Design Nested Vectored Interrupt Controller For 32-bit RISC Processor

Mitul S. Nagar [#1], Hitesh H. Mathukiya [*2], Kalpesh R Ranipa [#3]

[#]*Electronics & Communication Department,C. U. SHah University,*
C. U. Shah College of Engineering and Technology, Near Kothariya village, Wadhwan 363030, India

[1]mitul.nagar.08ec@gmail.com

[2]hitesh.mathukiya@gmail.com

[3]k.ranipa@gmail.com

***Abstract*— Modelled RISC processor design is ARM cortex M3. ARM cortex M3 has twelve blocks among them Nested vectored interrupt controller-NVIC is modelled. ARM cortex M3 is designed for embedded applications have properties low interrupt latency, low gate count, 3- stage pipelining, branch prediction, THUMB and THUMB-2 instruction set.**

NVIC is the interrupt controller serves 16 system interrupt and 240 external interrupt. The NVIC block serves features are low interrupts latency and low power consumption. Low interrupt latency of ARM cortex M3 is due to the late arrival and tail chaining of the NVIC. Low power consumption feature serves by sub blocks of processor like Wakeup interrupt Controller-WIC and Power Management Unit-PMU. These two blocks directly integrated with NVIC. Low Power consumes by reducing or blocking clock to ARM cortex M3 core during sleep modes SLEEPDEEP and SLEEPING. Interrupt occur during sleep mode clock will resume in ARM cortex M3 core through WAKEUP process.

NVIC is interrupt serving purpose but it also handles signalling related to sleep mode with WIC. WIC handles the signalling and PMU handles the actual power management. Design of sub blocks developed using verilog.

Keywords: Sub block, modelling, NVIC, WIC, PMU, low interrupt latency, low power, verilog.

I. INTRODUCTION

The ARM CORTEX M3 is rationalized, simplifies and improved upon the ARM7TDMI processor design [1]. The device blends the best features from the 32-bit ARM architecture with Thumb-2 instruction set design whilst adding several new capabilities. Design developed from some of the novel features of the ARM series processors like non-mask-able interrupts for critical tasks, highly deterministic nested vector interrupts and low power consumption. Design also depicted features for power management.

II. NESTED VECTORED INTERRUPT CONTROLLER(NVIC)

NVIC main functions [1]

1. Facilitates low-latency exception
2. Interrupt handling
3. Controls power management
4. Implements System Control Registers.

The NVIC supports up to 240 dynamically re-prioritizable interrupts each with up to 256 levels of priority. The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked (nested) interrupts to enable tail-chaining of interrupts. NVIC can fully access in privileged mode, but interrupts can pend in user-mode if Configuration of Control Register is enabled. Any other user-mode access causes a bus fault. All NVIC registers are accessible using byte, half word, and word unless otherwise stated. All NVIC registers and system debug registers are little endian regardless of the endianness state of the processor.

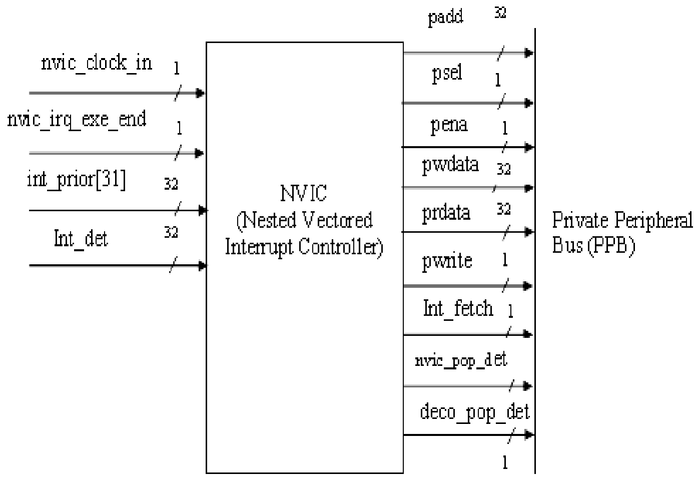

Fig. 1 NVIC interfacing

Fig 1 manifests the interface of NVIC with the private peripheral bus, core and instruction decoder unit. nvic_clock_in is the clock signal feeds to the NVIC from core. nvic_irq_exe_end if it is set to 1 will show that execution of interrupt is ended. int_prior[31] will give priority to 32 different interrupts. int_det[32] carry the status of current interrupt having capacity can handle interrupts up to 496. Padd will hold the address of the stack where status of register will be stored. Psel will select the PPB for access if it is set to 1 NVIC can access rather no access of PPB. Pwdata[32] will hold the data to be write in the memory same way Prdata[32] will hold the data fetched from memory. Pwrite if it is 1 it will show that writing operation is performed and if 0 fetching operation is performed. int_fetch if it is 1 it will inform that now IRQ fetch can start. nvic_pop_det if it is 1 it will stats popping from the stack

III. Wakeup Interrupt Controller

The ARM Cortex-M3 NVIC has logic dedicated to determining whether at any point in time a newly received interrupt is of higher priority than the current priority and must therefore be taken over the current execution context or priority. This priority scheme also operates during Wait For Event-WFE, Wait For Interrupt-WFI, and sleep-on-exit to determine when the core must resume execution of instructions after sleeping. Actually as part of core NVIC handles the signals from WIC and handles the sleep mode. For ultra-low power applications, it is desirable to be able to significantly reduce the dynamic and static power of the processor while in very-deep-sleep modes.

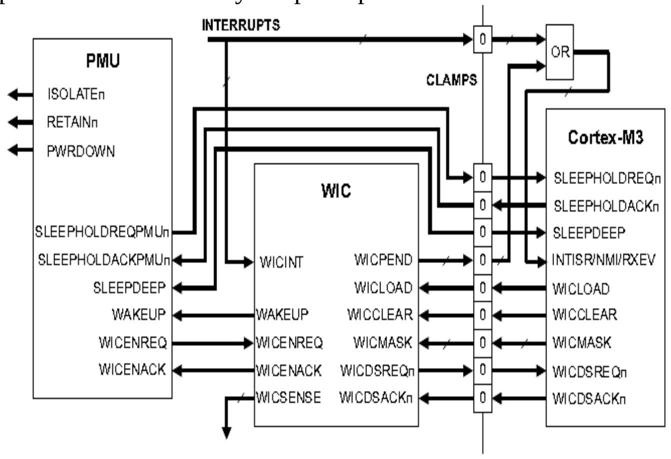

Fig 2 WIC Interfacing diagram

Fig 2[3][4] demonstrates the interfacing of WIC with power management unit (PMU) and core. This can be achieved by stopping clocks or removing power from the processor, or both. When powered off, the NVIC is unable to prioritize or detect interrupts. This means that knowing when to come out of very-deep-sleep becomes problematic. The Wake-up Interrupt Controller (WIC) provides significantly reduced gate count interrupt detection logic that can take over and emulate the full NVIC behavior when correctly primed by the full NVIC on entry to very-deep-sleep. The small size of the WIC ensures that its power requirements fit the budget available while in very-deep-sleep mode enabling it to always remain powered. Unlike the NVIC, the WIC has no prioritization logic. It implements a rudimentary interrupt masking system, signaling for wake-up as soon as a non-masked interrupt is detected. The WIC contains no programmer's model visible state and is therefore invisible to end users of the device other than through the benefits of reduced power consumption while sleeping [3][4].

IV. Design Development Process

Design development process is comprehension of five processes interrupt handling, pre-emption, interrupt return SLEEPING and WAKEUP. Interrupt handling will execute the current interrupt till higher priority interrupt will not occur. Pre-emption occur when execution pre-empt the current execution. How the processor restores the stacked ISR or tail-chains to a late-arriving interrupt with higher priority than the stacked ISR.

These three processes of interrupt will also integrated the tail chaining and late arrival effect of interrupt handling. Tail chaining is the process of two back to back interrupt serving by which interrupt serve without wasting the 6 clock cycle in pop up and push back the register status and serve them. Late arrival is the process by which higher priority interrupt from active interrupt will served first though it will detect later than active lower priority interrupt. Pr-empting is the process which will fetch the ISR from the vector table. Late arrival and tail chaining both are the effects which will help to improve the latency of interrupt. That is why the interrupt handling capabilities are more compare to other processor.

A. Interrupt handling

First step is reset and then loading of SP and PC from the location where registers reset value is stored, at location 0, 4.

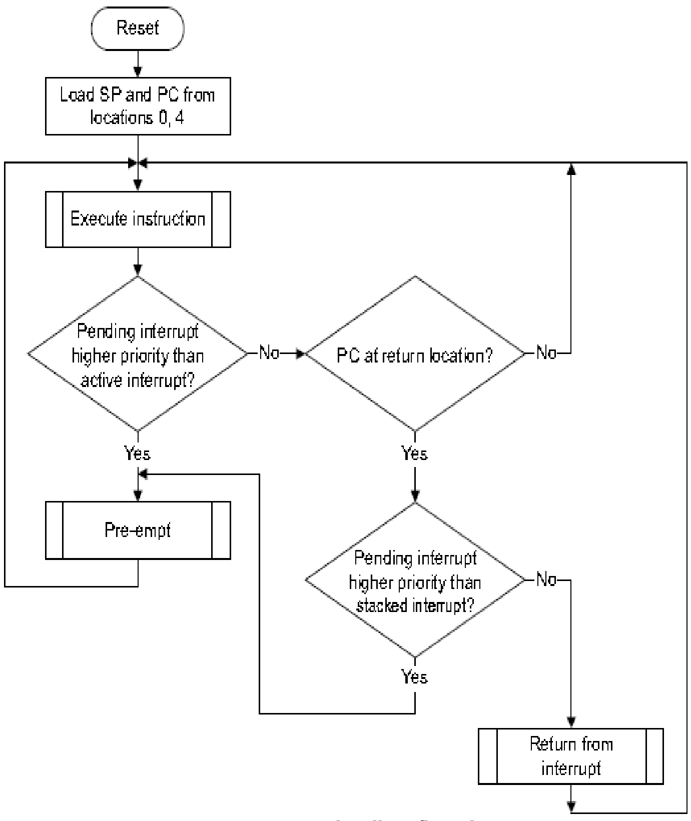
Reset
Load SP and PC from locations 0, 4
Execute instruction
Pending interrupt higher priority than active interrupt?
No
PC at return location?
No
Yes
Yes
Pre-empt
Pending interrupt higher priority than stacked interrupt?
No
Yes
Return from interrupt

Fig 3 interrupt handling flow chart

Fig 3[3] depicting the flow chart for interrupt handling. Then execution of instruction will be start if interrupt is still pended of higher priority than active interrupt then processor will pre-empt it. If not than active interrupt will continue to serve by loading the PC to new value or it should return to old PC. If still any higher priority interrupt will occur than the active interrupt if then it will pre-empt it rather it will serve and return from the interrupt.

B. Re-emption

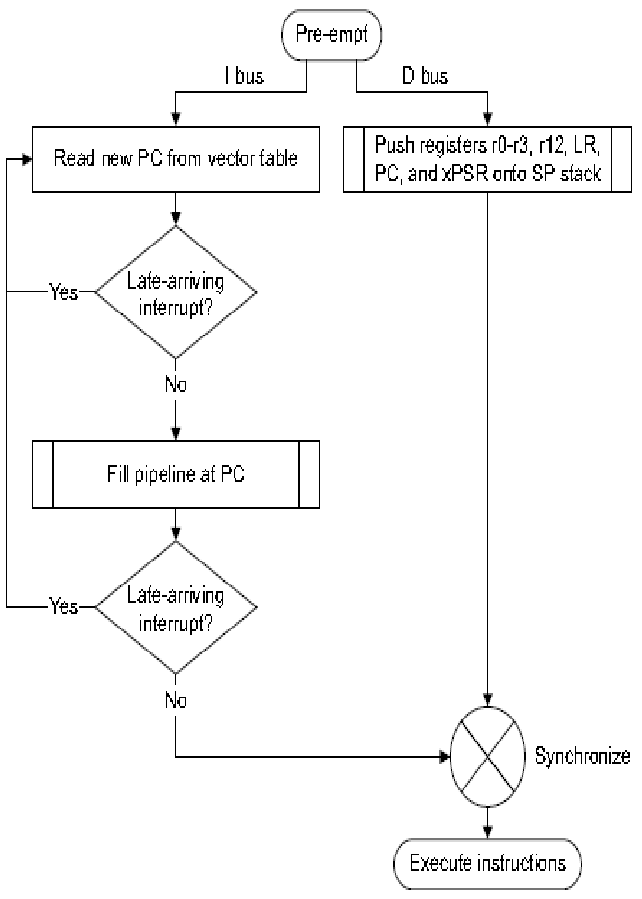

Fig 4 pre-emption flow chart

Fig 4[3] depicting the flow chart for pre-emption process. Pre-emption is the process by which interrupt will fetch from the vector table. Late arrival process by which interrupt will serve if higher priority interrupts will occur than fetch the PC for them from vector table. This process will store the status of the registers as well as fetch interrupt from vector table because of haward architecture.

C. Interrupt return

Interrupt return will be executed when nvic_pop_det signal will be high it will start poping of register status. It still check if interrupt will occur than it will serve by using tail chaining method. In tail chining it will use link register for serve new interrupt. It will again return and will pop register status and will end the execution of interrupt

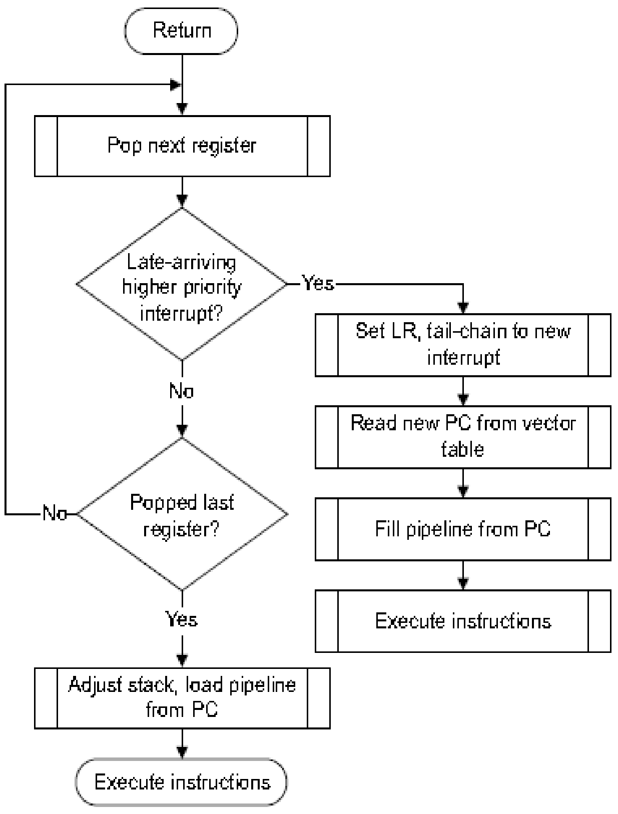
Return
Pop next register
Late-arriving higher priority interrupt?
Yes
Set LR, tail-chain to new interrupt
Read new PC from vector table
Fill pipeline from PC
Execute instructions
No
Popped last register?
No
Yes
Adjust stack, load pipeline from PC
Execute instructions

Fig 5 interrupts return flow chart

D. SLEEPDEEP

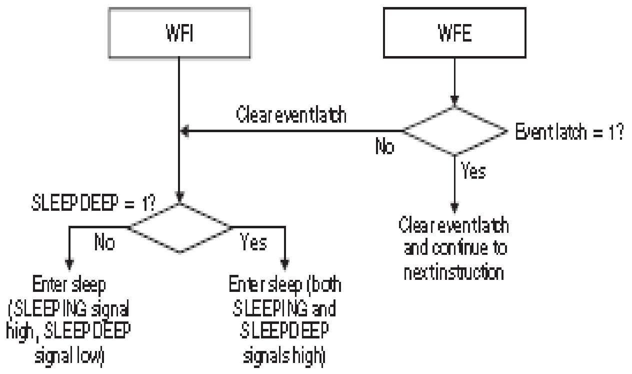
WFI
WFE
Clear event latch
No
Event latch = 1?
Yes
Clear event latch and continue to next instruction
SLEEPDEEP = 1?
No
Yes
Enter sleep (SLEEPING signal high, SLEEPDEEP signal low)
Enter sleep (both SLEEPING and SLEEPDEEP signals high)

Fig 6 SLEEPDEEP flow chart

You can use the WIC to gate FCLK and also to switch off power to the processor, if required. The Power Management Unit (PMU) first communicates to the WIC by asserting an enable called WICENREQ to the WIC. The WIC then sends a request to the processor to agree to WIC mode sleep. If the processor acknowledges then the WIC in turn acknowledges the PMU. When all acknowledges are set, then the next SLEEPDEEP mode is agreed to be a WIC mode sleep. The next time the deep sleep mode is entered either by WFI, WFE, or sleep-on-exit then the processor loads the WIC with a suitable mask using WICLOAD and WICMASK to enable the required interrupts and events, or both, to cause a wake-up. A logic-1 in the WICSENSE and WICMASK vector, or both, indicates that the WIC must wake up in response to the corresponding WICINT signal. WICPEND is a vector of pending interrupts captured by the WIC block. This provides a latched indication of the occurrence of any enabled and detected interrupts which occurred while the NVIC was sleeping. This enables pulse-interrupts to be used in combination with WIC based sleep methods. The PMU must assert SLEEPHOLDREQn to prevent the processor from waking up during a power-down sequence. When it has been acknowledged by SLEEPHOLDACKn then the PMU can proceed to power down the system.

E. *WAKEUP*

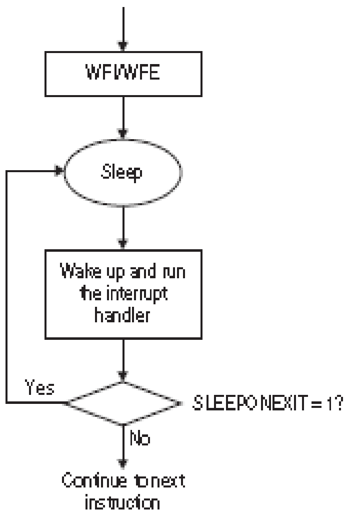
WFI/WFE
Sleep
Wake up and run the interrupt handler
Yes
SLEEPONEXIT = 1?
No
Continue to next instruction

Fig 7 WAKEUP flow chart

When the WIC detects a wake-up trigger from an interrupt or an event then it signals to the PMU to power up the processor using the WAKEUP pin. When the power is restored, the processor processes the event or interrupts or both given by the WICPEND signal that has stored the interrupts that have occurred including ones that do not cause a wake-up event because their priorities are not sufficient. When awake, the processor asserts WICLEAR to clear the mask contents stored in the WIC. It also shows the driving of ISOLATEn, RETAINn and PWRDOWN for use with state retention cells.

V. SIMULATION RESULT

The development process of design is done in the verilog and simulation process carried out in xlinx. Simulation of design features are carried out. The features like interrupt entry, interrupt exit, late arrival, tail chaining, control register, SLEEPDEEP, SLEEPHOLD req and WAKEUP are elaborated with result in further description.

A. *Interrupt entry*

Interrupt entry is also known as interrupt serving and waveform is shown in figure 8 when interrupt detected at int_det[32] it will generate the two signals interrupt_detect and int_fetch. Interrupt_detect shows that interrupt is occurred at int_det[32]. NVIC can serve up to 496 different interrupts. Next step is to save the status of the registers R0, R1, R2, R3, R12 and R13. Because these interrupts can affect during the interrupt execution. There is parallel process of interrupt service routine (ISR) is also fetched through the

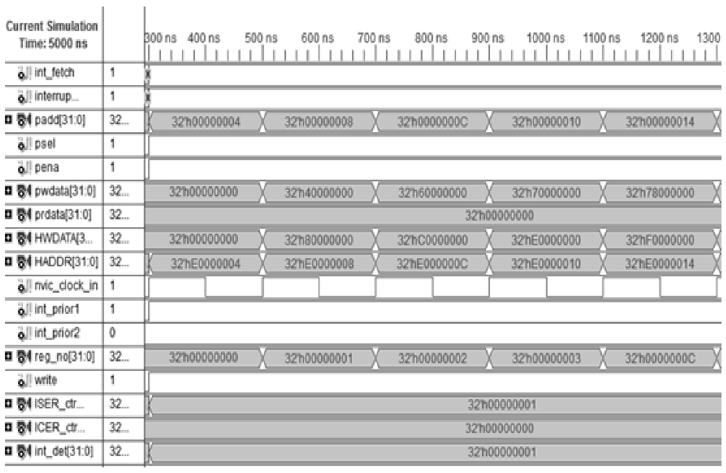

Fig 8 interrupt entry

private peripheral bus (PPB). Priority will be provided by system if it is system interrupt or if it is external interrupt it is mandatory to provide the priority with interrupt.

B. *Interrupt exit*

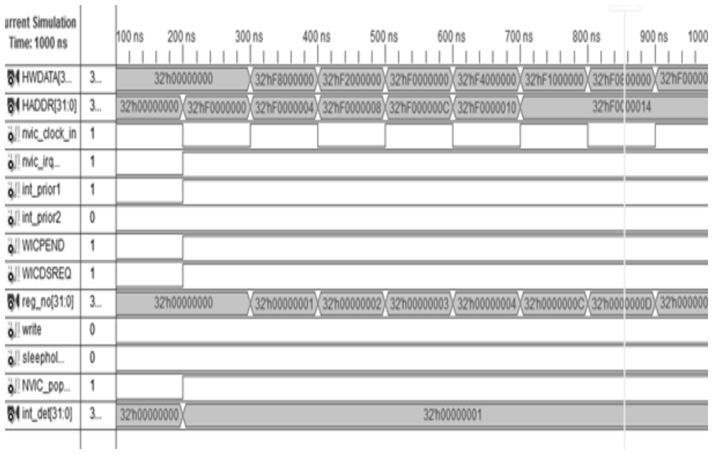

Fig 9 interrupt exit

Interrupt exit will perform when interrupt execution performed and then processor will generate the signals one is from the decoder unit of the core which is nvic_irq_exe_end if it is 1 it will and one from the pre-fetch unit it is nvic_pop_det if these signals goes high shows now interrupt exit process start now. Interrupt exit will pop registers content back to the general purpose registers.

C. *Late arrival*

Late arrival is the process of back to back interrupt serving but if highest priority interrupts will occur than it will serve first than lower priority interrupt. The process is same as the tail chaining which reduce the latency of the interrupt, Fig 10 shows the process of late arrival where two interrupt serve based on the priority which is provided by system.

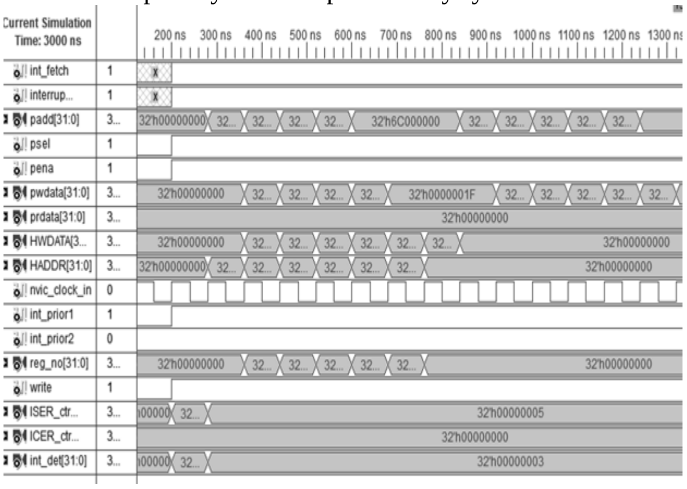

Fig 10 late arrival

D. Tail chaining

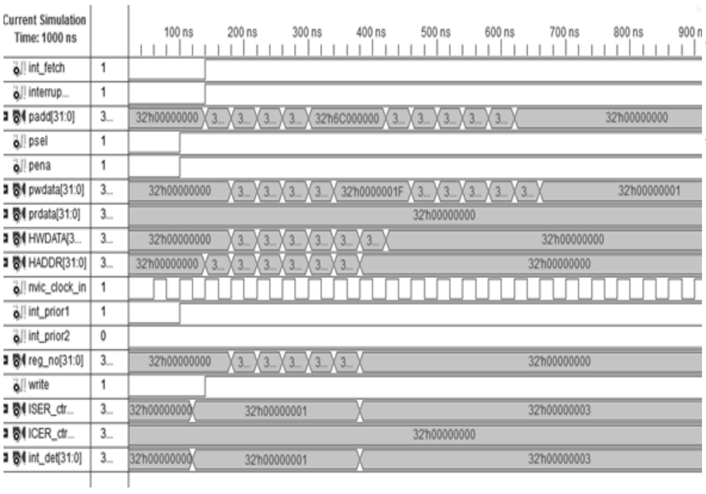

Fig 11 tail chaining

Tail chaining is the process of back to back serving of two interrupts without wasting the 12 clock cycles. It will waste just 6 clock cycle and will serve the two interrupts. Here in Fig 11 there is shown that second interrupt fetching will done after first serve. There is status register named ISER_ctrl_reg it allow interrupt this particular interrupt is going to serve or not. There is back to back interrupt occur is also shown in Fig 11.

E. Control registers

There is control registers supported by the NVIC which store the status of the interrupt arrived. Registers are ICTR_ctrl_reg, ISER_ctrl_reg, ICER_ctrl_reg, ISPR_ctrl_reg, ICPR_ctrl_reg, IPR_ctrl_reg and IABR_ctrl_reg which will control the interrupt execution process. ISPR_ctrl_reg which will store interrupt pending status of lower priority exceptions. ICPR_ctrl_reg will clear interrupt after interrupt served. IABR_ctrl_reg will show the active status of the interrupt to be served. This is shown in Fig 12.

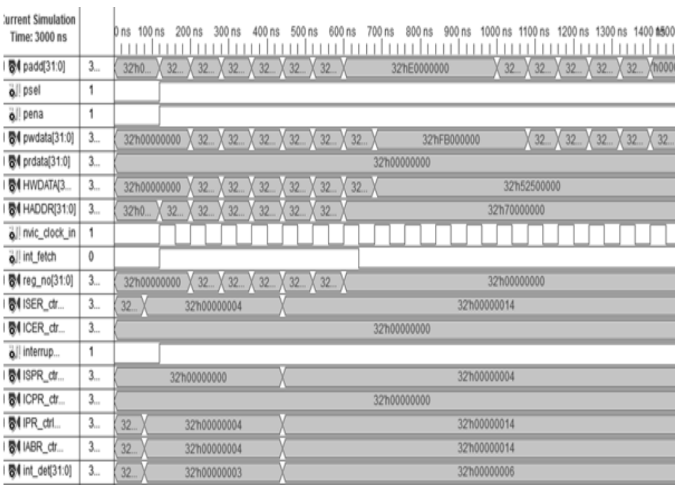

Fig 12 control registers

F. SLEEPDEEP

SLEEPDEEP is the effect when processor goes into the sleep mode. If no interrupt then processor is in ideal state. There is WICDSREQ will be generated from the power management unit (PMU) and is pass through the WIC to NVIC. NVIC reply the request by WICDSACK and will active and WICMASK generated.

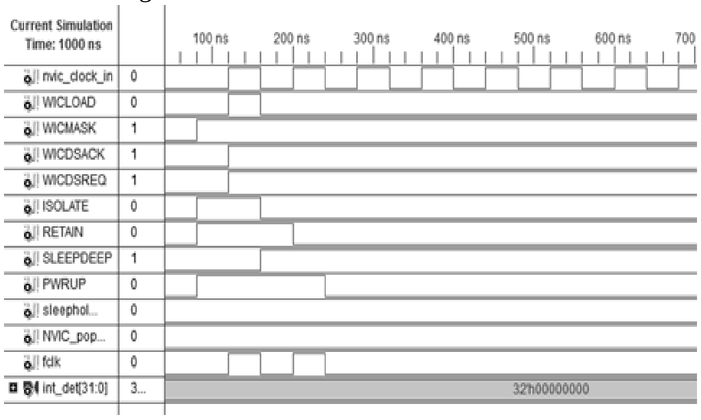

Fig 13 SLEEPDEEP

In sequence SLEEPDEEP will become high. ISOLATE become low so PMU isolate power domain. RETAIN will goes low PMU drive core state retention. PWRUP goes low which shows that PMU power down the core. Fig 13 will show the SLEEPDEEP mode of WIC.

G. SLEEPHOLD req

SLEEPHOLDREQ is the signal which is used to increase the sleep mode. WAKEUP signal will not generated till SLEEPHOLDREQ is 1 though interrupt arrive. After SLEEPHOLDREQ goes low it will do the process of wakeup as same as described in previous chapter. WAKEUP will done and will resume the FCLK. Waveform is shown in Fig

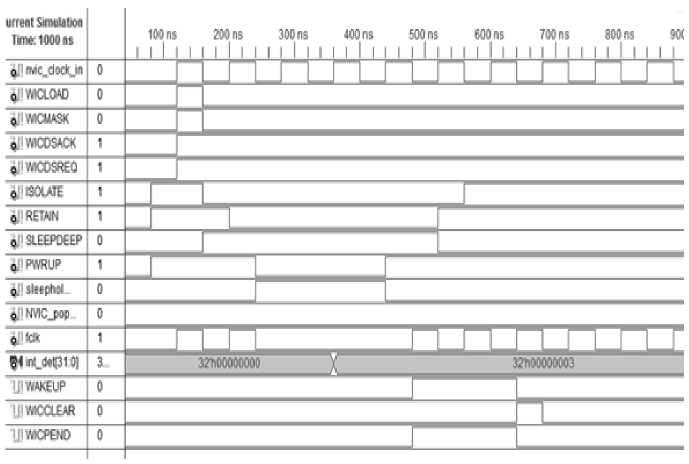

Fig 14 SLEEPHOLD req

H. WAKEUP

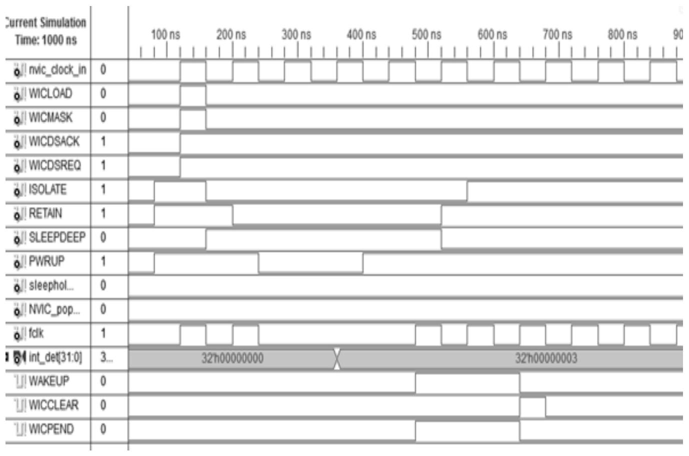

Fig 15 WAKEUP

Interrupt occurred at the int_det it will initiate core getting out of sleep mode. WAKEUP generated which shows that wakeup from sleep mode. WICPEND sows that interrupt is pending to serve. PWRUP will become high shows that PMU power up the core. RETAIN will become high shows that PMU drives state restoration. ISOLATE will goes high PMU take core out of isolation. After that FCLK will resumed. WICCLEAR will become high will clear the SLEEP mode. Waveforms are shown in Fig 15.

VI. CONCLUSION

The design of NVIC supports 256 interrupts and features like late arrival, tail chaining, SLEEPDEEP and WAKEUP. Designed block NVIC is perfectly communicates with WIC and PMU for low power management. Design supports all features specified by design architect.

REFERENCES

[1] Josheph yui, “the definitive guide to ARM cortex-M3” Newnes publishers ISBN: 978-0-7506-8534-4

[2] Samir palnitkar “verilog HDL aguide in digital design and synthesis” sun soft press.

[3] Cortex™-M3 r2p0 Technical Reference Manual , 2005-2008 ARM Limited. ARM DDI 0337G

[4] “Cortex™-M3 Revision: r1p1 Technical Reference Manual” 2005, 2006 ARM Limited.ARM DDI 0337E

[5] “ARM®v7-M Architecture Reference Manual”, 2006-2008, 2010 ARM Limited. ARM DDI 0403D (ID021310)

[6] “ARM Cortex-M3 Processor Software Development for ARM7TDMI Processor Programmers“, Joseph Yiu and Andrew Frame July 2009, White Paper – ARM

[7] “ARM Core Cortex-M3 / Cortex-M3 with ETM (AT420/AT425)”, Errata Notice, Feb-2011