NetSpeed SYSTEMS

NetSpeed NoC Clock Gating Specification

NetSpeed NoC Clock Gating Guidelines

About This Document

This document describes coarse clock gating guidelines for a NoC implementation.

Audience

Document Organization

This document is organized as follows:

Chapter 1, Introduction to NoC coarse grained clock gating
Chapter 2, Coarse clock gating of Router
Chapter 3, Coarse clock gating of Steaming Bridge
Chapter 4, Coarse clock gating of AXi Master Bridge
Chapter 5, Coarse clock gating of AXI Slave Bridge
Chapter 6, Clock gate control through RegBus Slave Bridge
Chapter 7, RTL instantiations of clock gate module in different Noc elements

Related Documents

The following documents can be used as a reference to this document.

- NocStudio User Manual

Customer Support

For technical support about this product, please email contact@netspeedsystems.com

For general information about NetSpeed products refer to: www.netspeedsystems.com

Contents

List of Figures

List of Tables

Acronyms

NoC	Network on Chip
SoC	System on Chip
Host	An IP core, component, or device sitting in an SoC
Host Port	A port of a host that connects to NoC router's injection and ejection port via a bridge to be able to inject traffic into NoC or eject traffic from NoC
Router	A hardware switch at the cross point of a mesh connecting to up to 4 of its neighboring routers and one or more bridges to connect to one or more host ports
Bridge	Sits between a router's port (often injection/ejection) and a host port of a host to convert the host ports signal protocol (such as Streaming or AMBA AXI4) to NoC packet format and vice-versa and additional operations needed by the signaling protocol such as width conversion, etc
Link/Port	Physical channel between two routers or between a router and a bridge
Channel	Physical or virtual channel between two routers or between a router and a bridge
Virtual Channel (VC)	Virtual channel between two routers or between a router and a bridge
Cell	A node in a 2D grid or mesh; A NoC router is associated with every cell
Node	A router or cell of the 2D NoC grid
Mesh/grid	A 2-dimensional (2D) grid organization of routers; a router at each cross point of the grid; each router having 4 links in north-, south-, east-, and west-directions; adjacent routers being interconnected with each other using one of the four links
Multi-layer NoC	Multiple parallel mesh NoCs each forming a layer; routers in each layer operate independently of each other; two NoC layers have no connection between their routers; a bridge at a cell connects the injection/ejection port to a router in each layer and transmits each host port transaction to one of the layer's routers, and receives transactions from all layer's routers delivering them to the host port
Liberty lib files	Standard Cells Timing Characterization .lib files provided by the foundry
LEF	Library Exchange Format. These files include the design rules for routing and the Abstract of the cells
DEF	Design Exchange Format. It represents the netlist and circuit layout. DEF is used in conjunction with LEF to represent complete physical layout

1 Introduction to NoC Coarse Grained Clock Gating

Noc can support up to maximum 8 layers for user specified traffic and 1 Register layer for Configuration. Each one of these layers is a Mesh comprising of Routers, which are connected to hosts through bridges. Traffic is initiated by hosts and injected into the NoC through bridges. The bridges forward the traffic to its attached router. The traffic goes through subsequent routers as per routing information embedded in its packet, until it reaches the intended destination. So until traffic or transactions arrive and route through a particular NoC element, it can be clock gated off to save power. But it needs to be turned on, just enough time ahead so that the incoming packets are not dropped.

The idea behind coarse clock gating of different NoC elements is to save power when the Noc elements are in idle condition for substantially long period of time. This is different from fine grained clock gating (usually covered by synthesis logic) which controls clock gating on a cycle by cycle basis. Coarse grained clock gating shuts off entire branches of associated clock tree, saving power in the suppressed un-clocked flops and also the power needed to drive the clock network itself. This can lead to big power savings. This document provides a scheme to adopt coarse grained clock gating for the following NoC elements.

- Router
- Streaming Bridge
- AXI Bridge
- Pipeline stages

The scheme opportunistically shuts off the NoC elements under the following conditions

- There are no transactions buffered or being processed internally and all credits have been returned from its neighboring block
- There are no transactions buffered and inbound from a neighboring block

The scheme allows over-riding or bypass coarse clock gating under certain conditions

- A System override to universally disable clock gating across all NoC elements
- Fast path overrides to selectively disable clock gating for certain NoC elements falling in fast paths

Additionally the scheme takes into account a *System Clock Gate Enable* signal to shut off all the NoC elements on a certain NoC Layer.

The general equation for coarse clock gating of a NoC element is as follows

$$NoC\_element\_clk\_enable = System\_clk\_enable \,\&\, (Clock\_gate\_override \mid (\sim Self\_idle \mid \sim Ifce\_idle));$$

$$Clock\_gate\_override = System\_override \mid Fast\_path\_override;$$

Since *ifce_idle* signal is sent from the neighboring element and arrives late in the clock cycle, care must be taken to take it through minimum number of gates to generate the *NoC_element_clk_enable*.

The coarse clock gating scheme described here makes an important assumption, that the worst case latency to turn off or turn on a clock tree (to reach the deepest flop), native on a NoC element is no more than one clock cycle.

2 Coarse clock gating of a Router

A Router has 4 directional (N, W, S, E) ports, 1 host port and 1 Regbus port. It is statistically hard to reach no traffic condition on all its 6 ports in order to coarse clock gate the whole router off. It is a better strategy to coarse clock gate the router on a port by port basis. If all of the six ports reach quiescent stage, the whole router can be turned off. Each of these ports can be selectively coarse clock gated (turned off) based on empty condition of all of its associated input VC FIFOs and an input signal indicating idle condition on neighboring element connected to that port.

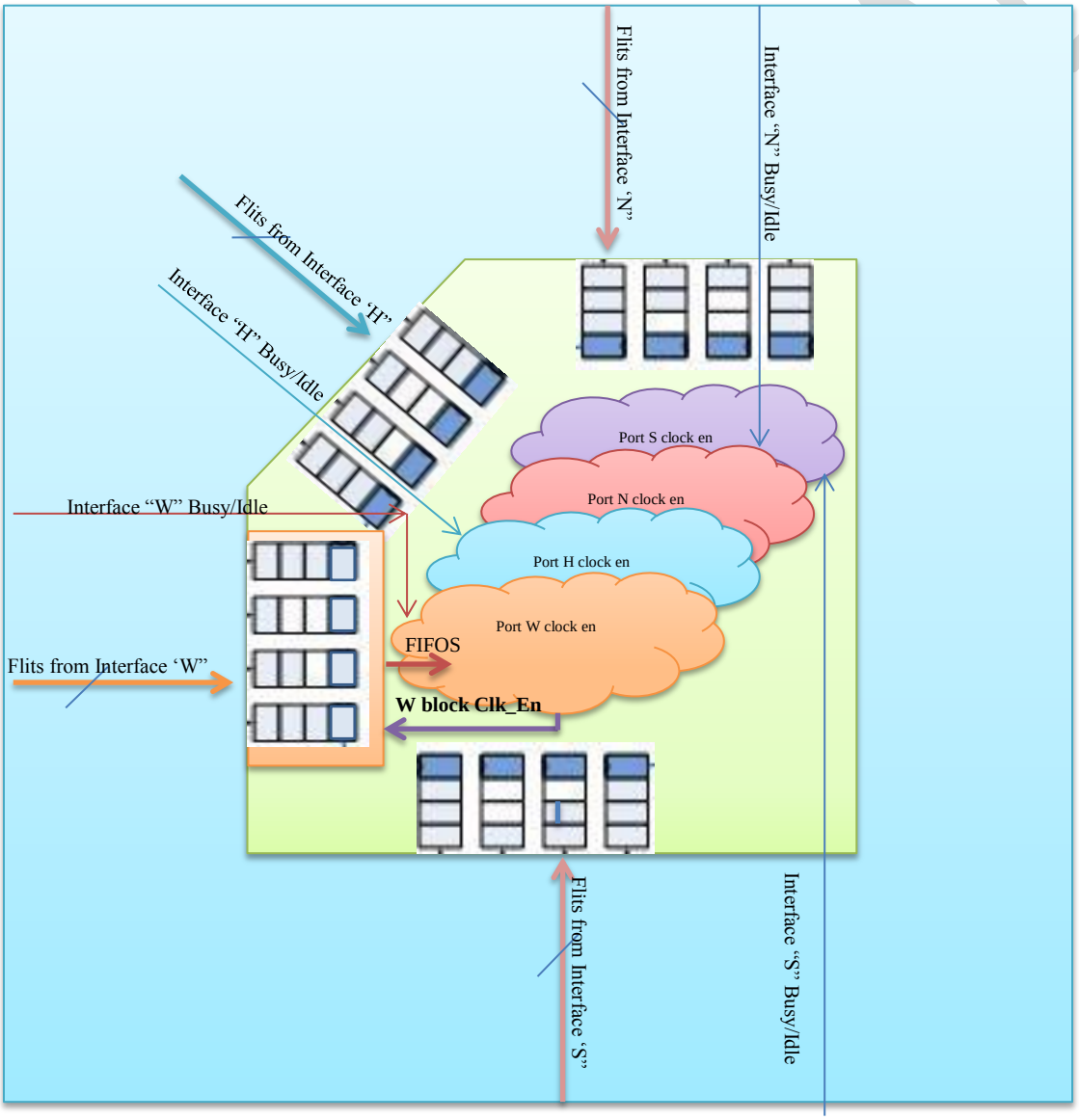

Figure 1 Coarse clock Gating Router

2.1 Coarse clock gating of a Router based on self-idle condition on a particular port

Evaluating this condition should be very straight forward. Once all the input VC FIFOs associated with that port of the router are empty, the *self-idle* signal for that port can be asserted high. For timing reason, it is important that this self-idle signal is output of a flop. The FIFO occupancy evaluation can be accomplished in the previous cycle. The empty state of the input VC FIFOs also indicate that all credits for the NoC element (Router, Bridge) connected to that port have been returned. Thus turning off the logic for that port won't cause any side effects for that neighboring NoC element.

2.2 Coarse clock gating of a Router based on busy/idle condition of neighboring element connected to a particular port

This section discusses coarse clock gating of a router based on traffic information from each of its *neighboring agents.* Various kinds of *neighboring agents* are discussed here. The *neighboring agents* generate the busy signal and is one per input port (N,E,W,S,H and Regbus) of the target router, which intends to exercise coarse clock gating. If the interface *busy* signal is low, that interface is considered by the router to be in *idle* condition. The interface idle conditions are used by the router to make two decisions:

I. To shut off clock and flops in the input block connected to that interface
II. To shut off the entire router if the rest of interfaces are already in idle condition and there are no other transactions ongoing or pending inside the router.

The *neighboring agent* will generate and transmit a dedicated *busy* signal, corresponding to the output port connected to the router. The busy signal indicates that the *neighboring agent* has transactions pending and ongoing for the router. The conditions for transitions on the *busy* signal are described below.

I. Assertion – The *neighboring agent* should set up this transition at least one cycle ahead of the first flit, sent to the destination router.
II. De-assertion – The *neighboring agent* should cause this transition, after a fixed number of cycles after all the transactions for the destination router has been exhausted. The number of cycles is programmed into a register, residing within the *neighboring agent*.

2.2.1 Interface busy signal from neighboring router with unregistered flits outputs

For sake of convenience, the *neighboring router* will be referred to as "**Router B**" and the target router (clock gating router) will be referred to as "**Router A**". Since a router has five output ports for traffic within a single NoC layer, there will be five busy signals from Router B. The particular output port of Router B, which is connected to input port of Router A will be referred to as Output port X (X corresponds to one of N,W,S,E or H). The corresponding busy signal will be named as *Output Port X Busy*. For Router B, early detection of outgoing port for an incoming flit is critical for setting up *Output Port X Busy* ahead of time.

2.2.1.1 Neighboring router with Synchronous Fifo inputs

Each router already has a dedicated input "*outp*" signal per "Router port" {N,W,S,E or H}. The information in "*outp*" signal is used to determine which output port the flit is destined for. In this scheme, the "*outp*" signal will be registered in addition to being written to its dedicated input VC Fifo. The registered "outp" signal will be read and decoded every cycle. Also, a status register *Output Port X Status* will be maintained for each output port of Router B.

If the decoded registered "*outp*" signal matches Output Port X, the signal "*Output Port X match*" will assert high. Router B will then perform the following two tasks:

- Set the input of "*Output Port X Status*" register to high. If "Output Port X Status" register output was low, it will transition to high in the next cycle. If the "*Output Port X Status*" register output had already been set to high, it will continue to maintain that state in the next cycle.
- Set the output signal "*Output Port X Busy"* high in the same cycle. The "Output Port X Busy" is OR function of "*Output Port X Status*" and "*Output Port X match*".

The register "*Output Port X Status*" is required to let a flit (at the head of the VC Fifo) know that corresponding input port and logic in Router A is already clock enabled. A flit can only arbitrate for Output Port X in Router B, if "*Output Port X Status*" is set to high. After an incoming flit to Router B gets written to its VC Fifo and happens to be in bypass condition, it cannot arbitrate in the same cycle for the *Output Port X* if "*Output Port X Status*" has **not** been set to high. That cycle will be utilized in sending "*Output Port X Busy*" to Router A, and to wake up and clock enable its corresponding input block in next cycle. This is the **cold start condition** of sending "*Output Port X Busy*" (clock enable) signal to Router A, ahead of the flit and it incurs a penalty of 1 cycle. In rest of the cases, "*Output Port X Status*" will have already been set high due to flits from other input ports and VCs contending for the same *Output Port X*. Only the first flit setting up "*Output Port X Status*" high for the first time, will incur 1 cycle penalty.

The "*Output Port X Status*" register will be cleared after a fixed number of cycles after all the transactions through "Output Port X" have been exhausted and the credits for all transactions through "Output Port X" have been returned from Router A. A counter can be maintained to keep track number of outstanding flits to be routed through "Output Port X". There will be five input "outp" signals to each Router corresponding to each port. Every time, the registered "outp" signal, corresponding to each input port matches "Output Port X" the counter is incremented. It could happen that in a given cycle, "outp" signal for all four input ports (excluding the "Input Port X") match "Output Port X". In this case the counter needs to be incremented by 4. The counter is decremented every time a flit exits through "Output Port X" in Router B. A second counter "*Output Port X power down counter*" is maintained inside each Router. Once the counter reaches zero, a fixed number of cycles can be counted before de-asserting "*Output Port X Status*". This is under the condition, that in the intervening period, there are no incoming flits destined for "Output Port X". If in fact the Router B sees an incoming flit bound for "Output Port X", the "*Output Port X power down counter*" is reset to zero.

The whole scheme is illustrated in the following diagram Figure 2

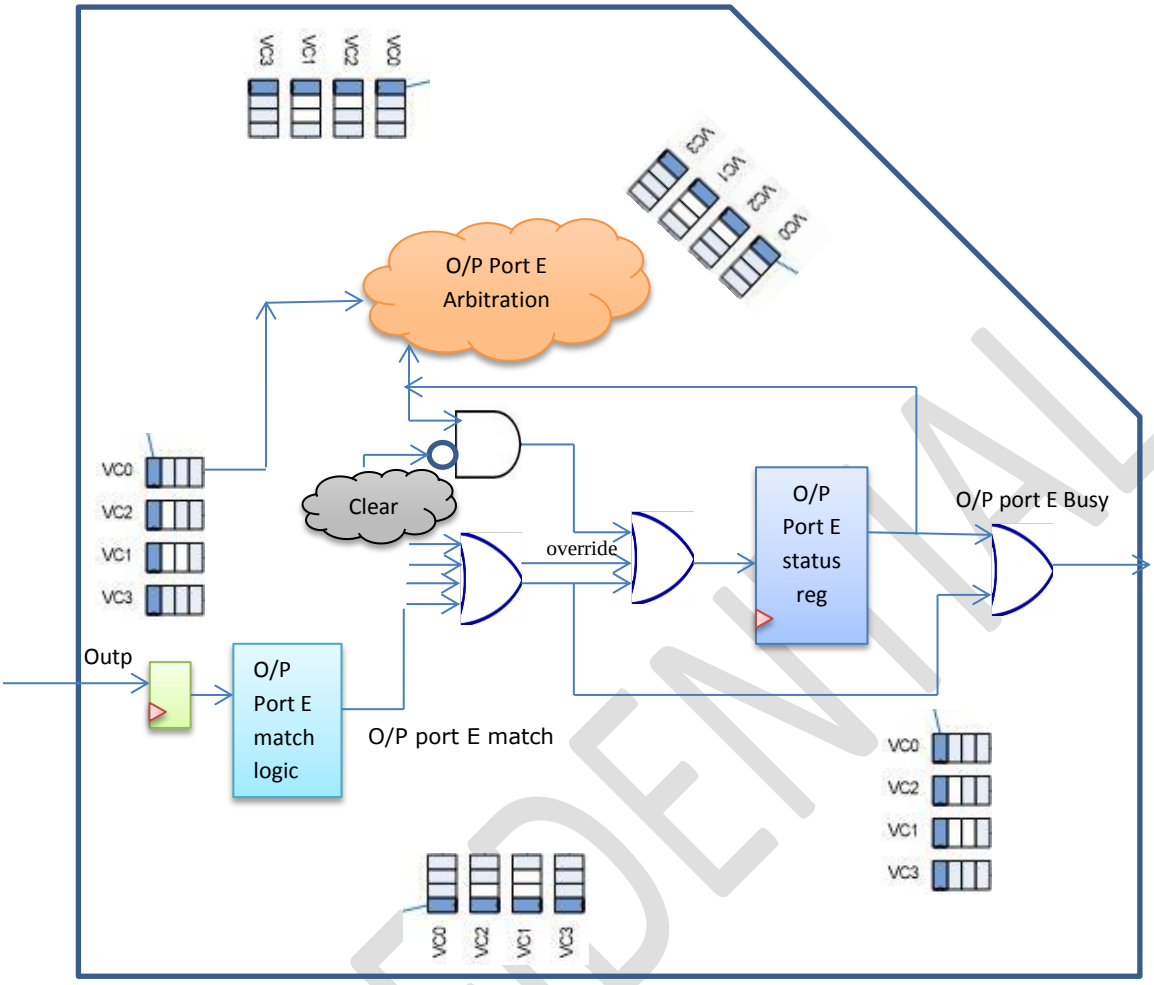

Figure 2 Synchronous interface: Generating Busy signal from neighboring router (Router B) with outputs (flits) un-registered

2.2.1.2 Neighboring router with Asynchronous Fifo inputs

In this case, *outp* information has to be read out from each Asynchronous Input VC FIFO. The existing Asynchronous FIFOs have logic to read an entry ahead (look ahead) of the current Read Pointer, provided that entry is valid. This look ahead read of *outp* information from each Asynchronous FIFO would potentially provide mechanism of setting up *Output Port X Busy* ahead of time. Since each of its 4 Input VC Fifos would be read out, there could be potentially 4 different values of *outp* every cycle from a particular input port.

In case of an input VC Fifo having just one entry, a look ahead read for *outp* is not possible. The *outp* information for that input VC Fifo is extracted from head of the queue. In addition, if "*Output Port X Status*" for that flit is in de-asserted state, a one cycle penalty is incurred. This is the cold start condition for Asynchronous case. This is for the case where, the Asynchronous Input VC FIFO is empty and the first flit that is written into the FIFO wants to access "*Output Port X*". In this case, arbitration is stalled for one cycle and is utilized to send "*Output Port X Busy"* to the destination router.

Since each of the 4 Input VC Fifos are read each cycle to extract *outp* information, the "*Output Port X match*" and "*Output Port X Busy"* signals have two additional levels of logic (2->1 Mux and 4 input OR) compared to synchronous case. This is shown below in Figure 3. This would make generating "*Output Port X Busy"* more timing critical and could limit the maximum operating frequency.

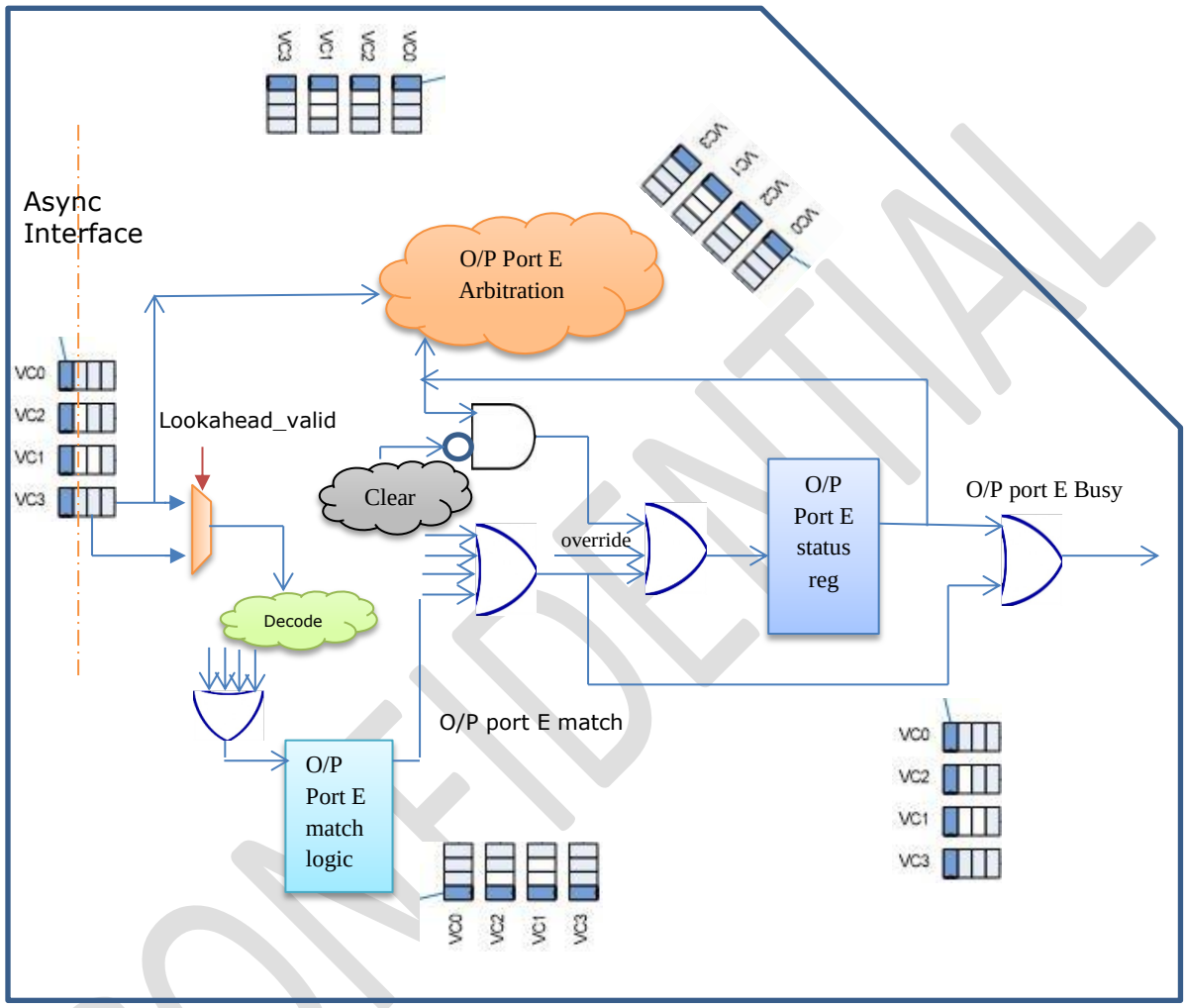

Figure 3 Asynchronous interface: Generating Busy signal from neighboring router (Router B) with outputs (flits) un-registered

2.2.2 Interface busy signal from neighboring router with registered flits outputs

The scheme is similar to one described in Section 2.2.1. The only difference is that the outgoing flits are registered whereas *Output Port X Busy* is not registered.

2.2.2.1 Neighboring router with Synchronous Fifo inputs

So there is no penalty for cold start condition. The one cycle penalty of sending the *Output Port X Busy* ahead of the flits is hidden by registering the flits at the output and effectively delaying the flits by a cycle.

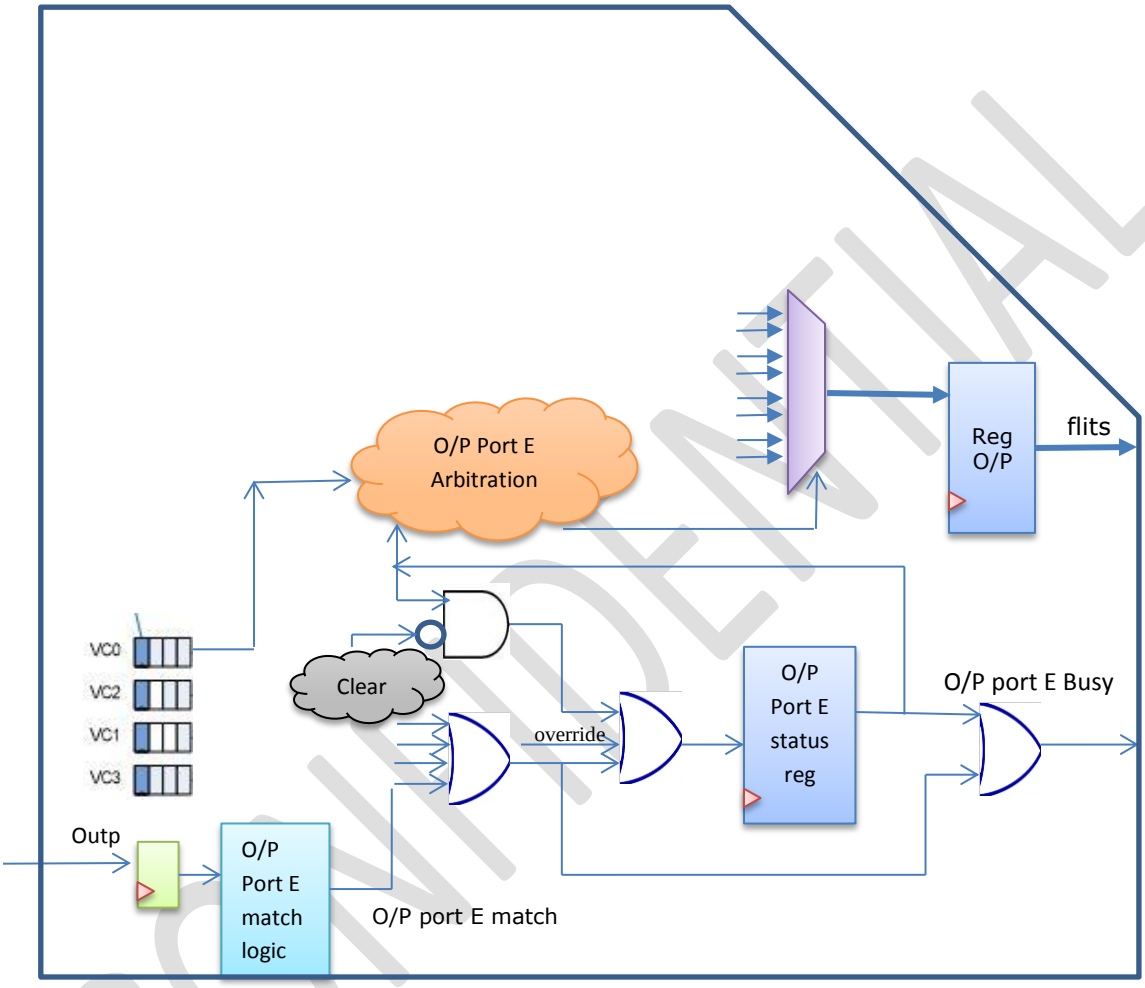

Figure 4 Synchronous Interface : Busy signal and flits from neighboring router (Router B) with O/P registered

2.2.2.2 Neighboring router with Asynchronous Fifo inputs

This is similar to the Asynchronous interface case described in 2.2.1.b There would be no one cycle penalty for cold start as well.

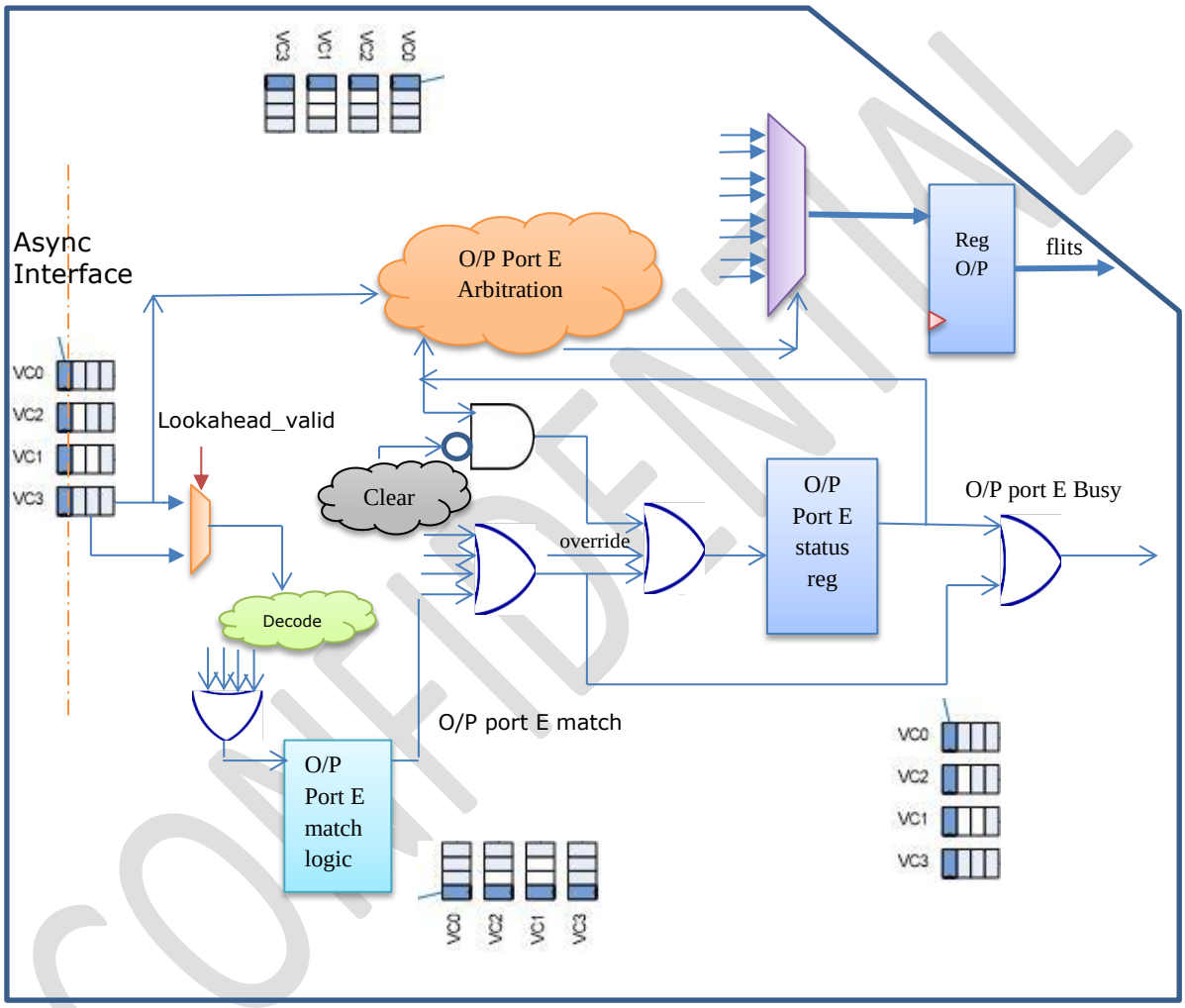

Figure 5 Asynchronous Interface: Busy signal and flits from neighboring router (Router B) with O/P registered

2.2.3 Interface busy signal from neighboring router with unregistered flits outputs, followed by pipeline stages

Router B is not O/P registered and there are n pipeline stages in between. *Output Port X Busy* signal needs be pipelined for *n* stages, before it hits Router A. There is a one cycle penalty for cold start as in Section 2.2.1. The *pipeline registers for flits* would be coarse grained clock gated too. The *pipeline registers for flits* would be clock enabled one cycle ahead of the actual flits in a cascaded fashion. For each pipeline stage, the local clock enable for the flit registers would be generated from registered "*Output Port X Busy*" for that stage. Each Pipeline stage will have its own local Coarse Clock gating control logic.

2.2.4 Interface busy signal from neighboring router with unregistered outputs for flits, followed by pipeline stages

The scheme is same as Section 2.2.3. The only difference is that Router B has registered outputs. There would be no penalty for cold starts.

2.2.5 Interface busy signal from attached Streaming Bridge with unregistered outputs

This scheme is similar to the one described in Section 2.2.1. The difference is that instead of directional output ports like routers, the Streaming Bridge has output ports per NoC Layer, through which it sends traffic. On each NoC layer through which it sends traffic, the streaming bridge is attached to the host port of a router. A dedicated busy/idle signal is required to be sent to each one of these routers for coarse clock gating purpose. Thus the busy signal is going to be one per NoC layer. The busy signal can be set, once the outgoing layer information corresponding to an incoming message beat is obtained. The outgoing layer information is obtained through look up of a mapping table. The look up is based on Source Interface and *QoS value, Destination Interface Id, Destination Host port Id signals* provided by the host through that particular Source Interface. In case of router, the incoming "outp" signal is registered, decoded and matched to a particular Output Port X. In case of Streaming Bridge, the incoming *QoS value, Destination Interface Id, Destination Host port Id signals* (per host interface) are required to be registered and subsequently used as key to the mapping table. If the output of the look up matches to a particular layer, busy signal for that particular layer will be set.

Currently, there is one mapping table per Streaming Bridge. Each of the four interface logic does a parallel look up in the mapping table corresponding to the entry at the head of its input FIFO~~. This additional look up for generating a busy signal per NoC layer will require a duplicate mapping table. The duplicate mapping table can be concurrently used by all four interfaces to do early look up. This will avoid aggravating the timing path for per NoC layer arbitration logic~~. Unlike router case, there is no plan to implement an early look-up for an incoming flit. An early look up in case of a Steaming Bridge involves instantiating a duplicate mapping table. This has area penalty and potential timing implication.

Similar to the case in Section 2.2.1, a "*Outgoing Layer X Match"* signal will be generated for each Noc Layer. The "*Outgoing Layer X Match"* is set by the message entry at the head of the input interface Fifo, doing a lookup of the mapping table and the look up output matching a particular Noc Layer X. An "*Outgoing Layer X Status"* is maintained for each Noc Layer and is set high a cycle after "*Outgoing Layer X Match"* is asserted. "*Outgoing Layer X Busy"* signal is OR function of "*Outgoing Layer X Match"* and "*Outgoing Layer X Status".*

The register "*Output Layer X Status*" is required to let a message (at the head of the interface FIFO) know that corresponding host port and logic in Router on Noc Layer X is already clock enabled. A message can only arbitrate for Output Layer X in Streaming Bridge, if "*Output Layer X Status*" is set to high. After an incoming message beat to Streaming Bridge gets written into FIFO and happens to be in bypass condition, it cannot arbitrate in the same cycle for the *Output Layer X* if "*Output Layer X Status*" has **not** been set to high. That cycle will be utilized in sending "*Output Layer X Busy*" to its attached Router on NoC Layer X, and to wake up and clock enable its corresponding input block in next cycle. This is the **cold start condition** of sending "*Output Layer X Busy*" (clock enable) signal to its attached Router on NoC Layer X, ahead of the message beat and it incurs a penalty of 1 cycle. In rest of the cases, "*Output Layer X Status*" will have already been set high due to message beats from other interface ports contending for the same *Output Layer X*. Only the first message beat setting up "*Output Layer X Status*" high for the first time, will incur 1 cycle penalty.

The "*Output Layer X Status*" register will be cleared after a fixed number of cycles after all the transactions for "Output Layer X" have been exhausted and the credits for all transactions through "Output Layer X" have been returned from its attached Router on NoC layer X. A counter can be maintained to keep track number of outstanding message beats to be switched to "Output Layer X". There will be four "*Outgoing Layer X Match"* signals inside the Streaming bridge, corresponding to four host interface a,b,c and d. Every time, each one of these four signals goes high, the counter is incremented. In case all the four signals go high in a given cycle, the counter needs to be incremented by 4. The counter is decremented every time a message beat switches to "Output Layer X". A second counter "*Output Layer X power down counter*" is maintained inside the Streaming Bridge. Once the counter reaches zero, a fixed number of cycles can be counted before de-asserting "*Output Layer X Status*". This is under the condition, that in the intervening period, there are no incoming message beats destined for "Output Layer X". If in fact the Streaming Bridge sees an incoming message beat bound for "Output Layer X", the "*Output Layer X power down counter*" is reset to zero.

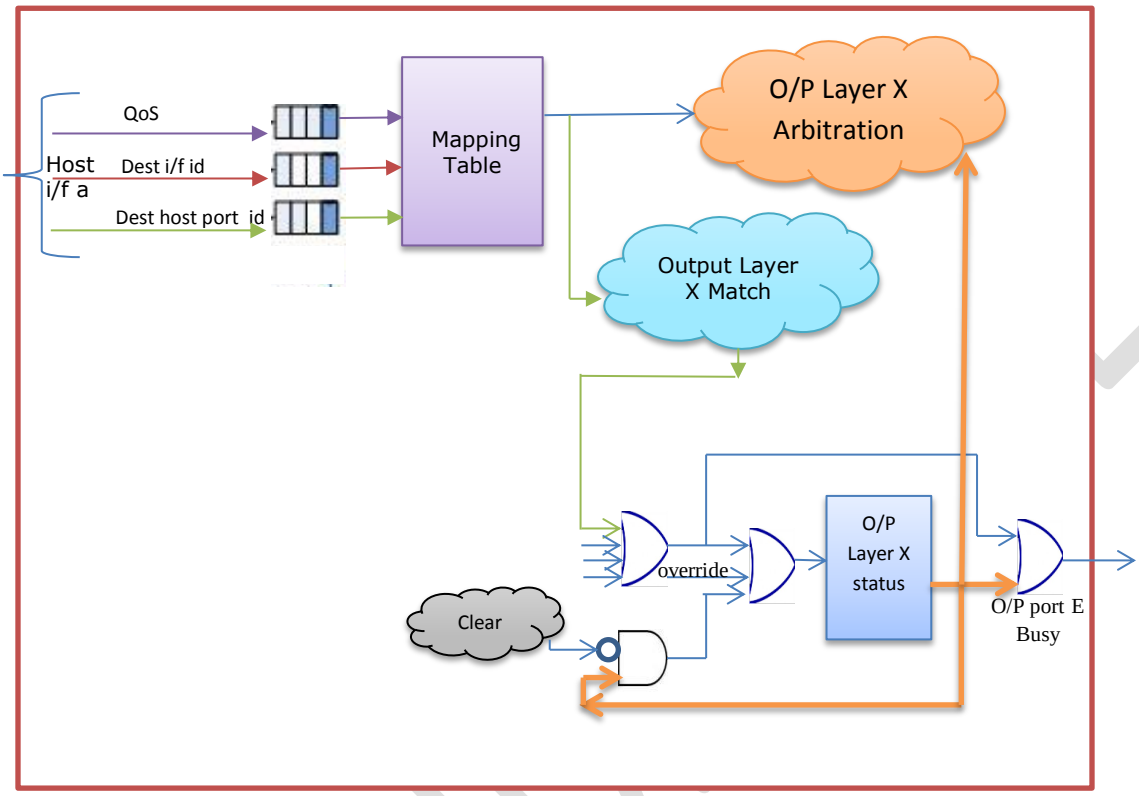

Figure 6 Busy signal generation from Streaming Bridge to its attached router

2.2.6 Interface busy signal from attached Streaming Bridge with registered flits outputs

The scheme is same as described in Section 2.2.6. The only difference is that the outgoing flits are registered whereas *Output Layer X Busy* is not registered. So there is no penalty for cold start condition. The one cycle penalty of sending the *Output Layer X Busy* ahead of the flits is hidden by registering the flits at the output and effectively delaying the flits by a cycle.

2.2.7 Interface busy signal from attached Streaming Bridge with unregistered flits outputs, followed by pipeline stages

Streaming Bridge is not O/P registered and there are n pipeline stages in between. *Output Layer X Busy* signal needs be pipelined for *n* stages, before it hits Router A on NoC Layer X. There is a one cycle penalty for cold start as in Section 2.2.5. The *pipeline registers for flits*

would be coarse grained clock gated too. The *pipeline registers for flits* would be clock enabled one cycle ahead of the actual flits in a cascaded fashion. For each pipeline stage, the local clock enable for the flit registers would be generated from registered "*Output Layer X Busy*" for that stage. Each Pipeline stage will have its own local Coarse Clock gating control logic.

2.2.8 Interface busy signal from attached Streaming Bridge with registered flits outputs, followed by pipeline stages

The scheme is same as Section 2.2.7. The only difference is that Streaming Bridge has registered outputs. There would be no penalty for cold starts.

2.2.9 Interface busy signal from attached AXI Master Bridge with unregistered flits outputs

Similar to Streaming Bridge, an *Output Layer X Busy* signal will be maintained for each layer and will be sent out to the attached routers on each Layer. On the input side on Master Interface, AXI Master Bridge has two channels e.g. Read Address Channel, Write Address/Write Data Channel. A general principle can be adopted that if both ARVALID and WVALID are low for specific number of cycles and there are no internal transactions pending inside AXI Master Bridge for an attached Router (on a specific NOC Layer X), *Output Layer X Busy* signal will be de-asserted *and* that particular Router can be shut off.

2.2.9.1 Synchronous interface with AXI Master

The input (AXI master input commands) to output (NoC flits) latencies, are different for AXI Read and Write commands within AXI Master Bridge. Hence the timing behavior, in terms of asserting *Output Layer X Busy* signal high and subsequently transmitting the flits are different. So AXI Read and Write cases are being discussed separately as below.

a. **Interface busy signal for AXI Read**: The AXI Read command signals from AXI Master are not input registered. For AXI Bridges having un-registered outputs, the SOP to NoC is sent out in the same cycle as the AXI Read Input commands if the arbitration for that specific NoC layer is won by the Read Command. The request for arbitration for access to Noc Layer X is set up after the AXI Read Address is decoded and look up of mapping table is done. For **cold start case**, where *Output Layer X Status* has not been set for the requested Noc X layer, arbitration and hence access to that NoC layer is stalled in the cycle the Read Request is received. The AXI Read commands can be stored in the already existing holding flop stations. Meanwhile that cycle can be utilized to assert *Output Layer X Busy* high and wake up the target router on NoC Layer X. Similar to Streaming Bridge case, the *Output Layer X Status* would be set high in the next cycle and the Read Command from previous cycle can arbitrate and access the target router on Layer X. For **cold start**, there will be a penalty of 1 cycle.

b. **Interface busy signal for AXI Write:** The AXI Write command signals from AXI Master are not input registered as well. The address look up mechanism to find out outgoing Noc Layer and routing information is same as AXI Read command. But corresponding SOP is not sent out to the target Noc Layer till the third cycle. So there is no penalty incurred even for **cold start case**.

The condition to de-assert *Output Layer X Status* is same as Streaming Bridge case as explained in Section 2.2.5.

2.2.9.2 Asynchronous interface with AXI Master

The coarse clock gating scheme would be similar to the Synchronous case as described in 2.2.9.1 The difference is the AXI Read command signals would be read from the head of the Input Async Fifo instead of un-registered inputs from AXI Master. Just as in synchronous case, there would be one cycle penalty in Reads and no penalty in AXI Writes, for cold start case.

2.2.10 Interface busy signal from attached AXI Master Bridge with registered flits outputs

a) **Interface busy signal for AXI Read**: The scheme is same as AXI Read case in Section 2.2.9. But since the output of the AXI Master Bridge is registered, arbitration does not need to be stalled for 1 cycle in cold start case. The flits are automatically delayed by 1 cycle due to output registering of the flits. As in case of Streaming Bridge, the *Output Layer X Busy* is not registered at AXI Master Bridge output.
b) **Interface busy signal for AXI Write:** The scheme is same as AXI Write case in Section 2.2.9. No penalty is incurred for cold start case.

2.2.11 Interface busy signal from attached AXI Master Bridge with unregistered flits outputs, followed by pipeline stages

AXI Master Bridge is not O/P registered and there are n pipeline stages in between. *Output Layer X Busy* signal needs be pipelined for *n* stages, before it hits Router A on NoC Layer X. There is a one cycle penalty for cold start as in Section 2.2.7. The *pipeline registers for flits* would be coarse grained clock gated too. The *pipeline registers for flits* would be clock enabled one cycle ahead of the actual flits in a cascaded fashion. For each pipeline stage, the local clock enable for the flit registers would be generated from registered "*Output Layer X Busy*" for that stage. Each Pipeline stage will have its own local Coarse Clock gating control logic

2.2.12 Interface busy signal from attached AXI Master Bridge with registered flits outputs, followed by pipeline stages

The scheme is same as Section 2.2.8. The only difference is that Streaming Bridge has registered outputs. There would be no penalty for cold starts.

2.2.13 Interface busy signal from attached AXI Slave Bridge with un-registered flits outputs

Similar to AXI Master Bridge, an *Output Layer X Busy* signal will be maintained for each layer and will be sent out to the attached routers on each Layer. On the input side from Slave Interface, AXI Slave Bridge has two channels e.g. Read Response Channel and B Response Channel. A general principle can be adopted that if both RVALID and BVALID are low for specific number of cycles and there are no internal transactions pending inside AXI Slave Bridge for an attached Router (on a specific NOC Layer X), *Output Layer X Busy* signal will be de-asserted and that particular Router can be shut off.

2.2.13.1 Synchronous Interface with AXI Slave

In synchronous case, the inputs from AXI Slave are not registered for both Read Response and B Response. The input (AXI Slave input Responses) to output (NoC flits) latencies, are different for AXI Read Responses and B Responses within AXI Slave Bridge. Hence the timing behavior, in terms of asserting Output Layer X Busy signal high and subsequently transmitting the flits are different. So AXI Slave Read Response and B Response cases are being discussed separately as below.

a) **Interface busy signal from AXI Slave Read Response :**
   AXI Slave Read Response is 1 cycle without De-interleaver and 2 cycles with De-interleaver. The first cycle is always address look up. There will be one cycle penalty for Read Response in De-interleaver case, if corresponding *Output Layer X Status* has already not been set. For Read Response with De-interleaver, the 1 cycle penalty can be hidden behind the 2 cycle latency.

b) **Interface busy signal from AXI Slave B Response:**
   AXI Slave B Response is 1 cycle. There will be one cycle penalty for B Response if corresponding *Output Layer X Status* has already not been set.

2.2.13.2 Asynchronous Interface with AXI Slave

The coarse clock gating scheme would be similar to the Synchronous case as described in 2.2.13.1 The difference is the AXI Read Response signals would be read from the head of the Input Async Fifo instead of un-registered inputs from AXI Slave.

2.2.14 Interface busy signal from RegBus Slave

The Regbus Slave chain protocol requires that Cmd and Data packets be sent in back to back cycles. For this reason, the RegBus Slave Bridge does not initiate transaction to the

Slave Ring until both SOP and EOP have been received. But the "*Clock Enable*" signal could be sent out on Slave Chain as soon as SOP is received at RegBus Slave Bridge (from the source router) on RegBus Layer. The Routers on the Slave chain would get time at least a clock cycle ahead to enable it's control and status register logic. An additional signal "Slave Chain Busy" could be sent out on the Slave Chain to coarse clock gate enable or disable CSR logic in each Router. The mechanism to clear "Slave Chain Busy" would be same as in case of Routers and Bridges. The control logic to set "Slave Chain Busy" would reside in Regbus Slave Bridge.

3 Coarse clock gating of Streaming Bridge

3.1 Transmit Block clock gating

The coarse clock gating in Streaming Bridge could be done on per input interface (a,b,c,d) basis. Once the input FIFOs for a particular interface are empty, they can be clock gated off, if there is no activity on that input interface from the host side for programmable number of cycles. The counter (per interface) should only start running after its input FIFOs have drained out and all associated credits returned to the host from that interface. The Streaming Bridge will wake up and clock-enable that input interface block after it receives the first *hst_strtxbrdg_dbeat_vld* signal from the host. The wake up signal for the FIFOs are required one cycle ahead, before the actual message beats are written into the FIFOs. For this purpose, a conditional register with free running clock, is used to capture the first message beats. In the same cycle, the input FIFO is clock enabled. This is illustrated in Figure 7

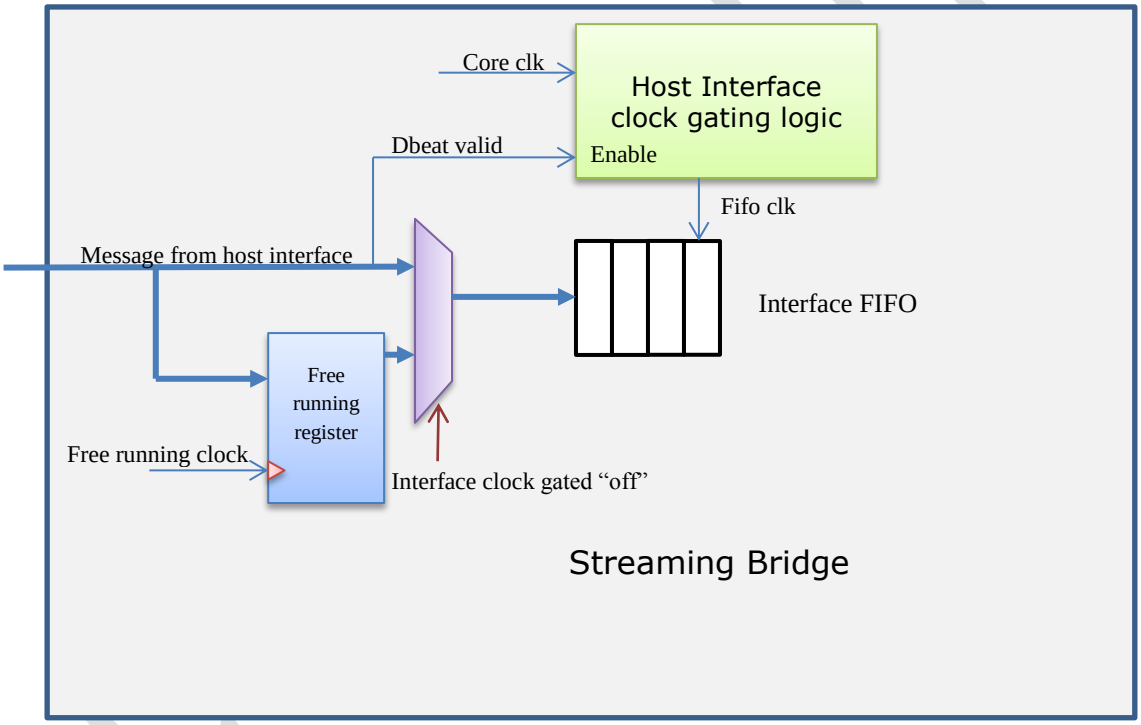

Figure 7 Streaming Bridge conditional register to capture the first message beats from host interface in clock gated mode

The whole Transmit block can be clock gated off, once all the host interface logic has been clock gated off and all the credits from the NoC side have been returned.

3.2 Receive Block clock gating

The Streaming Bridge Receive block (traffic from NoC) could be clock gated off based on traffic condition on a layer by layer basis. This scheme is slightly different than clock gating scheme for Transmit Block. For each NoC layer, there will be associated coarse clock gating logic. Once the input VC Fifos for a particular NoC layer have been drained, the clock gating logic will wait for the "*Input Layer X Busy*" signal from its attached router on that layer to be de-asserted (as described in Section 2). Once the condition is reached that the input VC Fifos for that NoC layer X are empty (credits returned back to the router) and "*Input Layer X Busy*" signal have been de-asserted, the associated logic could be clock gated off.

The whole Receive Block can be clock gated off, once all the input logic for all NoC Layers have been clock gated and all credits have been returned from the host to the Streaming Bridge.

4 Coarse clock gating of AXI Master Bridge

4.1 Transmit Block Clock Gating: AR and AW channels

The coarse clock gating in AXI Master Bridge could be done on per input interface basis eg Read (AR) and Write command (AW) channels with AXI Master. If there is no activity on a particular input channel from the master side for programmable number of cycles, the associated logic can be turned off. The associated counter should only start running only if there are no outstanding Read/Write commands inside the AXI Master Bridge and all dependencies on R and B channels have been resolved. For AXI Master Bridge with asynchronous interface with AXI Master, additional qualification is required that the input Async Fifo for that channel is empty. The AXI Master Bridge already has AREADY and WREADY signals to the host side. These will be de-asserted when those input channel interfaces need to be clock gated off due to prolonged non-activity. The AXI Master Bridge will wake up and clock-enable the input channel interface blocks after it receives AVALID and WVALID signals from the host.

4.2 Receive Block Clock Gating

4.2.1 Noc Rx VC Fifos

The AXI Master Receive block (traffic from NoC) could be clock gated off based on traffic condition on a layer by layer basis. For each NoC layer, there will be associated coarse clock gating logic. Once the input Rx VC Buffers for a particular NoC layer have been drained, the clock gating logic will wait for the "Input Layer X Busy" signal from its attached router on that layer to be de-asserted (as described in Section 2). Once the condition is reached that the input Rx VC Buffers for that NoC layer X are empty (credits returned back to the router) and "*Input Layer X Busy*" signal have been de-asserted, the associated logic could be clock gated off. The whole Receive Block can be clock gated off, once all the input logic for all NoC Layers have been clock gated and all credits have been returned from the Master to the AXI Master Bridge.

4.2.2 R and B channels

The R and B Response channels could be clock gated off once Noc Rx VC Fifos are empty and there are no outstanding transactions within the response channels. There are wide data paths for de-framing, width conversion and Response re-ordering logic inside the Response channels. Significant power could be saved by clock gating the Response channels if they are idle over prolonged period of time. In case of asynchronous interface with the AXI Master, additional qualification is required that the "AXI to Noc" clock async decoupling FIFOs are empty before clock gating the Response channels off.

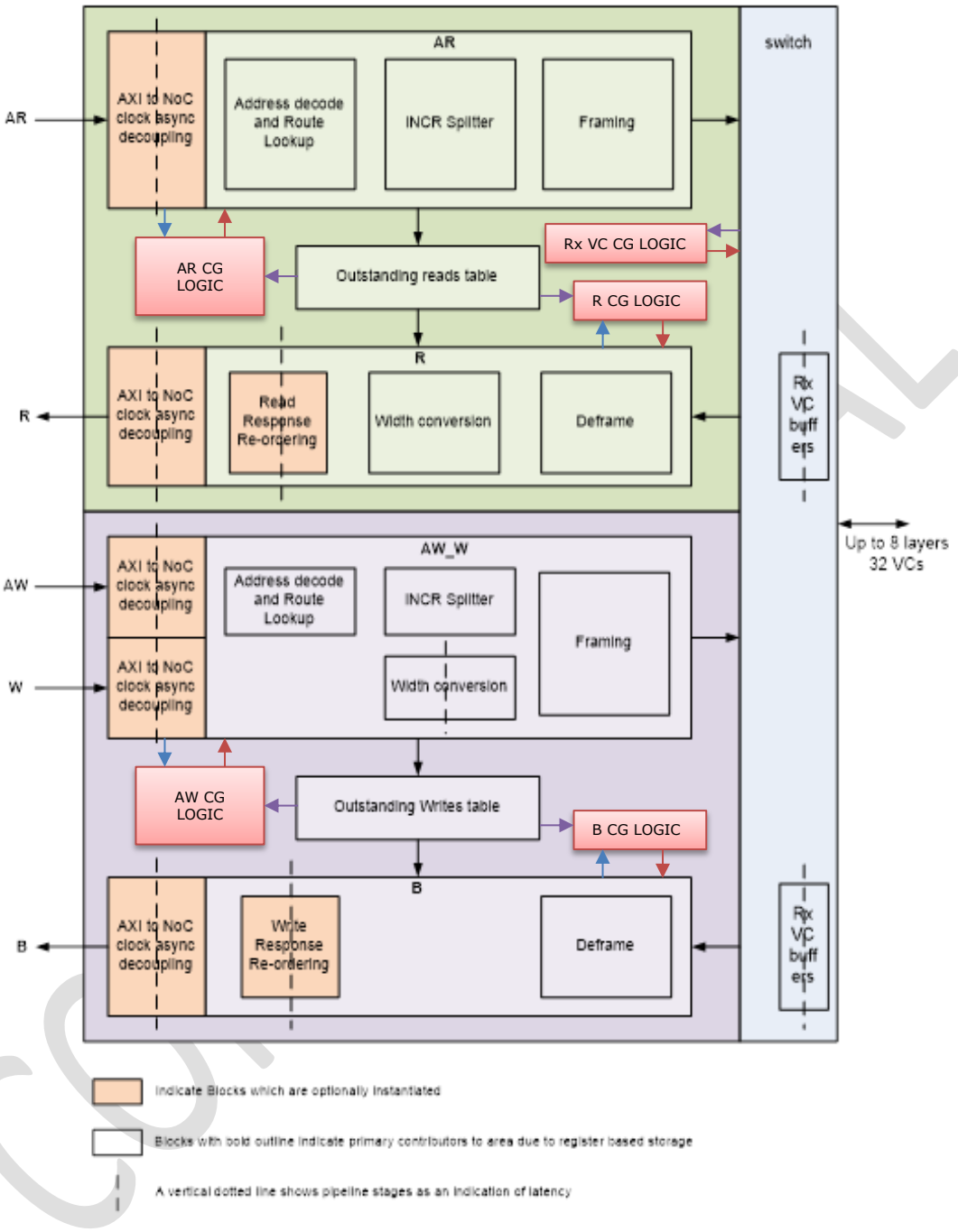

Figure 8 AXI Master coarse clock gating scheme

5 Coarse clock gating of AXI Slave Bridge

This is similar and identical to the scheme described for AXI Master Bridge in Section 4.

5.1 Transmit Block Clock Gating: R and B channels

This is similar to the scheme in Section 4.1 except that input channels for Transmit block are Read Response and B Response. The clock gating scheme is same.

5.2 Receive Block Clock Gating

5.2.1 Noc Rx VC Fifos

This is also similar to the scheme in Section 4.2.1. Clock Gating would be done on per NoC Layer basis. The per Noc Layer input interface logic would be clock gated off when the Rx VC buffers are empty and "*Input Layer X Busy*" signal have been de-asserted from the attached router on that NoC layer.

5.2.2 AR and AW channels

This is similar to the scheme in Section 4.2.2 The AR and AW channels could be clock gated off once Noc Rx VC Fifos are empty and there are no outstanding transactions within AR and AW channels. There are wide data paths for de-framing and width conversion inside the these channels. Significant power could be saved by clock gating the channels if they are idle over prolonged period of time. In case of asynchronous interface with the AXI Slave, additional qualification is required that the "AXI to Noc" clock async decoupling FIFOs are empty before clock gating each of AR and AW channels.

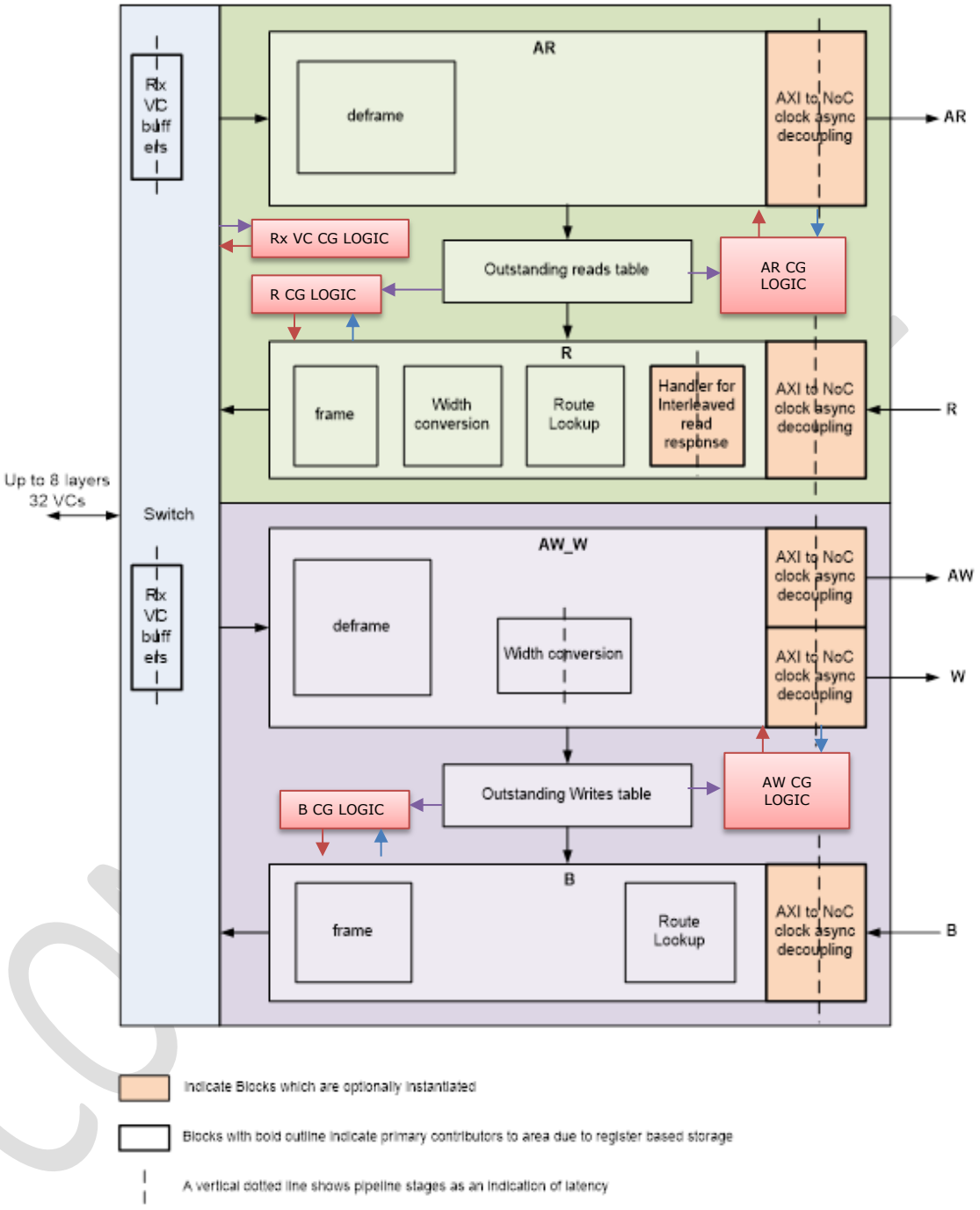

Figure 9 AXI Slave Bridge coarse clock gating scheme

6 RegBus Layer coarse clock gating considerations

The RegBus layer and all elements on RegBus Layer will not be coarse grained clock gated. All NoC elements on RegBus layer would be free running. Regbus layer may also run on a different clock frequency from the rest of the Noc, so synchronizers are recommended at the noc-elements before using any signals on RegBus Slave chain.

7 System Clock Enable and System Coarse Clock Gate Override signal

Each NoC element has a clock input pin, which corresponds to the root of the clock distribution within the NoC element. System Level clock gating allows a "*System_clk_en*" pin to exist for each clock input on a bridge or router. Also, there would be a system level Clock Gate Override signal provided to each NoC element. Each NoC element would have a *System_cg_or* (*System_override*) pin.

There are two options on how to drive the "*System_clk_en*" and "*System_cg_or*" for each NoC element.

- In absence of RegBus Layer, they would be driven by dedicated input pins from NoC.
- In presence of RegBus Layer, they would be driven by RegBus Slave Bridge. The values on these signals will depend on the value of dedicated registers programmed through RegBus Layer.

7.1 System Level Clock Gating and Override in absence of RegBus Layer

1) When a NoC does not have a RegBus Layer, the clock enable signal for the NoC element is going to be generated and controlled by a system clock controller external to the NoC. There are going to be two input pins *System_noc_clk_en* and *System_reg_clk_en* at the NoC top level. *System_noc_clk_en* is going to be connected to "*System_clk_en*" pin of NoC elements on non-Regbus layer. *System_reg_clk_en* is going to be connected to "*System_clk_en*" pin of NoC elements on RegBus Layer.
2) Similarly, there are going to be two input pins *System_noc_cg_or* and *System_reg_cg_or* to the NoC for Clock Gate override purpose. They are going to be generated and controlled by a system clock controller external to the NoC. *System_noc_cg_or* is going to be connected to "*System_cg_or*" pin of NoC elements on non-Regbus layer. *System_reg_cg_or* is going to be connected to "*System_cg_or*" pin of NoC elements on RegBus Layer.

7.2 System Level Clock Gating and Override through RegBus Layer

When a NoC has a Regbus Layer, the enable signal and clock gate override signal for each NoC element is going to be controlled by the RegBus Slave Bridge on that node. In this case, granularity is provided to control *system_clk_en* and *system_cg_or* pins of each NoC element individually. This is done by programming "system clock gating" dedicated registers in RegBus slave bridge. The Regbus slave bridge on each node contains 16 registers, each mapping to one of 9 Routers and 6 Bridges (N,E,W,S,H and Regbus Slave Bridge) on that node. Each register contains 2 R/W bits, one bit for *System_clk_en* and the other for *System_cg_or* for that particular NoC element. These

register outputs will drive output pins from RegBus slave Bridge and connect to *System_clk_en* and *System_cg_or* pins of the appropriate NoC elements on that node. 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 this scheme. NocStudio should also dump out information for the user mapping the address of the "System clock gate control register for Bridges" to actual Bridge ID. This is required so that user knows which register in the RegBus Slave Bridge should be written to so as to control the *System_clk_en* and *System_cg_or* pins of a certain Bridge. Following are the 16 registers provided in the RegBus slave bridge for controlling *system_clk_en* and *system_cg_or* pins of each NoC element on a particular node. Additional address space in RegBus Slave Bridge should be reserved for 4 more Bridges to be introduced in the future.

Table 1 System clock gate control register (RBSLVCG) for a Slave Agent, physically located in RegBus slave Bridge (RegBus Ring Master)

31 ... 2	1	0
	Sys_Cg_or for Slave Agent	Sys_Clk_En for Slave Agent

There are 32 such registers physically residing in RegBus Slave Bridge, to support and control clock gating for a maximum of 32 Slave Agents on the Regbus Slave Ring on a particular node.

7.3 Turning off System Clock Enable for a node

When the *System_Clk_en* for a node on the NoC is turned off, an interrupt is sent to the Regbus Slave Bridges on adjacent nodes. The interrupt will be conveyed on RegBus Layer, through point to point signals between two RegBus Slave Bridges on adjacent nodes.

Once the interrupt is received by a RegBus Slave Bridge, it will convey that information to all its NoC elements (routers) on that node through Regbus Slave chain. This is to let the routers on neighboring nodes (on non-regbus NoC layers) know that the link to disabled node is not valid. Hence those routers should not forward any traffic to the disabled node. An additional set of signals are required for this purpose which will go as side band signals on Slave chain.

8 Latency Penalty on Coarse Clock Enable

The following table summarizes latency incurred at different NoC elements to wake up their interface neighbors, from clock gated mode to clock enabled mode.

Table 2 Coarse Clock Gating Latency in different NoC elements as interface neighbors

#	Neighboring element	Neighboring element flit output registered	Neighboring element Input Async Boundary	Target 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 (?? Should be N)	
4	Router	N	Y	Router	Y	
5	Str Bridge	Y	N	Router	N	
6	Str Bridge	N	N	Router	Y	
7	Str Bridge	Y	Y (not supported)	Router	N	
8	Str Bridge	N	Y (not supported)	Router	Y	
9	Router	Y	N	Str Bridge	N	
10	Router	N	N	Str Bridge	Y	
11	Host	n/a	N	Str Bridge	N	Free running input register on bridge when clk-gated
12	AXI Br 'AR'	N/Y	N/Y	Router	Y	AR channel must de-assert 'ready' when neighbour is clock gated
13	AXI Br 'AR'	Y	N/Y	Router	Y	AR channel must de-assert 'ready' when neighbour is clock gated
14	AXI Br AW,W	Y/N	Y/N	Router	N	AW,W takes 2 cycles within bridge
15	Router	Y	N	AXI 'B'	N	
16	Router	N	N/Y	AXI 'B'	Y	
17	Router	Y	N	AXI 'R'	N	
18	Router	N	N/Y	AXI 'R'	Y	
19	Regbus Br	n/a	N	Router/ SBridge/ AXi Br	N	

9 Implementation #1: Instantiation of clock gating module inside NoC element wrapper.

Based on customer preference, there would be two flavors of implementation for coarse clock gating. In both modes of implementation, a native clock module *u_ns_<element>_cg* would be instantiated inside each NoC element wrapper. In the first mode of implementation, the *u_ns_<element>_cg* module would provide gated clocks to the NoC element inside the wrapper as well generate *Interface Busy* signals for NoC elements on adjacent nodes, based on condition of signals from the NoC element. Inside the NoC element wrapper, the *u_ns_<Noc_element>_cg* and *u_ns_<Noc_element>* modules would be instantiated at the same hierarchy. In addition to new input/output pins (listed in this Section) at Noc element wrapper level, some additional input and output ports needs to be created at u_ns_<Noc_element> module level. These signals will communicate with *u_ns_<element>_cg* to enable/disable clock gating for u_ns_<Noc_element> and also to convey busy/idle status to its neighboring NoC element.

9.1 Implementation #1: Router Clock Gate module instantiation and hierarchy

The following diagram illustrates where the clock gate module u_ns_router_cg will be instantiated inside the Router wrapper and the corresponding signal interactions.

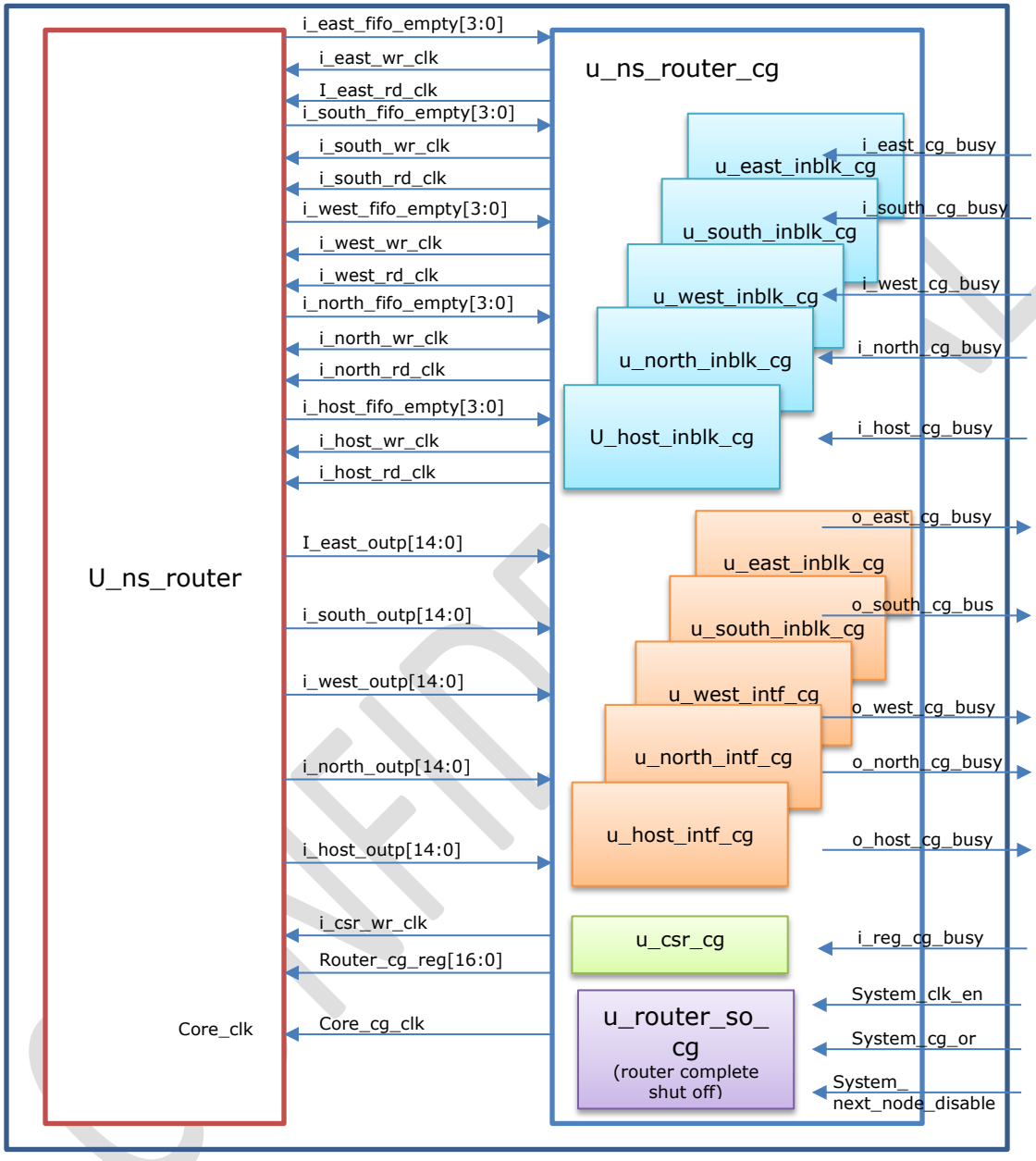

Figure 10 Implementation #1: Router hierarchy for clock gating

9.1.1 Implementation #1: Router clock gating module interface signals

Table 3 Implementation #1: Router Clock gate module Interface signals

Signal Name	Width	Direction	Description
System Signals (Wrapper I/O)			
System_noc_clk_en	1	Input	System Clock Enable from NoC IO
System_noc_cg_or	1	Input	System Clock Gate Override from NoC IO
System_reg_clk_en	1	Input	System Clock Enable from Regbus register
System_reg_cg_or	1	Input	System Clock Gate Override from Regbus Register
System_next_node_disabled	4	Input	Next node is disabled (from RegBus Slave Chain) 0 : Node on "North" disabled 1 : Node on "East" disabled 2 : Node on "West" disabled 3 : Node on "South" disabled
Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
Scan_mode	1	Input	Scan mode pin for clock gate cell
NoC [port] Signals (Wrapper I/O) where [port] is one of {north, east, west, south, host, reg}			
o_[port]_cg_busy	1	Output	Output Port Busy (to neighboring NoC)
i_[port]_cg_busy	1	Input	Input Port Busy (from neighboring NoC)
Router [port] Signals (Wrapper Internal) where [port] is one of {north, east, west, south, host}			
i_[port]_clk	1	Input	Input Write clock (from neighboring router) – **Async Case**
i_[port]_fifo_empty	1	Input	Input [port] VC FIFOs are empty
i_[port]_outp	15	Input	Bit[2:0] – outp at head of VC0 Fifo Bit[5:3] – outp at head of VC1 Fifo Bit[8:6] – outp at head of VC2 Fifo Bit[11:9] – outp at head of VC3 Fifo Bit[14:12] – Registered outp input (**Synchronous case else drive it to zero)**
Router_cg_reg	17	Input	17 bit register read value from clock gate control register (Section 9.1.2)
o_[port]_flits_empty	1	Input	All flits bound for output [port] have been transmitted (no flits outstanding) and all credits returned

			from the destination NoC element
o_[port]_cg_status	1	Output	NoC Element on [port] Status 1: Clock Enabled 0: Clock Disabled
i_[p]_wr_cg_clk	1	Output	Write side gated clk for I/P VC Fifos Use in Async case only : based on *Interface_Busy* condition
i_[p]_rd_cg_clk	1	Output	Read Side gated clk Use in Sync case only : Based on I/P *Fifo empty* condition
i_[p]_cg_clk	1	Output	Sync case only : Based on both *Interface_Busy* signal condition and *Fifo empty* condition
Core_cg_clk	1	Output	Gated clock for arbitration and u_[port]_outblk. An additional port is required in the router for this clock. **Core_clk will still be an input to the router for free running signals.**

Note for NoC Studio : There would be two System clock enable signals at NoC top level - *System_noc_clk_en* and *System_reg_clk_en*.

1) For Non RegBus NoC layers, the *System_clk_en* pin of the NoC element would be connected to *System_noc_clk_en* signal
2) For RegBus NoC layer, the *System_clk_en* pin of the NoC element would be connected to *System_reg_clk_en* signal.

Similarly there would be two System Clock Gating override signals at NoC top level - *System_noc_cg_or* and *System_reg_cg_or*

1) For Non RegBus NoC layers, the *System_cg_or* pin of the NoC element would be connected to *System_noc_cg_or* signal
2) For RegBus NoC layer, the *System_cg_or* pin of the NoC element would be connected to *System_reg_cg_or* signal.

Table 4 Implementation #1 : New Router module pins to be added (NoC Studio to add hooks)

Signal Name	Width	Direction	Description
System Signals			
System_clk_en	1	Input	System Clock Enable - Non Regbus Layer : to be connected to *System_noc_clk_en* Regbus Layer : to be connected to System_reg_clk_en
System_cg_or	1	Input	System Clock Gate Override - Non Regbus Layer : to be connected to System_noc_cg_or RegBus Layer : to be connected to *System_reg_cg_or*
System_next_node_disabled	4	Input	Next node is disabled (from RegBus Slave Chain) 0 : Node on "North" disabled 1 : Node on "East" disabled 2 : Node on "West" disabled

			3 : Node on “South” disabled
Scan_mode	1	Input	Scan mode override
Router [port] Signals where [port] is one of {north, east, west, south, host, reg}			
o_[port]_cg_busy	1	Output	Output Port Busy (to neighboring NoC)
i_[port]_cg_busy	1	Input	Input Port Busy (from neighboring NoC)

9.1.2 Implementation #1: Router clock gate control registers

This register would reside inside the u_ns_router_csr**. The read logic for this register should always be free-running** (verification should take note of this).

Table 5 Implementation #1: Router clock gate hysteresis count register

31								0
Count Value after which o_[port]_busy is de-asserted								

Table 6 Implementation #1: Router clock gate override register

31							1	0
Reserved								Fast Path override

9.1.3 Implementation #1: Router clock gate module parameters

The following parameter is required by the clock gating module to conditionally instantiate logic for Synchronous/Asynchronous input interface.

Table 7 Implementation #1: Router clock gate module parameter

Parameter Name	Bits	Description
P_[p]_IVCBUF_TYPE	8	4 groups of 2 bits for each VC in the order {3, 2, 1, 0}. Determines the maximum upsize ratio from that VC of the input port 00=> 1x 01=> 2x 10=> 4x 11=> Async mode (1x)
P_[p]_IN_ENB	1	Enable input port [p]. 0=> disabled, 1=> enabled

P_[p]_OUT_ENB	1	Enable output port [p]. 0=> disabled, 1=> enabled
P_[p]_OUTPUT_REG_ENA	1	Output port is registered enabled
P_SYSTEM_CK_EN_VIA_REGBUS	1	1 => System Clock Enable and System Clock Gate Override signals to be provided by the RegBus 0 => System Clock Enable and System Clock Gate Override signals to be provided by NoC IO Pins

Table 8 ~~Implementation #1: New Router parameter to be added (NoC Studio to add hooks)~~

~~P_SYSTEM_CK_EN_VIA_REGBUS~~	~~1~~	~~1 => System Clock Enable and System Clock Gate Override signals to be provided by the RegBus~~ ~~0 => System Clock Enable and System Clock Gate Override signals to be provided by NoC IO Pins~~

9.2 Implementation #1: Streaming Bridge Clock Gate module instantiation and hierarchy

The following diagram illustrates where the clock gate module ns_strbrdg_cg will be instantiated inside the Streaming Bridge wrapper and the corresponding signal interactions.

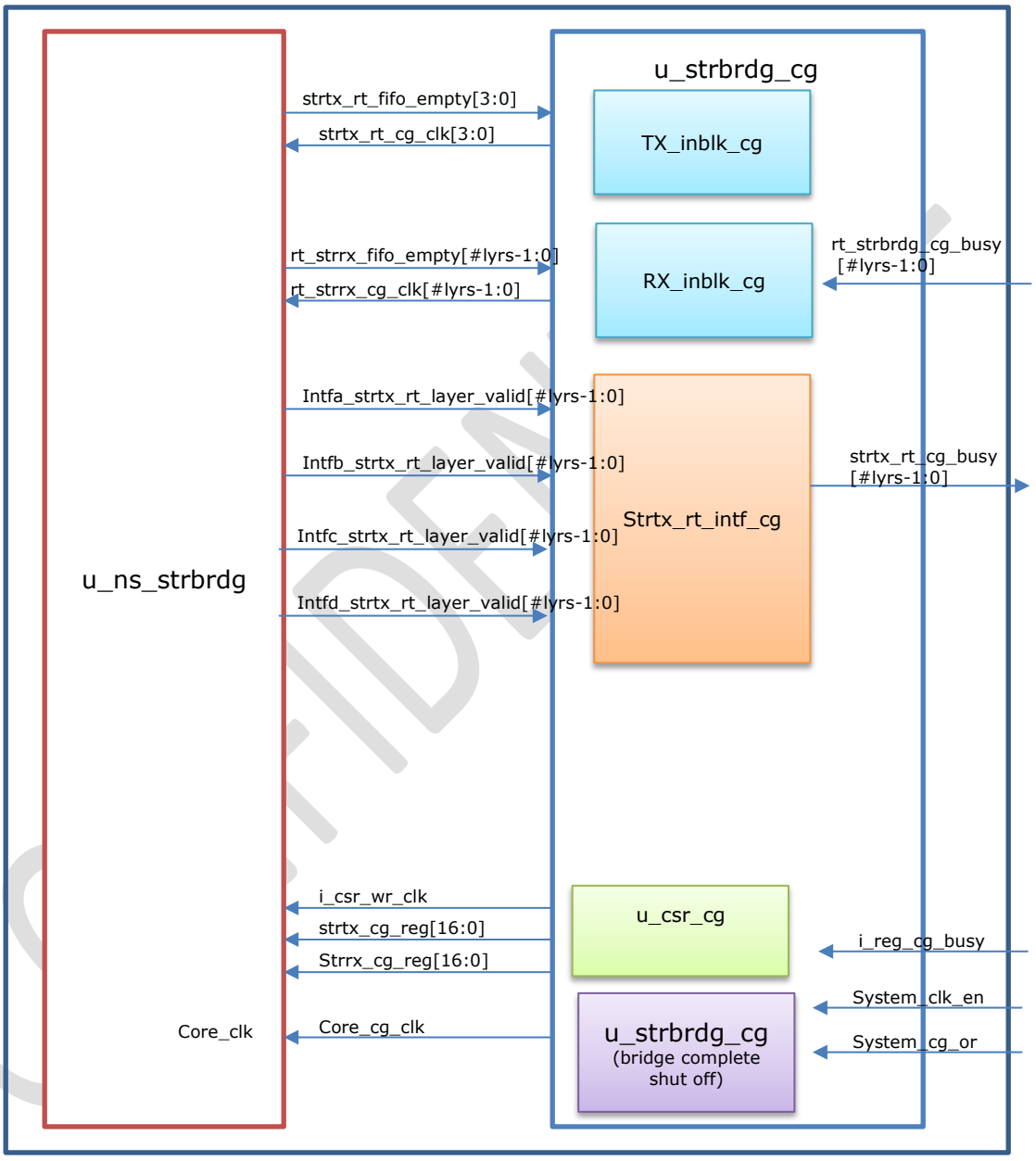

Figure 11 Implementation #1: Streaming Bridge clock gating hierarchy

9.2.1 Implementation #1: Streaming Bridge clock gating module interface

Table 9 Implementation #1: Streaming Bridge clock gating module interface

Signal Name	Width	Direction	Description
System Signals (Wrapper I/O)			
System_noc_clk_en	1	Input	System Clock Enable from NoC IO
System_noc_cg_or	1	Input	System Clock Gate Override from NoC IO
System_reg_clk_en	1	Input	System Clock Enable from RegBus register
System_reg_cg_or	1	Input	System Clock Gate Override from Regbus register
Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
Scan_mode	1	Input	Scan mode pin for clock gate cell
NoC Signals (Wrapper I/O)			
strtx_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC interface)
rt_strrx_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Input Port Busy (from NoC interface)
Streaming Bridge Signals (Wrapper Internal)			
strtx_rt_fifo_empty	[P_STR_TX_HST_INTERFACES-1:0]	Input	Host interface Fifos are empty
Intfa_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from Intf a
Intfb_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from Intf b
Intfc_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from Intf c

Intfd_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from Intf d
rt_strrx_fifo_empty	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	NoC side Input VC fifos are empty
strtx_rt_cg_status	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Router on [layer] Status 1: Clock Enabled 0: Clock Disabled
Strtx_rt_flits_empty	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	All flits bound for output [layer] have been transmitted (no flits outstanding) and all credits returned from the destination NoC element
strtx_rt_cg_clk	[P_STR_TX_HST_INTERFACES-1:0]	Output	Gated Input clock for Transmit Block
rt_strrx_cg_clk	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Gate Input clock for Receive Block
strtxbrdg_cg_reg	17	Input	17 bit register read value from Transmit block clock gate control register (Section 9.2.2)
strrxbrdg_cg_reg	17	Input	17 bit register read value from Receive block clock gate control register (Section 9.2.2)
Core_cg_clk	1	Output	Gated clock for complete shut-down of the Streaming Bridge. **Core_clk will still be an input to the router for some free running signals.**

Note for NoC Studio : There would be two System clock enable signals at NoC top level - *System_noc_clk_en* and *System_reg_clk_en*.

3) For Non RegBus NoC layers, the *System_clk_en* pin of the NoC element would be connected to *System_noc_clk_en* signal
4) For RegBus NoC layer, the *System_clk_en* pin of the NoC element would be connected to *System_reg_clk_en* signal.

Similarly there would be two System Clock Gating override signals at NoC top level - *System_noc_cg_or* and *System_reg_cg_or*

3) For Non RegBus NoC layers, the *System_cg_or* pin of the NoC element would be connected to *System_noc_cg_or* signal
4) For RegBus NoC layer, the *System_cg_or* pin of the NoC element would be connected to *System_reg_cg_or* signal.

Table 10 Implementation #1 : New Streaming Bridge module pins to be added (NoC Studio to add hooks)

Signal Name	Width	Direction	Description
System Signals			
System_clk_en	1	Input	System Clock Enable - Non Regbus Layer : to be connected to *System_noc_clk_en* Regbus Layer : to be connected to System_reg_clk_en
System_cg_or	1	Input	System Clock Gate Override - Non Regbus Layer : to be connected to System_noc_cg_or RegBus Layer : to be connected to *System_reg_cg_or*
Scan_mode	1	Input	Scan mode pin for clock gate cell
NoC Signals			
strtxbrdg_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC interface)
rt_strrxbrdg_cg_busy	[P_STR_RX_RT_NOOFLAYERS-1:0]	Input	Input Port Busy (from NoC interface)

9.2.2 Implementation #1: Streaming Bridge clock gate control registers

This register would reside inside the u_ns_csr**. The read logic for this register should always be free-running** (verification should take note of this).

Table 11 Implementation #1: Streaming Transmit Bridge clock gate hysteresis register

31									0
Count Value after which o_[layer]_busy is de-asserted									

Table 12 Implementation #1: Streaming Transmit Bridge fast path override register

31									0
									Fast path override

Table 13 Implementation #1: Streaming Receive Bridge clock gate hysteresis register

31									0
Count Value after which o_[layer]_busy is de-asserted									

Table 14 Implementation #1: Streaming Receive Bridge fast path override register

31									0
									Fast path override

Table 15 Implementation #1 : Streaming Bridge clock gate module parameter

Parameter Name	Bits	Description
P_STR_TX_OUTREGENB	1	Tx Bridge Output port is registered enabled
P_SYSTEM_CK_EN_VIA_REGBUS	1	1 => System Clock Enable and System Clock Gate Override signals to be provided by the RegBus 0 => System Clock Enable and System Clock Gate Override signals to be provided by NoC IO Pins

Table 16 Implementation #1: New Streaming Bridge parameters to be added (NoC Studio to add hooks)

~~P_SYSTEM_CK_EN_VIA_REGBUS~~	~~1~~	~~1 => System Clock Enable and System Clock Gate Override signals to be provided by the RegBus~~ ~~0 => System Clock Enable and System Clock Gate Override signals to be provided by NoC IO Pins~~

9.3 Implementation #1: AXI Master Bridge Clock Gate module instantiation and hierarchy

The following diagram illustrates where the clock gate module ns_acemstrbrdg_cg will be instantiated inside the AXI Master Bridge (u_aximstrbrdg) wrapper and the corresponding signal interactions

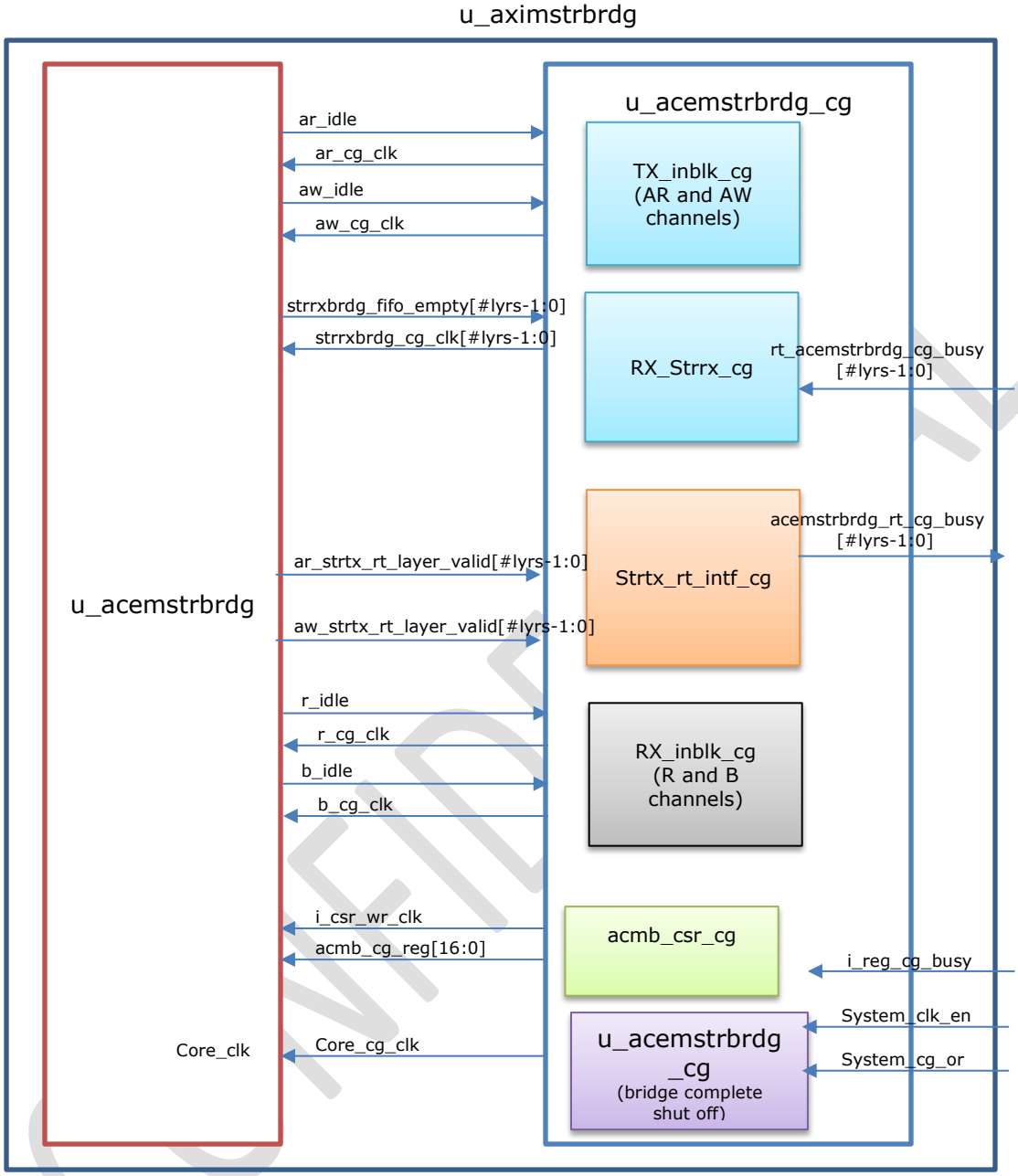

Figure 12 Implementation #1: AXI Master Bridge Clock Gating hierarchy

9.3.1 Implementation #1: AXI Master Bridge clock gating module interface

Table 17 Implementation #1: AXI Master Bridge clock gating module interface

Signal Name	Width	Direction	Description
System Signals (Wrapper I/O)			
System_clk_en	1	Input	System Clock Enable

System_cg_or	1	Input	System Clock Gate Over ride
Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
NoC Signals (Wrapper I/O)			
acemstrbrdg_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC side interface)
rt_acemstrbrdg_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Input Port Busy (from NoC side interface)
AXI Master Bridge signals : AR and AW channels (wrapper internal)			
ar_idle	1	Input	AR channel is idle For ASYNC intf with AXI Master : Additionally Async Fifos are empty
aw_idle	1	Input	AW channel is idle For ASYNC intf with AXI Master : Additionally Async Fifos are empty
ar_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from AR channel
aw_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from AW channel
ar_cg_clk	1	Output	Gated Input clock for AR Block
aw_cg_clk	1	Output	Gated Input clock for for AW Block
acemstrbrdg_rt_cg_status	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Router on [Layer] Status 1: Clock Enabled 0: Clock Disabled
AXI Master Bridge signals : R and B channels (wrapper internal)			
r_idle	1	Input	R channel is idle For ASYNC intf with AXI Master : Additionally Async Fifos are empty
b_idle	1	Input	B channel is idle For ASYNC intf with AXI Master : Additionally

			Async Fifos are empty
r_cg_clk	1	Output	Gated input clock for R channel
b_cg_clk	1	Output	Gated input clock for B channel
AXI Master Bridge signals : CSR (wrapper internal)			
acmb_cg_reg	17	Input	17 bit register read value from AXI Master Bridge clock gate control register (Section 9.2.2)
StrrxBrdg signals (wrapper internal)			
strrxbrdg_fifo_empty	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	NoC side Input VC fifos are empty
strrxbrdg_cg_clk	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Gated Input clock for NoC side Rx VC FIFOs.
Misc signal (wrapper internal)			
Core_cg_clk	1	Output	Gated clock for complete shut-down of the AXI Master Bridge. **Core_clk will still be an input to the router for some free running signals and logic**

9.3.2 Implementation #1: AXI Master Bridge clock gate control registers

This register would reside inside the u_ns_acmb_csr**. The read logic for this register should always be free-running** (verification should take note of this).

Table 18 Implementation #1: AXI Master Bridge clock gate control register

31			17	16	15				0
				Fast Path override	Count Value after which o_[port]_busy is de-asserted				

9.3.3 Implementation #1: AXI Master Bridge clock gating module parameters

Table 19 Implementation #1: AXI Master Bridge clock gate module parameters

Name	Bits	Description
P_ASYNC_MODE	1	Asynchronous input interface with AXI Master

9.4 Implementation #1: AXI Slave Bridge Clock Gate module instantiation and hierarchy

The following diagram illustrates where the clock gate module ns_aceslvbrdg_cg will be instantiated inside the AXI Slave Bridge wrapper (u_axislvbrdg) and the corresponding signal interaction.

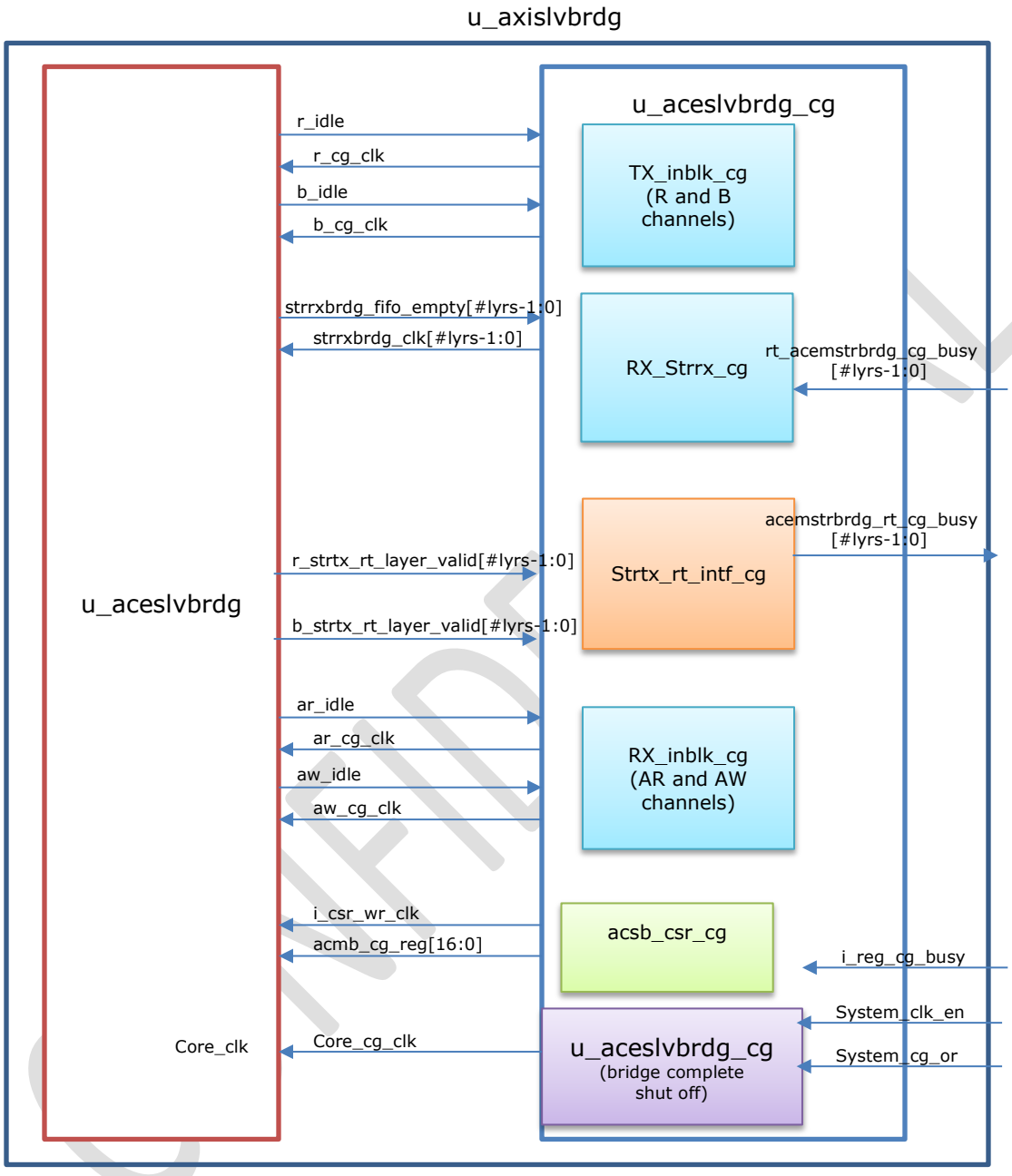

Figure 13 Implementation #1: AXI Slave Bridge clock Gating hierarchy

9.4.1 Implementation #1: AXI Slave Bridge clock gating module interface

Table 20 Implementation #1: AXI Slave Bridge clock gating module interface

Signal Name	Width	Direction	Description
System Signals (Wrapper I/O)			
System_clk_en	1	Input	System Clock Enable
System_cg_or	1	Input	System Clock Gate Over ride
Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
NoC Signals (Wrapper I/O)			
aceslvbrdg_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC side interface)
rt_aceslvbrdg_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Input Port Busy (from NoC side interface)
AXI Slave Bridge : R and B channels (Wrapper Internal)			
r_idle	1	Input	R channel is idle For ASYNC intf with AXI Slave : Additionally Async Fifos are empty
b_idle	1	Input	B channel is empty For ASYNC intf with AXI Slave : Additionally Async Fifos are empty
r_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from R channel
w_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from W channel
r_cg_clk	1	Output	Gated Input clock for R Block
w_cg_clk	1	Output	Gated Input clock for for W Block
aceslvbrdg_rt_cg_status	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Router on [layer] Status 1: Clock Enabled 0: Clock Disabled
AXI Slave Bridge : AR and AW channels (Wrapper Internal)			
ar_idle	1	Input	AR channel is idle For ASYNC intf with AXI Slave :

			Additionally Async Fifos are empty
aw_idle	1	Input	AW channel is idle For ASYNC intf with AXI Slave : Additionally Async Fifos are empty
ar_cg_clk	1	Output	Gated input clock for AR channel
aw_cg_clk	1	Output	Gated input clock for AW channel
AXI Slave Bridge : CSR (Wrapper Internal)			
acsb_cg_reg	17	Input	17 bit register read value from clock gate control register (Section 9.2.2)
StrrxBrdg signals (Wrapper Internal)			
strrxbrdg_fifo_empty	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	NoC side Input VC fifos are empty
strrxbrdg_cg_clk	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	**Gated Input clock for NoC side Rx VC FIFOs**
Misc signals (Wrapper Internal)			
Core_cg_clk	1	Output	Gated clock for complete shut-down of the AXI Slave Bridge. **Core_clk will still be an input to the router for some free running signals and logic**

9.4.2 Implementation #1: AXI Slave Bridge clock gate control registers

This register would reside inside the u_ns_acsb_csr**. The read logic for this register should always be free-running** (verification should take note of this).

Table 21 Implementation #1: AXI Master Bridge clock gate control register

31			17	16	15				0
				Fast Path override	Count Value after which o_[port]_busy is de-asserted				

9.4.3 Implementation #1: AXI Master Bridge clock gating module parameters

Table 22 Implementation #1: AXI Master Bridge clock gate module parameters

Name	Bits	Description
P_ASYNC_MODE	1	Asynchronous input interface with AXI Master

9.5 Implementation #1: Pipeline "System Clock Gating" Interface signals

Following are the list of interface signals related to "Coarse Clock Gating " in Pipeline Stages

9.5.1 Implementation #1: Data Pipeline

Table 23 Implementation #1: Data Pipeline clock gating module interface

Signal Name	Width	Direction	Description
System Signals (Wrapper I/O)			
System_noc_clk_en	1	Input	System Clock Enable from NoC IO
System_noc_cg_or	1	Input	System Clock Gate Override from NoC IO
System_reg_clk_en	1	Input	System Clock Enable from RegBus register
System_reg_cg_or	1	Input	System Clock Gate Override from Regbus register
Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
Scan_mode	1	Input	Scan mode pin for clock gate cell
NoC Signals (Wrapper I/O)			
i_cg_busy	1	Input	Input busy signal for destination

			NoC element
o_cg_busy	1	Output	Staged output busy signal for destination NoC element
Pipeline signals (Wrapper Internal)			
busy_pl_reg	1	Input	Internal Registered i_cg_busy signal to control clock gating pipeline registers
pl_cg_clk	[P_NUM_OF_STAGES-1:0]	Output	Gated Internal clocks to "clock gate" Flit pipeline registers.

Note for NoC Studio : There would be two System clock enable signals at NoC top level - *System_noc_clk_en* and *System_reg_clk_en*.

5) For Non RegBus NoC layers, the *System_clk_en* pin of the NoC element would be connected to *System_noc_clk_en* signal
6) For RegBus NoC layer, the *System_clk_en* pin of the NoC element would be connected to *System_reg_clk_en* signal.

Similarly there would be two System Clock Gating override signals at NoC top level - *System_noc_cg_or* and *System_reg_cg_or*

5) For Non RegBus NoC layers, the *System_cg_or* pin of the NoC element would be connected to *System_noc_cg_or* signal
6) For RegBus NoC layer, the *System_cg_or* pin of the NoC element would be connected to *System_reg_cg_or* signal.

Table 24 Implementation #1: New Data Pipeline pins to be added (NoC Studio to add hooks)

System_clk_en	1	Input	System Clock Enable - Non Regbus Layer : to be connected to *System_noc_clk_en* Regbus Layer : to be connected to System_reg_clk_en
System_cg_or	1	Input	System Clock Gate Override - Non Regbus Layer : to be connected to System_noc_cg_or RegBus Layer : to be connected to *System_reg_cg_or*
Scan_mode	1	Input	Scan mode pin for clock gate cell

NoC Signals			
i_cg_busy	1	Input	Input busy signal for destination NoC element
o_cg_busy	1	Output	Staged output busy signal for destination NoC element

9.5.2 Implementation #1: Data Pipeline clock gating module parameters

Table 25 Implementation #1: Data Pipeline clock gating module parameter

Name	Bits	Description
P_NUM_OF _STAGES	1	Number of stages within Pipeline Data wrapper

Table 26 Implementation #1: New Data Pipeline parameters to be added (NoC Studio to add hooks)

Parameter Name	Bits	Description
~~P_SYSTEM_CK_EN_VIA_REGBUS~~	~~1~~	~~1 => System Clock Enable and System Clock Gate Override signals to be provided by the RegBus~~ ~~0 => System Clock Enable and System Clock Gate Override signals to be provided by NoC IO Pins~~

9.5.3 Implementation #1: Credit Pipeline coarse clock gating

Since the credit pipeline stages are 4 bits wide, there would not be coarse clock gating implemented

10 Implementation #2: Instantiation of clock gating module outside NoC element and outside NoC

As discussed before, there would be two flavors of implementation for instantiating the clock gating module. Customers could choose between one. In the second mode of implementation, control logic for generating gated clock for each NoC element would be instantiated outside the NoC. The customer would have their own logic to control clock gating from outside the NoC. Lot of customers would actually want have this option since they design their own clock tree for the NoC, and have the precise knowledge of clock skew management and associated clock tree delays. Thus they will know at which points of clock tree to tap, qualify and propagate coarse grained gated clock signal. Thus all the signals associated with coarse grained clock gating are required to be brought out from the NoC element wrapper as well as top level NoC. NoCStudio needs to have the property *Clock_gate_control_external* 'ON' to use this option.

Inside the NoC element wrapper, a clock logic module *u_ns_<Noc_element>_cg* still needs to be instantiated for generating Interface busy signal for the neighboring NoC element. The module *u_ns_<Noc_element>_cg* would not generate gated clock signal and would be instantiated at the same level of hierarchy as *u_ns_<Noc_element>* module. In addition to new input/output pins (listed in this Section) at Noc element wrapper level, some additional input and output ports needs to be created at u_ns_<Noc_element> module level. These signals will communicate with *u_ns_<element>_cg* to convey busy/idle status for its neighboring NoC element.

10.1 Implementation #2: Router Clock Gate module instantiations and hierarchy

The following diagram illustrates where the clock gate module u_ns_router_cg will be instantiated inside the Router wrapper and the corresponding signal interactions.

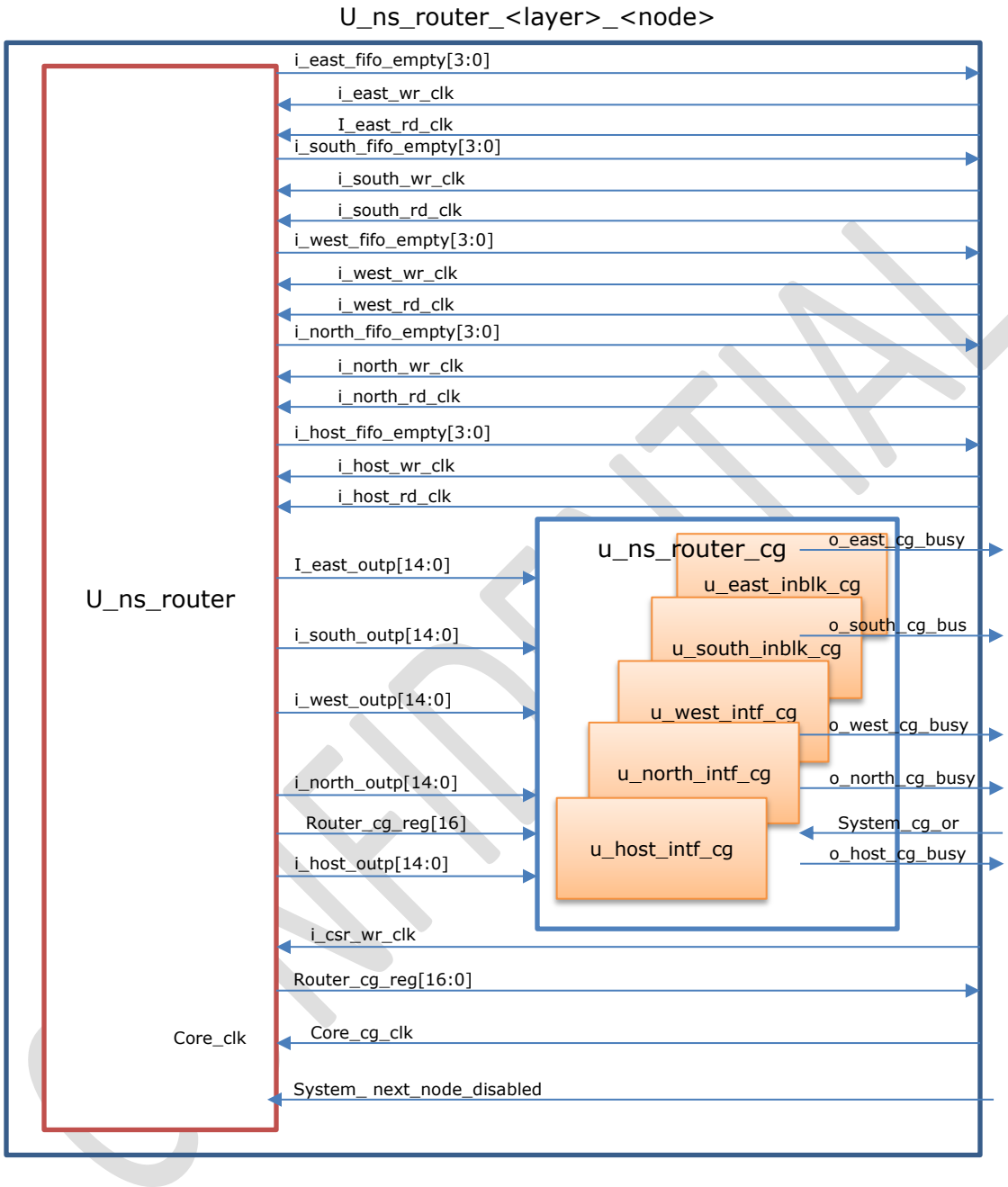

Figure 14 Implementation #2: Router hierarchy for clock gating

10.1.1 Implementation #2: Router clock gating interface signals

Table 27 Implementation #2: Router Wrapper Clock gating Interface signals

Signal Name	Width	Direction	Description
System Signals (Wrapper I/O)			
~~System_clk_en~~	~~1~~	~~Input~~	~~System Clock Enable~~
~~System_cg_or~~	~~1~~	~~Input~~	~~System Clock Gate Over ride~~
System_next_node_disabled	4	Input	Next node is disabled (from RegBus Slave Chain) 0 : Node on "North" disabled 1 : Node on "East" disabled 2 : Node on "West" disabled 3 : Node on "South" disabled
NoC [port] Signals (Wrapper I/O) where [port] is one of {north, east, west, south, host, reg}			
o_[port]_cg_busy	1	Output	Output Port Busy (to neighboring NoC)
~~i_[port]_cg_busy~~	~~1~~	~~Input~~	~~Input Port Busy (from neighboring NoC)~~
~~i_[port]_clk~~	~~1~~	~~Input~~	~~Input Write clock (from neighboring router) – **Async Case**~~
i_[port]_fifo_empty	4	Output	Bit[0] – I/P VC0 Fifo is empty Bit[1] – I/P VC1 Fifo is empty Bit[2] – I/P VC2 Fifo is empty Bit[3] – I/P VC3 FIfo is empty
Router_cg_reg	17	Output	17 bit register read value from clock gate control register (Section10.1.2)
i_[p]_wr_cg_clk	1	Output	Write side gated clk for I/P VC Fifos (common for both Async and Sync I/P interface cases)
i_[p]_rd_cg_clk	1	Output	Async Interface – Read Side gated clk Sync interface – This signal can be un-connected
Core_cg_clk	1	Output	Gated clock for arbitration and u_[port]_outblk. An additional port is required in the router for this clock. **Core_clk will still be an input to the router for free running signals.**

Table 28 Implementation #2: u_ns_router_cg Interface signals (Wrapper internal)

Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
System_cg_or	1	Input	System Clock Gate Over ride
i_[port]_outp	15	Input	Bit[2:0] – outp at head of VC0 Fifo Bit[5:3] – outp at head of VC1 Fifo Bit[8:6] – outp at head of VC2 Fifo Bit[11:9] – outp at head of VC3 Fifo Bit[14:12] – Registered outp input (**Synchronous case else drive it to zero)**
o_[port]_cg_busy	1	Output	Output Port Busy (for neighboring NoC element)
o_[port]_cg_status	1	Output	NoC Element on [port] Status 1: Clock Enabled 0: Clock Disabled
Router_cg_reg	17	Output	17 bit register read value from clock gate control register (Section10.1.2)

10.1.2 Implementation #2: Router clock gate control registers

This register would reside inside the u_ns_router_csr**. The read logic for this register should always be free-running** (verification should take note of this).

Table 29 Implementation #2: Router clock gate control register

31	17	16	15	0
		Fast Path override	Count Value after which o_[port]_busy is de-asserted	

10.1.3 Implementation #2: Router clock gate module parameters

The following parameter is required by the clock gating module to conditionally instantiate logic for Synchronous/Asynchronous input interface.

Table 30 Implementation #2: Router clock gate module parameter

Parameter Name	Bits	Description
P_[p]_IVCBUF_TYPE	8	4 groups of 2 bits for each VC in the order {3, 2, 1, 0}. Determines the maximum upsize ratio from that VC of the input port 00=> 1x 01=> 2x 10=> 4x 11=> Async mode (1x)

10.2 Implementation #2: Streaming Bridge Clock Gate module instantiation and hierarchy

The following diagram illustrates where the clock gate module ns_strbrdg_cg will be instantiated inside the Streaming Bridge wrapper and the corresponding signal interactions.

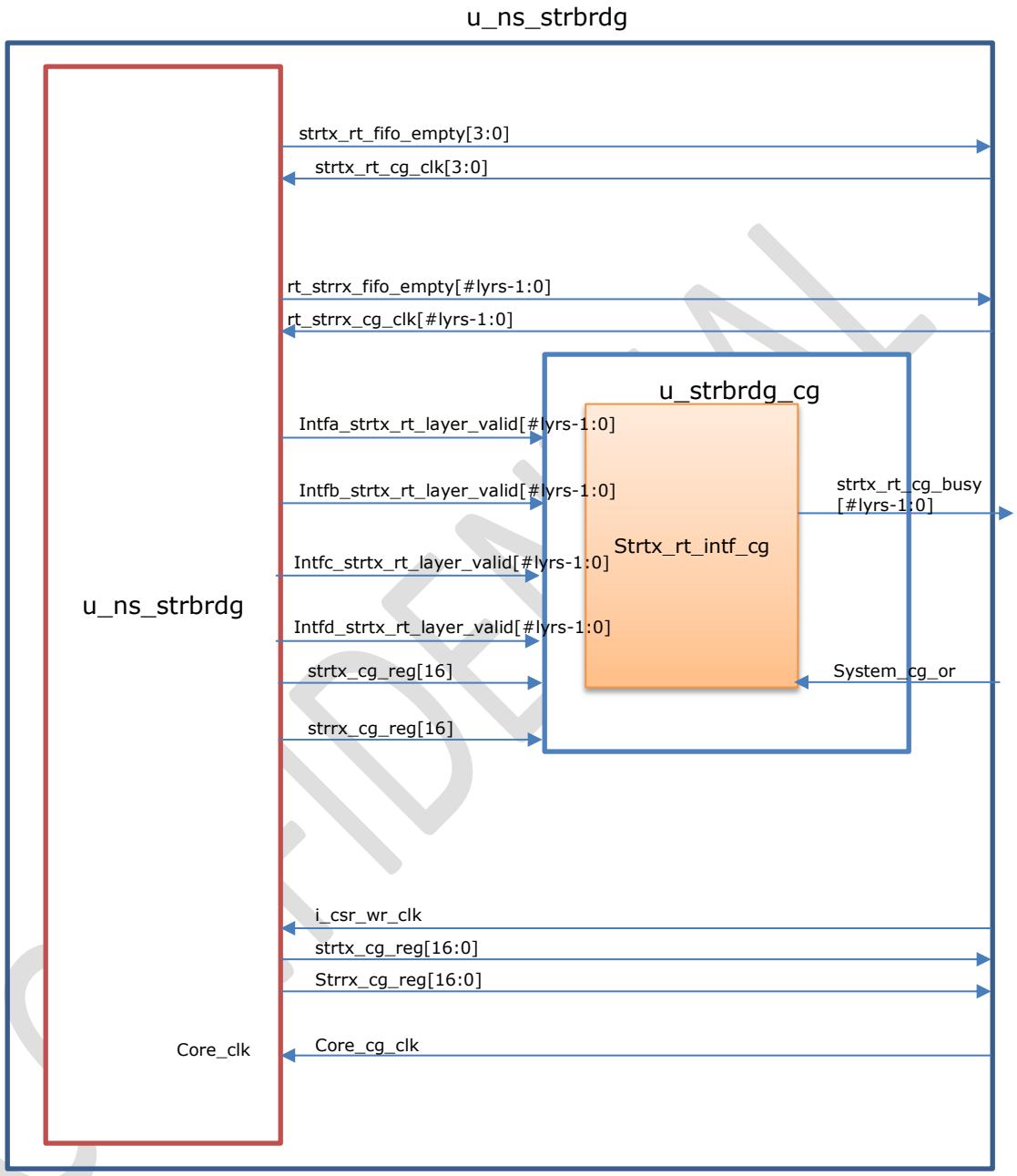

Figure 15 Implementation #2: Streaming Bridge hierarchy for clock gating

10.2.1 Implementation #2: Streaming Bridge clock gating interface signals

Table 31 Implementation #2: Streaming Bridge clock gating interface signals

Signal Name	Width	Direction	Description
System Signals (Wrapper I/O)			
~~System_clk_en~~	~~1~~	~~Input~~	~~System Clock Enable~~
~~System_cg_or~~	~~1~~	~~Input~~	~~System Clock Gate Over ride~~
NoC Signals (Wrapper I/O)			
strtx_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC interface)
~~rt_strrx_cg_busy~~	~~[P_STR_TX_RT_NOOFLAYERS-1:0]~~	~~Input~~	~~Input Port Busy (from NoC interface)~~
strtx_rt_fifo_empty	[P_STR_TX_HST_INTERFACES-1:0]	Output	Bit[0] – Intf a VC Fifo is empty Bit[1] – Intf b VC Fifo is empty Bit[2] – Intf c VC Fifo is empty Bit[3] – Intf d VC FIfo is empty
rt_strrx_fifo_empty	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	NoC side Input VC fifos are empty
strtx_rt_cg_status	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Router on [layer] Status 1: Clock Enabled 0: Clock Disabled
strtx_rt_cg_clk	[P_STR_TX_HST_INTERFACES-1:0]	Input	Gated Input clock for Transmit Block
rt_strrx_cg_clk	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Gate Input clock for Receive Block
strtxbrdg_cg_reg	17	Output	17 bit register read value from Transmit block clock gate control register (Section 10.2.2)
strrxbrdg_cg_reg	17	Output	17 bit register read value from Receive block clock gate control register (Section

			10.2.2)
Core_cg_clk	1	Input	Gated clock for complete shut-down of the Streaming Bridge. **Core_clk will still be an input to the router for some free running signals.**

Table 32 Implementation #2: u_strbrdg_cg Interface signals (Wrapper Internal)

Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
System_cg_or	1	Input	System Clock Gate Over ride
Intfa_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from Intf a
Intfb_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from Intf b
Intfc_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from Intf c
Intfd_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from Intf d
strtx_rt_cg_status	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Router on [layer] Status 1: Clock Enabled 0: Clock Disabled
strtx_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC interface)
strtxbrdg_cg_reg	17	Output	17 bit register read value from Transmit block clock gate control register (Section 10.2.2)
strrxbrdg_cg_reg	17	Output	17 bit register read value from Receive block clock gate control register (Section 10.2.2)

10.2.2 Implementation #2: Streaming Bridge clock gate control registers

This register would reside inside the u_ns_csr**. The read logic for this register should always be free-running** (verification should take note of this).

Table 33 Implementation #2: Streaming Transmit Bridge clock gate control register

31			17	16	15				0
				Fast Path override	Count Value after which o_[port]_busy is de-asserted				

Table 34 Implementation #2: Streaming Receive Bridge clock gate control register

31			17	16	15				0
				Fast Path override	Count Value after which o_[port]_busy is de-asserted				

10.3 Implementation #2: AXI Master Bridge Clock Gate module instantiation and hierarchy

The following diagram illustrates where the clock gate module ns_acemstrbrdg_cg will be instantiated inside the AXI Master Bridge (u_aximstrbrdg) wrapper and the corresponding signal interactions

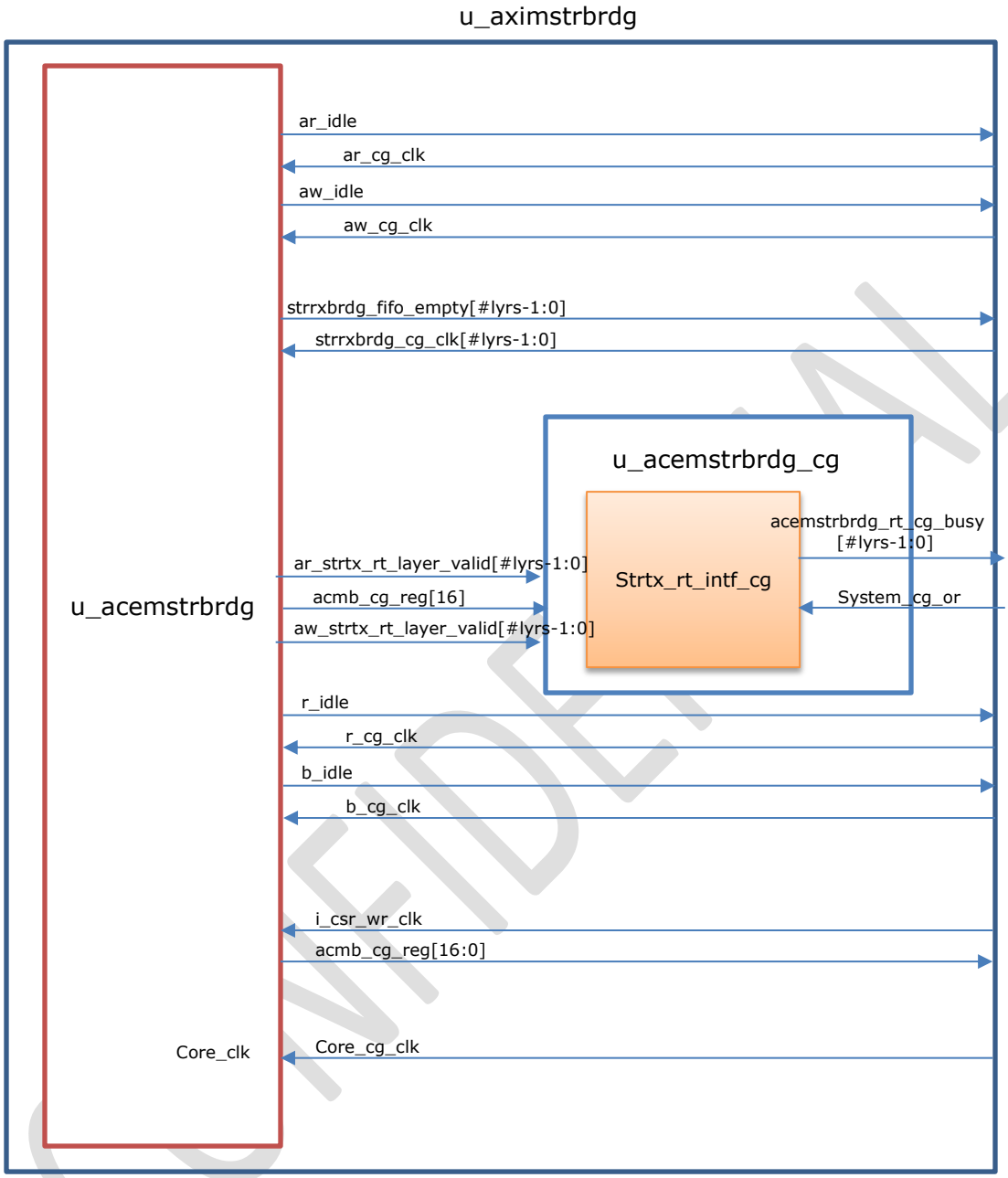

Figure 16 Implementation #2: AXI Master Bridge Clock Gating hierarchy

10.3.1 Implementation #2: AXI Master Bridge clock gating module interface

Table 35 Implementation #2: AXI Master Bridge Wrapper clock gating interface signals

Signal Name	Width	Direction	Description
System Signals			
~~System_clk_en~~	~~1~~	~~Input~~	~~System Clock~~

~~Enable~~			
~~System_cg_or~~	~~1~~	~~Input~~	~~System Clock Gate Over ride~~
NoC Signals			
acemstrbrdg_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC side interface)
~~rt_acemstrbrdg_cg_busy~~	~~[P_STR_TX_RT_NOOFLAYERS-1:0]~~	~~Input~~	~~Input Port Busy (from NoC side interface)~~
AXI Master Bridge signals : AR and AW channels			
ar_idle	1	Output	AR channel is idle For ASYNC intf with AXI Master : Additionally Async Fifos are empty
aw_idle	1	Output	AW channel is idle For ASYNC intf with AXI Master : Additionally Async Fifos are empty
ar_cg_clk	1	Input	Gated Input clock for AR Block
aw_cg_clk	1	Input	Gated Input clock for for AW Block
AXI Master Bridge signals : R and B channels			
r_idle	1	Output	R channel is idle For ASYNC intf with AXI Master : Additionally Async Fifos are empty
b_idle	1	Output	B channel is idle For ASYNC intf with AXI Master : Additionally Async Fifos are empty
r_cg_clk	1	Input	Gated input clock for R channel
b_cg_clk	1	Input	Gated input clock for B channel
AXI Master Bridge signals : CSR			
acmb_cg_reg	17	Output	17 bit register read value from AXI Master Bridge clock gate control register

			(Section 10.3.2)
StrrxBrdg signals			
strrxbrdg_fifo_empty	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	NoC side Input VC fifos are empty
strrxbrdg_cg_clk	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Gated Input clock for NoC side Rx VC FIFOs.
Misc signal			
Core_cg_clk	1	Input	Gated clock for complete shut-down of the AXI Master Bridge. **Core_clk will still be an input to the router for some free running signals and logic**

Table 36 Implementation #2: u_acemstrbrdg_cg interface signals (Wrapper internal)

Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
System_cg_or	1	Input	System Clock Gate Over ride
acemstrbrdg_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC side interface)
acemstrbrdg_rt_cg_status	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Router on [Layer] Status 1: Clock Enabled 0: Clock Disabled
ar_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from AR channel
aw_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from AW channel
acmb_cg_reg	17	Output	17 bit register read value from AXI Master Bridge clock gate control register (Section 10.3.2)

10.3.2 Implementation #2: AXI Master Bridge clock gate control registers

This register would reside inside the u_ns_acmb_csr**. The read logic for this register should always be free-running** (verification should take note of this).

Table 37 Implementation #2: AXI Master Bridge clock gate control register

31			17	16	15				0
				Fast Path override	Count Value after which o_[port]_busy is de-asserted				

10.3.3 Implementation #2: AXI Master Bridge clock gating module parameters

Table 38 Implementation #2: AXI Master Bridge clock gate module parameters

Name	Bits	Description
P_ASYNC_MODE	1	Asynchronous input interface with AXI Master

10.4 Implementation #2: AXI Slave Bridge Clock Gate module instantiation and hierarchy

The following diagram illustrates where the clock gate module ns_aceslvbrdg_cg will be instantiated inside the AXI Slave Bridge wrapper (u_axislvbrdg) and the corresponding signal interaction.

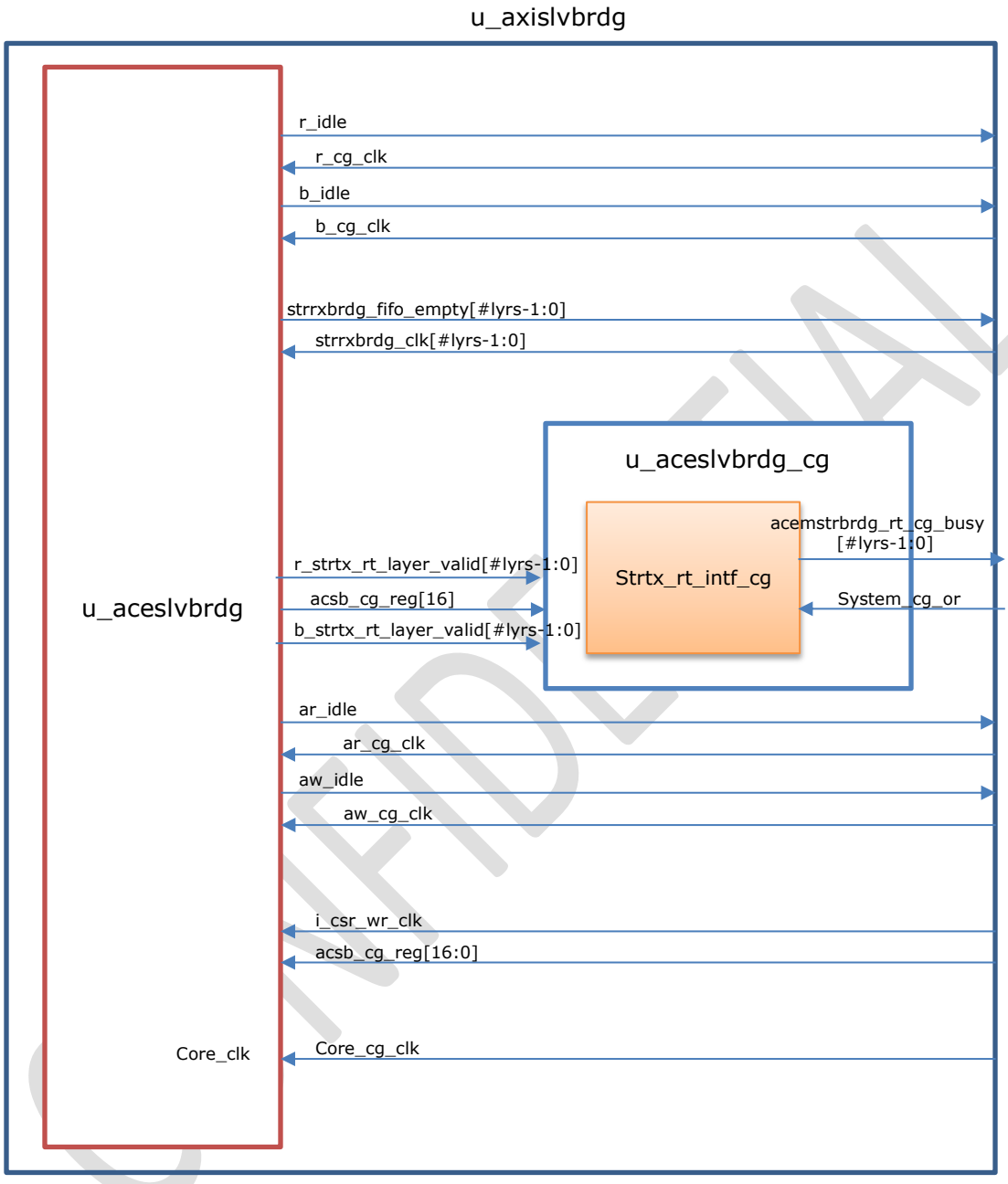

Figure 17 Implementation #2: AXI Slave Bridge clock Gating hierarchy

10.4.1 Implementation #2: AXI Slave Bridge clock gating module interface

Table 39 Implementation #2: AXI Slave Bridge clock gating signal interface

Signal Name	Width	Direction	Description
System Signals			
~~System_clk_en~~	~~1~~	~~Input~~	~~System Clock Enable~~
~~System_cg_or~~	~~1~~	~~Input~~	~~System Clock Gate Over ride~~
NoC Signals			
aceslvbrdg_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC side interface)
~~rt_aceslvbrdg_cg_busy~~	~~[P_STR_TX_RT_NOOFLAYERS-1:0]~~	~~Input~~	~~Input Port Busy (from NoC side interface)~~
AXI Slave Bridge : R and B channels			
r_idle	1	Output	R channel is idle For ASYNC intf with AXI Slave : Additionally Async Fifos are empty
b_idle	1	Output	B channel is empty For ASYNC intf with AXI Slave : Additionally Async Fifos are empty
r_cg_clk	1	Input	Gated Input clock for R Block
w_cg_clk	1	Input	Gated Input clock for for W Block
AXI Slave Bridge : AR and AW channels			
ar_idle	1	Output	AR channel is idle For ASYNC intf with AXI Slave : Additionally Async Fifos are empty
aw_idle	1	Output	AW channel is idle For ASYNC intf with AXI Slave : Additionally Async Fifos are empty
ar_cg_clk	1	Input	Gated input clock for AR channel
aw_cg_clk	1	Input	Gated input clock for AW channel

AXI Slave Bridge : CSR			
acsb_cg_reg	17	Output	17 bit register read value from clock gate control register (Section 10.4.2)
StrrxBrdg signals			
strrxbrdg_fifo_empty	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	NoC side Input VC fifos are empty
strrxbrdg_cg_clk	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	**Gated Input clock for NoC side Rx VC FIFOs**
Misc signals			
Core_cg_clk	1	Input	Gated clock for complete shut-down of the AXI Slave Bridge. **Core_clk will still be an input to the router for some free running signals and logic**

Table 40 Implementation #2: u_aceslvbrdg_cg interface signals (Wrapper Internal)

Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
System_cg_or	1	Input	System Clock Gate Over ride
aceslvbrdg_rt_cg_busy	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Output Layer Busy (to NoC side interface)
aceslvbrdg_rt_cg_status	[P_STR_TX_RT_NOOFLAYERS-1:0]	Output	Router on [layer] Status 1: Clock Enabled 0: Clock Disabled
r_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from R channel
w_strtx_rt_layer_valid	[P_STR_TX_RT_NOOFLAYERS-1:0]	Input	Outgoing layer valid from W channel
acsb_cg_reg	17	Input	17 bit register read value from

			clock gate control register (Section 10.4.2)

10.4.2 Implementation #2: AXI Slave Bridge clock gate control registers

This register would reside inside the u_ns_acsb_csr**. The read logic for this register should always be free-running** (verification should take note of this).

Table 41 Implementation #2: AXI Master Bridge clock gate control register

31			17	16	15				0
				Fast Path override	Count Value after which o_[port]_busy is de-asserted				

10.4.3 Implementation #2: AXI Master Bridge clock gating module parameters

Table 42 Implementation #2: AXI Master Bridge clock gate module parameters

Name	Bits	Description
P_ASYNC_MODE	1	Asynchronous input interface with AXI Master

10.5 Implementation #2: Pipeline "System Clock Gating" Interface signals

Following are the list of interface signals related to "Coarse Clock Gating " in Pipeline Stages

10.5.1 Implementation #2: Data Pipeline

Table 43 Implementation #2: Data Pipeline clock gating signal interface

Signal Name	Width	Direction	Description
System Signals			
~~System_clk_en~~	~~1~~	~~Input~~	~~System Clock Enable~~
System_cg_or	1	Input	System Clock Gate Over ride
Core_clk	1	Input	Free running clock input
Noc_reset	1	Input	Global reset
NoC Signals			
i_cg_busy	1	Input	Input busy signal for destination

			NoC element
o_cg_busy	1	Output	Staged output busy signal for destination NoC element
Pipeline signals			
busy_pl_reg	1	Output	Registered i_cg_busy signal to external clock gating module to control clock gating pipeline registers
pl_cg_clk	[P_NUM_OF_STAGES-1:0]	Input	Gated clocks from Registered clock module to "clock gate" Flit pipeline registers.

10.5.2 Implementation #2: Data Pipeline clock gating module parameters

Table 44 Implementation #2: Data Pipeline clock gating module parameter

Name	Bits	Description
P_NUM_OF _STAGES	1	Number of stages within Pipeline Data wrapper

10.5.3 Implementation #2: Credit Pipeline coarse clock gating

Since the credit pipeline stages are 4 bits wide, there would not be coarse clock gating implemented

11 RegBus Slave Bridge “System Clock Gating” interface signals

Following are the list of interface signals from RegBus Slave related to System Coarse clock gating.

Table 45 RegBus Slave Bridge

Signal Name	Width	Direction	Description
Tx interface on Slave chain			
System_clk_en	1	Output	System Clock Enable
System_cg_or	1	Output	System Clock Gate Over ride
System_next_node_disabled	4	Output	Next node is disabled 0 : Node on “North” disabled 1 : Node on “East” disabled 2 : Node on “West” disabled 3 : Node on “South” disabled
Tx interface on RegBus Layer (to neighboring nodes)			
O_Self_disabled_north	1	Output	To RegBus Slave on adjacent node in north direction
O_Self_disabled_west	1	Output	To RegBus Slave on adjacent node in west direction
O_Self_disabled_south	1	Output	To RegBus Slave on adjacent node in south direction
O_Self_disabled_east	1	Output	To RegBus Slave on adjacent node in east direction
Rx interface on RegBus Layer (to neighboring nodes)			
i_Self_disabled_north	1	Output	From RegBus Slave on adjacent node in north direction
i_Self_disabled_west	1	Output	From RegBus Slave on adjacent node in west direction
i_Self_disabled_south	1	Output	From RegBus Slave on adjacent node in south direction
i_Self_disabled_east	1	Output	From RegBus Slave on adjacent node in east direction

12 NoC Studio requirements

The requirements from NoC Studio are as below:

- A mesh property *coarse_clock_gating_enabled* is required to disable clock gating of NoC elements from within NoC level. When the property is set to 0, following actions are required
  - Tie of System_clk_en pin of Noc element to 1
  - Tie of System_cg_or pin of NoC element to 1
- A new parameter P_HYST_COUNT is to be assigned to each Noc element. This parameter would be used to set the reset/default value of the hysteresis count register(BTCGC, BRCGC, RCGC etc) of routers and bridges. The default value of this parameter is set 'd100.
- A router property *router_hysteresis_prop* could be specified by the user to override the default value of P_HYST_COUNT parameter of the router. The value should be restricted to be less than 2^{31} .

  `Eg. router_prop R0.5 router_hysteresis_prop 1000`

- A Bridge property *bridge_hysteresis_prop* could be specified by the user to override the default value of P_HYST_COUNT parameter of the Bridge. The value should be restricted to be less than 2^{31} .

  `Eg. bridge_prop h1/p1 bridge_hysteresis_prop 1000`
- Specify mesh property *System_clk_en_via_regbus* to drive *system_clk_en* and *system_cg_or* pins pf each Noc element through Regbus Slave Bridge
  - NocStudio should also dump out information for the user, mapping the address of the "System clock gate control register for Bridges" to the actual Bridge ID. This is required so that user knows which register in the RegBus Slave Bridge should be written to so as to control the *System_clk_en* and *System_cg_or* pins of a certain Bridge.

- ~~Specify clock frequency and clock crossing per NoC element~~
  - ~~AXI/ACE Bridges : Sync and Async interface~~
  - ~~Routers : Need Async specification, only Sync currently exists~~
  - ~~Streaming Bridges : Sync exists, no plan for Async interface~~
  - ~~TBD: Pseudo sync or meso-sync interface props are needed~~
  - ~~TBD: 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 'ON' to support generation of *System_clk_en* and *System_cg_or* through RegBus Slave.~~
  - ~~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 neighbor 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 wrapper. Outputs to be tied off in wrapper module.
- Specify *Clock_gate_control_external* property to choose location of coarse clock gating control module.
    - If the property is OFF, the clock logic module instantiated inside the NoC element wrapper would generate "coarse grained" gated clock signal for itself as well as *Interface busy* signal for NoC element on adjacent node.
    - If the property is ON, "coarse grained" gated clock signal for the NoC element would be generated from control logic, external to NoC. The clock logic module instantiated inside the NoC element wrapper would generate only *Interface busy* signal for NoC element on adjacent node.

- RegBus Slave bridge has some added outputs and needs a new wrapper
- Pipeline stage clock gating module needs to know the parameter P_NUM_OF_STAGES for the depth of the pipeline.
- The clock gating module needs to be passed the parameter to declare if the corresponding NoC element is registered at the output interface.

Document Revision A

Update 1.

2670 Seely Ave
Building 11
San Jose, CA 95134
(408) 914-6962
www.netspeedsystems.com