The Interrupt Controller (INTC)

These notes are based on the Xilinx OPB Interrupt Controller (v1.00c) Specification
Reading

[opb_intc.pdf] on Blackboard and on your PC
Why use an Interrupt Controller?

- A complete system requires multiple devices:
  - A timer to generate the system real-time clock
  - Serial communication
  - Maybe more devices like audio/video etc.

- The MicroBlaze provides only one *Interrupt* port

- The Interrupt Controller (IntC) allows multiple peripheral devices to connect to this single port
IntC Overview

- Consists of an Interrupt Controller (IntC) core and a OPB bus interface

- Can connect directly to the OPB for easy integration into the system

- Supports data bus widths of 8-, 16- or 32-bits for the OPB interface

- Number of interrupt pins are configurable
IntC Structure

- **Interrupt Detection**
- **Interrupt Request Generation**
- **Bus Interface**
- **Software Interface**
IntC Programmable Registers

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Abbreviation</th>
<th>OPB Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt Status Register</td>
<td>ISR</td>
<td>0 (00h)</td>
</tr>
<tr>
<td>Interrupt Pending Register</td>
<td>IPR</td>
<td>4 (04h)</td>
</tr>
<tr>
<td>Interrupt Enable Register</td>
<td>IER</td>
<td>8 (08h)</td>
</tr>
<tr>
<td>Interrupt Acknowledge Register</td>
<td>IAR</td>
<td>12 (0Ch)</td>
</tr>
<tr>
<td>Set Interrupt Enable Bits</td>
<td>SIE</td>
<td>16 (10h)</td>
</tr>
<tr>
<td>Clear Interrupt Enable Bits</td>
<td>CIE</td>
<td>20 (14h)</td>
</tr>
<tr>
<td>Interrupt Vector Register</td>
<td>IVR</td>
<td>24 (18h)</td>
</tr>
<tr>
<td>Master Enable Register</td>
<td>MER</td>
<td>28 (1Ch)</td>
</tr>
</tbody>
</table>
### IntC Register Organization

<table>
<thead>
<tr>
<th></th>
<th>32-bit Implementation</th>
<th>LSB</th>
<th>OPB Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ISR</td>
<td></td>
<td></td>
<td>BAR + 0</td>
</tr>
<tr>
<td>IPR</td>
<td></td>
<td></td>
<td>BAR + 4</td>
</tr>
<tr>
<td>IER</td>
<td></td>
<td></td>
<td>BAR + 8</td>
</tr>
<tr>
<td>IAR</td>
<td></td>
<td></td>
<td>BAR + 12</td>
</tr>
<tr>
<td>SIE</td>
<td></td>
<td></td>
<td>BAR + 16</td>
</tr>
<tr>
<td>CIE</td>
<td></td>
<td></td>
<td>BAR + 20</td>
</tr>
<tr>
<td>IVR</td>
<td></td>
<td></td>
<td>BAR + 24</td>
</tr>
<tr>
<td>MER</td>
<td></td>
<td></td>
<td>BAR + 28</td>
</tr>
</tbody>
</table>
INTC Registers: ISR (offset = 0x0)

Interrupt Status Register

- Checked first to see who is asking for service in an Interrupt Service Routine (ISR)
Mapping Interrupts to the IntC

- Each interrupt signal from a peripheral maps to a particular bit in each IntC register

- The LSB is assigned the highest priority

- Interrupts can be enabled/disabled by writing 1/0 to the corresponding bit in the IER
INTC Registers: IER (offset = 0x8)

![Diagram of Interrupt Enable Register](image)

- A ‘1’ written to bit position \( n \) will enable Interrupt \( \#n \).
INTC Registers: IAR (offset = 0xc)

- Write a ‘1’ to the corresponding position to indicate that you have handled the corresponding interrupt condition.
- Usually one of the last steps done in an ISR
Write a “0x3” to this register as part of the INTC initialization routine.
IntC Operation (1 of 2)

- Peripheral asserts its interrupt request IRQ

- When enabled, the INTC will forward the request to the MicroBlaze via the IRQ pin

- If interrupts are enabled, MicroBlaze PC will jump to 0x10 where the programmer should include a jump to the corresponding service routine
IntC Operation (2 of 2)

- The ISR on the MicroBlaze acknowledges the interrupt to the corresponding device by writing a 1 to the IAR bit
  - This clears the interrupt
  - Note that writing to the IAR is the only way of clearing the interrupt
- If the corresponding interrupt signal remains asserted after acknowledgment, a new IRQ will be generated
Using Interrupts

- Interrupt Initialization Sequence
- Interrupt Service Sequence
Interrupt Initialization Sequence

- Memory location 0x10 should contain a branch to your ISR
- Write the appropriate mask to the Interrupt Enable Register (IER)
- Write a “0x3” to the Master Enable Register (MER)
- Enable MicroBlaze to respond to interrupts by writing a ‘1’ to the Interrupt Enable bit in the Machine Status Register (MSR)
Interrupt Service Sequence

When an interrupt occurs

- Read the INTC Interrupt Status Register to determine the source of the interrupt
- Acknowledge the interrupt to the INTC by writing a ‘1’ to the appropriate interrupt source in the Interrupt Acknowledge Register (IAR)
- Reset the interrupt state in the interrupting device
  - Usually entails writing a ‘1’ to the appropriate bit in Interrupt Status Register (ISR) of the interrupting device
    - Timer: TCSR
    - GPIO: ISR
More Information

Xilinx OPB Interrupt Controller (v1.00c) Datasheet available on Blackboard