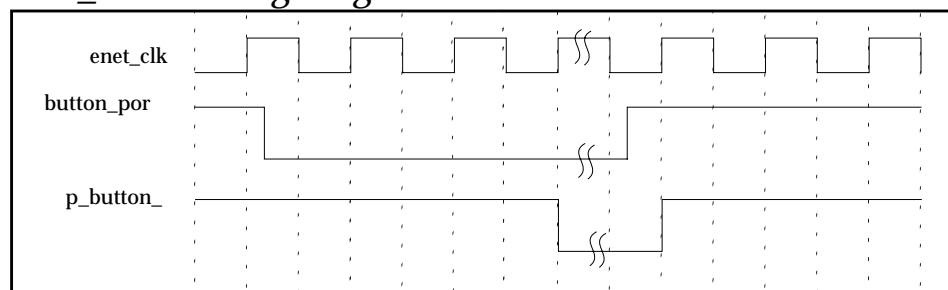


Figure 4-7 Button\_POR Timing

### 4.3.3 Button\_XIR Description

#### 4.3.3.1 Overview

The Button\_XIR block is responsible for generating a reset when it gets a signal from lab console button xir reset switch or a SCAN\_XIR command from Scan controller.

#### 4.3.3.2 Button\_XIR Block Diagram

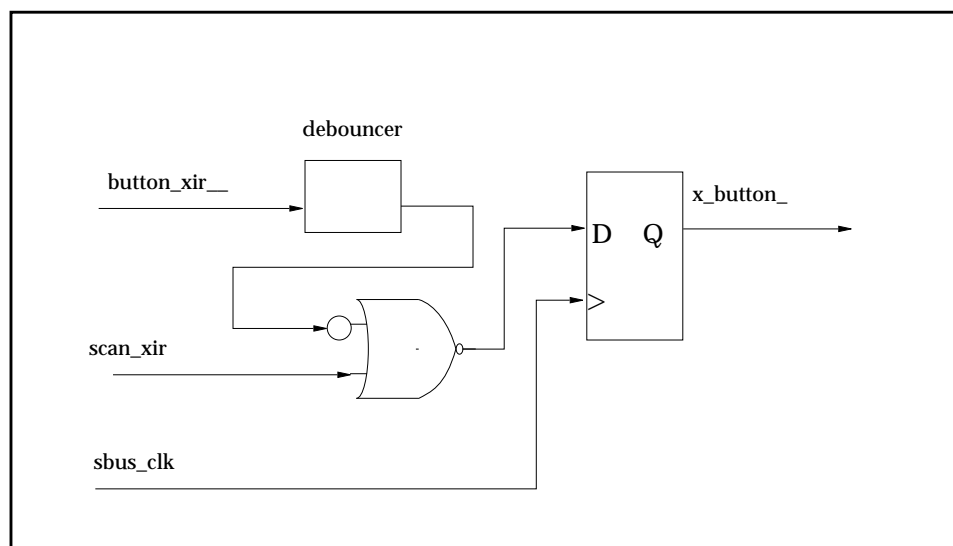


Figure 4-8 Button\_XIR - Detailed Block

#### 4.3.3.3 Button\_XIR State Descriptions

The button\_xir\_ signal coming from the switch will go through a debouncer circuit and combinatorial logic before it goes into a flip flop. So when scan\_xir or button\_por\_ are asserted, the flip flop will assert the p\_button\_ for at least one clock cycle.

### 4.3.3.4 Button\_XIR Timing Diagram

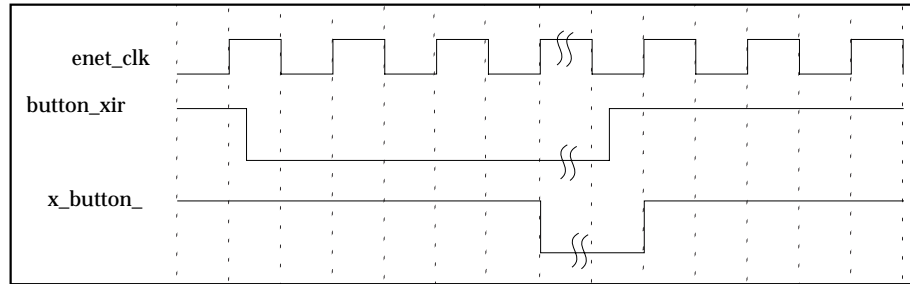


Figure 4-9 Button\_XIR Timing



## 5.1 Overview

The function of the interrupt concentrator is to detect all the active interrupts and provide the U2S with interrupt numbers. The major goal of designing this logic inside the RIC chip is to support the large number of interrupt sources and reduce the pin count of the U2S chip.

### 5.1.1 Interrupt Concentrator Overview

The interrupt concentrator is basically a two level round robin priority encoder. There are two major blocks in the Interrupt Concentrator: Encoder and Dispatcher.

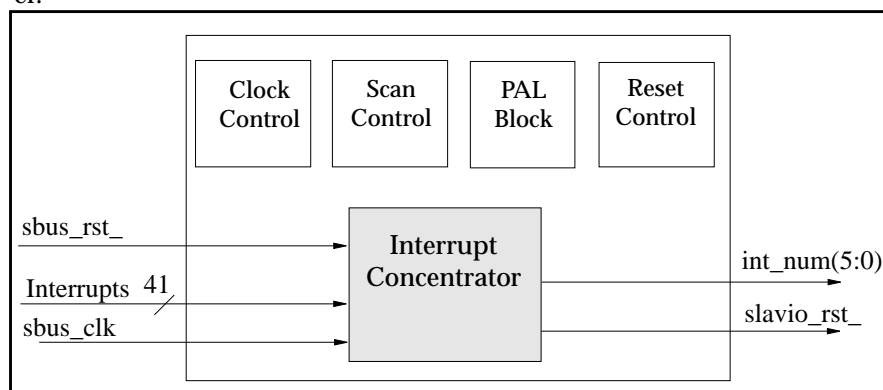


Figure 5-1 Interrupt Concentrator System Block Diagram

### 5.1.2 Interrupt Concentrator Detailed Block Diagram

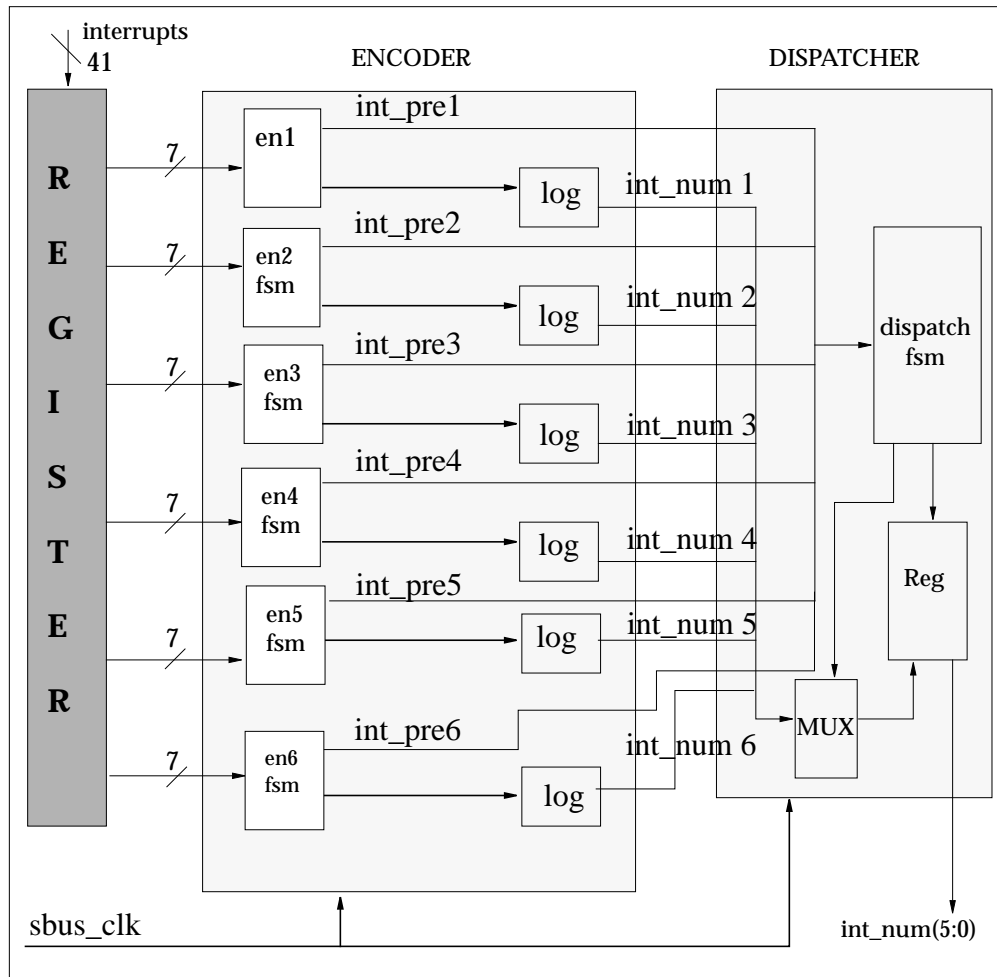


Figure 5-2 Interrupt Concentrator Block

### 5.1.3 Interrupt Concentrator Gate Count Estimates

Current gate count is approximately 1800 gate equivalents

## 5.2 Signal Descriptions

The Interrupt controller block can be shown as follows:

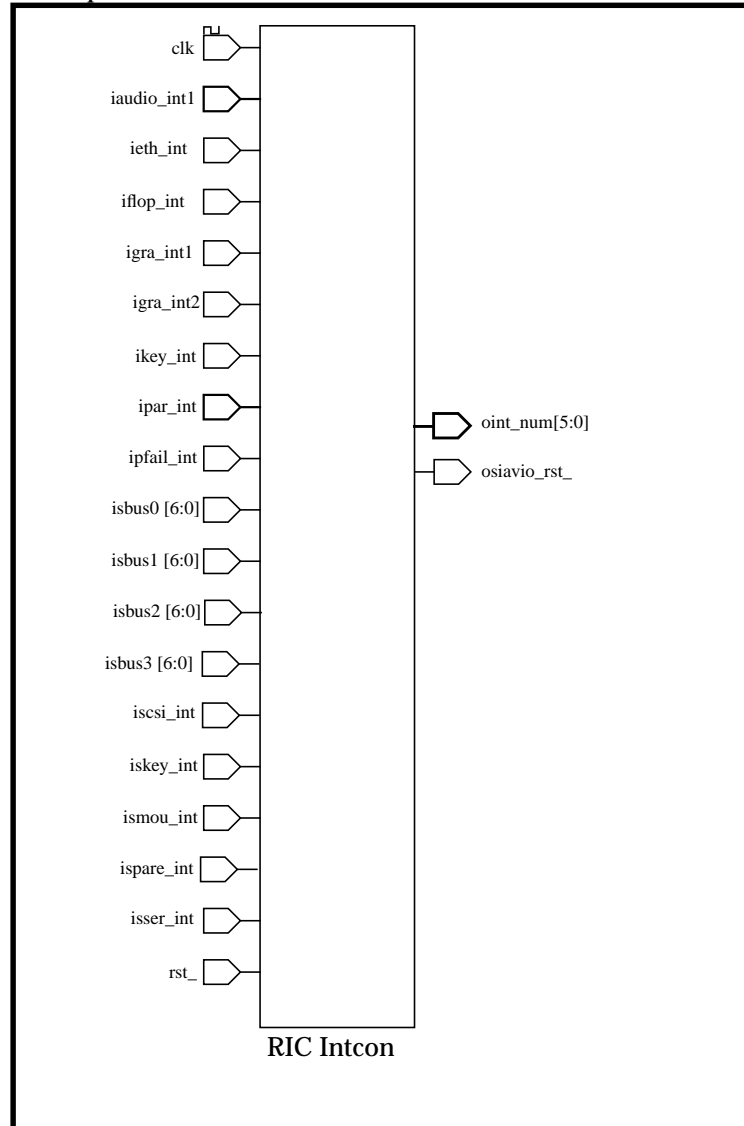


Figure 5-3 Ric Intcon

*Table 5-1*      **Interrupt Signals**

Signal name	Signals	I/O	Description
isbus_clk	1	I	25 Mhz input clock
iaudio_int_	1	I	interrupt from audio device
ieth_int_	1	I	interrupt from ethernet
iflop_int	1	I	interrupt from a floppy device
igra_int1_	1	I	interrupt from on board graphics
igra_int2_	1	I	interrupt from graphics mod
ikey_int	1	I	interrupt form key board
ipar_int_	1	I	interrupt from parallel port
ipfail_int_	1	I	interrupt from power supply
isbus0_int_(6:0)	7	I	interrupts from sbus slot 0
isbus1_int_(6:0)	7	I	interrupts from sbus slot 1
isbus2_int_(6:0)	7	I	interrupts from sbus slot 2
isbus3_int_(6:0)	7	I	interrupts from sbus slot 3
iscsi_int_	1	I	interrupt from scsi
iskey_int_	1	I	spare int for keyboard
ismou_int_	1	I	spare int for mouse
ispare_int_	1	I	interrupt from a spare device
isser_int	1	I	spare int for serial port
isbus_rst_	1	I	sbus reset
oint_num(5:0)	6	O	interrupt number to U2S
oslavio_rst_	1	O	Registered sbus reset to slavio

### ***5.3 Interrupt Concentrator Functional Description***

The encoder receives interrupts from SBUS slots, SLAVIO, and the UPA expansion slots. There are a total of 41 interrupts. The encoder constantly monitors for an active interrupt and generates a interrupt number associated with the device. It will signal the interrupt dispatcher state machine that there is a valid interrupt



## *5. Interrupt Concentrator*

present. There is a three clock cycle latency after the interrupt is detected before it is sent out to the U2S. Since all the interrupts coming are asynchronous, it will be double clocked and stored in a register before any encoding takes place.

The outputs of this six sub groups are fed into the second stage of decoding. After the arbitration is complete at second stage, the interrupt vector is sent to the U2S on the next clock edge.

### *5.3.1 Encoder Description*

#### *5.3.1.1 Overview*

The incoming interrupts are divided into seven sub groups before any encoding is done. This has to be done to simplify the encoding logic. Each subgroup is comprised of a state machine and some combinatorial logic to do the arbitration. The state machine uses round robin arbitration scheme to service an interrupt. Once the interrupt source has been determined, the interrupt number will be loaded into a register associated with that particular sub group. Also an interrupt valid signal is asserted to inform the dispatcher (second stage) state machine.

### 5.3.1.2 Encoder Block Diagram

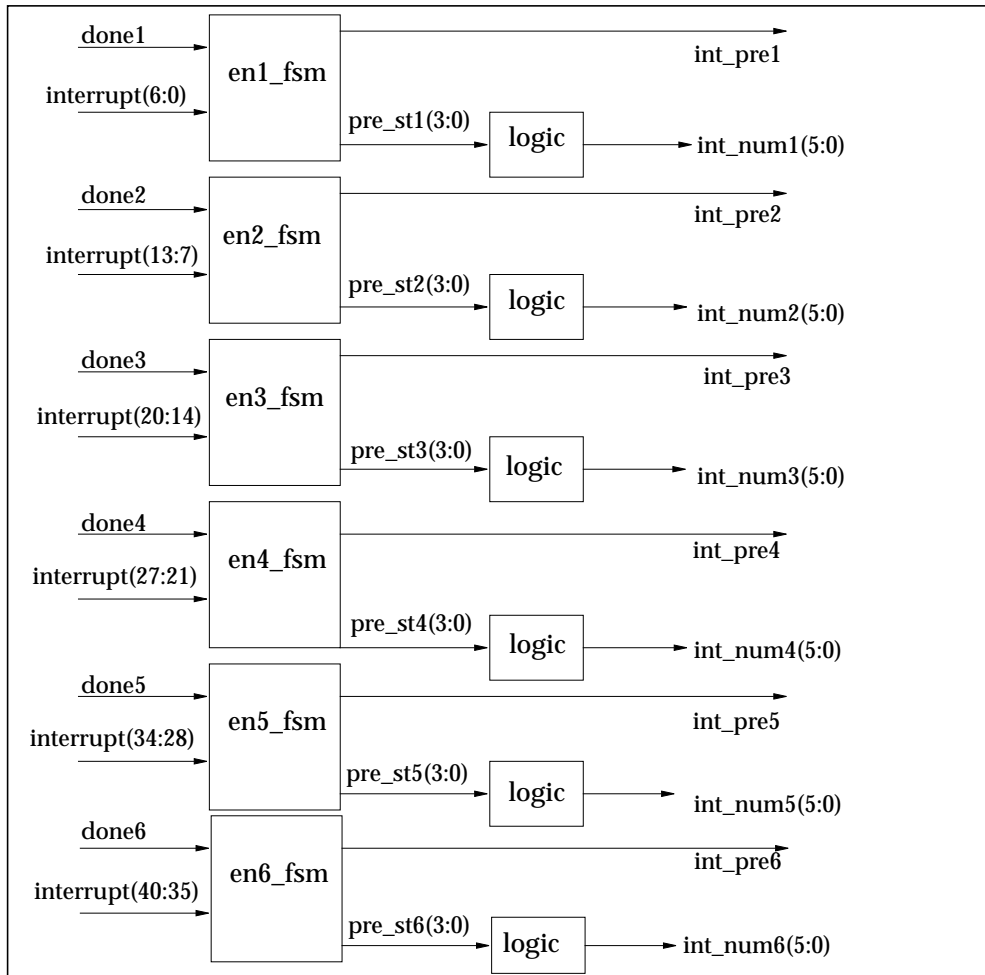


Figure 5-4 Encoder Block

### 5.3.1.3 EN1\_FSM State Descriptions

The state machine could be in any one of the 16 different states when an interrupt occurs. There are seven idle states and seven dispatch states corresponding to each of the eight interrupts coming in. Each idle state waits in that state until an interrupt occurs. Once it detects an interrupt, it goes to the dispatch state of that interrupt which is being serviced. In that state, it does two things. It drives an int\_pre signal to the second stage, which is the dispatch state machine. Also it drives control signals to some combo logic which enables the right interrupt number get loaded into the register on the following clock edge. Then it waits in that state until it sees a done signal from the dispatcher state machine from the second stage. Once it sees the done signal, it goes to the idle state of the next higher interrupt from the one being serviced.

Table 5-2 en1\_fsm State Description

State	Description
eid1	waiting for an interrupt, 1 has the highest pri.
eid2	waiting for an interrupt, 2 has the highest pri.
eid3	waiting for an interrupt, 3 has the highest pri.
eid4	waiting for an interrupt, 4 has the highest pri
eid5	waiting for an interrupt, 5 has the highest pri
eid6	waiting for an interrupt, 6 has the highest pri
eid7	waiting for an interrupt, 7 has the highest pri
edi1	send out int number 1, goes to eid 2 on "done"
edi2	send out int number 2, goes to eid3 on "done"
edi3	send out int number 3, goes to eid4 on "done"
edi4	send out int number 4, goes to eid5 on "done"
edi5	send out int number 5, goes to eid6 on "done"
edi6	send out int number 6, goes to eid7 on "done"
edi7	send out int number 7, goes to eid8 on "done"

### 5.3.1.4 EN\_FSM1 State diagram

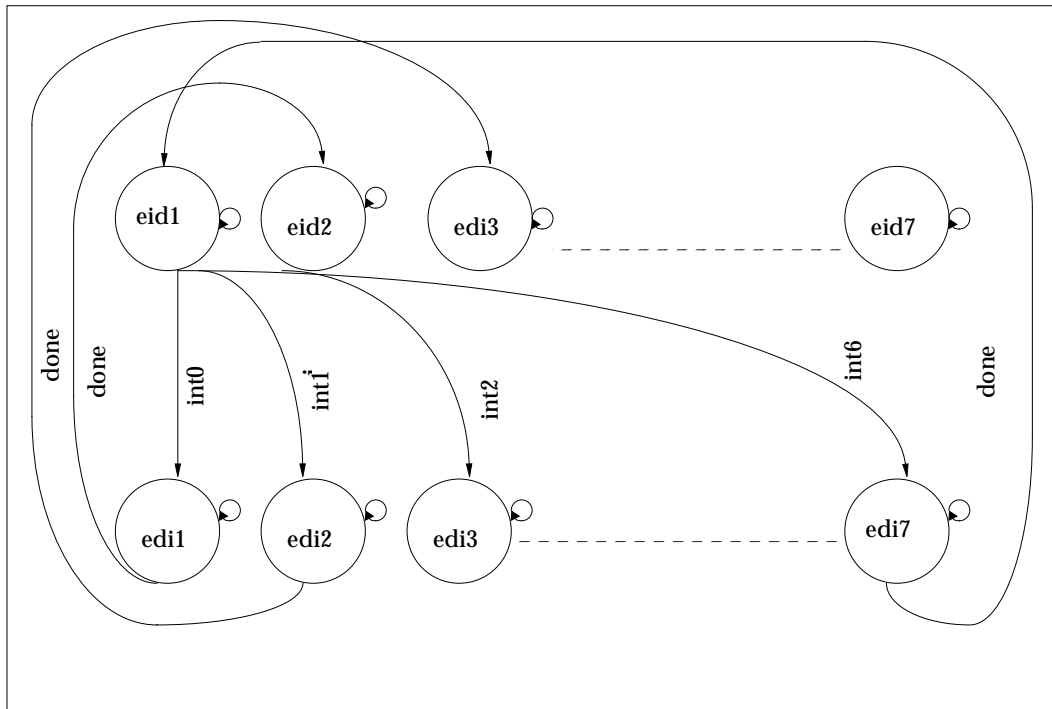


Figure 5-5 State Diagram of the First Stage encode

### 5.3.1.5 EN\_FSM1 State Machine Description

**EID1:** /\* Goes to this state on reset or whenever a when a transition is completed in state edi8 \*/

**SYNC OUTPUTS:**/\* Everything is inactive in EID \*/

**TRANSITIONS:**

**EDI1:**

/\*interrupt number 1 is being serviced\*/

If (int1 then go to state EDI1)

**EDI2:** /\* interrupt number 2 is being serviced\*/

If (!int1 & int2 then go to EDI2)

**EDI3:** /\*interrupt number3 is being serviced \*/

If (!int1 &!int2 & int3 then go to EDI3)

**EDI4:** /\* interrupt number 4 is being serviced\*/

If (!int1 &!int2 &!int3 & int4 then go to EDI4)

**EDI5:** /\* interrupt number 5 is being serviced \*/

If (!int1&!int2 &!int3 &!int4 & int5 then go to EDI5)

**EDI6:** /\*interrupt number 6 is being serviced \*/

If (!int1&!int2 &!int3 &!int4 &!int5 & int6 then go to EDI6)

**EDI7:** /\*interrupt number 7is being serviced \*/

If (!int1&!int2 &!int3 &!int4 &!int5 &!int6 & int7 then go to EDI7)

**EDI1:** /\*First stage of interrupt dispatch cycle\*/

**SYNC OUTPUTS :** /\*dispatch int valid and load the int register\*/

int\_pre1 = 1; int\_num1= 0;

**TRANSITIONS:**

**EID2:** /\* go to EID2 state and wait for the next interrupt\*/

If (done go to EID2)

**EDI1:** default

**EDI2:** /\*First stage of interrupt dispatch cycle\*/

**SYNC OUTPUTS:** /\*dispatch int valid and load the int register\*/

int\_pre1 = 1; int\_num1 = 1;

**TRANSITIONS:**

**EID3:** /\* go to EID3 state and wait for the next interrupt\*/

If (done go to EID3)

**EDI2:** default

**EDI3:**    /\*First stage of interrupt dispatch cycle\*/  
      *SYNC OUTPUTS:*       /\*dispatch int valid and load the int register\*/  
                                  int\_pre1 = 1; int\_num1= 2;.  
  
      **TRANSITIONS:**  
              **EID4:**       /\* go to EID3 state and wait for the next interrupt\*/  
                                  If (done go to EID4)  
              **EDI3:**       default

**EDI4:**    /\*First stage of interrupt dispatch cycle\*/  
      *SYNC OUTPUTS:*       /\*dispatch int valid and load the int register\*/  
                                  int\_pre1 = 1; int\_num1 = 3;  
  
      **TRANSITIONS:**  
              **EID5:**       /\* go to EID5 state and wait for the next interrupt\*/  
                                  If (done go to EID5)  
              **EDI4:**       default

**EDI5:**    /\*First stage of interrupt dispatch cycle\*/  
      *SYNC OUTPUTS:*       /\*dispatch int valid and load the int register\*/  
                                  int\_pre1 = 1; int\_num1 = 4;  
  
      **TRANSITIONS:**  
              **EID6:**       /\* go to EID6 state and wait for the next interrupt\*/  
                                  If (done go to EID6)  
              **EDI5:**       default

**EDI6:**    /\*First stage of interrupt dispatch cycle\*/  
      *SYNC OUTPUTS:*       /\*dispatch int valid and load the int register\*/  
                                  int\_pre1 = 1; int\_num1 = 5;  
  
      **TRANSITIONS:**  
              **EID7:**       /\* go to EID7 state and wait for the next interrupt\*/  
                                  If (done go to EID7)  
              **EDI6:**       default

**EDI7:**    /\*First stage of interrupt dispatch cycle\*/  
      *SYNC OUTPUTS:*       /\*dispatch int valid and load the int register\*/  
                                  int\_pre1 = 1; int\_num1 = 6;  
  
      **TRANSITIONS:**  
              **EID8:**       /\* go to EID3 state and wait for the next interrupt\*/  
                                  If (done go to EID1)  
              **EDI7:**       default

### 5.3.1.6 Encoder Timing Diagrams

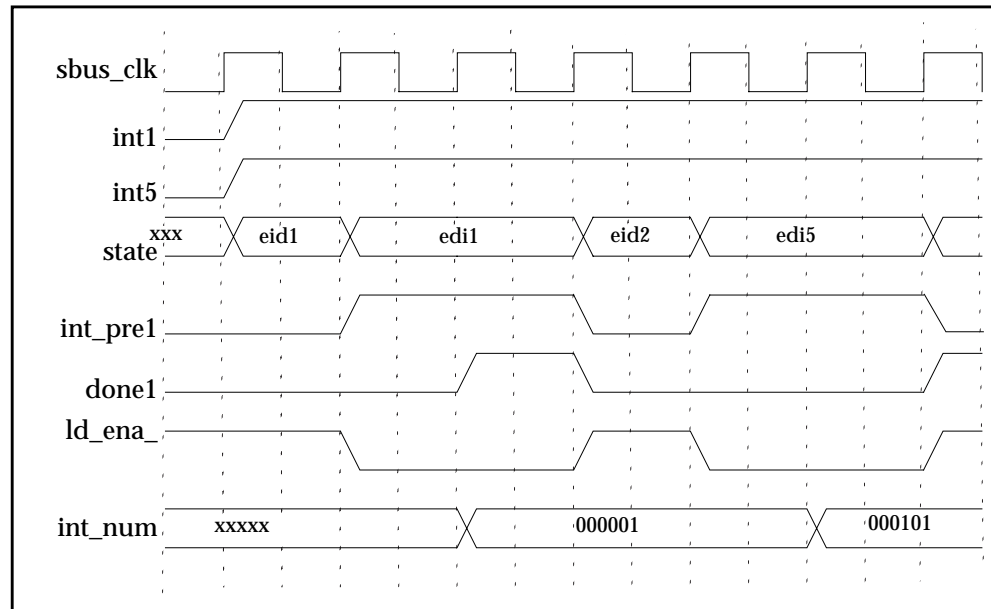


Figure 5-6 Encoder 1 Timing

### 5.3.2 Dispatcher Description

#### 5.3.2.1 Overview

The dispatcher state machine basically dispatches the active interrupt to the U2S chip. As with the encoder state machine, this state machine also uses round robin algorithm to select an active interrupt to be serviced. Potentially there could be six interrupts active from the stage one encoder state machines. Once the dispatcher selects the interrupt source, it loads its interrupt register with the interrupt number from the appropriate sub group. On the next clock cycle, the interrupt number is driven on the bus for U2S to service. Also the dispatcher generates a “done” signal for the first stage so that the encoder state machine (first stage) can continue with its arbitration. This process continues until there are no interrupts pending. When all interrupts have been serviced, the dispatcher will send idle interrupt number (all 1’s) to U2S.

### 5.3.2.2 Dispatcher Block Diagram

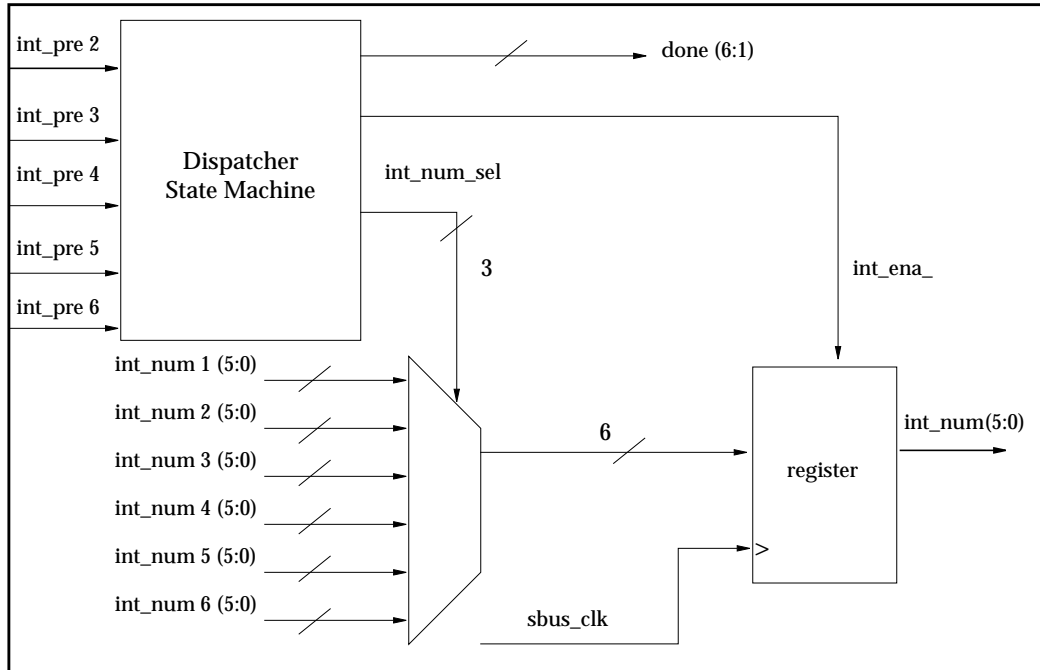


Figure 5-7 Dispatcher Block



### 5.3.2.3 Dispatcher State Description

The dispatcher state machine is comprised of twelve different states. Based on its present state, the next higher interrupt will be selected to be passed onto the U2S. There are six idle states and six dispatch states. The following table explains the different states.

Table 5-3 State Table description

State	Description
did1	Wait for interrupt, Enc1 output has highest pri
did2	Wait for interrupt, Enc2 output has highest pri
did3	Wait for interrupt, Enc3 output has highest pri
did4	Wait for interrupt, Enc4 output has highest pri
did5	Wait for interrupt, Enc5 output has highest pri
did6	Wait for interrupt, Enc6 output has highest pri
ddi1	dispatch ENC1 interrupt, go to DID2 state or DDI2 state
ddi2	dispatch ENC2 interrupt, go to DID3 state or DDI3 state
ddi3	dispatch ENC3 interrupt, go to DID4 state or DDI4 state
ddi4	dispatched ENC4 interrupt, go to DID5 state or DDI5 state
ddi5	dispatch ENC5 interrupt, go to DID6 state or DDI6 state
ddi6	dispatch ENC6 interrupt, go to DID1 state or DDI1 state

### 5.3.2.4 Dispatcher State Machine Diagram

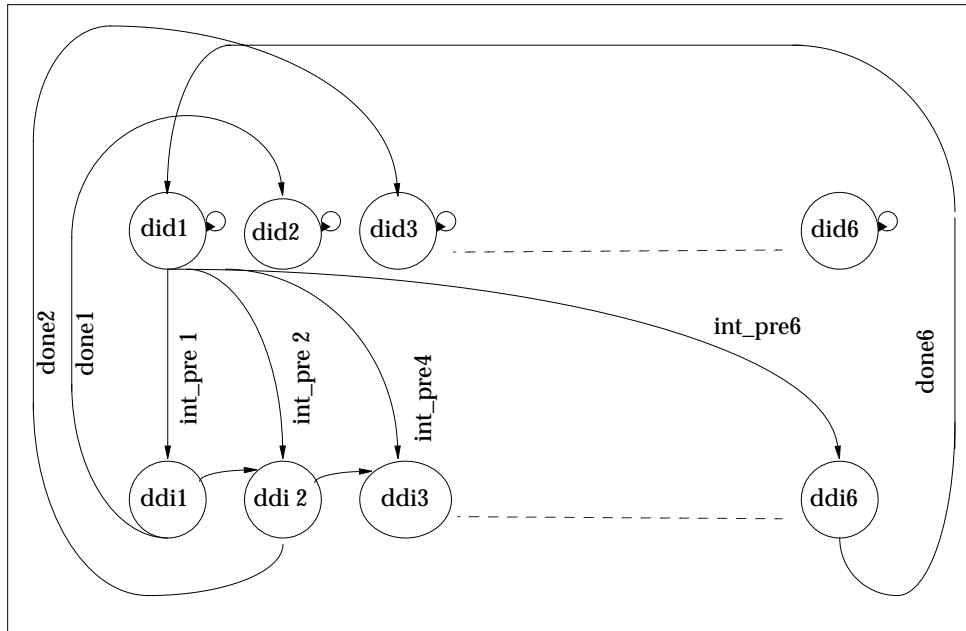


Figure 5-8 Dispatcher State Machine

### 5.3.2.5 Dispatcher State Machine description

**DDI1:** /\* Goes to this state on reset or whenever a when a transition is completed in state ddi5 \*/

**SYNC OUTPUTS:** /\* Everything is inactive in DID \*/

**TRANSITIONS:**

**DDI1:** /\*interrupt from ENC1 is being serviced\*/  
If (int\_pre1 then go to state DDI1)

**DDI2:** /\* interrupt number 2 is being serviced\*/  
If (!int\_pre1 & int\_pre2 then go to DDI2)

**DDI3:** /\*interrupt number3 is being serviced \*/  
If (!int\_pre1 & !int\_pre2 & int\_pre3 then go to DDI3)

**DDI4:** /\* interrupt number 4 is being serviced\*/  
If (!int\_pre1 & !int\_pre2 & !int\_pre3 & int\_pre4 then go to DDI4)

**DDI5:** /\* interrupt number 5 is being serviced \*/  
If (!int\_pre1 & !int\_pre2 & !int\_pre3 & !intPre4 & intPre5 then go to DDI5)

**DDI6:** /\* interrupt number6 is being serviced \*/  
If (!int\_pre1 & !int\_pre2 & !int\_pre3 & !intPre4 & !intPre5 & int\_pre6 then go to DDI6)

**DDI1:** /\*First stage of interrupt dispatch cycle\*/

**SYNC OUTPUTS:** /\* load the int register\*/  
int\_sel(2:0)= 1; int\_en\_=0; done1= 1;

**TRANSITIONS:**

**DDI2:** If (int\_pre2)

**DDI3:** If(int\_pre3)

**DDI4:** If(int\_pre4)

**DDI5:** if(int\_pre5)

**DDI6:** if(int\_pre6)

**DID2:** default

**DDI2:** /\*First stage of interrupt dispatch cycle\*/

**SYNC OUTPUTS:** /\*dispatch int valid and load the int register\*/  
int\_sel(2:0)= 2; int\_en\_=0; done2=1;

**TRANSITIONS:**

**DDI3:** If(int\_pre3)

**DDI4:** If(int\_pre4)

**DDI5:** if(int\_pre5)

**DDI6:** if(int\_pre6)

**DDI1:** If (int\_pre1)

**DID3:** default

**DDI3:**    /\*First stage of interrupt dispatch cycle\*/  
*SYNC OUTPUTS:*               /\*dispatch int valid and load the int register\*/  
                                   int\_sel(2:0)= 3; int\_en\_=0; done3=1;.

**TRANSITIONS:**

DDI4	If(int_pre4)
DDI5	if(int_pre5)
DDI6	if(int_pre6)
DDI1:	If (int_pre1)
DDI2:	If(int_pre2)
<b>DID4:</b>	default

**DDI4:**    /\*First stage of interrupt dispatch cycle\*/  
*SYNC OUTPUTS:*               /\*dispatch int valid and load the int register\*/  
                                   int\_sel(2:0)= 4; int\_en\_=0; done4=1;

**TRANSITIONS:**

DDI5	if(int_pre5)
DDI6	if(int_pre6)
DDI1:	If (int_pre1)
DDI2:	If(int_pre2)
DDI3	If(int_pre3)
<b>DID5:</b>	default

**DDI5:**    /\*First stage of interrupt dispatch cycle\*/  
*SYNC OUTPUTS:*               /\*dispatch int valid and load the int register\*/  
                                   int\_sel(2:0)= 5; int\_en\_=0; done5=1;

**TRANSITIONS:**

DDI6	if(int_pre6)
DDI1:	If (int_pre1)
DDI2:	If(int_pre2)
DDI3	If(int_pre3)
DDI4	if(int_pre4)
<b>DID6:</b>	default

**DDI6:**    /\*First stage of interrupt dispatch cycle\*/  
*SYNC OUTPUTS:*               /\*dispatch int valid and load the int register\*/  
                                   int\_sel(2:0)= 6; int\_en\_=0; done6=1;

**TRANSITIONS:**

DDI1:	If (int_pre1)
DDI2:	If(int_pre2)
DDI3	If(int_pre3)
DDI4	if(int_pre4)
DDI5	if(int_pre5)
<b>DID1:</b>	default

### 5.3.2.6 Dispatcher Timing Diagram

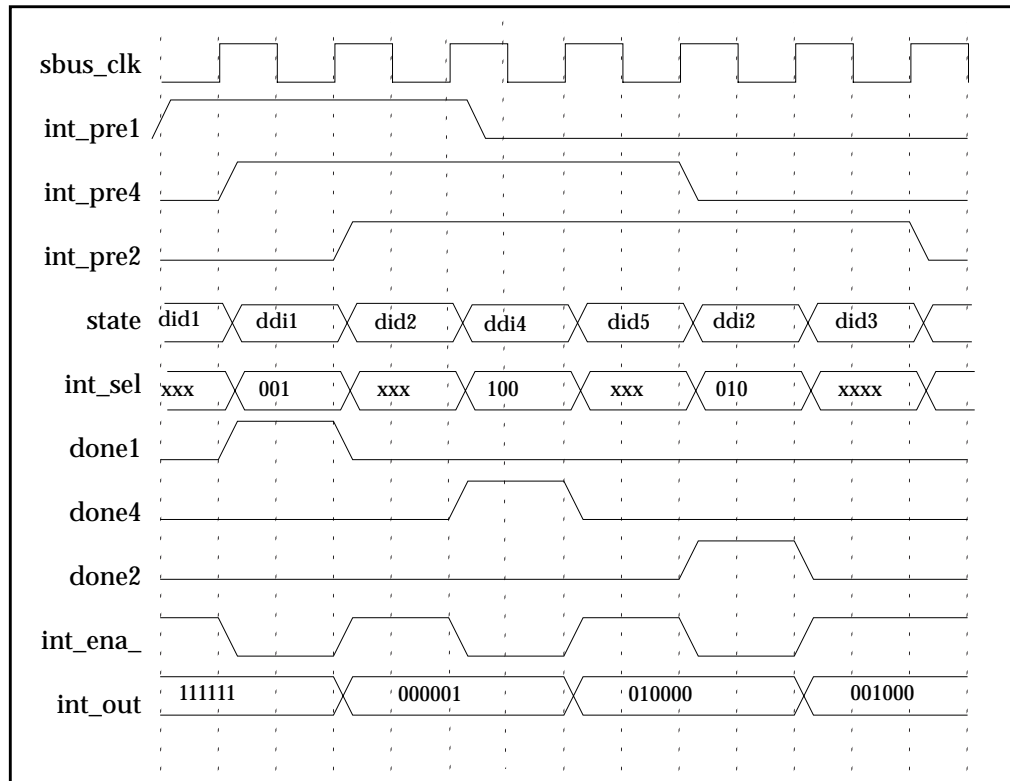


Figure 5-9 Dispatcher Timing



## 6.1 Overview

The scan controller is used for controlling the scan rings on the system mother boards. The major goal is to provide an efficient debug and test tool that could be used by both engineering and manufacturing personnel.

### 6.1.1 Scan Controller Overview

The Scan Controller has two major blocks associated with it: JTAG+ Interface and JTAG Internal Scan Ring.

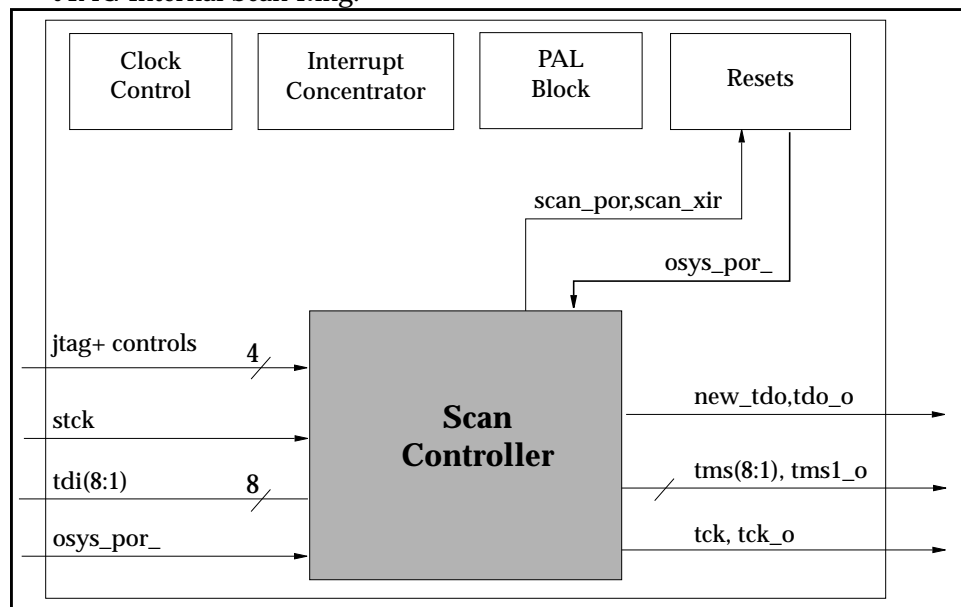
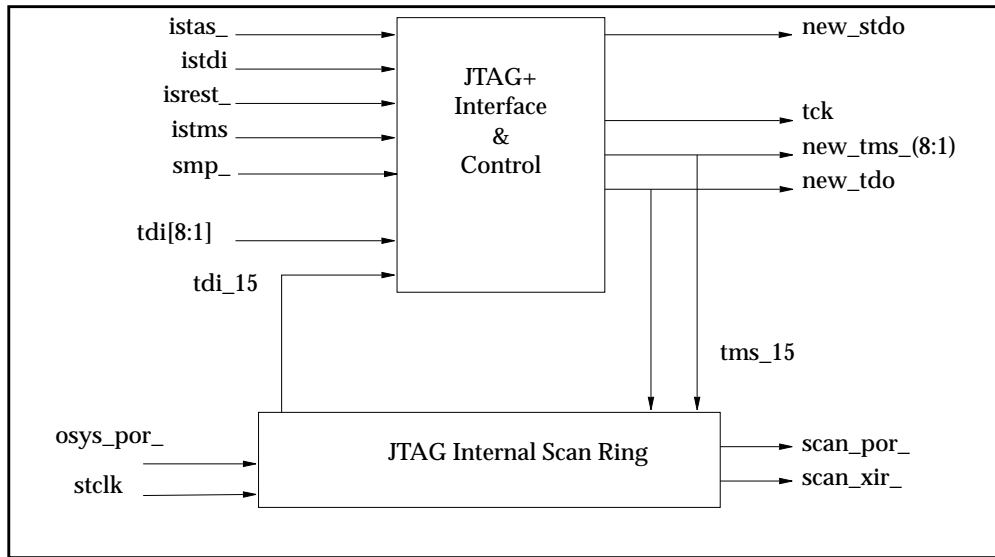


Figure 6-1 Scan Controller System Block



*Figure 6-2* Scan Controller Block Diagram

### **6.1.2 Scan Controller Gate Count Estimates**

The current gate count for the scan controller is 500 gate equivalents



## 6.2 Signal Descriptions

The block diagram for the scan controller can be shown as follows:

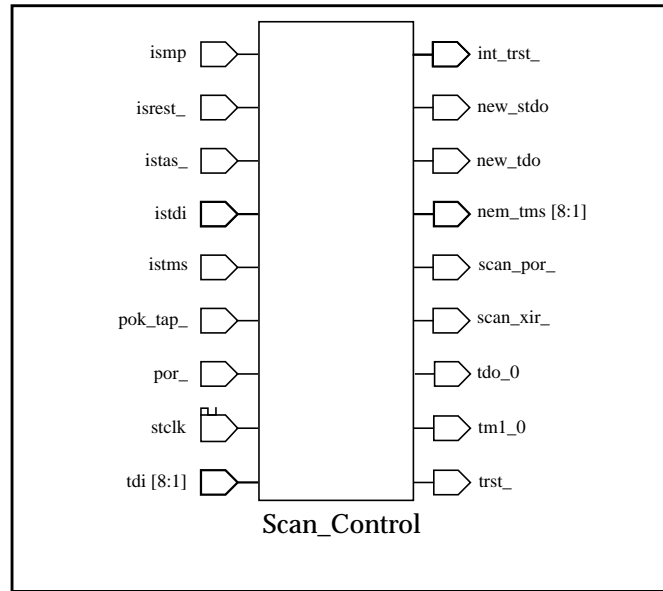


Figure 6-3 Scan\_Controller

## 6.3 Scan Controller Functional Description

Table 6-1 Scan Controller Signals

Signal Name	Signals	I/O	Description
ismp_	1	I	Service Processor Present signal
isrest_	1	I	Test Reset for Scan from Service Processor
istas_	1	I	Test Address Strobe from Service Processor
istdi	1	I	Test data in for Scan from Service Processor
istms	1	I	Test Mode Select for Scan from Service processor
pok_tap_	1	I	Power Ok tap (same signal as ipower_ok_)
por_	1	I	Power On Reset from reset block (same signal as osys_por_)
stclk	1	I	Test Clock for Scan from Service processor
tdi(8:1)	8	I	Test Data from ASICS on UUT
int_trst_	1	O	Internal trst for the internal tap (ring 15)
new_stdo	1	O	Test data out for Scan to Service Processor
new_tdo	1	O	Test data out to all ASICS on brd
new_tms(8:1)	8	O	Test Mode select for ASICS
scan_por_	1	O	scan power on reset
scan_xir_	1	O	scan xir reset
tdo_o	1	O	Test Data out (same as new_tdo)
tm1_0	1	O	Extra Test mode select 1 (same as new_tms(1))
trst_	1	O	Test mode Reset

The JTAG+ bus interface of the scan controller is used for communicating between the board under test and Service Processor. It supports two protocols. First, it supports the “Standard Test Access Port and Boundary-Scan Architecture” Specification from the IEEE 1149.1. This is known as JTAG. Second, it supports a special address protocol that was developed by Sundragon Project group. This special protocol will allow selection of up to 16 boards and up to 16 rings on that selected board. To support this protocol two signals (istas, ismp\_) were added to the JTAG interface. The Standard JTAG interface will be used to access the JTAG boundary and internal scan paths. All of this communication is done through the

TDI\_n, TDO, TMS\_n and TCK signals. For further information on JTAG, please refer to “Standard Test Access Port and Boundary-Scan Architecture” Specification from the IEEE 1149.1.

### 6.3.1 JTAG+ Interface Description

#### 6.3.1.1 Overview

When the `istas_` is asserted from the service processor, the `jtag+ fsm` state machine enables the 8 bit shift register. The board and the ring addresses are fed serially into this address register through `s_tdi` signal. The `s_tas` will stay active for 8 clocks of `s_tck`. In this implementation of the scan controller, the least significant four bits of the address register will be used to send `s_tms` signal to the right scan chain. All the other `tms` lines for the remaining scan rings will be kept high (see Note below). The scan bits are also used as input to a MUX to select the right `tdi` signal to be scanned back to the service processor.

Since ATE in manufacturing does not support multiple rings, the RIC chip on power up will default to scan chain 1. So all the ASICS which have to be tested on the ATE must be on a single scan chain. Also, the normal operation of the scan control logic in the RIC chip has a two clock delay associated with all the test signals which ATE can not account for. To work around this problem, the `tms1` and `tdo` signals are bypassed inside the RIC chip when the default ring is selected (for example: at power up). This is illustrated in the Figure 6-4, "JTAG+ Interface Detailed Block."

---

**Note:** Keeping the TMS lines high on a scan chain ring, will reset the unselected rings in five test clock cycles or less. This also prevents the RIC chip for being used for inter ring testing. To allow inter-ring testing, see Section 6.5, “Scan Block Bugs and Solutions,” for a possible fix .

---

### 6.3.1.2 JTAG+ Interface Block Diagram

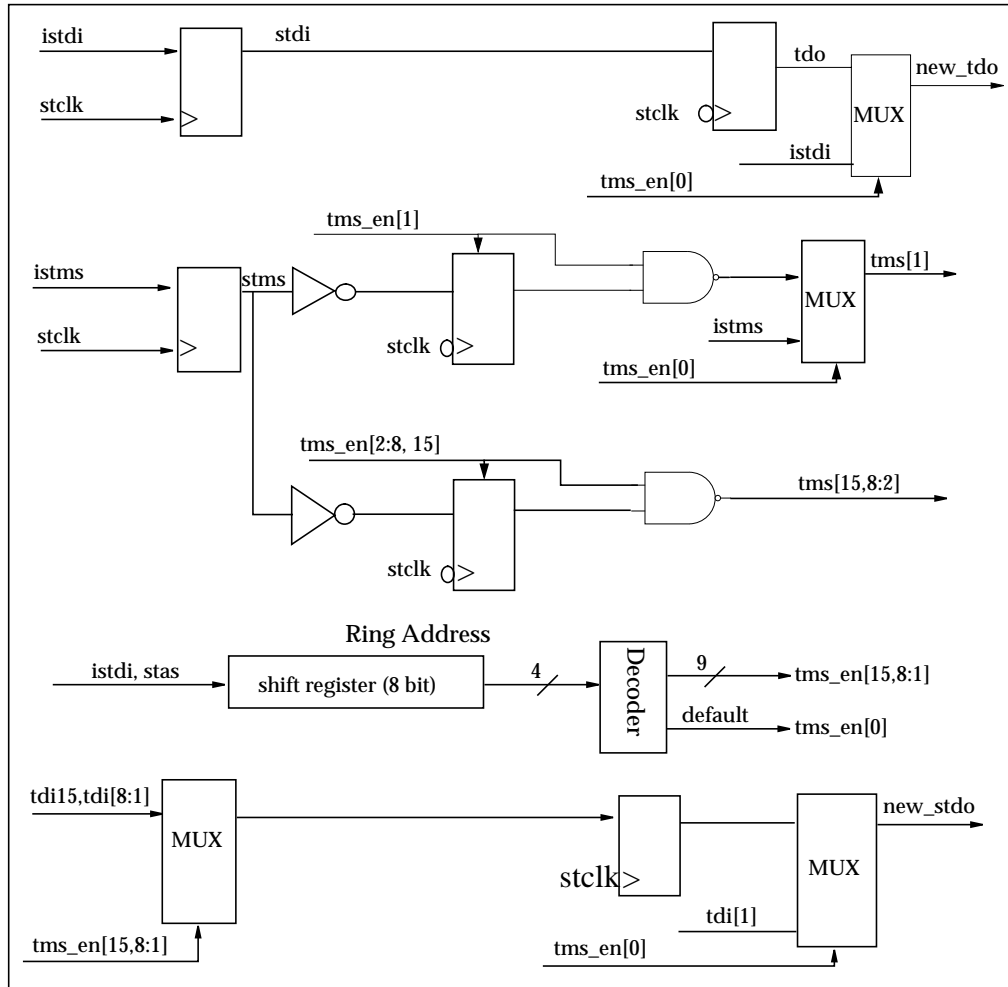


Figure 6-4 JTAG+ Interface Detailed Block

### 6.3.1.3 JTAG+ FSM State Descriptions

The JTAG+ state machine can be in two different states. On power up or on reset it will be in idle state. when s\_tas is asserted by the service processor, state machine in scan ring state. In this state, it will enable the address register to be scanned in with the board and ring addresses.

Table 6-2 JTAG FSM State

State	Description
idle	waiting for s_tas
shift	enable address register for shifting scan ring

### 6.3.1.4 JTAG+ FSM State Diagram

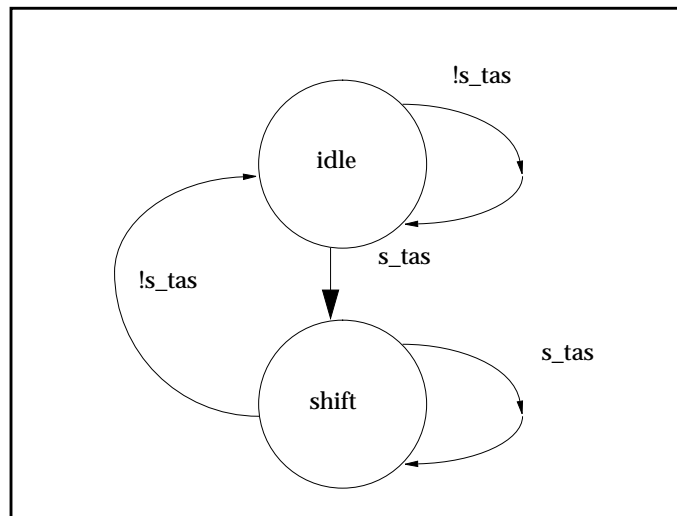


Figure 6-5 JTAG+ FSM State Machine

### 6.3.1.5 JTAG+ Timing Diagram

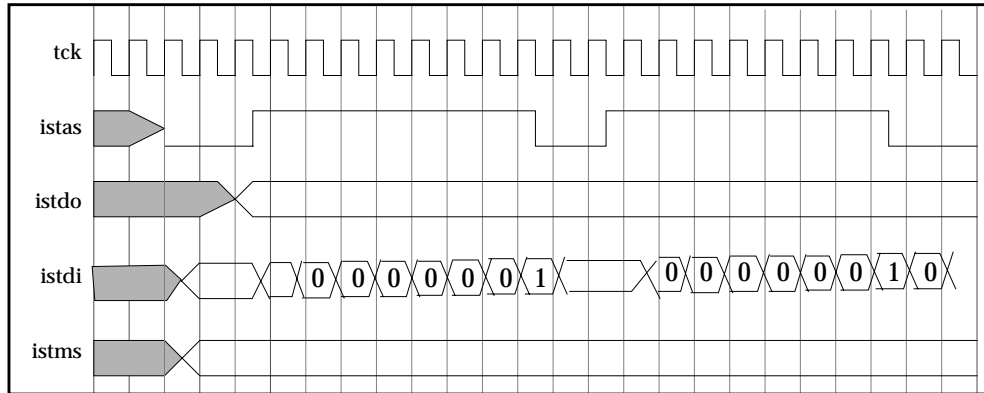


Figure 6-6 JTAG+ Interface Timing Diagram

### 6.3.1.6 JTAG+ Registers

#### Address Register

Table 6-3 JTAG+ Scan Address Register

Field	Bits	Description	R/W
Board ID	7:4	Board Address (not used by RIC)	RW
Ring Number	3:0	Selects one of the 8 scan rings, or ring 15 (inter RIC tap)	RW

The most significant four bits of the address register specify particular board to be selected (Board addressing is not supported in the RIC chip, use address 0 for the UltraSPARC system motherboards) and the least significant four bits specify the particular scan ring on the board.

### 6.3.2 JTAG Internal Scan Ring Description

#### 6.3.2.1 Overview

The internal Scan ring (ring 15) is used for generating POR and XIR resets. This JTAG ring will only support the data register (DR) portion of the JTAG state diagram. The instruction register (IR) portion is not used or not needed. Instead a dummy path was added in place of the IR path to meet the JTAG specifications. The JTAG Internal Scan Ring is 8 bit long register. Out of the bits, currently only two are defined. Two bits are used for generating power on and xir resets.

#### 6.3.2.2 JTAG Internal Scan Ring Block Diagram

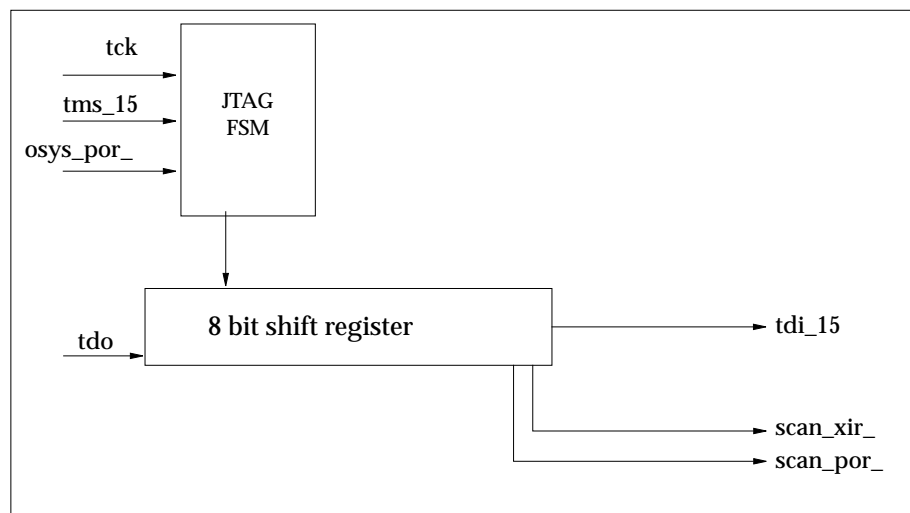


Figure 6-7 JTAG Internal Scan Ring Block

### **6.3.2.3 JTAG Scan Ring Register**

#### **Scan Ring Register**

*Table 6-4* JTAG Scan Ring Data/Instruction Register

Field	Bits	Description	R/W
N/A	07:03	Not Used	RW
Reserved	02	Reserved for future use	RW
POR	01	Scan Power On Reset	RW
XIR	00	Scan XIR reset	RW

The JTAG Scan Ring Register provides data to be loaded into the clock counter and control bits to generate resets.

### **6.3.2.4 JTAG FSM State Description**

*Table 6-5* Jtag state machine

State	Description
Test reset Logic	Test logic disabled
Run_Test/Idle	Controller State between scan operations
Select DR Scan	Test data selected by the current inst, retains their previous state
Capture DR	Data may be parallel loaded into test data registers
Shift DR	Data register shifts data one stage
Exit1 DR	Terminates scanning process
Pause DR	Allows shifting of test data register
Exit2 DR	Scanning process terminates
Update DR	Data register will be provided with latched parallel output



### 6.3.2.5 JTAG FSM State Diagram

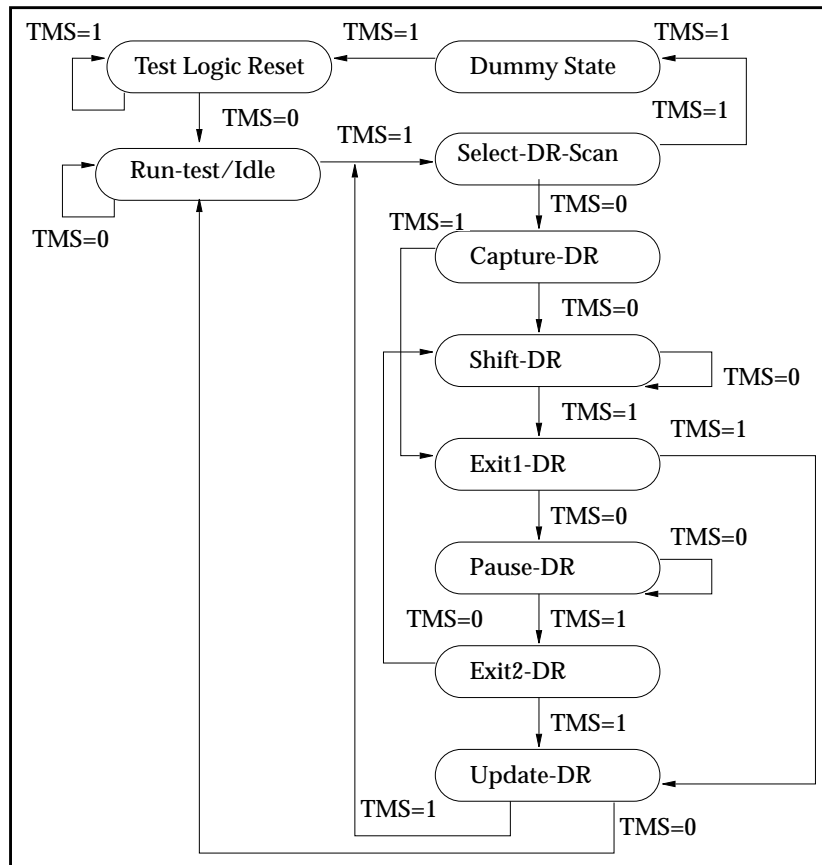


Figure 6-8 JTAG State Machine

## 6.4 Scan Controller Timing Diagrams

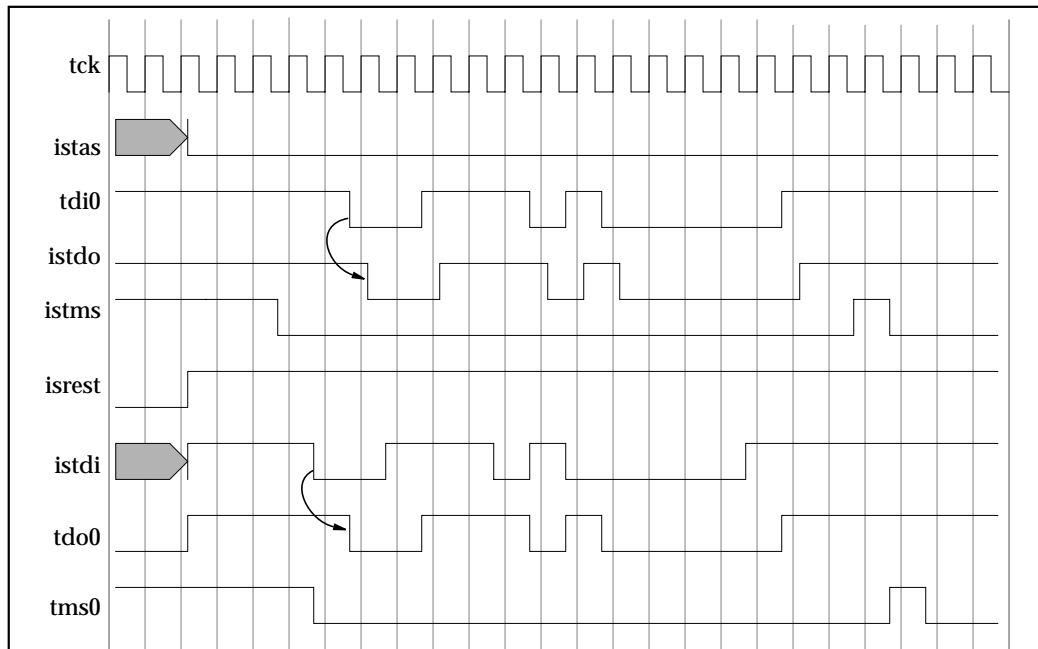


Figure 6-9 JTAG Scan Ring Timing Diagram

## 6.5 Scan Block Bugs and Solutions

The JTag+ implementation in the RIC has three known problems, these are discussed in detail below.

### 6.5.1 Unselected Rings Reset Automatically

All unselected TMS ring lines are driven high, this was done intentionally to make sure that all unused TMS lines were driven to a known state. Unfortunately driving TMS line high causes JTAG to reset after five clocks. Thus when attempting to do inter ring testing, it is not possible to setup one ring and then load the second ring as the first ring will have reset when you come back.

The solution to this problem is to do a gate level fix and tie the tms\_enable line on the nand gates that are used to drive the TMS signals to VCC, this will leave the TMS lines always enable and thus low when switching between rings. (See Figure Figure 6-4, "JTAG+ Interface Detailed Block.")

### ***6.5.2 Selecting the same ring, resets the ring***

While the TAS signal is active TMS[8:2] are driven to a high value, this causes the rings connected to TMS[8:2] to be reset, TMS[1] is a special case and is not reset when TAS is active.

The same solution as above will also fix this problem.

### ***6.5.3 Selecting non-existent rings gives inconsistent behavior***

Since the register in the JTAG controller has four bits for the ring address and only eight TMS lines, it is possible to select non-existent rings, in this case, the JTAG controller should consistently return all one's, however selecting non-existent rings causes the JTAG controller in the RIC to default to ring 0 and give inconsistent behavior.

The solution to this problem would be to better qualify the tms\_enable[0] signal, so that rather than default to tms\_enable[0], we would only drive tms\_enable[0] high after reset. This solution needs further investigation and verification.



## 7.1 Overview

This chapter describes how the clock controller of the RIC chip sets the system and CPU speeds

The clock controller in the RIC chip provides a magnitude comparator for multiple (up to four) CPU clock speeds and determines the slowest CPU speed. This information is then used to set the CPU speed for all CPU modules in the system and the system clock frequency. Unconnected speed select inputs are internally pulled high to allow the clock controller to function correctly even though only two or three processor modules are connected.

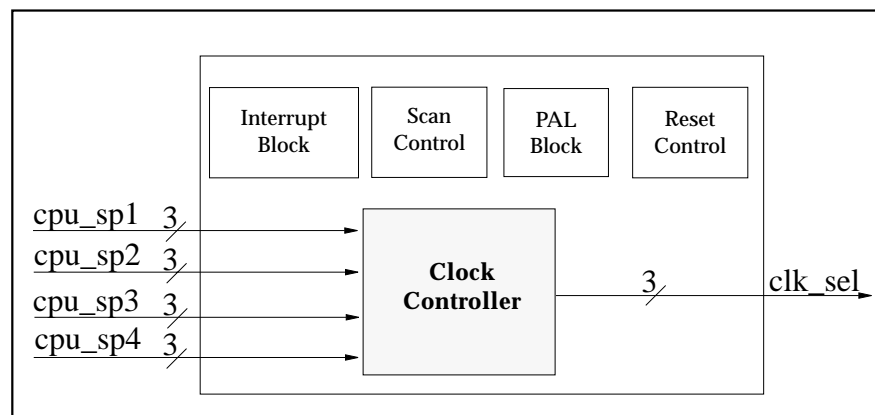


Figure 7-1 Clock Controller System Block

### 7.1.1 Clock Controller Gate Count Estimates

The current gate count for the Clock controller is fifty gate equivalents.

### 7.1.2 Signal Descriptions

The block diagram for the clock controller can be shown as follows:

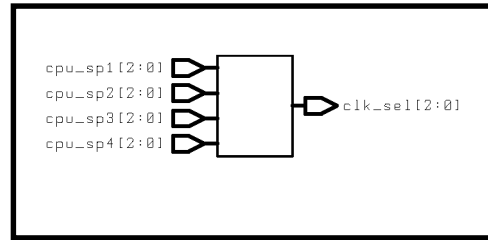


Figure 7-2 RIC\_Clogen

These signals connect from the clock controller to the IO cells.

Table 7-1 Clock Controller signal descriptions.

Signal Name	Signals	I/O	Description
icpu_sp1	3	I	Speed Inputs from CPU module 1
icpu_sp2	3	I	Speed Inputs from CPU module 2
icpu_sp3	3	I	Speed Inputs from CPU module 4
icpu_sp4	3	I	Speed Inputs from CPU module 4
clk_sel	3	O	Selection Code to the Clock chip

## 7.2 Clock Controller Function Descriptions

### 7.2.1 CPU Speed Selection

Each CPU module sends three signals to the clock generator chip to state what is the speed of the CPU. Based on that information, from all the CPUs, the CPU speed selection determines the system speed by selecting the slowest CPU speed. It then sends the three bit code associated with the slowest CPU speed to the output. The slowest CPU speed and the corresponding output is shown in the table below.

Table 7-2 System clock select

Slowest CPU Speed Code	Output Speed Select
000	000
001	001
010	010
011	011
100	100
101	101
110	110
111	111





### *8.1 Overview*

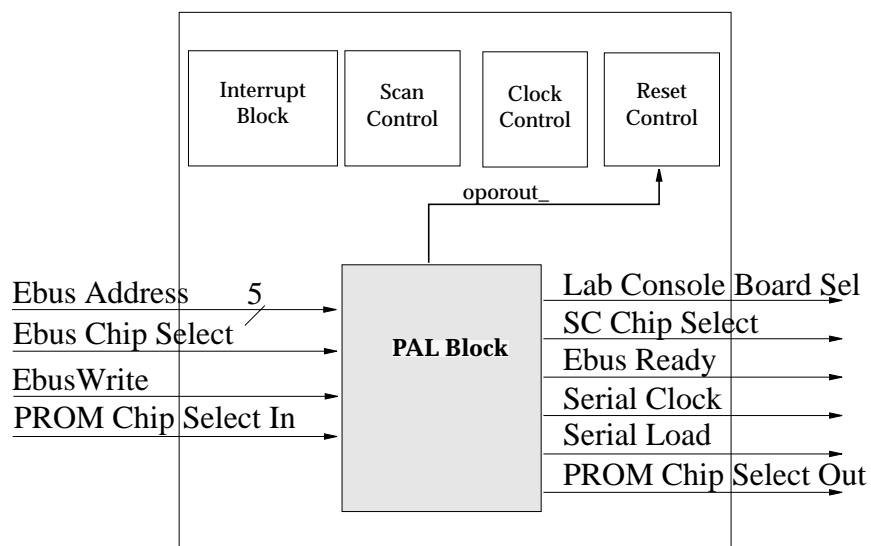
This chapter describes how the PAL block of the RIC chip works.

The PAL block was added late in the RIC development to provide misc decode logic services that did not fit anywhere else on the system.

The PAL block implements the following five functions:

1. Decode for the SC chip
2. Decode for lab console accesses
3. Decode and sequencing for clock generator parts
4. Decode for PROM writeable space
5. Xor Gate

Figure 8-1 PAL Block Diagram



### 8.1.1 PAL Block Gate Count Estimates

The current gate count for the PAL Block is 250 gate equivalents

### 8.1.2 Signal Descriptions

The Block diagram for the PAL Block can be shown as follows:

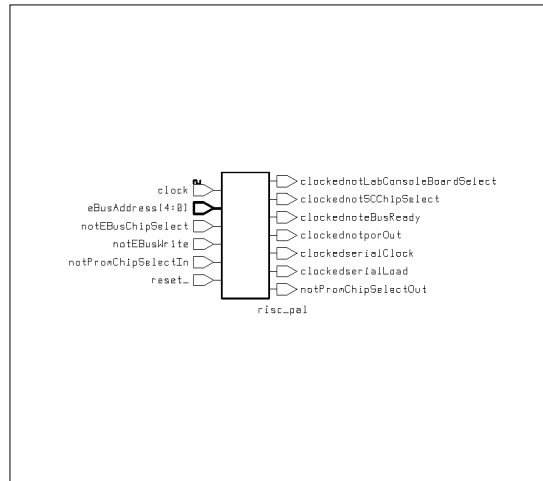


Figure 8-2 RIC\_PAL

These signals connect from the PAL Block to the IO cells.

Table 8-1 PAL Block signal descriptions.

Signal Name	Signals	I/O	Description
clock	1	I	Sbus Clock
eBusAddress	5	I	Ebus Address[19, 14, 13, 1, 0]
notEbusChipSelect	1	I	Ebus Chip Select
notEbusWrite	1	1	EbusWrite
norPromSelectIn	1	1	PROM Chip Select In
reset_	1	1	SBus reset
ClockednotLabConsoleBoardSelect	1	O	Lab Console Board Select
clockednotSCChipSelect	1	O	SC ChipSelect
clockednotEbusReady	1	O	Ebus Ready
clockednotporOut	1	O	Reset signal, oporout_ to reset block
clockedserialClock	1	O	Serial Clock out for Freq chip
clockedserialLoad	1	O	Serial Load out for Freq chip

*Table 8-1* PAL Block signal descriptions.

Signal Name	Signals	I/O	Description
notPromChipSlectOut	1	O	PROM Chip Select Out
ixorin1	1	I	Xor Input 1 (Not Shown above)
ixorin2	1	I	Xor Input 2 (Not Shown above)
oxorout	1	O	Xor Output (Not Shown above)

## 8.2 PAL Block Function Descriptions

### 8.2.1 PAL Logic

The PAL logic decodes the Ebus signals coming in, to provide access to the SC address and data register, Lab Console address and data registers (see Note), the frequency margining (Motorola MC12429 & MC12439) chips, and address range to write to the Flash PROM (Slavio only provides read access).

---

**Note:** It was decided not to implement the Lab Console in the UltraSPARC systems, although the support exists in the RIC PAL Block

---

The Address map for the PAL logic is shown in the table below

Table 8-2 PAL logic Address Map

Address Range	Description
0x0.0000	SC Address Register
0x0.0004	SC Data Register
0x0.2000	Lab Console Address Register (Not Used)
0x0.2001	Lab Console Data Register (Not Used)
0x0.4000	Freq Margining Serial Load and Reset
0x0.4001	Freq Margining serial load
0x0.4002	Freq Margining serial data in
0x8.0000 - 0xF.FFFF	Prom write address

### 8.2.2 XOR gate

Additionally the RIC chip make an xor gate available to select between the POST and OBP parts of the PROM.



## *Index*

---



### A

- address map 68
- Address Register 54
- Addresses 2
- AG Internal Scan Ring 55
- Assertion 2

### B

- Block 17
- bus interface signals 12
- Button PO 19
- Button XIR 19
- Button\_POR 24, 25
- Button\_XIR 26, 27
- byte stream 2
- Bytes 2

### C

- clock 17
- Clock Contro 16
- Clock Control 4, 17
- Clock Controller 11, 61, 63
- clock controller 61, 62
- Clock Controller Signals 11
- Clock Controller System 61
- clock generator 65
- clock select signals 17

- console accesses 65

- CPU Speed 63

### D

- data register 55
- De-assertio 2
- Decode 65
- Dispatcher 29, 39
- Dispatcher Block 40
- Dispatcher State 41, 42
- Dispatcher Timing 45

### E

- Ebus signals 68
- EN\_FSM1 37
- EN\_FSM1 State 36
- EN1\_FSM State 35
- Encoder 29, 33
- encoder 39
- Encoder 1 Timing 39
- Encoder Block 34
- Encoder Timing 39
- External interrupt Reset 2

### F

- frequency margining chip 5

### G

- Gate Count 21, 30, 48, 62, 66

gate level fix 58

H

halfword 2

I

Instruction Register 56

instruction register 55

Interface Signal 8, 11

Interface Signals 9

internal logics 13

Internal Scan Ring 17

Interrupt 4

Interrupt Concentrator 9, 16, 17, 29

Interrupt Signals 32

J

JTAG 50, 58

JTAG controller 59

JTAG FSM State 53, 56, 57

JTAG Internal Scan Ring 47, 48, 55

JTAG ring 55

JTAG Scan controller 19

JTAG Scan Ring Register 56

Jtag state machine 56

JTAG+ 50

JTAG+ Bus protocol 17

JTAG+ FSM 53

JTAG+ Interface 47, 48

JTAG+ Interface Block 52

JTAG+ Registers 54

JTAG+ Timing 54

L

Lab Console 68

Lab Console (LC) 5

Logical Block 6

M

MUX 51

P

PAL Block 5, 16, 18, 65, 66, 68

PAL Block Signals 12

PAL Logic 68

PAL Logic Address Map 68

PAL Signals 12

Partition 3

POR 2, 19, 23, 55

POR State 24

POR Timing 24

Power on Reset 2

PRO 5

PROM 65, 68

R

Reset Block 19, 21

reset block 21, 22

Reset Circuit 20

Reset Logic 19

reset logic 19

Resets 4, 8, 16, 17

RIC 1, 2, 8, 9, 10, 15, 19, 29, 51, 59

RIC Gate Counts 18

Ric Intcon 31

RIC\_Clogen 62

RIC\_PAL 66

RIC\_Reset 22

S

SC 2

SC chip 65

Scan 4

Scan Controller 17, 47

Scan Controller Signals 50

Scan Controller Timing 58

Scan Interface Signals 10

scan rings 47

Scan service processor 10

Scan\_Control 49

Signal Descriptions 31, 49

state machine 33

state machines 39

sub blocks 23



## X

## Index

system clock frequency 61

System clock select 63

System Controller 2

system controller 19

## T

TAS signal 59

TMS 51, 58

TMS lines 59

TMS ring 58

## U

U2S 2, 19, 29

Uniform Port Architectur 2

Unselected Rings 58

UPA 2

## W

word 2

## X

XIR 2, 55

XOR gate 5, 68

Xor Gate 65

