Q-Channel Protocol Implementation

Sanidhya Saxena, DESE, Indian Institute of Science, Bangalore-560012

I Q-Channel implementation

Typically a device and a controller are implemented with asynchronous clocks, that might be driven from the same source clock but with a significant phase difference. Below Figure 1shows the recommended implementation of the re-synchronization in this case.

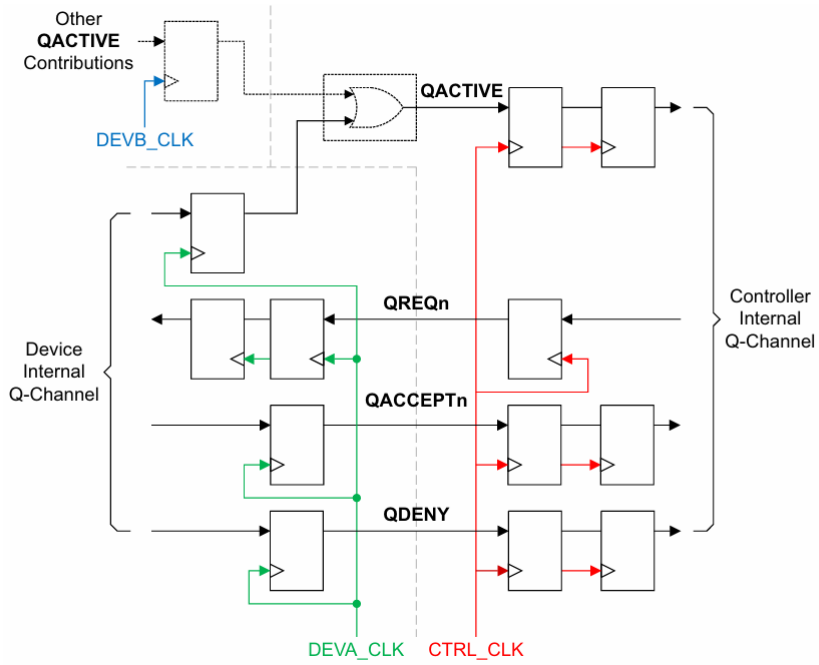

Figure 1: Recommended Implementation

For an asynchronous Q-Channel implementation ARM provides the following guidance to implementers:

- Re-synchronize all signals at the destination before use.
- Only remove clock or power when the interface is in the **Q_STOPPED** state.
- To ensure operation of the four-phase handshake, register all **QREQn**, **QACCEPTn**, and **QDENY** outputs.
- The QACTIVE signal is driven either directly by a register, or by a number of registers whose contributions are logically combined. ARM strongly recommends that this combining logic is limited to a logical OR, where possible, and is implemented using instantiated gates.
- When other logic is implemented, the implications of **QACTIVE** source register changes on the output **QACTIVE** signal must be carefully considered. Although the handshake protocol guarantees functionally correct behavior regardless of **QACTIVE** behavior, ARM recommends implementing the simplest possible logic to minimize the likelihood of introduced glitches at the **QACTIVE** output.

II CONTROLLER DESIGN

2.1 Low Power controller FSM

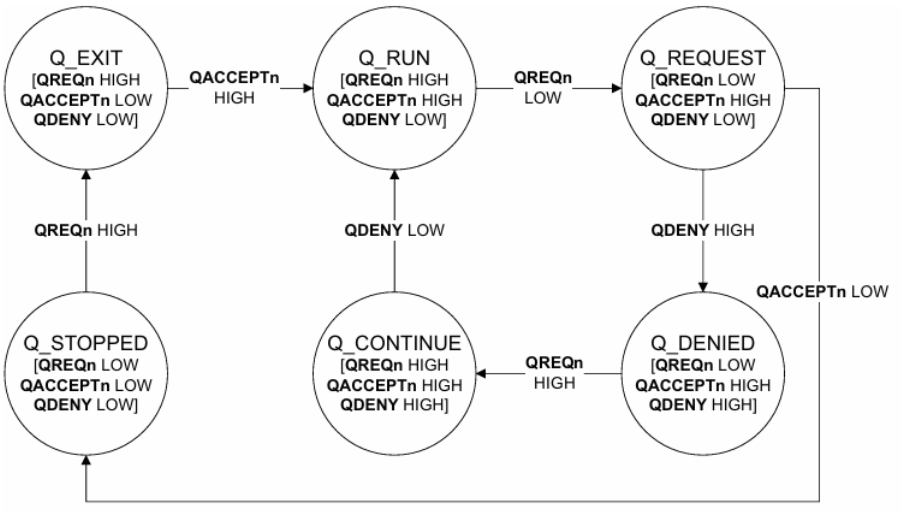

Figure 2: Low power FSM

2.2 QREQ_N Controller policies

Handshaking rules

- **QREQn** can only transition from **HIGH** to **LOW** when **QACCEPTn** is **HIGH** and **QDENY** is **LOW**.
- **QREQn** can only transition from **LOW** to **HIGH** when either:
  - **QACCEPTn** and **QDENY** are both **LOW**.
  - **QACCEPTn** and **QDENY** are both **HIGH**. •
- **QACCEPTn** can only transition from **HIGH** to **LOW** when **QREQn** is **LOW** and **QDENY** is **LOW**. •
- **QACCEPTn** can only transition from **LOW** to **HIGH** when **QREQn** is **HIGH** and **QDENY** is **LOW**. •
- **QDENY** can only transition from **HIGH** to **LOW** when **QREQn** is **HIGH** and **QACCEPTn** is **HIGH**. •
- **QDENY** can only transition from **LOW** to **HIGH** when **QREQn** is **LOW** and **QACCEPTn** is **HIGH**.

Asserting **QACTIVE HIGH** can be used as a stimulus for the controller to exit the **Q_STOPPED** state. The controller responds by driving **QREQn HIGH**, exiting the quiescent state.

Detection of qactive signal as high goes for 5 clock cycles and then the **up_active_signal** is aserted which will make the QREQn signal HIGH in Q_STOPPED state.

Detecting **QACTIVE LOW** can be used, by a controller in the **Q_RUN** state, as a criterion for initiating a quiescence request. However, the controller can change the state of QREQn from HIGH to LOW at any time while it is in the Q_RUN state.

Detection of qactive signal as low goes for 5 clock cycles and then the **down_active_signal** is aserted which will make the QREQn signal LOW in Q_RUN state.

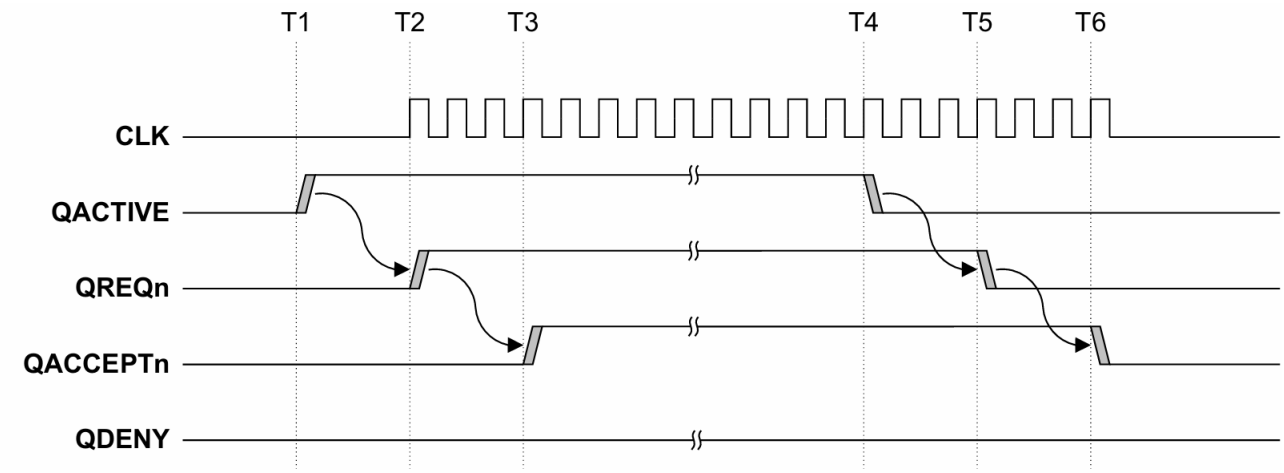

Figure 3: Device-led Q_EXIT and Q_REQUEST

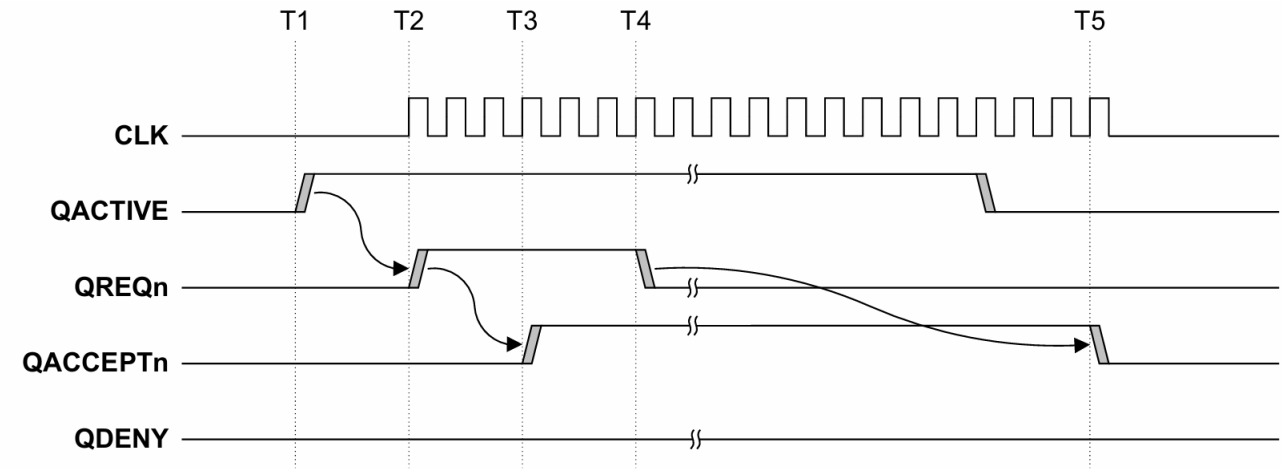

Figure 4: Device-led Q_EXIT and controller-led Q_REQUEST

2.3 DEVICE CLOCK GATING ENABLE

When the device will be in **Q_STOPPED** mode, the clock will be completely gated for the device using a signal **device_icg_enable** . This will enusre the power saving of the device.

III DEVICE DESIGN

3.1 Accepted quiescence request

Below Figure 5 shows a handshake sequence for an accepted quiescence request. It includes the guaranteed activity of an optional controller-supplied clock that is managed according to the interface semantics:

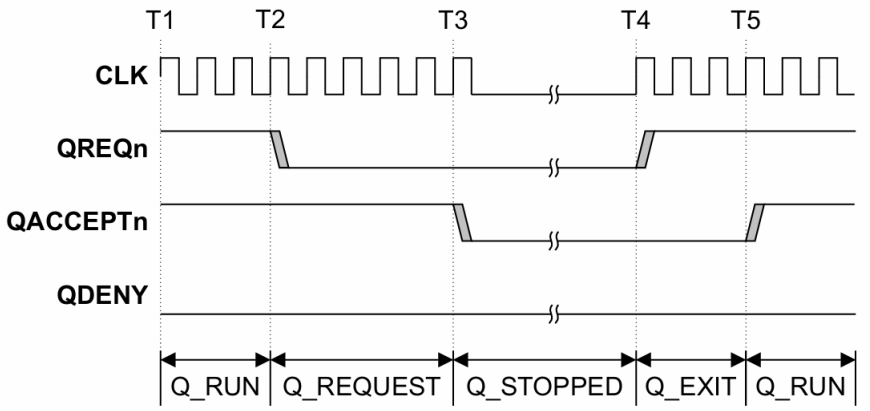

Figure 5: Q Channel Handshake

3.2 QACCEPT LOGIC

Qaccpet_o can only toggle if present state of machine is **Q_REQUEST** or **Q_EXIT.** Hence the qaccpet_enable is generated so that unwanted transition doesnt happen.

The nxt_qaccept transition from 1 to 0, when the FIFO is empty and Write to the FIFO is done and q_reqn is low. When the controller exits the Low power mode in **Q_EXIT** state, the qaccept signal goes high in next cycle and the state of machine again goes to **Q_RUN**.

```
assign qenable = (present_state == Q_REQUEST) | (present_state == Q_EXIT);
assign nxt_qaccept = ~( fifo_empty & wr_done_i & ~qreqn_i ) ;
always_ff @( posedge clk or posedge reset ) begin : QACCEPT_FLOP
        if(reset)
                qaccept_q <= 1'b1;
        else if(qenable)
                qaccept_q <= nxt_qaccept;
end
assign qacceptn_o = qaccept_q;
```

3.3 QACTIVE LOGIC

QACTIVE logic has two component, one is the flopped signal of qactive which is generated using the internal activity of FIFO and another comes from outside as an **asynchronous wakeup signal.**

For Activity, when either the **write_valid** or **read_valid** is 1 or if the FIFO is not empty. The external wakeup signal needs to be ensured that it is always driven from a flop.

```
assign qactive_o = if_wakeup_i | qactive_q ;
always_ff @( posedge clk or posedge reset ) begin
        if(reset)
                qactive_q <= 1'b0;
        else
                qactive_q <= (wr_valid_i | ~fifo_empty | rd_valid_i);
        end
```

3.4 FIFO_FLUSH LOGIC

When the qreq_n is asserted low, that means we have a request for Low power, Hence the read side needs to be informed that read all the data from FIFO, as we need to flush the FIFO. The write side revert back with a WRITE_DONE signal which ensures that no more writes will be on FIFO.

```
always_ff @( posedge clk or posedge reset ) begin : WRITE_FLUSH
        if(reset)
```

```
                wr_flush_o <= 1'b0;
        else begin
                if( !qreqn_i && !wr_flush_o) wr_flush_o <= 1'b1;
                if(wr_done_i) wr_flush_o <= 1'b0;
        end
end
```

IV Simulations

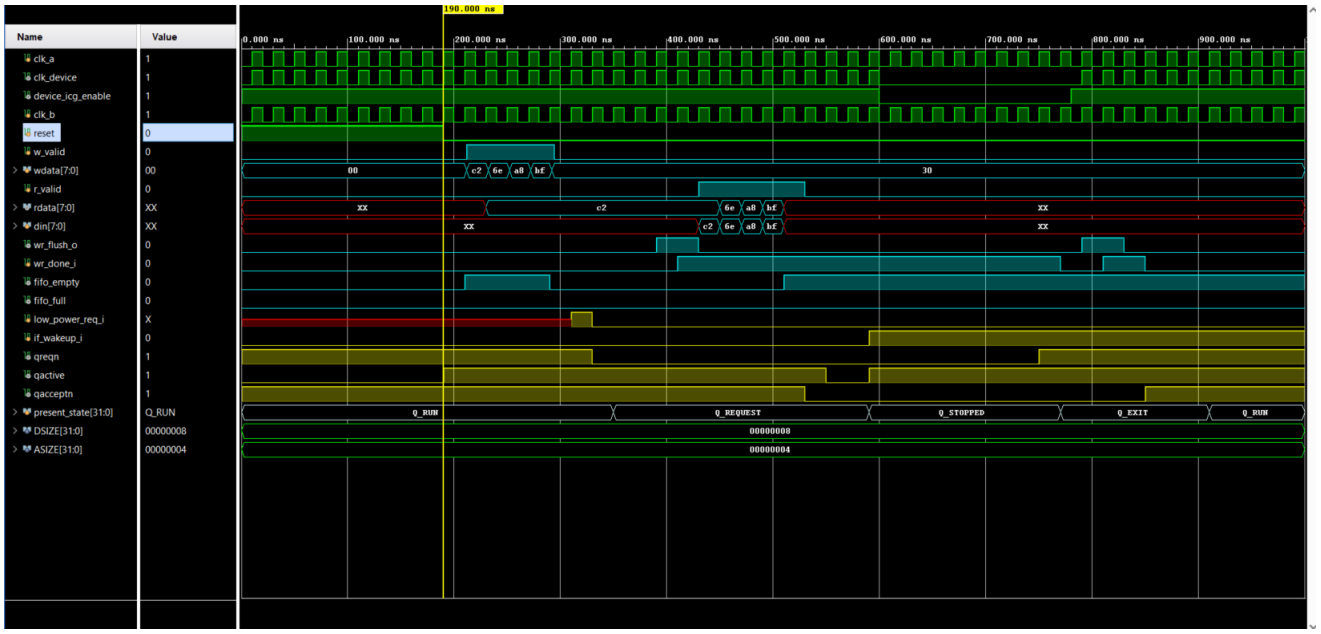

Figure 6: Simulations

Here the Green signals represents the GLOBAL SIGNALS like Clock and Resets. Clk_A is the global clock to device, Clk_B is the clock for Controller. Device_clk is the gated clock of Clk_A which is given to the device for Power saving applications.

1. When the device comes out of the reset, the write transation starts and master writes on to the FIFO.
2. When the external **low_power_req_i** is asserted the **qreqn** goes low and the state chnages to **Q_REQUEST**.
3. The **wr_fifo_flush** is asserted as we have to go in LP mode. The Write side gave the acknowledgement by asserting **wr_done** signal.
4. Since the we need to flush the FIFO, the read side starts reading the data present in the FIFO memory.
5. When the FIFO is empty that means it is safe to go in Low power mode, hence the **qacceptn** goes low and we enter the **Q_STOPPED** state.
6. In the **Q_STOPPED** state the clock to the device is gated.
7. When the external wakeup signal (**if_wakeup**) is asserted, the **qactive** goes high and after 5 cycles the **qreq_n** also goes high indicating the **EXIT** from the low power state.
8. In the **Q_EXIT** state the clock to the device is ensured.
9. When the **qaccept_n** is lifted high, it marks the entry of **Q_RUN** state and the device is back to functional state.