SOC EVENT GENERATOR

This APB peripheral device collects all the events which are presented to the CPU as IRQ11 (Machine interrupt). Each event is individually maskable by the appropriate bit in the REG\_MASKx register. When an enabled event (unmasked) is received it is placed in an event FIFO and the IRQ11 signal is presented to the CPU which can then read the REG\_FIFO to determine which event caused the interrupt. Each event has a queue of depth four to collect events if the queue for any event overflows an error is logged into the appropriate REG\_ERR register and IRQ31 is presented to the CPU.

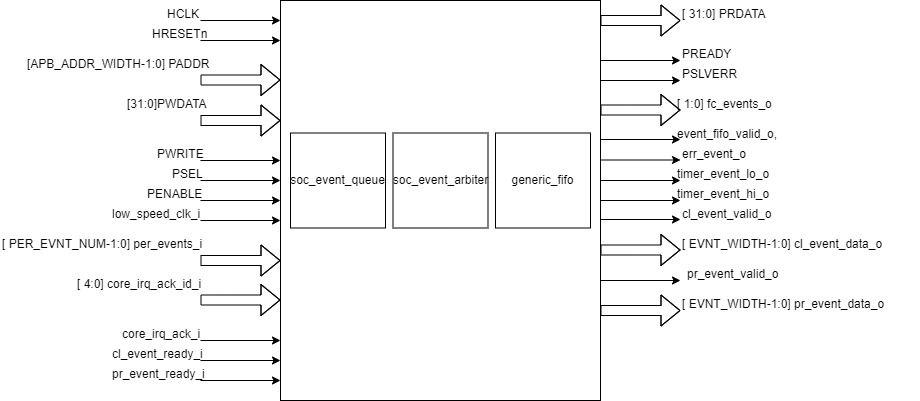
FEATURES:

* Events are generated from the cluster and peripherals and go to fabric control.
* Events are maskable.
* Error register to log error.

THEORY OF OPERATION:

It contains soc\_event\_queue and soc\_event\_arbitrator. Events are stored in an event fifo. The CPU can read the event fifo to determine which event caused the interrupt.

BLOCK DIAGRAM:



* fc\_events\_ o is a two bit output port which drives the third and fourth bit of peripheral event input (per\_events\_i).
* Driving valid output signals:
  + Fabric control event is said to be valid if there is at least one event which is granted by soc\_event\_arbiter and is not masked. Fabric control event valid data is sent to generic\_fifo which indicates there is valid data available for writing to fifo.
  + Cluster event is said to be valid if there is at least one event which is granted by soc\_event\_arbiter and is not masked. Cluster event valid data is driven through cl\_event\_valid\_o port.
  + Peripheral event is said to be valid if there is at least one event which is granted by soc\_event\_arbiter and is not masked. Peripheral event valid data is driven through pr\_event\_valid\_o port.
* Readiness of FC, cluster and peripheral:
  + Fabric control is said to be ready if the fabric control event is valid and generic fifo is ready to accept data.
  + Cluster is said to be ready if the cluster event is valid and cl\_event\_ready\_i is high.
  + Peripheral is said to be ready if the peripheral event is valid and pr\_event\_ready\_i is high.
* Register to store events: All the events from peripheral and cluster are stored in an internal register. Peripheral events will occupy the lower bits and cluster events will occupy the higher bits of this register.
  + For driving the timer\_event\_lo\_o event at REG\_TIMER1\_SEL\_LO index of this register is selected.
  + For driving the timer\_event\_hi\_o the event at REG\_TIMER1\_SEL\_HI index of this register is selected
  + All the events from this register are provided as input to the soc\_event\_queue.
* soc\_event\_queue:
  + soc\_event\_queue is instantiated for all the events.
  + Each event from the cluster and peripherals are provided as input to soc\_event\_queue.
  + An event acknowledgement input is provided which specifies acknowledgment of that event by soc\_event\_arbiter. It is high if the event is ready and is acknowledged by the arbiter.
  + Total number of events in the queue is updated at every positive edge of the clock if there is a new event or an event acknowledgement.
    - Event count is increased if there is an input event and an event is not acknowledged.
    - Event count is decreased if an event is acknowledged and there is no new input event.
    - Event count is reset at negedge of reset.
  + Error will be generated for an event if the event queue is full. Depth of the queue is four. err\_event\_o is asserted to indicate error in the input events.
  + A successful event is generated from the soc\_event\_queue if the queue is not empty.
* soc\_event\_arbiter:
  + soc\_event\_arbiter sets priority for the event. Out of the input events it selects the events to be granted.
  + It takes output events from soc\_event\_queue as the input.
  + It takes an acknowledgement input which is set to one if fabric control, cluster and peripheral are ready.
  + It uses the parallel prefix arbitration method to select events of higher priority. The granted events are sent as output.
* Cluster and peripheral event data output:
  + The index of the granted events are sent as output from the soc\_event\_generator through cl\_event\_data\_o and pr\_event\_data\_o.
* Reset: The model supports active low reset.
* Writing to registers:
  + Data can be written to registers if PSEL, PENABLE and PWRITE are enabled.
  + If PSEL and PENABLE is enabled and PWRITE is disabled then error registers(REG\_ERR\_X) are cleared.
* Generic\_fifo:
  + Push operation:
    - A valid high input is sent if there is at least one granted unmasked event. It indicates there is a valid event for writing into fifo.
    - A grant output is sent by the fifo which indicates fifo can accept new data.
      * Drive zero if fifo is full else drive one.
    - The index of the granted events from the soc\_event\_arbiter is sent as an input.
  + Pop operation:
    - A valid output is sent by the fifo which indicates there is valid data available in fifo for reading. It drives value to event\_fifo\_valid\_o port.
      * Drive zero if fifo is empty else drive one.
    - A grant input is sent which indicates if data can be read from fifo. It is set to 1 if acknowledgement from the core(core\_irq\_ack\_i) is positive and core acknowledgement id(core\_irq\_ack\_id\_i) is 3.
    - A data output is sent by the fifo. REG\_FIFO register can be read to get this data if the fifo is not empty.

APB EVENT CONTROL CSR’s

| Offset | Field | Bits | Type | Default | Description |
| --- | --- | --- | --- | --- | --- |
| 0x00 | REG\_EVENT | 15:0 | W | 0x00 | 16 bits of software generated event. |
| 0x04 | REG\_FC\_MASK\_0 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 0 - 31 of fc\_subsystem 1=mask event |
| 0x08 | REG\_FC\_MASK\_1 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 32 - 63 of fc\_subsystem 1=mask event |
| 0x0C | REG\_FC\_MASK\_2 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 64-95 of fc\_subsystem 1=mask event |
| 0x10 | REG\_FC\_MASK\_3 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 96-127 of fc\_subsystem 1=mask event |
| 0x14 | REG\_FC\_MASK\_4 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 128-159 of fc\_subsystem 1=mask event |
| 0x18 | REG\_FC\_MASK\_5 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 160 - 191 of fc\_subsystem 1=mask event |
| 0x1C | REG\_FC\_MASK\_6 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 192-223 of fc\_subsystem 1=mask event |
| 0x20 | REG\_FC\_MASK\_7 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 224-255 of fc\_subsystem 1=mask event |
| 0x24 | REG\_CL\_MASK\_0 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 0 - 31 of clock 1=mask event |
| 0x28 | REG\_CL\_MASK\_1 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 31 - 63 of clock 1=mask event |
| 0x2C | REG\_CL\_MASK\_2 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 64-95 of clock 1=mask event |
| 0x30 | REG\_CL\_MASK\_3 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 96-127 of clock 1=mask event |
| 0x34 | REG\_CL\_MASK\_4 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 128-159 of clock 1=mask event |
| 0x38 | REG\_CL\_MASK\_5 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 160-191 of clock 1=mask event |
| 0x3C | REG\_CL\_MASK\_6 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 192-223 of clock 1=mask event |
| 0x40 | REG\_CL\_MASK\_7 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 224-255 of clock 1=mask event |
| 0x44 | REG\_PR\_MASK\_0 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 0-31 of peripheral 1=mask event |
| 0x48 | REG\_PR\_MASK\_1 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 32-63 of peripheral 1=mask event |
| 0X4C | REG\_PR\_MASK\_2 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 64-95 of peripheral 1=mask event |
| 0X50 | REG\_PR\_MASK\_3 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 96-127 of peripheral 1=mask event |
| 0X54 | REG\_PR\_MASK\_4 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 128-159 of peripheral 1=mask event |
| 0X58 | REG\_PR\_MASK\_5 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 160-191 of peripheral 1=mask event |
| 0X5C | REG\_PR\_MASK\_6 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 192-223 of peripheral 1=mask event |
| 0X60 | REG\_PR\_MASK\_7 | 31:00 | R/W | 0xFFFFFFFF | individual masks for events 224-255 of peripheral 1=mask event |
| 0X64 | REG\_ERR\_0 | 31:00 | R/W | 0x0 | individual error bits to indicate event queue overflow for events 0 - 31 |
| 0X68 | REG\_ERR\_1 | 31:00 | R/W | 0x0 | individual error bits to indicate event queue overflow for events 0 - 31 |
| 0X6C | REG\_ERR\_2 | 31:00 | R/W | 0x0 | individual error bits to indicate event queue overflow for events 0 - 31 |
| 0X70 | REG\_ERR\_3 | 31:00 | R/W | 0x0 | individual error bits to indicate event queue overflow for events 0 - 31 |
| 0X74 | REG\_ERR\_4 | 31:00 | R/W | 0x0 | individual error bits to indicate event queue overflow for events 0 - 31 |
| 0X78 | REG\_ERR\_5 | 31:00 | R/W | 0x0 | individual error bits to indicate event queue overflow for events 0 - 31 |
| 0X7C | REG\_ERR\_6 | 31:00 | R/W | 0x0 | individual error bits to indicate event queue overflow for events 0 - 31 |
| 0X80 | REG\_ERR\_7 | 31:00 | R/W | 0x0 | individual error bits to indicate event queue overflow for events 0 - 31 |
| 0X84 | REG\_TIMER1\_SEL\_HI | 7:0 | R/W | 0x0 | Specifies which event should be routed to the lo timer. |
| 0X88 | REG\_TIMER1\_SEL\_LO | 7:0 | R/W | 0x0 | Specifies which event should be routed to the hi timer. |
| 0X90 | REG\_FIFO | 7:0 | R | 0x0 | ID of triggering event to be read by interrupt handler. |