

SAME 2009 Forum

Session : System & Design

Adding support for Power Management in OCP: The OCP Disconnect Protocol

Christophe Vatinel, Texas Instruments

Villeneuve Loubet, France

Abstract

According to Moore's law, System-on-Chips are continually becoming more complex and integrating more components, enabled by the continuing size reduction of silicon technologies. On the other hand, power consumption does not follow the same reduction ratio due to increasing leakage in deep sub-micron technologies. Hence new power management techniques are being progressively used within the SoC in order to reduce power dissipation as much as possible.

This paper exposes some of the new architectural problems that exist in today's complex SoCs involving power management techniques. The concept of interface disconnection is introduced as a possible solution and the OCP disconnect protocol is detailed as an implementation of this concept. Finally several examples of use in a power management framework are given.

Table of Contents of the paper

Introduction - Emerging Problems with Power Management

- Putting a Domain to Sleep
- Traffic towards a Sleeping Domain
- The Disconnection Concept

The OCP disconnect Protocol

- Protocol Signaling
- Integration into the Master
- Connection Status state chart
- Stopping the OCP Initiator
- Alternate Behavior

Linking the OCP Disconnect Protocol with Power Management

- Slave IP
- Master IP
- Interconnect IP

Conclusion

Introduction - Emerging Problems with Power Management

A typical SoC can be partitioned into several domains that can be managed more or less independently with respect to power management. Several power management techniques may be used such as lowering clock frequency, lowering voltage, gating the clock, switching to retention mode or switching off. One must realize that with some of these techniques, the affected components are not responsive to any input. At most, signals from a component in such a state can be forced to a pre-defined value thanks to a clamp technique. Such states will be referred to as "sleep" states in the rest of this paper.

On the other hand, the Open Core Protocol (OCP) defines a point-to-point fully synchronous interface protocol that can be used as an internal communication method. Following are some of the problems that appear in a SoC using OCP as interconnect protocol and involving power management techniques.

Putting a Domain to Sleep

Before putting a domain to sleep, one must ensure that any OCP interface crossing the domain boundary is in a state where all OCP transactions are completed (i.e. no outstanding transactions).

Not doing so can result in various SoC disorders. If a domain is being put to sleep while some transactions at its boundary are not completed, then the corresponding communicating entities will likely become stuck or even corrupted. In any cases, there is no easy way to recover from such a situation. Note also that the disorders may propagate via the interconnect throughout the entire SoC, ultimately resulting in a major failure.

Traffic towards a Sleeping Domain

Once a domain has been properly put to sleep, another concern is that some masters may still be able to send transactions towards this domain. Such

transactions will likely induce the same kind of disorder as exposed in the previous section: master or interconnect becoming stuck.

The Disconnection Concept

The above problems imply the need for a new concept: connection and disconnection. Connection is a state where both sides are functional and where communication can happen normally. Conversely, disconnection is a state where no communication is possible. Additionally, the disconnection state must only be reached after a clean termination of the communication interface; with all outstanding transactions completed.

With these definitions, the disconnection state becomes a pre-requisite state for all OCP interfaces at a domain boundary before performing a domain sleep transition.

Connection or disconnection has to be negotiated between master and slave. Because by definition one cannot have communication with a module (slave or master) in a sleeping domain, there will be a hierarchy between the connection and disconnection states. The connection state will only be established if both sides agree to be connected. However, both sides can independently request and cause a disconnection.

As a consequence, a disconnection can be the result of three different scenarios:

- Mutual disconnection request from both sides,
- Disconnection solely requested by master,
- Disconnection solely requested by slave.

In the first two cases, it can be assumed that the master does not need to communicate with the slave since master is requesting disconnection. In the third case however, it may be that the master has new traffic intended for the disconnected slave. A simple case is when the master is actually an interconnect which is routing traffic from several system initiators. It is then necessary to specify the behavior of this master with pending traffic. This behavior will be referred to as the “alternate behavior” of the master port.

The OCP disconnect protocol

The OCP disconnect protocol is a new protocol implementing the above connection/disconnection concept. It is primarily targeted at the existing OCP-IP protocol and will be specified as an optional add-on in version 3.0 of the OCP-IP standard. However, it can also be used for other communication interfaces, provided they have similar characteristics (one way, point-to-point interfaces).

Like the existing OCP protocol, the OCP disconnect protocol is a point-to-point protocol between master and slave. It is fully synchronous to the OCP clock, and its signals are registered, in order to ease timing closure at SoC level.

The protocol is based on the following principles:

- Both master and slave have a vote for entering the connection state.
- The slave vote is propagated to the master where the connection or disconnection transition is performed.
- The master generates the actual connection status, which is propagated back to the slave. Transactions can only be issued by the master and only when in the connected state. The slave is also committed to handling transactions normally while in the connected state, independent of its own vote.
- The slave as well as the master also have the ability to temporarily stall the connect/disconnect transition being performed on the master side.

The voting mechanism allows implementation of the connection/disconnection concept as defined in the previous section: a connection is established only in the case of mutual agreement. Note that either side can change its vote at any moment without restriction.

It is advantageous to perform the connection or disconnection at the master side for the following reason: in case an OCP-to-OCP bridge is present between the master and the slave (ex: asynchronous bridge), the emptying/idling operation of such a bridge can be included as part of a disconnection. It is assumed here that the bridge is knowledgeable about the OCP disconnect protocol and uses it for that purpose. One consequence of having the connection or disconnection performed at the master side is that the master’s vote is a piece of information that remains internal to the master. Only the slave’s vote is propagated from slave to master.

The master generates the actual connection status and propagates it to the slave. The slave monitors this connection status to determine when it is safe to perform a domain sleep transition. The connection status permits a disconnection state initiated solely by the slave, to be distinguished from a disconnection state initiated by the master. In the case of a disconnection state initiated solely by the slave, the protocol supports an alternate behavior of the master port for new traffic intended for the disconnected slave.

Protocol Signaling

The OCP disconnect protocol signaling consists of three signals described in following table.

Signal	Value	Description
MConnect[1:0] (Driven by the master)	'b00 (M_OFF)	This is a disconnect state, requested by the master. The OCP interface is cleanly stopped. The alternate behavior is also cleanly stopped. <i>This is the default reset state.</i>
	'b01 (M_WAIT)	This is a transient state. The OCP interface is cleanly stopped. The alternate behavior is also cleanly stopped.
	'b10 (M_DISC)	This is a disconnect state, requested solely by the slave. The OCP interface is cleanly stopped. The master alternate behavior is enabled.
	'b11 (M_CON)	This is the only connection state. The OCP interface is up and running. It can send transactions to the slave. The alternate behavior is cleanly stopped.
SConnect (Driven by the slave)	'b0 (S_DISC)	The slave is either requesting a disconnect sequence, or once disconnected, indicating it is unwilling to re-connect. <i>This is the default reset state.</i>
	'b1 (S_CON)	The slave is indicating a readiness to perform a re-connect sequence, or maintaining a connected state.
SWait (Driven by the slave)	'b0 (S_OK)	Used to indicate to the master that transition to any new state is allowed. <i>This is the default reset state.</i>
	'b1 (S_WAIT)	Used to indicate to the master that it can only transition to the M_WAIT state. That is, the slave temporarily stalls the transition at the master side.

Table 1: OCP disconnect protocol signals

SConnect constitutes the slave's vote and SWait is a stall request from the slave. MConnect[1:0] is the actual connection status from the master.

Integration into the Master

At the master side, a finite state machine is responsible for generating the OCP connection status. First of all, the interfaces between this FSM and the master's other internal components must be detailed.

It must have an interface with a local or distant controller, which can be part of what is commonly called the power manager. This delivers the master's side connection/disconnection vote as well

as an internal stall condition equivalent to the SWait signal for the master side.

An interface with the OCP traffic initiator must also be present in order to control its activity. The OCP connection status is intimately tied to the OCP master port activity.

Similarly, an interface with the master port's alternate behavior must also be present in order to control its activity. In the same way, the OCP connection status is intimately tied to the master port's alternate behavior activity.

The following table defines the signals used for interfacing the OCP connection status FSM with master's internal components.

Signals	Description
MDiscReq	Master's connection/disconnection vote from local or distant controller 1: vote for OCP disconnection 0: vote for OCP connection
MWaitReq	Internal stall condition from a local or distant controller 1: OCP connection state can only transition to M_WAIT state (stall) 0: OCP connection state can transition to any new state
OCPStopReq	Request to control OCP master port's activity 1: Request OCP master port to be or remain stopped 0: Allows OCP master port to be active and initiate transactions
OCPStopAck	Acknowledge from OCP master port to OCPStopReq request 1: OCP master port is cleanly stopped (no outstanding transactions) 0: outstanding or incomplete transactions may exist
ALTStopReq	Request to control master port's alternate behavior 1: Request alternate behavior to be or remain stopped 0: Allows alternate behavior to be active and handle new traffic intended for the disconnected slave
ALTStopAck	Acknowledge from alternate behavior to ALTStopReq request 1: alternate behavior is cleanly stopped 0: alternate behavior may have on-going activity

Table 2: Interfaces with other components

The following schematic gives a high-level view of the integration of the connection status FSM within the master module. The OCP master port is actually made up of three entities: the OCP traffic initiator, the alternate behavior and the connection status FSM.

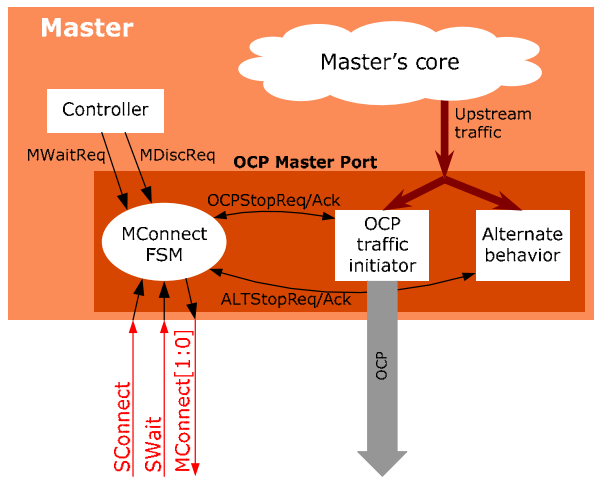


Figure 1: Master high-level architecture

Connection Status state chart

The following state chart defines how the connection status is generated at the master side, interacting with the controller, the OCP traffic initiator and the alternate behavior.

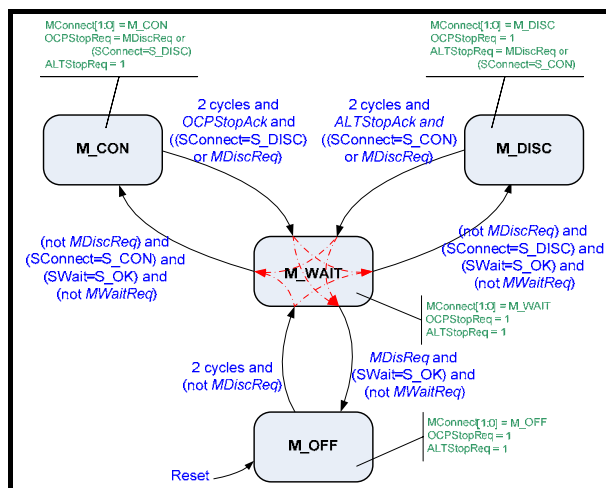


Figure 2: Connection Status FSM

This state chart deserves some explanations. It consists of four states. The green callout attached to each state defines the actions that are performed in each state (signal generation). Arrows and blue text define the transitions between states and their condition. Note that a minimum of two clock cycles are spent in the stable states (M_OFF, M_CON and M_DISC). This is to allow the slave to sample the current state and compute the next SWait value. This is a side effect of the registered output / fully synchronous timing requirement of the OCP disconnect protocol.

The red arrows inside the M_WAIT state are there to show that the M_WAIT state can be entered and exited simultaneously (i.e. transparent – no cycle spend) if an exit transition condition is true when

the state is entered. This facilitates a compact representation of the state chart.

The default value is the M_OFF state. Upon a connection vote from the controller, the FSM will transition to the M_WAIT state or directly to the M_CON or M_DISC state according to the slave's connection vote, SConnect, assuming neither slave nor master stalls the transition.

Upon reaching the M_CON state, the OCP initiator's activity is enabled (OCPStopReq is de-asserted). It stays enabled unless either the master or the slave votes for a disconnection (in which case OCPStopReq is asserted). It is interesting to note here that there is no restriction on when either side can change its vote on connection state. Only when the OCP initiator has stopped its activity (OCPStopAck=1), will the M_CON state be exited, assuming there is still a disconnection vote. The FSM will then transition to the M_WAIT state or directly to the M_DISC or M_OFF state according to the master's connection vote, assuming neither slave nor master stalls the transition.

Similarly, upon reaching the M_DISC state, the alternate behavior's activity is enabled (ALTStopReq is de-asserted). It stays enabled unless either the master votes for a disconnection or the slave votes for a reconnection (ALTStopReq is asserted). Only when the alternate behavior has completed its activity (ALTStopAck=1), will the M_DISC state be exited, assuming there is still a master disconnection vote or a slave reconnection vote. The FSM will then transition to the M_WAIT state or directly to the M_CON or M_OFF state according to the master's connection vote, assuming neither slave nor master stalls the transition.

Finally, the M_WAIT state is exited only when both slave and master do not stall the transition (MWaitReq=SWait=0). The FSM will then transition to the M_OFF state if the master votes for a disconnection (MDisReq=1), or to the M_DISC state if the slave votes for a disconnection (SConnect=0), or to the M_CON state if both master and slave vote for a connection.

Stopping the OCP Initiator

The OCP initiator has a hand-shake protocol with the connection status FSM (OCPStopReq/Ack). The OCP disconnect protocol ensures that the OCP initiator has cleanly stopped its traffic with the slave before exiting the M_CON state, by way of the OCPStopAck signal. Stopping the OCP traffic towards the slave involves two steps. The first is called "fencing" and consists of stopping sending

OCP commands at a transaction boundary. The second step is called “draining” and consists of waiting to receive the responses of all outstanding transactions. Only when both steps are completed, is the OCP initiator cleanly stopped: there are no more outstanding transactions, and all current transactions have been completed.

The OCP disconnect protocol does not actually specify when the fencing operation must be performed. Therefore, upon a request to stop the OCP initiator, the master has the freedom to decide when to perform fencing. In some cases it may be desirable to fence the OCP traffic at the next transaction boundary. However, in other cases, it may be desirable for the initiator to delay the fencing operation based on architectural choices that are beyond the scope of the OCP disconnect protocol. A master could for example perform fencing at a frame boundary, a frame consisting of several OCP transactions.

Alternate Behavior

The alternate behavior is a behavior that is defined for upstream traffic intended for a disconnected slave. This option may or may not be used in a master. The OCP disconnect protocol provides support for this alternate behavior by linking it to the M_DISC connection state. However, the OCP disconnect protocol does not define the alternate behavior. Multiple alternate behaviors can be implemented in the same master and switching between different behaviors is possible according to ad-hoc decisions. Specification of the alternate behavior and its control could be added to the OCP disconnect protocol in the future.

Several alternate behaviors could be imagined. The simplest one would be to do nothing regarding upstream traffic intended for a disconnected slave. This would be equivalent to not implementing an alternate behavior. Another possible behavior would be to reply autonomously to upstream traffic with a default ERROR response. This could be useful for example, when a processor in a SoC tries to access a slave that is asleep, and typically denotes a software problem. The processor would receive an error response that would be treated as an exception. This would be of considerable help in finding the cause of the problem and debugging the software. A third alternate behavior would be to reply autonomously to upstream traffic by a default DVA response. This could be useful in the case where the sleeping slave is a debug IP and it is not desirable for the processor to receive an error when it tries to access this module. A final example of alternate behavior would be the wakeup-on-demand behavior. When there is any pending traffic for a sleeping slave, the slave controller is signaled to

wake up and vote again for connection. Consequently, the OCP connection will be restored and any stalled traffic will be sent to the slave.

Linking the OCP Disconnect Protocol with Power Management

As established previously, the OCP disconnection state is a pre-requisite state for all OCP interfaces at a domain boundary, before performing a domain sleep transition. The OCP disconnect protocol can easily be linked to the power manager module to comply with this requirement.

Slave IP

Consider a power-managed slave IP, having a dedicated power-management protocol with the power manager, namely a SleepReq/SleepAck handshake.

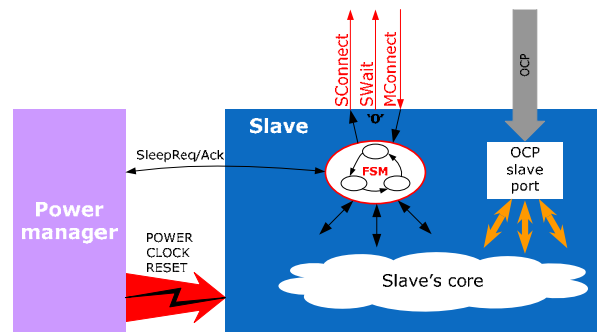


Figure 3: Power-managed Slave IP

Before performing a sleep transition, the power manager sends a sleep request to the slave, and waits for the corresponding acknowledge. Within the slave, a finite state machine performs any necessary actions before sending the acknowledge signal. One such action is to perform the OCP disconnection by de-asserting the SConnect signal and then waiting for a disconnected state (M_DISC or M_OFF). By using this sequence and the OCP disconnect protocol, we can be sure that the OCP socket is cleanly stopped before performing a sleep transition.

Conversely, the power manager will perform a wakeup transition with respect to power/clock/reset before requesting the slave to wake up. In turn, the FSM will ensure that the OCP slave port is functional before voting again for an OCP connection by asserting SConnect signal.

Note that in this particular case, the SWait signal is tied low as there is no need for the slave to stall the transition of the OCP connection status.

Master IP

Now consider a power-managed master IP, having a dedicated power-management protocol with the power manager, namely SleepReq/SleepAck handshake.

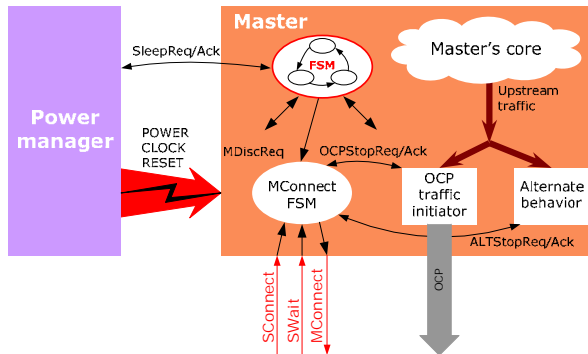


Figure 4: Power-managed Master IP

Before performing a sleep transition, the power manager sends a sleep request to the master, then waits for the corresponding acknowledge before performing the sleep transition. Within the master, a finite state machine performs any necessary actions before sending the acknowledge signal. One such action is to perform the OCP disconnection by asserting the MDiscReq signal and then waiting for the OCP connection status to reach the M_OFF state. Note that it is likely that the initiator was already in a quiet state before the power manager took the decision to put it to sleep (as the power manager knows the master's activity). Consequently, disconnection should be straightforward.

Conversely, the power manager will perform a wakeup transition with respect to power/clock/reset before requesting the master to wake up. The FSM will again vote for an OCP connection by de-asserting MDiscReq and turning on any other necessary components within the master.

Interconnect IP

A power-managed interconnect IP, having a dedicated power-management protocol with the power manager (SleepReq/SleepAck handshake) will be treated as a combination of the above two cases. The interconnect IP consists of N slave ports, M master ports and a core (switch fabric).

Before performing a sleep transition, the power manager sends a sleep request to the interconnect IP, then waits for the corresponding acknowledge. Within the IP, a finite state machine performs any necessary actions before sending the acknowledge signal. The sequence of actions could be: first

perform the OCP disconnection at the N slave ports, by de-asserting the SConnect signal and waiting for a disconnected state (M_DISC or M_OFF). Next, ensure the internal switch fabric is entirely empty of transactions. Finally, perform the OCP disconnection at the M master ports and wait for the OCP connection status to reach the M_OFF state.

Conversely, the power manager will perform a wakeup transition with respect to power/clock/reset before requesting the IP to wakeup. In turn, the FSM will ensure that the switch fabric is functional before voting again for an OCP connection on both its master and its slave ports.

Conclusion

There is a clear trend to use more and more power management techniques within SoCs. Multiple domain partitioning involves new architectural challenges. The OCP disconnect protocol has been primarily introduced to facilitate power management transitions at domain level. Conditioning the domain sleep transition to a prior OCP disconnection state ensures that the system will remain coherent and will not experience any disorder due to the OCP interfaces.

References

Open Core Protocol Specification 2.2 - Document Revision 1.0

About the Authors



Christophe Vatinel
Power Management System-on-Chip Architect
OMAP Platform Business Unit
Texas Instruments

Graduated from Ecole Polytechnique and Télécom Paris, France, in 1991. Before joining Texas Instruments Inc, Christophe was successively ASIC designer for Thomson-CSF, embedded DSP project lead for VLSI Technology Inc then Philips, and 2.5G base-band modem design lead for a startup in Wireless.