Low power specification

|  |  |  |  |
| --- | --- | --- | --- |
| Revision # | Author | Date | Notes |
| 0.1 | Poonacha Kongetira | 11 Feb 2014 | First version, Clock Gating |
| 0.2 | Poonacha Kongetira | 10 Mar 2014 | Added Low Power Spec for NocStudio, Master / Slave Power Sequencing |
| 0.5 | Poonacha Kongetira | 14 May 2014 | Added power registers, modified power sequences. |
| 0.7 | Poonacha Kongetira | 17 July 2014 | Added Composite bridge power gating |
| 0.75 | Rimu Kaushal | 25 Sep 2014 | Added description for level one FSM |

Table of Contents

[2 Overview 5](#_Toc399452549)

[3 Clock gating 5](#_Toc399452550)

[3.1 Fine grained clock gating 5](#_Toc399452551)

[3.2 System Level Clock Gating 5](#_Toc399452552)

[3.3 Coarse Grained clock Gating 6](#_Toc399452553)

[3.3.1 Override feature 6](#_Toc399452554)

[3.3.2 Logic for clk gating 7](#_Toc399452555)

[3.3.3 Latency Penalty on Coarse Clock enable 9](#_Toc399452556)

[3.4 NocStudio Requirements 10](#_Toc399452557)

[4 Power Gating 11](#_Toc399452558)

[4.1 Specification of Power, Voltage Domains & Profiles 11](#_Toc399452559)

[4.1.1 Specifying Noc Power/Voltage 13](#_Toc399452560)

[4.2 NocStudio requirements for Phase-1 13](#_Toc399452561)

[4.3 NocStudio Requirements for Phase-II 16](#_Toc399452562)

[4.4 Hardware Requirements (high level) 16](#_Toc399452563)

[4.5 Clock and Power Gating Sequences 18](#_Toc399452564)

[4.5.1 Host with single AXI4 master bridge. 18](#_Toc399452565)

[4.5.2 Host with single AXI4 Slave Bridge 21](#_Toc399452566)

[4.5.3 Power gating sequence for router 23](#_Toc399452567)

[4.5.4 AHB-lite master bridge 23](#_Toc399452568)

[4.5.5 AHB-lite Slave Bridge 30](#_Toc399452569)

[4.5.6 AXI-lite Master Bridge 33](#_Toc399452570)

[4.5.7 AXI-lite Slave Bridge 34](#_Toc399452571)

[4.5.8 APB Bridge 34](#_Toc399452572)

[4.6 Hardware Implementation 36](#_Toc399452573)

[4.6.1 Registers 36](#_Toc399452574)

[4.7 Power Sequences Using Regbus RD/WR 49](#_Toc399452575)

[4.7.1 Power Gate Sequence for Master Bridge 49](#_Toc399452576)

[4.7.2 Power Enable Sequence for Master Bridge 49](#_Toc399452577)

[4.7.3 Power Gate Sequence for Slave Bridge 50](#_Toc399452578)

[4.7.4 Power Enable Sequence for Slave Bridge 52](#_Toc399452579)

[4.7.5 Power Gating Sequence for Router 53](#_Toc399452580)

[4.7.6 Power Enable Sequence for Router 53](#_Toc399452581)

[4.7.7 Power Gate Sequences for a Simple Noc with power profiles 54](#_Toc399452582)

[4.7.8 Rules for Power Sequence 57](#_Toc399452583)

[4.8 Power ON vs Power Enable 57](#_Toc399452584)

[4.9 Power Sequencing Using Hardware State Machines. 59](#_Toc399452585)

[4.9.1 Register Description 60](#_Toc399452586)

[4.9.2 1st Level FSM operation 61](#_Toc399452587)

[4.9.3 2nd Level FSM Operation 63](#_Toc399452588)

[4.9.4 Sample Power sequence 63](#_Toc399452589)

# Overview

Low power features in the Netspeed noc include the following :

1. Fine Grained clock gating : at the flop level within a noc element
2. Coarse Grained clock gating : at the noc element level
3. System level clock gating : external system controller generated clock gating of noc element clocks
4. Power gating of individual noc elements, or a collection of noc elements
5. Voltage and Frequency scaling of a Noc or Sub-noc
6. Logic level power reduction options exist, but these will be considered at a later time.

Each of these schemes is explained in brief, followed by a description of the operation sequence required to engage the feature.

# Clock gating

[5/18/2014 Note : This section, the initial clock gating spec, is now out of date, and the reader should refer to Sandip’s Clock Gating Spec. It will be refreshed in due time]

## Fine grained clock gating

This is implemented using the Synthesis tools. Exists in the product today

## System Level Clock Gating

Each noc element has a clock input pin, which corresponds to the root of the clock distribution within the noc element. For the synchronous case, a single clock exists per noc element, while for the async case two clocks can exist for a bridge, and up to 5 clocks can exist for a router.

System Level clock gating allows a “system\_clk\_enable” pin to exist for each clock input on a bridge or router. This enable is generated or controlled by a system clock controller external to the NoC. There are two methods to provide such a system\_clk\_enable :

1. Primary input to the NoC, one per clock per noc element. Timing to be guaranteed by the customer
2. The regbus slave bridge contains a pair of 32b registers : SYSTEM\_CK\_EN and SYSTEM\_CK\_EN\_MASK. Each bit of the \*MASK register corresponds to a write enable for the SYSTEM\_CK\_EN register. The outputs of SYSTEM\_CK\_EN serve as the clock enables for each NoC clock. Writes to these registers are done via the Regbus Master Bridge by the System clock manager block.

NocStudio must support a prop called system\_ck\_en\_via\_regbus ‘on’ to implement #2. NocStudio also implements a layer-wise numbering scheme to populate the register and register mask in a pre-determined fashion with noc clock element enables.

TBD : If a valid packet is sent to a noc element which is clock gated by SYSTEM\_CK\_EN, an interrupt must be signaled. This may require SYSTEM\_CK\_EN to be routed to the neighbouring router/bridge to set a bit indicating LINK\_NOT\_AVAILABLE. The interrupt must be generated at the sending router/bridge. The signal must be routed to a the neighbouring element using flops which are not clock gated (get a free running clock).

Note that the Regbus slave bridge clock is not gated. Regbus may also run on a different clock from the rest of the Noc, so synchronizers are recommended at the noc-elements before using the signals.

## Coarse Grained clock Gating

In this feature, a noc-element ‘R’ is clock gated based on occupancy of its fifos and the occupancy of neighbouring element fifos with a flit destined for ‘R’.

### Override feature

This feature may be enabled by ‘COARSE\_GRN\_CG\_OVERRIDE’ signal. When set to logic ‘1’, coarse grain clock gating is overridden and the feature is disabled. This signal may be sourced in two ways :

1. Primary input to the NoC, one per noc element. Timing to be guaranteed by the customer
2. The regbus slave bridge contains a pair of 32b registers : COARSE\_GRN\_CG\_OVERRIDE and COARSE\_GRN\_CG\_OVERRIDE\_MASK. Each bit of the \*MASK register corresponds to a write enable for the register COARSE\_GRN\_CG\_OVERRIDE. The outputs of this register serve as the overrides for each NoC clock gating feature. Writes to these registers are done via the Regbus Master Bridge by the System clock manager block.

NocStudio must support a prop called system\_ck\_en\_via\_regbus ‘on’ to implement #2. NocStudio also implements a layer-wise numbering scheme to populate the register and register mask in a pre-determined fashion with noc clock element overrides.

Note that the Regbus slave bridge clock is not gated. Regbus may also run on a different clock from the rest of the Noc, so synchronizers are recommended at the noc-elements before using the signals.

### Logic for clk gating

* 1. A clock gating logic block is implemented for each bridge or router. Flops in this block should never be gated by local clock enables. Sytem\_ck\_enable
  2. Self\_Idle : the state of a noc element that has no transactions or flits to process in its internal buffers.
  3. Ifce\_idle : a signal from a neighbouring block to indicate that the neighbouring block has no transactions destined for the element.
     1. A router can have up to 6 neighbours – N,E,W,S,H and Regbus.
     2. A bridge can have up to 10 neighbour interfaces : 8 routers, 1 host, 1 regbus
     3. Ifce\_idle\_N is if no transaction for the element is present in the N element.
  4. If all Ifce\_idle are asserted and self\_idle is asserted, the element may be clock gated
  5. Ifce\_idles have to be valid one cycle ahead of the packet arriving at the element’s interface. If a valid transaction destined for ‘R’ enters R’s neighbour ifce element, in the very next cycle, R’s local\_clk\_enable must be asserted and R’s clk should be enabled.
  6. The critical timing path is in (d). On entering a neighbour element :
     1. CYC-1 : Tck2q (R\_nhbr\_flit\_valid) + T\_wire\_delay + Tsu(in R)
     2. CYC-2 : Tck2q(enable\_flop\_ck2q) + Tck\_enable\_gen\_delay + Tsu
     3. Both CYC-1 and CYC-2 must be balanced for no degradation to occur in noc timing.
     4. Pre-routing is recommended for these signals to guard against scenic routes and for good signal integrity.
  7. As the ifce\_idle signal which is the critical path mentioned in ” e “ above must be valid a cycle before data, for the case where the bridge or router flit output is not registered, the flit will have to be delayed by a cycle before being allowed to arbitrate for the output port. The ifce\_idle signal will not be registered. Therefore in the case where bridge/router flit outputs are registered, no penalty exists when having to wake up the downstream noc element.
  8. A programmable count is provided to hold ifce\_idle low (indicating busy) to the neighbour for some number of cycles. This is to provide some hysteresis value to the clock gating logic so that clock gating logic does not engage too often, and thereby increase the average latency of the traffic.
  9. System clk enable, Regbus ifce\_idle and R\_local\_clk\_gating\_override are external to the router or bridge ‘R’, but should not be critical paths

### Latency Penalty on Coarse Clock enable

The following table describes which cases incur an added cycle of penalty when the neighbouring noc-element is coarse grain clock gated.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Neighbour element | Flit o/p registered | Async Bdry | Clock Gated Element | Penalty on Clock Enable | Notes |
| 1 | Router | Y | N | Router | N |  |
| 2 | Router | N | N | Router | Y |  |
| 3 | Router | Y | Y | Router | Y |  |
|  | Router | N | Y | Router | Y |  |
|  | Str Bridge | Y | N | Router | N |  |
|  | Str Bridge | N | N | Router | Y |  |
|  | Str Bridge | Y | Y | Router | Y |  |
|  | Str Bridge | N | Y | Router | Y |  |
|  | Router | Y | N | Str Bridge | N |  |
|  | Router | N | N | Str Bridge | Y |  |
|  | Host | n/a | N | Str Bridge | N | Free running input register on bridge when clk-gated |
|  | AXI Br ‘AR’ | N/Y | N/Y | Router | Y | AR channel must deassert ‘ready’ when neighbour is clock gated |
|  | AXI Br ‘AR’ | Y | N/Y | Router | Y | AR channel must deassert ‘ready’ when neighbour is clock gated |
|  | AXI Br AW,W | Y/N | Y/N | Router | N | AW,W takes 2 cycles within bridge |
|  | Router | Y | N | AXI ‘B’ | N |  |
|  | Router | N | N/Y | AXI ‘B’ | Y |  |
|  | Router | Y | N | AXI ‘R’ | N |  |
|  | Router | N | N/Y | AXI ‘R’ | Y |  |
|  | Regbus Br | n/a | N | Router/ SBridge/ AXi Br | N |  |

## NocStudio Requirements

The list of requirements from NocStudio are below:

* Specify clock frequency per noc element, and clock crossing.
  + AXI/ACE Bridges : sync and async exists
  + Routers : need async specification, only sync currently exists
  + Streaming Bridges : sync exists, no plan for async interface
  + Pseudo sync or meso-sync interface props are needed
  + Eventual plan to have NocStudio decide which section of the Noc runs on what frequency, but this is far out.
  + To specify multi-clock Noc;
    - Support separate clock for a given layer
    - In ECO mode, specify / change router clocks and clock crossings, flow bandwidth adjustment factor is needed.
* Specify system\_ck\_en\_via\_regbus property
  + Enables system control of clocks via regbus
  + Instance enable, override and mask registers in regbus slave
  + Hookup \*CK\* slave bridge reg outputs to noc-element inputs
  + Route enable signals to neighbour to signal LINK\_NOT\_AVAILABLE. Use Regbus layer for this, along with flop repeaters (non clk gated).
* New clock logic module to be present in each router/bridge. Outputs to be tied off in wrapper
* Slave bridge has some added outputs and needs a new wrapper

# Power Gating

Definition of terms :

* Voltage Domain : Where the Vsupply to a collection of transistors can be varied independently of other Vsupplys. Srams or flip-flops may have transistors tied to different voltage domains. A level shifter may be required to cross the voltage domain boundary
* Power gated Domain : an area to which Vsupply can be interrupted by means of a power switch.
  + Power switches can be on-chip : always on cells and state retention allowed
  + Power switch is off chip : no always on cells
* A relationship exists between Voltage Domain and Power Gated Domain. A Voltage Domain may have several Power Gated Domains within it.
* In NocStudio terms, a Host may contain multiple Voltage Domains and each Voltage Domain may contain several Power Gated Domains.
* Power Profile : A use case which describes a collection of power gated and always on domains in ON, OFF states.
* Power State : An operating state which consists of a collection of agents working at certain V,f combinations

## Specification of Power, Voltage Domains & Profiles

Power domain specification is done relative to each host. An add\_host command should now be associated with a power domain. It is also possible in the case of large hosts to specify multiple power domains associated with a host. Essentially each bridge on a host can be **associated** with a power domain. The bridge to host power protocol will be described in a later section. The add\_host statement must be extended:

**Add\_host cpu pos 15 size 7 8 bridge m power\_domain Pcpu**

The noc must be able to handle hosts from different voltage domains. A cpu in a voltage domain that can be independently varied, must be able to a communicate with a memory controller running at a different voltage. NocStudio must know about this, and be able to insert the necessary level shifters to allow this. This is done by specifying the voltage\_domain of the host and the bridge associated with it.

**Add\_host cpu pos 15 size 7 8 bridge m power\_domain Pcpu voltage\_domain Vcpu**

**Add\_host mem pos 85 1 2 bridge s power\_domain Pmem voltage-domain VSoc**

Hosts must have a default voltage domain called Vsoc.

A VAO, PAO for always on, must exist.

The bridge inherits host power and voltage domain by default. Therefore a bridge must have a bridge\_prop called power\_domain and voltage\_domain. If the user desires the bridge to be in a different power domain, **bridge\_prop** is used to modify the values.

If a bridge or noc element is required to support state retention, it must expose an addition pair of properties : **power-domain-ret, voltage-domain-ret**. Specific noc elements like the CCC must have additional properties for sram voltages.

A power profile is a collection of power domains. A new construct is required :

**Add\_power\_profile N P0 P1 P3 P6** :

This specifies the list of power domains which are ‘ON’ for a specific use-case. ‘N’ can be a number or a string. Examples of different power profiles can be :

**Add\_power\_profile PP\_all P0,P1,..Pn** : a case where all power domains are ON

**Add\_power\_profile PP\_video\_dec Pcpu Pmem Pvdec Pdisp Pusb**

Power Profiles must be hierarchically specifiable :

**Add\_power\_profile PP\_camcorder Pisp, Pvenc, PP\_video\_dec**

The use of reserved strings to indicate power, voltage domains and power profiles is encouraged in NocStudio, for eg : V\_, P\_, PP\_ .

### Specifying Noc Power/Voltage

We have above specified how a given bridge can be assigned a power and a voltage domain. We now need to be able to specify power and voltage domains for routers and pipeline stages. Since these elements are generated by NocStudio, many possible algorithms are possible that take into account power as a cost function, to choose the best possible routes that allow connectivity to be maintained when power domains are gated off. Map\_opt may need to understand domain and profile information, as will tune\_route, so that nocStudio is able to **choose** routes based on power and voltage domain information.

The development of power aware features in NocStudio should be split into two stages :

1. Phase 1: Build a network of routers optimized for traffic profiles. Assign routers to domains and profiles based on the need to maintain connectivity.
2. Phase 2 : Build a network of routers that uses power as a cost function for both floorplan and routing.

## NocStudio requirements for Phase-1

The starting point is the existing set of optimizations present in NocStudio.

* Allow specification of Voltage Domain, Power Domain information as described above. Bridges can inherit P/V information from the host, OR have user specified P/V information for each bridge.
* Introduce the concept of power profiles through the add\_power\_profile construct
* Using traffic and floorplan information as before, create the network of routers using map/map\_opt etc
* Associate Regbus elements with Vsoc and treat this as a separate power domain Pregbus that is always ON, as long as Vsoc is ON. Have an option to put the regbus master bridge in a VAO domain if one exists in the SoC.
* For the non-Regbus layers in the network, P/V/PP information must be derived by NocStudio from the input specification and routes :
  + For each PP consider inactive Hosts as not sending any traffic, and based on this find the number of unused routers. Express this as a fraction of total routers (FOM) to provide some information to the User about possible power saving. Graphically show which routers are inactive in the GUI.
  + Introduce a command to add a set of active routers to a particular power profile. NocStudio can automatically do this.
    - Add\_routers\_to\_power\_profile
    - For PP\_x the power profile now contains the set of routers that are **active**.
  + Routers should be considered by default to belong to Vsoc voltage domain. If it interfaces to a Vcpu bridge for example, NocStudio infers the domain crossing.
  + If routers do belong to a different power domain, this property must be passed to the router by the User. 4 properties are needed :
    - Router\_voltage\_domain
    - Router\_voltage\_domain\_ret
    - Router\_power\_domain
    - Router\_power\_domain\_ret
  + Pipeline flops are not power gated. However a voltage\_domain property is required for pipeline flops, which can be user modified. ECO mode insertion of pipeline flops must now require a voltage domain property to be specified by the user. A power domain property of AO should be set by default.
  + CCC, DVM, NCB blocks must be identified by NocStudio as power gate-able for a given power profile. These blocks might require properties to indicate ram power supply. Since the retained state in these blocks can be larger than in bridges and routers, these block require special handling (TBD) to un-load and re-load context, or for state retention.
  + AHB master > AXI convertor and it’s associated AXI bridge must be treated as one voltage and power domain. There does not appear to be a need to treat them separately.
  + AXI > AHB Slave convertor and its associated slave bridge can talk to multiple AHB slaves. For the case where a single slave is present, it inherits the P,V, property of the slave. When multiple slaves are present, it must be assigned a unique (but user modifiable) P, V domain.
  + AXI-APB slave convertor should be handled the same way as the AXI>AHB Slave convertor.
* With the set of specifications above, a default or user specified power domain, voltage domain and power profile is present for each noc element. Signals between these elements can be seen to cross voltage or power domains, and the proper isolation should be inferred. Based on the domain crossings, Gen-ip must generate the right CPF files to allow proper isolation.
* Static power estimation from nocStudio is required
* Large hosts can be declared as transparent to through the block routing : insertion of routers as well as pipeline flops. These elements must be in the AO power domain.
* NocStudio assist to System power manager. For a given power profile, generate the following :
  + List of elements to be power gated
  + List of elements that must be informed to NOT send traffic to a soon to be power gated destination
  + Pseudo-code for power gate sequence & power enable sequence.
    - Must include host interaction (reset, stop\_traffic, etc)
    - Must have specific register read / write operations that set up the reset, clock gate, isolate, power\_shutoff, etc events, with appropriate wait times in between.
    - Must have these operations in the right sequence for a single element as well as for multiple elements.
    - Generate a sequencing micro-code which can be used to drive the RTL
      * This may require the user to specify the order in which hosts should be power up or down.
  + Ability to simulate regbus traffic. The interest here is to be able to simulate the power gate, enable sequences to determine the time taken to gate/enable blocks in a certain power profile. The time constant of the power grid should really be the limiting factor here, but often slow or poorly written firmware can add significantly to the time taken to gate/enable. Being able to estimate this is a significant value add to determining a power gating architecture.

## NocStudio Requirements for Phase-II

* Dynamic Power estimation from nocstudio simulator
* Ability to provide a weight or priority to a power profile
* Placement algorithm to take in power as a cost function, power profile weights and come up with optimal placements. Will need more sophistication to weight solutions between power and latency for eg.
* Routing algorithm to take in power as a cost function.
* Routing algorithm to take in power profiles, and ensure that connectivity is maintained with 2 turn constraint for each power profile.

## Hardware Requirements (high level)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal Name | Source | Destination | Purpose | Active value |
| System\_noc\_clock\_en | CK Mgr | Router/Br | Enable noc clk | 1 |
| System\_reg\_clock\_en | CK Mgr | Router/Br | Enable regbus clk | 1 |
| System\_noc\_cg\_ov | CK Mgr | Router/Br | Override HW CG for noc clk | 1 |
| System\_reg\_cg\_ov | CK Mgr | Router/Br | Override HW CG for regbus clk | 1 |
|  |  |  |  |  |
| Noc\_element\_pwr\_en | PM/Regs | Router/Br | Enable power switch | 1 |
| Noc\_element\_isolation\_en | PM/Regs | Router/Br | Enable output isolation for normal layers | 1 |
| Noc\_element\_regbus\_ifce\_isolation\_en | PM/Regs | Router/Br | Enable output isolation to regbus interface | 1 |
| Noc\_element\_pg\_req | PM/Regs | Router/Br | Power gate request to bridge | 1 |
| Noc\_element\_pg\_rdy | Router/Br | PM/Regs | Power Gate Status response from Bridge (1=> ready) | 1 |
|  |  |  |  |  |
| Link\_not\_available | Router/Br | Neighbouring Router/Br | Signal to a neighbouring to indicate that an interrupt must be raised in the event of a packet being routed to the PG bridge or router | 1 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Clock and Power Gating Sequences

An external power manager is assumed to exist, which could be a firmware engine. The power manager performs the sequence of operations with handshakes that are required for power gating. Portions of this sequence may be performed by dedicated FSM hardware that exists in the Ringmaster, if NocStudio is directed to enable it. The signal names used in this sequence may be signals originating directly from the PM, or from registers in the Ringmaster which are set by the PM through Regbus. A unique set of signals or Registers must exist per noc element. The sequences using registers are covered in section 4.7.

### Host with single AXI4 master bridge.

The example sequence below shows the steps to be taken when a host is power gated. The master bridge inherits the voltage and power domain properties from the host. Isolation must therefore be inserted between each bridge and the routers which are its immediate neighbours, including Regbus. Domain crossing isolation is not required between the host and its master bridge, though an interface protocol must exist between them, to enable reliable operation during power gating operations.

It is possible for the Bridge to have a different power domain from the host and NocStudio allows this. In this event, isolation will also be inserted on the host interface.

* PM decides to Power Gate master host
* PM waits for host to be idle
* PM requests Master Bridge to Power Gate
  + PM sends bridge power\_gate message by asserting ‘Noc\_element\_pg\_req’
  + Bridge signals ‘not ready’ on its AR, AW, W interface to Master to inhibit new requests. Any requests in progress must be accepted before signaling ‘not ready’. In case of Streaming Bridges, it must indicate Link\_not\_available to master.
    - If an AMBA master requires a ‘link\_not\_available’ indication to power gate, it can be provided on the AMBA bridges as well.
  + When all transactions outstanding in bridge are complete, and no responses remain to be delivered, it signals ‘Noc\_element\_pg\_rdy’ to power manager, and ‘Link\_not\_available’ to its neighbours on the NoC.
  + No nack of a power gate request will be supported as it adds to complexity in the PG sequence. 5/7 decision
* PM initiates PG sequence for Master Bridge
  + PM isolates the bridge by asserting ‘isolation\_en’ for primary layer paths.
  + If any register state is to be saved, PM reads those registers through Regbus and saves the state
    - Register state programmed by supervisor code need not be ‘saved’. It need only be reprogrammed.
  + PM isolates the bridge by asserting ‘regbus\_ifce\_isolation\_en’ for the regbus path.
  + PM clock gates the bridge by deasserting ‘clock\_en’.
  + PM power gates the bridge by deasserting ‘pwr\_en’
* The Bridge is now power gated. Bridge outputs are driven to their pre-decided power gated value by the isolation\_en signals. Bridge inputs are not isolated by the sequence above; if isolation is required, it must be done at the source block.
* The Master or Host must also be power gated by the same domain ‘pwr\_en’ as the Bridge. If several power enable signals exist in a power domain to limit in-rush current by staggering power up, the PM must isolate both Master and master Bridge from their neighbours until all power regions within the domain are up, are stable and reset has taken effect. In practice, the bridge power enable comes from Regbus, and the master power enable may not come from the same register bit. These should both be enabled before reset is deasserted for the bridges.
* Power manager decides to Power enable host and bridge
* PM initiates Power enable sequence
  + PM signals power\_enable to bridge and to master (separately, or through common signals as discussed above). PM must wait a finite time for voltage levels to reach nominal operating points.
  + PM asserts separate resets to bridge and master.
  + PM asserts separate clock\_enables for bridge and master. Bridge is no longer clock gated. Resets must be asserted for long enough after clock\_en assertion to clear any spurious X’s in host - bridge interface
  + PM deasserts reset. Bridge is still isolated and continues to indicate ‘not ready’ to the Master, and ‘link\_not\_available’ to its router neighbours. PM must guarantee that Master does not propagate X’s into the bridge after master reset is deasserted.
  + PM deasserts ‘regbus-ifce\_isolation\_en’ to Bridge. This allows the PM to program the master Bridge registers to its values prior to Power Gating.
* PM performs bridge register programming
  + During this time, the Master is not allowed to send requests to the Bridge. This is enforced by ‘isolation\_en’ causing ready to be forced to ‘0’ until bridge setup is complete. In case of Streaming Bridge, link\_not\_available on the Master interface performs the same function.
  + PM must program bridge address registers if different from initial values
  + PM must Demap any power gated slave ranges from master bridge address table
  + PM must program Bridge QoS registers if different from initial values
  + PM must program Coarse clock gating Hysteresis counter value if different from the initial value.
* Once bridge is programmed PM deasserts ‘isolation\_en’. This also causes ‘link\_not\_available’ to be deasserted, and allows the master Bridge to signal ‘ready’ to the AMBA master.
* The Bridge is now ready for use.

### Host with single AXI4 Slave Bridge

The example sequence below shows the steps to be taken when a host such as a memory controller with a slave bridge is power gated. The slave bridge inherits the voltage and power domain properties from the host. Isolation must therefore be inserted between each bridge and the routers which are its immediate neighbours, including regbus. Domain crossing isolation is not required between the host and its slave Bridge, though an interface protocol must exist between them, to enable reliable operation during power gating operations. The signal names used in this sequence may be signals originating directly from the PM, or from registers in the Regbus Slave Bridge which are set by the PM through Regbus. A unique set of signals are assumed to exist per noc element.

Detailed steps for portions of the sequence which are identical between Power Gating of master and slave Bridges are omitted below in the interest of brevity.

* PM decides to gate slave host, and the slave Bridge along with it.
* Each Master host that can request to the slave must be inhibited. This sequence may be skipped if for eg. all Masters that can request to this slave have already been inhibited earlier, and the Slave is the last host to be power gated.
* To disable slave address range on each master bridge :
  + The PM must issue a regbus write to disable each distinct address range in the master’s address table. No operation currently exists to disable all or ‘some’ address ranges in a master’s address table with a single write. The PM knows that the writes have taken effect when it receives ‘Bresp’, and can then move on to the next step in the sequence.
  + At this time the master Bridge may still have outstanding requests to the Slave which needs to be power gated. These requests must complete before the slave is power gated. Options to determine if outstanding requests are done :
    - PM issues special write to set a ‘slave-id+region’ register in the bridge, which checks (static compare) each cycle for older transactions and updates a status bit. When all transactions to that slave complete, the status ‘done’ bit is updated. PM issues polling reads to this register to determine if ‘done’ bit is set.
    - Can only do one slave disable at a time.
    - If there are no outstanding transactions, the sequence is :
      * Write slave-id-outstanding register
      * Read status register (should indicate ‘done’),
* When all Masters which can request to the slave have had their address tables updated to disable requests to the Slave, the PM can power gate the Slave host as well as the slave Bridge.
* The PM asserts ‘pg\_req’ to the Slave Bridge.
* After the PM polls and sees a Pg\_rdy status from the Bridge, it initiates and completes the PG sequence.
* The slave Host as well as the slave Bridge are now power gated.
* **Power manager decides to enable Slave and bridge**
* PM initiates power enable sequence starting with power\_en> reset > clk\_
* PM deasserts regbus\_ifce\_isolation\_en and programs slave Bridge registers
* PM deasserts isolation\_en for Bridge after programming Slave host.
* PM programs each Master bridge such that Slave is enabled as a destination by issuing writes over regbus to the address table~~.~~
* Once the PM is done updating address tables for each MBR of interest, the slave host and bridge are ready for use by the system.

While disabling masters, slaves and their respective Bridges, scenarios may exist where a bridge being isolated may mean a deadlock in a multi-hop traffic flow. These scenarios must be planned for by the programmer as it cannot be described in NocStudio.

**No Regbus read or write can have any dependency on transactions from the Normal layers. Regbus must poll when needed, but returning the polling response can never be gated by an event from the normal layer.**

### Power gating sequence for router

A router may only be power gated when all traffic routes through that router have been disabled, by programming the master bridge address tables appropriately. The sequence is :

* PM asserts PG\_request to router
* PM polls PG\_ready status of router before moving to the next step. PG\_ready in turn sets ‘link\_not\_available’ to neighbours
* PM asserts ‘isolation\_enable’ signals for regbus and normal layer
* PM deasserts ‘clock\_en’ to disable clock
* PM deasserts ‘pwr\_en’

The router is now power gated. To re-enable power, operations are in the reverse order by the PM :

* Assert ‘pwr\_en’
* Assert ‘ reset’
* Assert ‘clk\_en’
* Deassert ‘reset’ after a long enough interval to clear X’s
* Deassert ‘isolation\_en’ on the regbus interface
* Perform any register programming (hysteresis counter for eg) through Regbus
* Deassert ‘isolation\_en’ on layer interfaces and also therefore ‘link\_not\_available’

### AHB-lite master bridge

Constraints for AHB-lite that are different from AXI4:

* When reset is asserted : HREADY must be high.
* When master asserts IDLE or BUSY, slave must respond with HREADY high, HRESP ok.
* Implies we cannot use HREADY to backpressure master and then gate off the noc as we do with AXI4.
* For HREADY to back-pressure master, a transaction must always be PENDING in the convertor.

To meet these constraints a ‘shim’ layer must always be available to respond to the AHB master bridge. Since the convertor is small, it is proposed to keep the convertor on as long as the Host is on.

The two cases this presents are :

#### AHB-lite Master Bridge power gating sequence (Bridge & Host in same PD)

In this case no isolation is needed between host and bridge, but isolation is needed between bridge and router. **Convertor is allowed to be part of bridge & host power domain and is powered off when axi4-bridge powers off**.

Because AHB protocol is not set up for clock or power gating, a clear sequence must be defined at the system level to enable AHB power gating.

* PM decides to Power Gate master host
  + PM ensures Master host is in RESET, must not be in the middle of a transaction when RESET is asserted.
* PM requests Master Bridge to Power Gate
  + PM sends **AXI4 bridge component a** power\_gate message by asserting ‘Noc\_element\_pg\_req’
  + AXI4 Bridge signals ‘not ready’ on its AR, AW, W interface to **AHB2AXI Convertor** to inhibit new requests. Any requests in progress must be accepted before signaling ‘not ready’.
    - If an AMBA master requires a ‘link\_not\_available’ indication to power gate, it can be provided on the AMBA bridges as well.
  + When all transactions outstanding in AXI4 bridge are complete, and no responses remain to be delivered, it signals ‘Noc\_element\_pg\_rdy’ to power manager, and ‘Link\_not\_available’ to its neighbours on the NoC.
  + During this time, the AHB2AXI Convertor may have a valid request from the AHB-lite master, which is indicated as PENDING to the AHB-lite master. **This request will be lost by the convertor when it powers off.**
* PM initiates PG sequence for Master Bridge
  + PM isolates the bridge by asserting ‘isolation\_en’ for primary layer paths. In this case, isolation is required between Bridge and router, but not between Bridge and Convertor OR between Convertor and Host.
  + If any AXI4 register state is to be saved, PM reads those registers through Regbus and saves the state. Convertor register state, if it exists, may also be saved.
    - Register state programmed by supervisor code need not be ‘saved’. It need only be reprogrammed.
  + PM isolates the bridge by asserting ‘regbus\_ifce\_isolation\_en’ for the regbus path.
  + PM clock gates the convertor by deasserting ‘conv\_clock\_en’
  + PM clock gates the bridge including convertor by deasserting ‘clock\_en’
  + PM power gates composite bridge by writing PWR\_EN register
  + PM completes the power gating Host by deasserting pwr\_enable for the Host.

The Master or Host must also be power gated by the same domain ‘pwr\_en’ as the Bridge. If several power enable signals exist in a power domain to limit in-rush current by staggering power up, the PM must isolate both Master and master Bridge from their neighbours until all power regions within the domain are up, are stable and reset has taken effect. In practice, the bridge power enable comes from RingMaster, and the master power enable may not come from the same ringmaster register bit.

**Power manager decides to Power enable host and bridge :**

**PMGR must ensure Master host is not enabled before AHB-lite bridge is ready!**

* PM initiates Power enable sequence
  + PM signals power\_enable to AHB-lite bridge and to Host (separately, or through common signals as discussed above). PM must wait a finite time for voltage levels to reach nominal operating points.
  + PM asserts RESET to Host , and enables clock to Host. Master host stays in RESET until both bridge and convertor are ready.
  + PM asserts RESET to Convertor by writing PDR-conv register
  + PM asserts RESET to AXI4 MBr by writing PDR
  + PM asserts clock\_enables for Convertor
  + PM asserts clock\_enables for Bridge. Resets must be asserted for long enough after clock\_en assertion to clear any spurious X’s in host – conv - bridge interface.
  + PM deasserts reset to AXI4 bridge component.
    - AXI4 component of the AHB-lite Bridge will now assert READY to the AHB2AXI Convertor component. The Convertor MUST stay in RESET.
  + PM deasserts ‘regbus-ifce\_isolation\_en’ to Bridge. This allows the PM to program the master Bridge registers to its values prior to Power Gating.
* PM performs bridge register programming of AXI4 MBr component.
  + PM must program AXI4 Mbr bridge address registers if different from initial values
  + PM must Demap any power gated slave ranges from AXI4 master bridge address table
  + PM must program AXI4 MBr Bridge QoS registers if different from initial values
  + PM must program Coarse clock gating Hysteresis counter value if different from the initial value.
* PM deasserts RESET to Convertor by writing PDR-conv. Convertor is now ready.
* PM deasserts RESET to Master host. Host may come out of RESET and request to convertor.
* PM deasserts ‘isolation\_en’. This also causes ‘link\_not\_available’ to be deasserted, and allows the flow of traffic.
* The AHB-lite Master Bridge is now ready for use.

#### AHB-lite Master Bridge power gating sequence (Bridge & Host in different PD)

In this case, **the AHB-lite convertor is ALWAYS ON, but the AXI4 Master bridge** may be power gated. Isolation therefore must exist between host and convertor as well as between convertor and AXI4 MBr.

* PM decides to Power Gate AHB-lite master bridge
* PM requests Master Bridge to Power Gate
  + PM sends **AXI4 bridge component a** power\_gate message by asserting ‘Noc\_element\_pg\_req’
  + AXI4 Bridge signals ‘not ready’ on its AR, AW, W interface to **AHB2AXI Convertor** to inhibit new requests. Any requests in progress must be accepted before signaling ‘not ready’.
    - If an AMBA master requires a ‘link\_not\_available’ indication to power gate, it can be provided on the AMBA bridges as well.
  + When all transactions outstanding in AXI4 bridge are complete, and no responses remain to be delivered, it signals ‘Noc\_element\_pg\_rdy’ to power manager, and ‘Link\_not\_available’ to its neighbours on the NoC.
  + During this time, the AHB2AXI Convertor may have a valid request from the AHB-lite master, which is indicated as PENDING to the AHB-lite master. **This request will be retained by the convertor as it never powers off – it is ALWAYS ON.**
* PM initiates PG sequence for Master Bridge
  + PM isolates the bridge by asserting ‘isolation\_en’ for primary layer paths. In this case, isolation is required between Bridge and router, AND between Bridge and Convertor AND between Convertor and Host. Inputs to Convertor (AO) must be isolated.
  + If any AXI4 register state is to be saved, PM reads those registers through Regbus and saves the state. Convertor register state, if it exists, may also be saved, though since Convertor is responding to Host, the state may be dynamic.
    - Register state programmed by supervisor code need not be ‘saved’. It need only be reprogrammed.
  + PM isolates the AXI4 bridge by asserting ‘regbus\_ifce\_isolation\_en’ for the regbus path. If the Convertor is AO, the regbus interface to the Convertor may remain ON.
  + PM must now ensure that Host does not expect responses from the NoC, as the Convertor cannot complete any NONSEQ/SEQ transactions.
  + PM clock gates the AXI4 MBr bridge but not the AHB2AXI convertor by deasserting ‘clock\_en’.
  + PM power gates AXI4 MBr. Convertor is not power gated.

Power manager decides to Power enable host and bridge

* PM initiates Power enable sequence
  + PM signals power\_enable to AHB-lite bridge. PM must wait a finite time for voltage levels to reach nominal operating points.
  + PM asserts resets to AXI4 MBr component. If the Host is also coming up from PG state, the convertor should also be reset to clear any PENDING transaction.
  + PM asserts separate clock\_enables for bridge. Resets must be asserted for long enough after clock\_en assertion to clear any spurious X’s in host - bridge interface.
    - Note that since Convertor is AO, clock was not disabled to it.
  + PM deasserts reset to AHB-lite bridge.
    - AXI4 component of the AHB-lite Bridge is still isolated and continues to indicate ‘not ready’ to the AHB2AXI Converter, and ‘link\_not\_available’ to its router neighbours.
    - The AHB2AXI Convertor may receive a valid request from the Host at this time over the AHB-lite interface. The Convertor must be able to respond to IDLE/BUSY requests with HREADY/HRESP = ready/ok. It must also be able to receive a NON-SEQ request and respond with PENDING, while the AXI4 MBr goes through its power up sequence.
    - PM must guarantee that Host does not propagate X’s into the Convertor after master reset is deasserted.
  + PM deasserts ‘regbus-ifce\_isolation\_en’ to Bridge. This allows the PM to program the master Bridge registers to its values prior to Power Gating. Note that the Convertor is working at this time, and may not be programmed if it is holding valid requests from the Host.
* PM performs bridge register programming of AXI4 MBr component.
  + During this time, the Master is not allowed to send requests to the Bridge. This is enforced by ‘isolation\_en’ causing ready to be forced to ‘0’ until bridge setup is complete.
  + PM must program AXI4 Mbr bridge address registers if different from initial values
  + PM must Demap any power gated slave ranges from AXI4 master bridge address table
  + PM must program AXI4 MBr Bridge QoS registers if different from initial values
  + PM must program Coarse clock gating Hysteresis counter value if different from the initial value.
* Once bridge is programmed PM deasserts ‘isolation\_en’. This also causes ‘link\_not\_available’ to be deasserted, and allows the master Bridge to signal ‘ready’ to the AHB2AXI Convertor.
* The AHB-lite Master Bridge is now ready for use.

### AHB-lite Slave Bridge

The AHB-lite SBr = AXI4-SBr + AXI2AHB Convertor. Both components are always in the same power domain.

The AHB-lite SBr may or may not be in the same power domain as the Slave Host(s), especially for the case with multiple slave hosts.

#### AHB-lite Slave Host Power gating (Slave host and Bridge are in different PDs)

The AHB-lite Bridge can have upto 16 connections to multiple slave hosts. Therefore one or more of these hosts can be power gated while the bridge is still powered on. The sequence to power off an AHB-lite slave is below, from the point of view of the NoC.

1. Power manager decides to power off an AHB-lite slave.
2. Before power gating the AHB-lite slave, PMgr must :
   1. Demap slave address from all master bridges
   2. Ensure outstanding transactions in the master bridges to the slave have completed.
   3. This is done using the methods described in 4.5.2
3. Any subsequent transaction to this AHB-lite slave will cause a DEC\_ERR at the MBr.
   1. If a demap is not done, and a transaction is issued, the convertor will send the transaction on to the power gated slave. This may cause a hang or timeout if the AHB-lite slave is powered down.
4. The AXI2AHB Convertor ensures :
   1. HSEL to the power gated slave is not asserted
5. The Power gated AHB-lite slave must ensure:
   1. HREADYOUT is driven HIGH & HRESP = OK, to indicate that it has no transactions in a PENDING state. This can be misleading to the convertor as it can independently conclude that the slave is available. Therefore PMGR control of address demap is critical for correct operation.
6. When the AHB-lite slave is power enabled by the PMGR and available for use, the address range for this slave must be enabled at the MBr. No register programming action is needed at the convertor.

#### AHB-lite Bridge Power gating (Slave host and Bridge are in different PDs)

This is a composite bridge : AXI4 SBr + AXI2AHB-lite Convertor. Similar to the AXI4 Sbr and AXI-lite bridge, the AXI4 SBr component of the composite bridge is aware of all outstanding transactions to AHB-lite slave hosts. Therefore a similar sequence to AXI4 SBr power gating can be employed.

In practice, the expectation is that the system PMGR would first power gate all slave hosts before power gating the AHB-lite slave bridge.

* If not already done as part of AHB-lite slave host power gating :
  + Demap AHB-lite slave host address ranges from all master bridges
  + Ensure all outstanding transactions at the master bridge(s) are complete for all the AHB-lite slaves connected to the AHB-lite slave bridge.
* Issue power-gate request to the AXI SBr. In this case the SBr is not expected to have any transactions.
  + Convertor ensures HSELs are pulled low at the AHB-lite interface to the slave host.
  + Convertor drives IDLE to all slaves. This should also be the isolated (clamped) value driven.
  + It is not required to pull READY low for the AXI4 SBr, though this may be a don’t care.
* Complete the rest of the power gating sequence exactly as for AXI4 SBr.
  + Separate reset and clock enable for the Convertor is not required.

Power Manager decides to power enable bridge

* Power Manager must ensure that when the AHB-lite SBr is enabled, the AHB-lite Slave hosts are able to respond to transactions driven by the AXI2AHB convertor. As power gated AHB-lite slaves drive HREADYOUT=1, HRESP=OK, there is no way for the convertor to know from interface signals if the AHB-lite slave is power gated or not.
* Once the PMGR has power-enabled AHB-lite slaves, the same sequence used to power enable an AXI4 slave bridge may be used to enable the AHB-lite SBr.

#### AHB-lite Bridge Power gating (Slave host and Bridge are in same PD)

In this case, the AHB-lite SBr and the AHB-lite Slave host can be in the same power domain. This is a case where the AHB-lite SBr is connected to a single AHB-lite Slave host.

* When all Masters which can request to the slave have had their address tables updated to disable requests to the AHB-lite Slave host, the PM can power gate the Slave host as well as the slave Bridge.
* The PM asserts ‘pg\_req’ to the Slave Bridge.
* After the PM polls and sees a Pg\_rdy status from the Bridge, it initiates and completes the PG sequence : interface\_isolation, state save if needed, regbus interface isolation, clock disable, power disable.
* The slave Host as well as the slave Bridge are now power gated.
* **Power manager decides to enable Slave and bridge**
* Power Manager must first enable AHB-lite Slave and ensure it is active before enabling AHB-lite SBr. The AHB-lite protocol does not allow the slave host to back-pressure the bridge.
* PM initiates power enable sequence for AHB-lite SBr starting with power\_en> reset > clk
* PM deasserts regbus\_ifce\_isolation\_en and programs slave Bridge registers
* PM deasserts isolation\_en for Bridge
* PM programs each Master bridge address table such that Slave is enabled as a destination.
* Once the PM is done updating address tables for each MBR of interest, the slave host and bridge are ready for use by the system.

### AXI-lite Master Bridge

AXI-lite Master bridge = AXI4 Master bridge with input signals tied off to AXI4-lite pinout. It is not technically a composite bridge. Therefore there is no difference between the power gating sequence of an AXI4 MBr and the AXI4-lite MBr.

### AXI-lite Slave Bridge

AXI-lite Slave Bridge = AXI4 SBr + AXI4-to-AXI4-lite convertor

The AXI4 SBr tracks all outstanding transactions, including any that are in the AXI2AXI-lite convertor. As the AXI-lite interface to the slave also has the same valid ready protocol as does the AXI SBr, the same sequence can be used. The entire AXI-lite SBr is in the same power domain, which might be the same or different as the host. In brief this sequence is :

* Demap slave address range from master bridge
* Ensure all outstanding transactions at the master bridge(s) are complete for the AXI-lite slave to be power gated.
* Issue power-gate request to the AXI SBr. In this case the SBr is not expected to have any transactions. READY is pulled low at the AXI-lite interface to the slave. It is not required to pull READY low for the AXI4 SBr, though this may be a don’t care.
* The rest of the power sequence is completed as for the AXI4 SBr. Isolation happens at the AXI-lite outputs to the host if it is a different power domain.

### APB Bridge

#### APB Slave Host Power gating

The APB Bridge can have upto 16 connections to multiple slave hosts. Therefore one or more of these hosts can be power gated while the bridge is still powered on. The sequence to power off an APB slave is below, from the point of view of the NoC.

* Power manager decides to power off an APB slave.
* Before power gating the APB slave, PMgr must :
  + Demap slave address from all master bridges
  + Ensure outstanding transactions in the master brdiges to the slave have completed.
  + This is done using the methods described in 4.5.2
* Any subsequent transaction to this APB slave will cause a DEC\_ERR at the MBr.
* When the APB slave is power enabled and is available for use, the address range for this slave must be enabled at the MBr.

#### APB Bridge Power gating

This is a composite bridge : AXI4 SBr + AXi2AXI-lite filter + APB Bridge. Similar to the AXI4 Sbr and AXI-lite bridge, the AXI4 SBr component of the composite bridge is aware of all outstanding transactions. Therefore a similar sequence can be employed :

* Demap slave address range from master bridge
* Ensure all outstanding transactions at the master bridge(s) are complete for the APB slave to be power gated.
* Issue power-gate request to the AXI SBr. In this case the SBr is not expected to have any transactions. PSEL is pulled low at the APB interface to the slave. It is not required to pull READY low for the AXI4 SBr, AXI-lite filter though this may be a don’t care.

## Hardware Implementation

The goal is to have power control signals implemented through Regbus registers. This is more efficient from the perspective of global signal routing, as NocStudio will ensure that all signals are pipelined along with Regbus, and no random global long wire signals need to be routed at the top level. Also implementing power controls through Regbus registers allows NocStudio to synthesize the power sequence using Regbus RD/WR.

Signals between the Power Control Registers in the Ringmaster and the Noc Elements on the node should be declared Multi-cycle paths to eliminate any timing issues from wire routing and possibly clock domain crossing. This requires the following relationship to hold true for signal values for ports that are isolated : reset\_value == isolation\_value == idle\_value.

APB Bridge

Router

AXI Bridge

Router

Pwr Ctl Regs

Regbus Router

Ring Master

Regbus Router

Regbus Master

### Registers

A complete set of registers is defined per noc element on a node. Creating registers this way is more natural for NocStudio.

No registers are presently defined that need retention across PG event.

#### Power Gate Request (PGR)

|  |  |  |  |
| --- | --- | --- | --- |
| Bit Position | 31 | 30:2 | 0 |
| Field | Chain\_Op | Reserved | PG Request |
| Reset Value | 1 | --- | 0 |
| Driven by | Regbus Master |  | Regbus Master |
| Received by | RingMaster FSM |  | Bridge |

PG Request : When set to ‘1’ a Power Gate request is made to the Bridge

Chain\_Op : when set, the FSM chains this operation to the next operation. When ‘0’ the next Operation is initiated by the PM using Regbus RD/WR.

Operation : PM initiates a PG request by writing ‘PG Request’ to ‘1’. Master Bridge will assert not\_ready, process existing transactions and when they are all complete, will set PG\_Ready in the PSR Register. PG\_Ready also drives link\_not\_available status to the Link\_status\_register in the neighbouring bridge or router.

FSM Op (prelim) : If Chain op is set, FSM should monitor PG\_ready status in Power Status Reg (PSR), and when set, can initiate the layer isolation operation which is the next step in the sequence.

#### Check Outstanding to Specified Slave (AM\_OSSLV) and Status Flags Register (AM\_STS)

Location : ACE Master Bridge

Operation : PM initiates a Demap Slave operation by writing to the Address table registers in the Master Bridge to disable the Slave range as a destination. Subsequent transactions to the demapped range will result in a decode error. If this is an operation to power gate the slave, then older transactions to that Slave in the master bridge must be first allowed to finish. The PM must write the Slave and optionally Region IDs in AM\_OSSLV via regbus. When all older transactions to the slave of interest are complete, the bridge logic updates the awo, aro status bit in AM\_STS register. The PM must poll the AM\_STS register and when no transactions are found to be outstanding to the Slave of interest, it can proceed to the next step in its power gating sequence.

#### Isolation Enable (IEN) & Isolation Enable Counter (IEC)

Location : Ringmaster

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IEC Bit Position | 31 | 30:17 | 16:1 | 0 |
| Field | Chain\_Op | RSVD | Counter Val | Counter\_en |
| Reset Value | 1 | 0 | 0 | 0 |
| Driven by | Regbus Master | -- | Regbus Master | Regbus Master |
| Received by | RingMaster FSM | -- | RingMaster FSM | Ringmaster FSM |

|  |  |  |
| --- | --- | --- |
| IEN Bit Position | 31:18 | 0 |
| Field | RSVD | Isol\_en |
| Reset Value | 0 | 0 |
| Driven by | -- | Regbus Master / FSM |
| Received by | -- | Bridge |

Counter\_en : ‘1’ use counter value for interval, ‘0’ do not use counter value, but explicit PM access for the next step in the sequence.

Counter\_value : count down from the value to ‘0’ before FSM sets ‘Isol\_en’.

Isolation\_enable : when set to ‘1’, isolates noc element outputs by clamping to reset == idle value

Chain\_Op : when set, the FSM chains this operation to the next operation. When ‘0’ the next Operation is initiated by the PM

Operation : IEC values can be setup at boot.

* If Counter\_en is set ‘1’, FSM loads the counter value, and counts down to ‘0’. At ‘0’, it sets Isol\_en.
* If Counter\_en is at ‘0’, PM must count down the appropriate interval, and then issue a WR over Regbus to set Isol\_en.

#### Isolation Enable Regbus (IENR)and Isolation Enable Counter Regbus(IECR)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IECR Bit Position | 31 | 30:17 | 16:1 | 0 |
| Field | Chain\_Op | RSVD | Counter Val | Counter\_en |
| Reset Value | 1 | 0 | 0 | 0 |
| Driven by | Regbus Master | -- | Regbus Master | Regbus Master |
| Received by | RingMaster FSM | -- | RingMaster FSM | Ringmaster FSM |

|  |  |  |
| --- | --- | --- |
| IENR Bit Position | 31:18 | 0 |
| Field | RSVD | Isol\_en\_reg |
| Reset Value | 0 | 0 |
| Driven by | -- | Regbus Master / FSM |
| Received by | -- | Bridge /Router |

Count\_en : ‘1’ use counter value for interval, ‘0’ do not use counter value, but use explicit PM access for the next step in the sequence.

Counter\_value : count down from the value to ‘0’ before FSM sets isol\_en\_reg.

Isolation\_enable\_regbus : when set to ‘1’, isolates noc element **regbus** outputs by clamping to reset == idle value

Chain\_Op : when set, the FSM chains this operation to the next operation. When ‘0’ the next Operation is initiated by the PM

Operation : IECR values can be setup at boot

* If Counter\_en is set ‘1’, FSM loads the counter value, and counts down to ‘0’. At ‘0’, it sets Isol\_en\_reg.
* If Counter\_en is at ‘0’, PM must count down the appropriate interval, and then issue a WR over Regbus to set Isol\_en\_reg.

#### Clock Enable (CKEN) and Clock Enable Counter (CKEC) (tbd : match Sandip’s spec)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CKEC bit pos | 31 | 30:17 | 16:1 | 0 |
| Field | Chain\_Op | RSVD | Counter Val | Counter\_en |
| Reset Value | 1 | 0 | 0 | 0 |
| Driven by | Regbus Master | -- | Regbus Master | Regbus Master |
| Received by | RingMaster FSM | -- | RingMaster FSM | Ringmaster FSM |

|  |  |  |
| --- | --- | --- |
| CKEN Bit Position | 31:1 | 0 |
| Field | RSVD | Clk\_en |
| Reset Value | 0 | 0 |
| Driven by | -- | Regbus Master / FSM |
| Received by | -- | Bridge |

Counter\_en : ‘1’ use counter value for interval, ‘0’ do not use counter value, but explicit PM access for the next step in the sequence.

Counter\_value : count down from the value to ‘0’ before setting clk\_en.

Clk\_en : when set to ‘1’, clock is enabled, when ‘0’ clock to the noc element is disabled

Chain\_Op : when set, the FSM chains this operation to the next operation. When ‘0’ the next Operation is initiated by the PM

Operation :

* If Counter\_en is set ‘1’, FSM loads the counter value, and counts down to ‘0’. At ‘0’, it sets Clk\_en.
* If Counter\_en is at ‘0’, PM must count down the appropriate interval, and then issue a WR over Regbus to set clk\_en.
* When clk\_en is set, it factor into and sets link\_not\_available in the neighbouring elements link\_sts\_register

#### Power Enable Register (PER) and Power Enable Counter (PEC)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PEC Bit Pos | 31 | 30:17 | 16:1 | 0 |
| Field | Chain\_Op | RSVD | Counter Val | Counter\_en |
| Reset Value | 1 | 0 | 0 | 0 |
| Driven by | Regbus Master | -- | Regbus Master | Regbus Master |
| Received by | RingMaster FSM | -- | RingMaster FSM | Ringmaster FSM |

|  |  |  |
| --- | --- | --- |
| PER Bit Position | 31:1 | 0 |
| Field | RSVD | Pwr\_en |
| Reset Value | 0 | 0 |
| Driven by | -- | Regbus Master / FSM |
| Received by | -- | Bridge |

Count\_en : ‘1’ use counter value for interval, ‘0’ do not use counter value, but explicit PM access for the next step in the sequence.

Counter\_value : count down from the value to ‘0’ before setting Pwr\_en.

Pwr\_enable : when set to ‘1’, power is enabled, when ‘0’ power to the noc element is disabled

Chain\_Op : when set, the FSM chains this operation to the next operation. When ‘0’ the next Operation is initiated by the PM

Operation : PEC values can be setup at boot

* If Counter\_en is set ‘1’, FSM loads the counter value, and counts down to ‘0’. At ‘0’, it sets PSR.Pwr\_good.
* If Counter\_en is at ‘0’, PM must count down the appropriate interval, and then issue a WR over Regbus to set PSR.Pwr\_good. It is not strictly necessary to do this if the HW FSM is not involved in the next step.

#### Power Domain Reset (PDR) and Power Domain Reset Counter (PDC)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PDC Bit Pos | 31 | 30:18 | 16:1 | 0 |
| Field | Chain\_Op | RSVD | Counter Val | Counter\_en |
| Reset Value | 1 | 0 | 0 | 0 |
| Driven by | Regbus Master | -- | Regbus Master | Regbus Master |
| Received by | RingMaster FSM | -- | RingMaster FSM | Ringmaster FSM |

|  |  |  |
| --- | --- | --- |
| PDR Bit Position | 31:1 | 0 |
| Field | RSVD | PD\_reset\_n |
| Reset Value | 0 | 0 |
| Driven by | -- | Regbus Master / FSM |
| Received by | -- | Bridge |

Count\_en : ‘1’ use counter value for interval, ‘0’ do not use counter value, but explicit PM access for the next step in the sequence.

Counter\_value : count down from the value to ‘0’ before setting PDR.

PD-reset\_n : when set to ‘0’, reset is asserted, when ‘1’ reset is deasserted

Chain\_Op : when set, the FSM chains this operation to the next operation. When ‘0’ the next Operation is initiated by the PM

Operation : PDC values can be setup at boot

* Reset is set to ‘0’ by the FSM or by a Regbus Write. If Counter\_en is set ‘1’, FSM loads the counter value, and counts down to ‘0’. At ‘0’, it sets PD\_reset\_en to ‘1’.
* If Counter\_en is at ‘0’, PM must issue a WR to assert reset (PD-reset\_n =0) PM must count down the appropriate interval, and then issue a WR over Regbus to set PD-reset\_n.
* **TBD : Must differentiate Cold Power-on reset from Power Domain reset.**

#### Power Status Register (PSR)

Location : Ringmaster

One PSR per Bridge or Router

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bit Position | Field | Reset Value | Driven by | Received by | Function |
| 0 | PG Ready | 0 | BR/RT | PM/FSM | This status bit is updated by Bridge or Router, in response to a PG\_req. When set, the BR/RT is ready for PG. |
| 1 | IEN\_sts | 0 | BR/RT | PM/FSM | Set by BR/RT in response to setting isol\_en |
| 2 | IENR\_sts | 0 | BR/RT | PM/FSM | Set by BR/RT in response to setting isol\_en\_reg |
| 3 | Pwr\_Good | 0 | PM/FSM | FSM | Set by FSM or PM after sufficient interval has elapsed after asserting power enable (PER.pwr\_en). Does not have to be set if HW FSM is not used. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

#### Link Status Register Router (LSRR)

Location : Router

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bit Position | 31:5 | 4 | 3 | 2 | 1 | 0 |
| Field | RSVD | N\_sts | E\_sts | W\_sts | S\_sts | H\_sts |
| Reset Value |  | 1 | 1 | 1 | 1 | 1 |
| Driven by |  | Neighbor | neighbour | neighbour | neighbour | neighbour |
| Received by |  | Interrupt logic | Interrupt logic | Interrupt logic | Interrupt logic | Interrupt logic |

Link Status Register Bridge (LSRB)

Location : Bridge

|  |  |  |
| --- | --- | --- |
| Bit Position | 31:8 | 7:0 |
| Field | RSVD | Layer[7:0] |
| Reset Value |  | 1 |
| Driven by |  | Neighbor |
| Received by |  | Interrupt logic |

The NEWSH and Layer [7:0] link status bits are continuously driven by the neighbor elements in the NoC.

The Link\_status bits are fed by the link\_not\_available signals. Link\_not\_available can be driven high by pg\_ready, isolation\_enable, ~clk\_en as all of these can cause the neighbouring element to not be available.

#### FSM Instruction Register (FIR)

Location : ringmaster, one per NoC element on node

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bit Position | Field | Values | Notes | Field | Value | FSM\_en |
| 31 | FSM\_en | 1 | Enable FSM op | FSM\_en | 1 | Enable FSM operation |
| 30:28 | Inst\_type | 001 | Power Gate M Br | Inst type | 100 | Pwr Enable MBR |
|  |  |  |  |  |  |  |
|  | WR PGR | 1 | #1 Set PGR[0] | WR PER | 1 | #1 Set PGR[0] |
|  | RD PSR | 1 | #2 If PSR[0], do ne xt step | WR PDR | 1 | #2 Clear PDR[0] |
|  | WR IEN | 1 | #3 Set IEN[0] | WR PGR | 1 | #3 Clear PGR[0] |
|  |  |  |  |  |  |  |
|  | WR IENR | 1 | #4 Set IENR[0] | WR CKEN | 1 | #4 Set CKEN[0] |
|  | WR CKEN | 1 | #5 Set CKEN[0] | WR PDR | 1 | #5 Set PDR[0] |
|  | WR PER | 1 | #6 Clear PER[0] | WR IENR | 1 | #6 Clear IENR[0] |
|  |  |  |  | WR IEN | 1 | #7 Clear IEN[0] |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

* Exact bit positions are left open for implementation convenience.
* Two instructions are defined above : Power Gate and Power Enable
* A value of ‘1’ against a field like WR PGR implies the associated operation is to be performed by the HW FSM
* A value of ‘0’ against the field implies the FSM will not perform the operation. If the operation is performed by FW, the FSM will monitor the result and perform the next operation if it is enabled (has ‘1’ next to it).
* A delay equal to the counter value associated with the register must be introduced in the sequence. For eg : performing the WR IEN operation should incur IEC # cycle delay.
* If in the PE sequence above, the IEN step is to be done after the PMGR does some register programming, the IEN register must be written by FW after completing the programming operation.
* For the simple case where no save/restore of register values is required, all bits can be ‘1’, and a power gating sequence reduces to a single instruction : WR FIR %inst\_value which then kicks off the state machine.
* If the gate count of the logic is tiny, the logic should be replicated conditionally for each noc element at the node. This allows multiple operations to happen in parallel.
* Power Enable and Gate sequences for master and slave bridges appear to be identical. **An exception is AHB-lite master bridge which requires separate control of PDR and CKEN for the convertor and axi4 bridge. Sequence and register definition for this is TBD.**
* DONE state of the operation may be monitored by ‘RD PSR’ from the PMGR. **PSR may need to be extended to store more status bits.**
* Other operations associated with MBr :
  + Demap address : this is one write per demapped address range, so no FIR instruction is required. No polling is needed from PMGR, so no benefit from implementing this operation through FIR.
  + Tracking completion of older transactions in MBr to a slave that needs to be power gated : This requires the following two operations :
    - PM : WR AM\_OSSLV
    - PM : RD AM\_STS : polling step to assess completion before moving to next step which could be demapping another slave. Polling for each slave to be PG-ed is expensive.
    - Defining a HW instruction to track completion for several slaves would be very useful. Upto 3-4 slaves could be tracked sequentially in a 64b FIR register. However 12 bits are needed per master bridge to set each AM\_OSSLV register. This is a direct connect not through ringmaster.

## Power Sequences Using Regbus RD/WR

### Power Gate Sequence for Master Bridge

PM decides to Power Gate master host & optionally waits for host to be idle.

* Power Gate Request & Layer Isolation Macro (PG-rq)
  + PM : WR PGR : make power gate request to Master Bridge
  + PM : RD PSR : poll to see if Slave Bridge is ready to PG. If yes, move to next step
  + PM : WR IEN : enable layer isolation
* Unload Bridge State (optional) Macro (PG-unload)
  + PM : RD %r1 Bridge/Router register
  + PM : RD %r2 Bridge/Router register
  + ….
* Enable Regbus isolation and Power Gate Macro (PG-pg)
  + PM : WR IENR : enable Regbus isolation
  + PM : WR CKEN : Gate clock to Bridge
  + PM : WR PER : Gate power to Bridge

Master Bridge is now Power Gated. The sequence in terms of macros are :

* **PG-rq** : Make power gate request and isolate bridge to stop normal traffic
* **PG-unload** (optional) : Unload register state
* **PG-pg** : Isolate regbus layer and power gate.

### Power Enable Sequence for Master Bridge

* Power Enable Macro (PE-pe)
  + PM : WR PER : enable power to bridge
  + PM : Wait N cycles for power up until voltage levels are good for cmos operation
  + PM : WR PDR (assert reset for the power domain)
  + PM : WR CKEN (Enable clock)
  + PM : Wait ‘M’ cycles for reset to propagate and clear X’s.
  + PM : WR PDR : deassert Reset
  + PM : WR IENR : deassert Regbus isolation to allow bridge programming
* Remap Bridge State Macro (PE-remap) : bridge register programming
  + PM : WR %r1 : Bridge address registers programming
  + PM : WR %r2 : Bridge QoS / counter programming
  + PM : WR %r3 : Add Demapped power gated slave range
  + …
* Enable Traffic on Bridge Macro (PE-et)
  + PM : WR IEN : deassert layer isolation

The Bridge is now ready for use.

The sequence in terms of instruction macros is :

* **PE-pe** : Enable power to deassertion of Regbus isolation
* **PE-remap** : Bridge Register programming, including power gated slave range Demap
* **PE-et** : de-assert Layer isolation and enable traffic through Bridge

### Power Gate Sequence for Slave Bridge

PM decides to Power Gate slave host and optionally waits for host to be idle

* Demap Slave Address ranges in Master Bridges, Macro : PG-demap
  + PM : WR Master Address register : Disable slave address in master bridge
  + PM : WR AM\_OSSLV : setup tracking for older transactions to the PG slave
  + PM : RD AM\_STS : poll for older transactions to be complete, if yes move to next step
* Power Gate Request & Layer Isolation Macro (PG-rq)
  + PM : WR PGR : make power gate request to Slave Bridge
  + PM : RD PSR : poll to see if Slave Bridge is ready to PG. If yes, move to next step
  + PM : WR IEN : enable layer isolation
* Unload Bridge State (optional) Macro (PG-unload)
  + PM : RD %r1 Bridge/Router register
  + PM : RD %r2 Bridge/Router register
  + ….
* Enable Regbus isolation and Power Gate Macro (PG-pg)
  + PM : WR IENR : enable Regbus isolation
  + PM : WR CKEN : Gate clock to Bridge
  + PM : WR PER : Gate power to Bridge

Master Bridge is now Power Gated. The sequence in terms of macros are :

* **PG-demap** : Demap Slave address range from Master Bridges before power gating
* **PG-rq** : Make power gate request and isolate bridge to stop normal traffic
* **PG-unload** (optional) : Unload register state
* **PG-pg** : Isolate regbus layer and power gate bridge.

### Power Enable Sequence for Slave Bridge

* Power Enable Macro (PE-pe)
  + PM : WR PER : enable power to bridge
  + PM : Wait N cycles for power up until voltage levels are good for cmos operation
  + PM : WR PDR (assert reset for the power domain)
  + PM : WR CKEN (Enable clock)
  + PM : Wait ‘M’ cycles for reset to propagate and clear X’s.
  + PM : WR PDR : deassert Reset
  + PM : WR IENR : deassert Regbus isolation to allow bridge programming
* Load Bridge State Macro (PE-load) : bridge register programming
  + PM : WR %r2 : Bridge register programming
  + …
* Enable Traffic on Bridge Macro (PE-et)
  + PM : WR IEN : deassert layer isolation
* Remap Bridge State Macro (PE-remap) : **Master** bridge register programming
  + PM : WR %r3 : Re-enable power gated slave range in master m1
  + Repeat for other masters as needed

The Bridge is now ready for use. The sequence in terms of instruction macros is :

* **PE-pe** : Enable power to deassertion of Regbus isolation
* **PE-load** : Bridge Register programming
* **PE-et** : de-assert Layer isolation and enable traffic through Bridge
* **PE-remap :** enable address ranges in master bridges for Slave

### Power Gating Sequence for Router

A router is power gated only after it is known that no traffic can pass through it. Expressed in terms of Instruction macros for brevity :

* **PG-rq** : Make power gate request and isolate router at layer interface
* **~~PG-unload~~** ~~(optional)~~ : Unload register state, very unlikely for router
* **PG-pg** : Isolate regbus layer and power gate router.

### Power Enable Sequence for Router

* **PE-pe** : Enable power > deassertion of Regbus isolation
* **PE-load** : Register programming (hysteresis counter?),
* **PE-et** : de-assert Layer isolation and enable traffic through Router

### Power Gate Sequences for a Simple Noc with power profiles

M2

M1

R3

R2

R1

S2

S1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Profile | M1 | M2 | S1 | S2 | R1 | R2 | R3 |  |
| PP0 | ON | ON | ON | ON | ON | ON | ON |  |
| PP1 | ON | ON | ON |  | ON | ON | ON |  |
| PP2 |  |  | ON | ON |  |  |  |  |
| PP3 | ON |  | ON |  | ON |  |  |  |
| PP4 |  | ON |  | ON |  |  | ON |  |

For brevity, the detailed steps in the previous sequence are reduced to macros.

PG : refers to the entire PG sequence,

Demap and Remap refers to disabling and enabling a slave from a masters address map. Demap complete also includes the completion of all outstanding transactions to the slave.

PE refers to the entire PG sequence, and sub-bullets may call out specific steps

An operation in one bullet must complete before the operation in the next bullet

PG M1 , PG M2 written in one line means the two ops can happen in parallel

#### PP0 => PP1 => PP0

PP1 requires S1 to be power gated.

Power Gate sequence

* PG-demap S1 in M1, PG-demap S1 in M2 : Demap S1 from M1, M2’s address tables
* PG S2 (rq > unload > pg)

Power Enable sequence

* PE S2 (pe > load > et)
* PE-Remap S2 in M1, PE-Remap S2 in M2

#### PP0 => PP2 => PP0

PP2 requires M1 & M2 to be power gated

Power Gate Sequence

* PG M1, PG M2
* PG R1, R2, R3

Power Enable Sequence

* PE R1, PE R2, PE R3
* PE M1, PE M2

Router power up must be complete before powering up masters. Though it may be possible to power all up together, it is necessary to ensure masters do not send traffic prior to routers being available. A useful simplification is to wait for routers to be powered up first. However, the sequence below is certainly faster :

* PE-pe (M1, M2, R1, R2, R3) in parallel
* PE-remap (M1, M2), (PE-load, PE-et (R1, R2 R3))
* PE-et (M1, M2)

Therefore all steps are done in parallel except for de-asserting layer isolation in M1, M2, which is done last. This ensures no traffic is sent before routers are ready.

#### PP0 => PP3 => PP0

PP3 requires M2, S2 to be power gated

Power gate sequence

* PG-demap S2 from M1, PG\_demap S2 from M2
* PG-rq M2, PG-rq S2
* PG R2, PG R3, (PG-unload, PG-pg M2, S2)

Simple Power Enable Sequence (high latency due to serialized power grid charge up time)

* PE R2, PE R3
* PE S2
* PE M2

Parallelism can be achieved by PE R2, R3, S2, M2 together, but prior to deasserting isolation-enable for layer outputs on M2, PM must ensure that R2, R3, S2 PE sequences are complete.

* PE (R2, R3, S2), (PE-pe, PE-remap M2)
* PE-et M2

#### PP3 => PP4

PP3 requires M2, S2 to be power gated, and PP4 requires M1, S1 to be power gated.

PP3 to PP4 sequence : power gate before power enable

* PG-demap S1 from M1
* PG M1, PG S1
* PG R1
* PE R3, PE S2
* PE M2 (including demap of S1 range from M2)

**The important step above is to ensure M2 never comes up with S1 enabled in its address map.**

Though the sequence above is ‘safe’, the disadvantage is that the long wait time for power to reach operating voltage levels in M2, S2, R3 is serialized with power gating of the other units. If the power delivery system can handle all units drawing current at the same time, the following sequence is preferred :

* (PE-pe, PE-remap M2), PE (R3, S2), PG-demap S1 from M1
* PG M1, PG S1, PG R1, PE-et M2

### Rules for Power Sequence

1. When Power Gating M&S always Demap Slave from Master address tables, and then Power Gate M, S, R in parallel if possible.
2. When Power Enabling M & S,
   1. Completion of Master Power Enable sequence must always be gated by Power enable sequence completion of Slave and Routers
   2. Deassertion of isolation-enable for Master bridge layers (not regbus) must be only be done after Demap of any power gated slaves.
3. When switching power profiles
   1. Starting Power Enabling in parallel with Power gating is preferred. If this is not allowed by the power delivery system constraints, then Power gating must be completed before Power Enabling. This will need to be a nocstudio switch.

## Power ON vs Power Enable

Power ‘ON’ or ‘Cold’ power ON is the event where power is delivered to the entire SoC by switching on the regulated external power supply. At power ON, the boot path, microboot sequence, boot handler and OS take control of the hardware. This requires the entire chip to be enumerated. During this sequence, the NoC must be power on as quickly as possible as this enables all on chip communication. Therefore RESET is propagated to the NoC through the RESETn pin at the NoC level to all NoC elements. This is the quickest possible method to enable communication through the NoC. As the NoC is small in area as compared to the other large IP blocks on an SoC like CPU, GPU, Memory etc, it also should be acceptable to enable Clock to all portions of the NoC simultaneously. This can be done through System\_clock\_en – a NoC level pin, or by having the register values at each RingMaster for system\_clk\_en come up enabled.

Power Enable may be thought of as an event where a block or a collection of blocks that have been power gated through on-chip switches are once again power enabled. Since this is less likely to require full enumeration of the SoC, the path of enabling reset and other control propagation through Regbus accesses to control registers in individual node Ringmasters is used.

A noc element therefore receives a NoC reset from the primary RESETn pin as well as a power domain reset from its individual RingMaster register.

## Power Sequencing Using Hardware State Machines.

Power state switching for NoC elements involves a sequence of operations that need to be performed per element along with sequencing/synchronization of operations for multiple elements. This can be achieved using individual sw register accesses, as described in Section 4.7.

Power state control logic in ring-master also implements hardware state machines that can perform low level pre-defined sequences in controlled manner. This removes the need for multiple reg-access per element for both power gate or enable sequence. Each element that has power switch control has a dedicated FSM controlled through FIR register. In addition to FIR register, there are PDC/CKEC/PEC registers to control wait times between steps of power sequences. The FSM can be instructed to Power gate or Power Enable corresponding element by writing FIR register.

The FIR register provides control for individual steps in the sequence, which can be skipped.

***<Abort:TBD>***

***<Commit:TBD>***

### Register Description

#### FSM instruction Register (FIR)

Location : ringmaster, one per NoC element on node

|  |  |  |  |
| --- | --- | --- | --- |
| Bit Position | Field | Reset Value | Description |
| [15:14] | Update PER | 2’b0 | update\_per [1:0] :  [1] – Enable wait (PEC == Counter)  [0] – Enable set/reset per |
| [13:12] | Update CKEN | 2’b0 | update\_cken [1:0] :  [1] – Enable wait (CKEC == Counter)  [0] – Enable set/reset cken |
| [11:10] | Update PDR | 2’b0 | update\_pdr [1:0] :  [1] – Enable wait (PDC == Counter)  [0] – Enable set/reset pdr |
| [9:8] | Update IENR | 2’b0 | update\_ienr [1:0] :  [1] – Enable poll PSR.IENR\_sts  [0] – Enable set/reset ienr |
| [7:6] | Update IEN | 2’b0 | update\_ien [1:0] :  [1] – Enable poll PSR.IEN\_sts  [0] – Enable set/reset ien |
| [5:4] | Update PGR | 2’b0 | update\_pgr [1:0] :  [1] – Enable poll PSR.PG\_RDY  [0] – Enable set/reset pgr |
| [3:1] | FSM Instruction | 3’b0 | FSM instruction :   * 3’b000 : Power Enable element * 3’b001 : Power Gate element |
| [0:0] | FSM Enable | 1’b0 | WR : enable or disable FSM  RD : FSM\_BUSY : FSM in non-idle state |

#### Aliased Registers

As the power sequencing of elements involve access to single bit status or enable bits distributed across register definition, aliasing provides a simple mechanism to group similar access to multiple element registers. This in turn reduces the reg access required for power switching.

Examples of aliased registers :

1. SET\_PGR : Writing 1 to the bit positions of this register will set PGR bit for the element that has the ring-id same as bit position.
2. STATUS\_PG\_RDY : Reading this register will provide status for PG\_RDY for all elements on particular ring.
3. STATUS\_FSM : Reading this register will provide status for FSM busy for all the elements on particular ring.

### 1st Level FSM operation

Each element whose power sequence is controlled at the ring master has a corresponding Level -1 FSM.

The FSM can update following signals/registers for the corresponding NoC element:

1. PER
2. CKEN
3. PDR
4. IENR
5. IEN
6. PGR

Each FSM has a counter to add defined delays post the update of certain registers based on corresponding counter control registers. (PDC/CKEC/PEC)

To minimize the width of counters in each FSM instance, the counter are based on common event-ticks which are generated at ring master level and shared for all the elements.

Based on the actual power profile requirements, some the steps can could be configure to be skipped.

For example, a master bridge being power enabled may need some slave(s) de-mapped from the valid address ranges as the slave is powered down. The FIR register could in this case be updated to keep master core logic (IEN) in isolation after the power up. Once the master is up, the slaves can be de-mapped by register accesses and then the isolation could be removed by writing to IEN register.

#### Power Gate Sequence

When the FIR register is written to power gate an element (FIR.instr == 3’b001), the FSM performs following steps.

* *Step 1: Set PGR and wait for status.*
* *Step 2: Set IEN and wait for status.*
* *Step 3: Set IENR and wait for status.*
* *Step 4: Assert PDR and wait for PDC count.*
* *Step 5: Reset CKEN and wait for CKEC count.*
* *Step 5: Reset PER and wait for PEC count.*
* *Step 6: FSM idle, sequence completed.*

For each step, the update or corresponding wait can be skipped by not setting corresponding bits in the FIR.

#### Power Enable Sequence

When the FIR register is written to power enable an element (FIR.instr == 3’b000), the FSM performs following steps.

* *Step 1: Set PER and wait for PEC count.*
* *Step 2: Reset PGR and wait for status.*
* *Step 3: Set CKEN and wait for CKEC count.*
* *Step 4: De-assert PDR and wait for PDC count.*
* *Step 5: Reset IENR and wait for status.*
* *Step 6: Reset IEN and wait for status.*
* *Step 6: FSM idle, sequence completed.*

For each step, the update or corresponding wait can be skipped by not setting corresponding bits in the FIR.

#### Abort Sequence

#### Commit Sequence

### 2nd Level FSM Operation

### Sample Power sequence