



APB Interconnect Core User Guide

UG-CORE-APB-INTERCONNECT-v1.0
December 2024
www.efinixinc.com



Contents

- Introduction..... 3**
- Features.....3**
- Device Support..... 3**
- Resource Utilization and Performance.....4**
- Release Notes..... 5**
- Functional Description.....6**
 - Ports..... 7
 - APB Interconnect Operations.....10
 - Write Operation..... 11
 - Read Operation.....13
 - Optional Pipeline Register..... 15
 - Clock and Reset.....17
 - Arbitration Mode..... 19
- IP Manager..... 21**
- Customizing the APB Interconnect Core..... 22**
- APB Interconnect Core Example Design.....24**
- APB Interconnect Testbench..... 25**
- Revision History.....25**

Introduction

The APB Interconnect core provides solution to connect multiple APB masters to a single APB slave. The APB Interconnect core supports two arbitration modes to determine the grant access of the masters. The APB Interconnect core operates based on the Advanced Microcontroller Bus Architecture (AMBA) 3 Advanced Peripheral Bus (APB) protocol specifications version 1.0.

Use the IP Manager to select IP, customize it, and generate files. The APB Interconnect core has an interactive wizard to help you set parameters. The wizard also has options to create a testbench and/or example design targeting an Efinix® development board.

Features

The APB Interconnect core includes the following features:

- Compliant with AMBA 3 APB Protocol specifications
- Data width up to 32 bits [1, 2, ..., 32]
- Address width up to 32 bits [1, 2, ..., 32]
- **Fixed-Priority** and **Round-Robin** arbitration modes
- Supports arbitration up to 32 masters to a single slave
- Optional pipeline registers to optimize latency or/and performance
- Include indicators on which master is being granted the request

Device Support

Table 1: APB Interconnect Core Device Support

FPGA Family	Supported Device
Trion	All
Titanium	All
Topaz	All

Resource Utilization and Performance



Note: The resources and performance values provided are based on some of the supported FPGAs. These values are just guidance and can change depending on the device resource utilization, design congestion, and user design.

Table 2: Titanium Resource Utilization and Performance on PCIe Application

FPGA	Arbitration Mode	Resource			f_{MAX} (MHz) ⁽¹⁾	Efinity Version ⁽²⁾
		Logic Elements (Logic, Adders, Flipflops, etc.)	Memory Block	DSP Block		
Ti375 N1156 C4	Fixed-Priority	1905/362,880 (0.52%)	0/2,688 (0%)	0/1,344 (0%)	210	2024.2
	Round-Robin	2070/362,880 (0.57%)	0/2,688 (0%)	0/1,344 (0%)	194	



Note:

- The resource utilization is gathered from a design where the APB Interconnect core arbitrates 32 masters (the maximum number) with the quad 0 APB in Ti375N1156C4. The APB Interconnect core is configured with zero pipeline register to optimize latency.
- You may choose to optimize f_{max} to meet your targeted performance by enabling any (or all) pipeline registers at Masters-APB Interconnect interface and APB Interconnect-slave interface.

Table 3: Titanium Resource Utilization and Performance on Soft IP Application

FPGA	Arbitration Mode	Resource			f_{MAX} (MHz) ⁽¹⁾	Efinity Version ⁽²⁾
		Logic Elements (Logic, Adders, Flipflops, etc.)	Memory Block	DSP Block		
Ti375 N1156 C4	Fixed-Priority	896/362,880 (0.52%)	0/2,688 (0%)	0/1,344 (0%)	261	2024.2
	Round-Robin	1633/362,880 (0.03%)	0/2,688 (0%)	0/1,344 (0%)	248	



Note:

- The resource utilization is gathered from a design where the APB Interconnect arbitrates 32 masters (the maximum number) with another soft IP as a slave in the design. Here, the APB Interconnect core is configured with zero pipeline register to optimize latency.
- You may choose to optimize f_{max} to meet your targeted performance by enabling any (or all) pipeline registers at Masters-APB Interconnect interface and APB Interconnect-slave interface.

⁽¹⁾ Using default parameter settings.

⁽²⁾ Using System Verilog 2005.

Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes are available on the [Efinity Downloads](#) page under each Efinity software release version.



Note: You must be logged in to the Support Center to view the IP Core Release Notes.

Functional Description

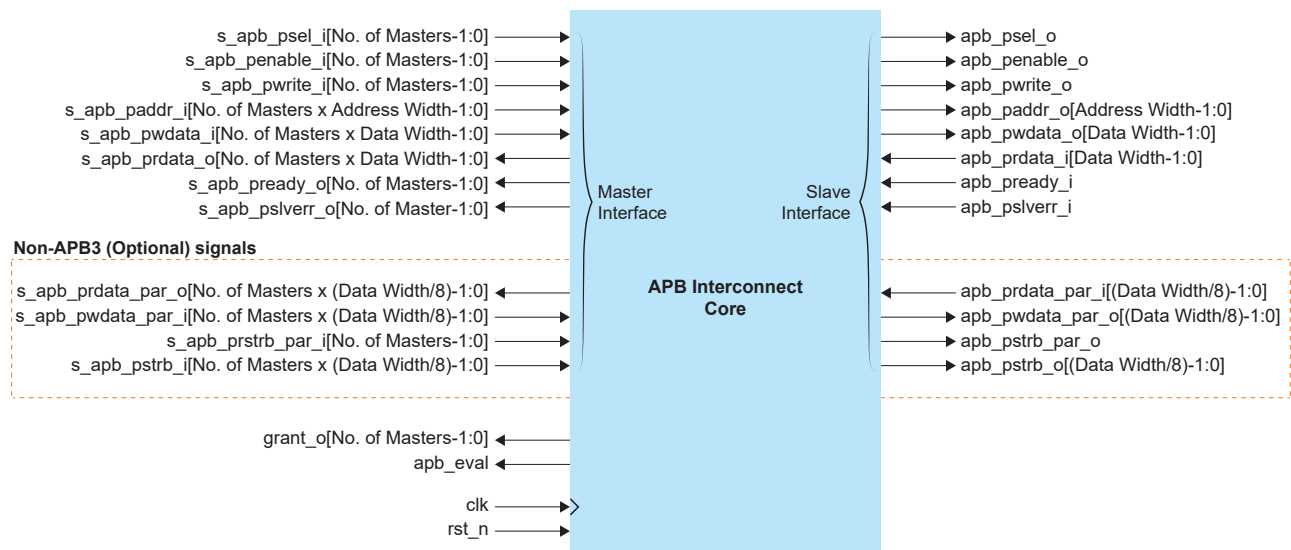
The APB Interconnect core is a bridge to arbitrate among multiple masters to a single slave based on priority level. The APB Interconnect core grants request to master with the selected priority. The APB Interconnect core has two main interfaces:

- *APB Master Interface*—Provides 32 master interfaces to the APB Interconnect core.
- *APB Slave Interface*—Supports single slave interface.

The APB Interconnect core operates based on AMBA 3 APB protocol specifications. The following describes the general operation of the APB Interconnect core:

- When `apb_eval` is asserted, it indicates that APB Interconnect is in EVALUATION state, where it analyzes all masters' request and decide which master to grant.
- During EVALUATION state (i.e., `apb_eval = 1`), any assertion of `PSEL` marks the start of the APB request.
 - For an APB write request, `PWRITE` needs to assert too. `PADDR` and `PWDATA` contain destination address and the intended `DATA` to be written.
 - For an APB read request, `PWRITE` needs to be 0 while `PADDR` contains the address of the register to be read.
- The `PENABLE` signal needs to be 1 at one clock cycle after the assertion of `PSEL` for the APB request to be successful. The `PENABLE` assertion marks the start of transfer of the APB request.
- The `PREADY` signal is a return or response signal from the slave. The assertion `PREADY` indicates that the slave has completed the APB request.
- The `PRDATA` signal contains the result of the APB read request and it is valid when `PREADY` is 1.
- Masters' input signals (`PSEL`, `PENABLE`, `PWRITE`, `PADDR`, and `PWDATA`) need to be maintained throughout the APB request, i.e., from the assertion of `PSEL` until the assertion of `PREADY`.
- Upon receiving `PREADY`, master may change the `PSEL`, `PENABLE`, `PWRITE`, `PADDR`, and `PWDATA` signals for the next request or transfer.

Figure 1: APB Interconnect Core Block Diagram



Ports

Table 4: APB Interconnect General Ports

Port	Direction	Clock Domain	Description
clk	Input	-	Clock source for APB Interconnect core.
rst_n	Input	-	Active low asynchronous reset.
apb_eval	Output	clk	The assertion of this signal indicates that the APB Interconnect core is in EVALUATION state. When this signal is asserted, APB Interconnect core evaluates and decides which master to be granted a request. When this signal is de-asserted, it means that APB Interconnect core has exited the EVALUATION state and is actively granting a request to one of the masters.
grant_o[No. of Master-1:0]	Output	clk	This is a one-hot signal. The binary position of 1 in this signal indicates which master is being granted a request. For example, <ul style="list-style-type: none"> grant_o = 4'b0001 means Master_0 is being granted a request. grant_o = 10'b00_0010_0000 means Master_5 is being granted a request. When grant_o is all-zeros, it means no master is being granted a request.

Table 5: APB Interconnect Master Interface Ports

Port	Direction	Clock Domain	Description
s_apb_psel_i[No. of Master-1:0]	Input	clk	APB select signal from the master, to request an APB transfer.
s_apb_penable_i[No. of Master-1:0]	Input	clk	APB enable signal from the master, to indicate the start of the APB request.
s_apb_pwrite_i[No. of Master-1:0]	Input	clk	APB write signal from the master, to indicate whether the APB request is a write or read request.
s_apb_paddr_i[No. of Master*Address Width-1:0]	Input	clk	APB address signal from the master, which contains the location of the destination register.
s_apb_pwdata_i[No. of Master*Data Width-1:0]	Input	clk	APB write data signal from the master, which contains the data content to be written into the destination register. This signal is only effective if APB write signal is 1.
s_apb_pwdata_par_i[No. of Master*(Data Width/8)-1:0]	Input	clk	APB write data parity signal from the master, which contains the data parity of the APB write data. This signal is only effective if APB write signal is 1.
s_apb_prdata_o[No. of Master*Data Width-1:0]	Output	clk	APB read data signal to the master, which contains data read from destination register (slave) as the result of APB read request. This signal is cycle accurate with APB ready signal.
s_apb_prdata_par_o[No. of Master*(Data Width/8)-1:0]	Output	clk	APB read data parity signal to the master, which contains data parity from the slave as the result of APB read request. This signal is cycle accurate with APB read data and APB ready signal.
s_apb_pready_o[No. of Master-1:0]	Output	clk	APB ready signal from the slave to the master. The assertion of this signal indicates that the active APB transfer has been completed.
s_apb_pslverr_o[No. of Master-1:0]	Output	clk	APB error signal from the slave to the master. The assertion of this signal indicates that an error has occurred during the APB transfer.
s_apb_pstrb_i[No. of Master*(Data Width/8)-1:0]	Input	clk	<p>APB write strobe signal from the master. This signal enables the write request with byte resolution. Each bit of this signal correlates with a byte of APB write data.</p> <p>For example, when APB write strobe is 4'b1001, both APB Write Data[31:24] and APB Write Data[7:0] are intended to be written to the Destination Register[31:24] and Destination Register[7:0], whereas APB Write Data[23:16] and APB Write Data[15:8] are not intended to be processed, result in Destination Register[23:16] and Destination Register[15:8] retaining their original content.</p> <p>This signal is effective only during write request.</p>
s_apb_pstrb_par_i[Number of Master-1:0]	Input	clk	APB write strobe parity signal from the master, which is effective during write request only.

Table 6: APB Interconnect Slave Interface Ports

Port	Direction	Clock Domain	Description
apb_psel_o	Output	clk	APB select signal from the APB Interconnect core to the slave, to trigger an APB request.
apb_penable_o	Output	clk	APB enable signal from APB Interconnect core to slave, to start an APB transfer at 1 clock cycle after APB select. To comply with APB protocol, the APB Interconnect core evaluates the masters' APB enable at 1 clock cycle after the assertion of masters' APB select, to determine the start of the APB transfer. Refer to APB Interconnect Operations on page 10.
apb_pwrite_o	Output	clk	APB write signal from the APB Interconnect core to the slave. When this signal is 1, it refers to an APB write request. When this signal is 0, it refers to an APB read request.
apb_paddr_o[Address Width-1:0]	Output	clk	APB address from the APB Interconnect core to slave. Contains the address of the destination location.
apb_pwdata_o[Data Width-1:0]	Output	clk	APB write data signal from the APB Interconnect core to the slave, contains data content to be written into destination registers. This signal is effective only during a write request.
apb_pwdata_par_o[(DataWidth/8)-1:0]	Output	clk	APB write data parity signal from the APB Interconnect core to the slave, contains the data parity of the APB write data. This signal is only effective if an APB write signal is 1.
apb_pstrb_o[(Data Width/8)-1:0]	Output	clk	APB write strobe signal from the APB Interconnect core to the slave. This signal enables the write request with byte resolution. Each bit of this signal correlates with a byte of the APB write data. For example, when APB write strobe is 4'b1001, both APB Write Data[31:24] and APB Write Data[7:0] are intended to be written to the Destination Register[31:24] and Destination Register[7:0], whereas APB Write Data[23:16] and APB Write Data[15:8] are not intended to be processed, results in Destination Register[23:16] and Destination Register[15:8] retaining their original content. This signal is effective only during a write request.
apb_pstrb_par_o	Output	clk	APB write strobe parity signal from the APB Interconnect core to the slave, which is effective only during a write request.
apb_prdata_i[Data Width-1:0]	Input	clk	APB read data signal from the slave to the APB Interconnect core which contains data read from the destination register (slave) as the result of an APB read request. This signal is cycle accurate with the APB ready signal.
apb_prdata_par_i[(Data Width/8)-1:0]	Input	clk	APB read data parity signal from the slave to the APB Interconnect core which contains data parity from the slave as a result of APB read request. This signal is cycle accurate with APB read data and APB ready signal.
apb_pready_i	Input	clk	APB ready signal from the slave to the APB Interconnect core. The assertion of this signal indicates that the active APB transfer has been completed.
apb_pslverr_i	Input	clk	APB error signal from the slave to the APB Interconnect core. The assertion of this signal indicates that an error has occurred during the APB transfer.

APB Interconnect Operations

Based on the APB protocol, a successful APB transfer starts with asserting a group of APB signals such as `PSEL`, `PWRITE`, `PADDR`, `PWDATA`, `PWDATA_PAR`, `PSTRB`, and `PSTRB_PAR`. The `PENABLE` signal needs to assert at 1 clock cycle after asserting these signals. All these signals need to maintain their states throughout the entire transfer. The assertion of `PREADY` from the slave indicates that the active APB request is completed and marks the end of the APB transfer. At the end of the APB transfer, the master needs to de-assert the `PENABLE` signal and may change the signal state of `PSEL`, `PWRITE`, `PADDR`, `PWDATA`, `PWDATA_PAR`, `PSTRB`, and `PSTRB_PAR` for the next APB transfer (or no APB request).

The assertion of `apb_eval` signal indicates that the APB Interconnect core is in an `EVALUATION` state, where it evaluates all available assertions of `PSEL` from any master. It then decides and selects which master to be granted the APB request based on the selected priority.

To comply with the APB protocol, the APB Interconnect core evaluates the granted master's `PENABLE` at 1 clock cycle after the assertion of the master's `PSEL`, to determine the start of the APB transfer. There are two situations:

- If the granted master's `PENABLE` is 1, it marks the start of APB transfer, where the APB Interconnect core exits the `EVALUATION` state to enter the `TRANSFER` state. Concurrently, the `grant_o` signal updates to indicate which master is being granted a request.
- If the granted master's `PENABLE` is 0, there is no APB transfer. The `apb_eval` signal maintains 1, but the APB Interconnect core starts to evaluate the master of next priority. If the master of the next priority fulfills the `PENABLE` criteria, the APB Interconnect core exits the evaluate state and updates `grant_o`. Otherwise, the APB Interconnect core stays in the `EVALUATION` state and moves to the master of next priority.

The assertion of `PREADY` from the slave indicates the end of active APB transfer. If the slave does not assert `PREADY`, the APB Interconnect core continues to wait since there is no timeout mechanism in the APB Interconnect core. When this occurs, you may need to reset this deadlock by asserting the `rst_n`.

Tip: You may implement an independent timeout mechanism to determine whether a deadlock has occurred and to decide when to perform the reset.

Write Operation

The write transfer is determined by the assertion of the granted master's `s_apb_pwrite_i`. The `s_apb_pwrite_i` needs to be asserted together with the granted master's `s_apb_psel_i`, `s_apb_paddr_i`, `s_apb_pwdata_i`, `s_apb_pwdata_par_i`, `s_apb_pstrb_i`, and `s_apb_pstrb_par_i`.

At 1 clock cycle after the assertion of the granted master's `s_apb_psel_i`, the granted master's `s_apb_penable_i` needs to be asserted to start the write transfer.



Note: If the granted master's `s_apb_penable_i` does not assert or assert at ≥ 1 clock cycle after the assertion of the `s_apb_psel_i`, the granted master loses the active priority.

The minimum latency between the granted master's input signals and the slave's interface output signals is 2 clock cycles. Upon receiving a write transfer request from the granted master, the slave may need some time to process the request.

While waiting for the slave to complete the write transfer, the granted master needs to maintain the signal state of `s_apb_psel_i`, `s_apb_penable_i`, `s_apb_pwrite_i`, `s_apb_paddr_i`, `s_apb_pwdata_i`, `s_apb_pwdata_par_i`, `s_apb_pstrb_i`, and `s_apb_pstrb_par_i`.

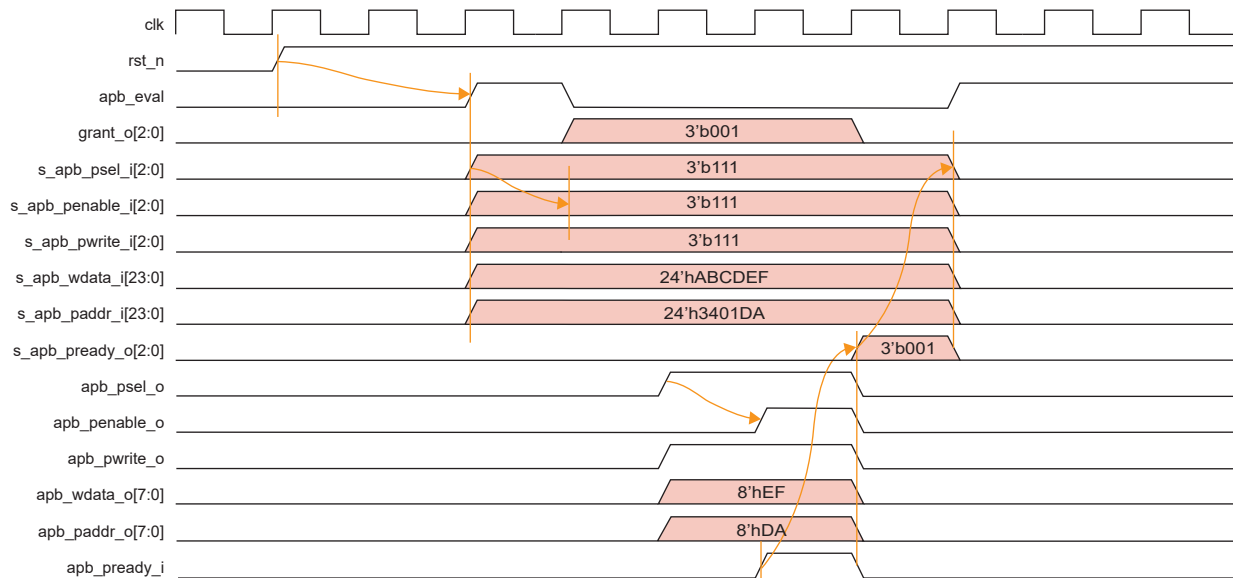
When the slave finishes the write transfer, it asserts `apb_pready_i`. The APB Interconnect core takes a minimum of 1 clock cycle to propagate the asserted `apb_pready_i` to the granted master's `s_apb_pready_o`. The assertion of granted master's `s_apb_pready_o` marks the end of the current write transfer.

At the end of the write transfer, the granted master needs to deassert `s_apb_penable_i` and may change the signal state of `s_apb_psel_i`, `s_apb_pwrite_i`, `s_apb_paddr_i`, `s_apb_pwdata_i`, `s_apb_pwdata_par_i`, `s_apb_pstrb_i`, and `s_apb_pstrb_par_i` for the next APB request (or no request).



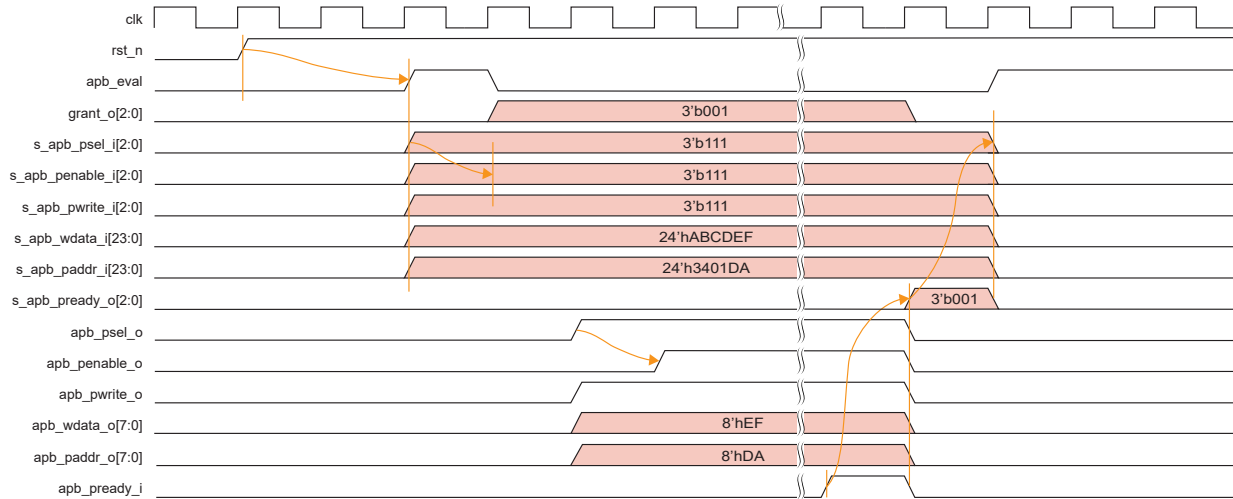
Note: Figure 2 to Figure 5 is without pipeline register. The master priority is set as default.

Figure 2: Write Operation with No Wait State



The write operation with no wait state illustrates a situation where the slave responds immediately to the granted master's request, i.e., the slave asserts `apb_pready_i` at the same clock cycle as the assertion of `apb_penable_o`.

Figure 3: Write Operation with Wait State



The write operation with wait state is a typical situation, where upon receiving the granted master's request, the slave may take ≥ 1 clock cycle to response.

Read Operation

The read transfer is determined by the 0 state of the granted master's `s_apb_pwrite_i` during the APB request. The 0 state of `s_apb_pwrite_i` is evaluated together with the assertion of the granted master's `s_apb_psel_i`, `s_apb_paddr_i`, `s_apb_pwdata_i`, `s_apb_pwdata_par_i`, `s_apb_pstrb_i`, and `s_apb_pstrb_par_i`.

At 1 clock cycle after the assertion of the granted master's `s_apb_psel_i`, the granted master's `s_apb_penable_i` needs to be asserted to start the read transfer.



Note: If the granted master's `s_apb_penable_i` does not assert or assert at ≥ 1 clock cycle after the assertion of the `s_apb_psel_i`, the granted master loses the active priority.

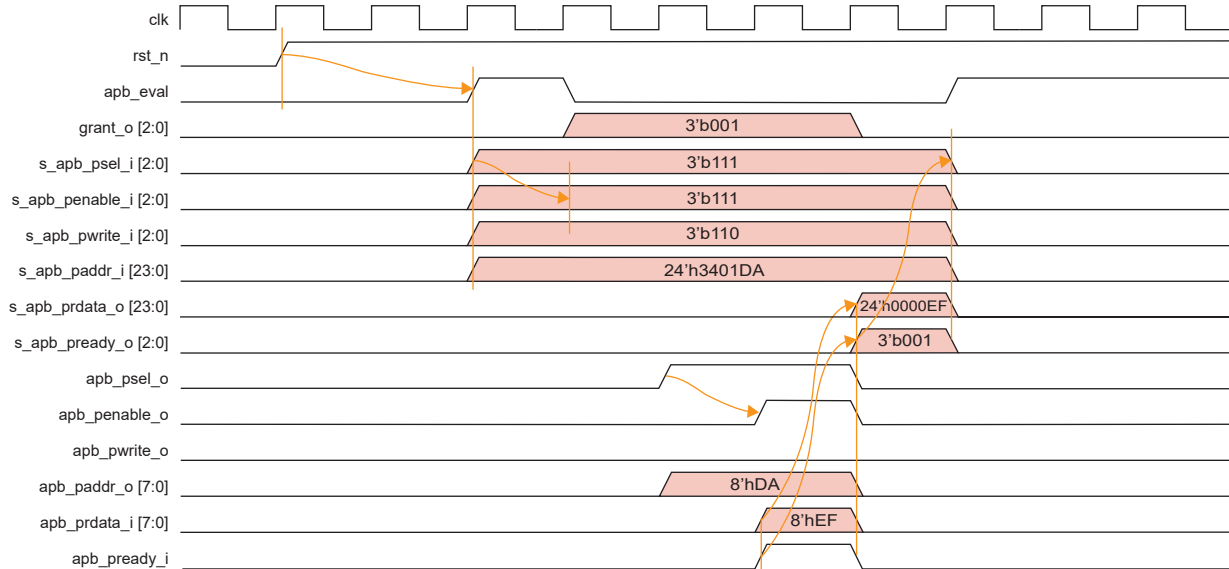
The minimum latency between the granted master's input signals and the slave's interface output signals is 2 clock cycles. Upon receiving a read transfer request from the granted master, the slave may need some time to process the request.

While waiting for the slave to complete the read transfer, the granted master needs to maintain the signal state of `s_apb_psel_i`, `s_apb_penable_i`, `s_apb_pwrite_i`, `s_apb_paddr_i`, `s_apb_pwdata_i`, `s_apb_pwdata_par_i`, `s_apb_pstrb_i`, and `s_apb_pstrb_par_i`.

When the slave finishes the read transfer, it asserts `apb_pready_i`. The APB Interconnect core takes a minimum of 1 clock cycle to propagate the asserted `apb_ready_i` to the granted master's `s_apb_pready_o`. This marks the end of the current read transfer.

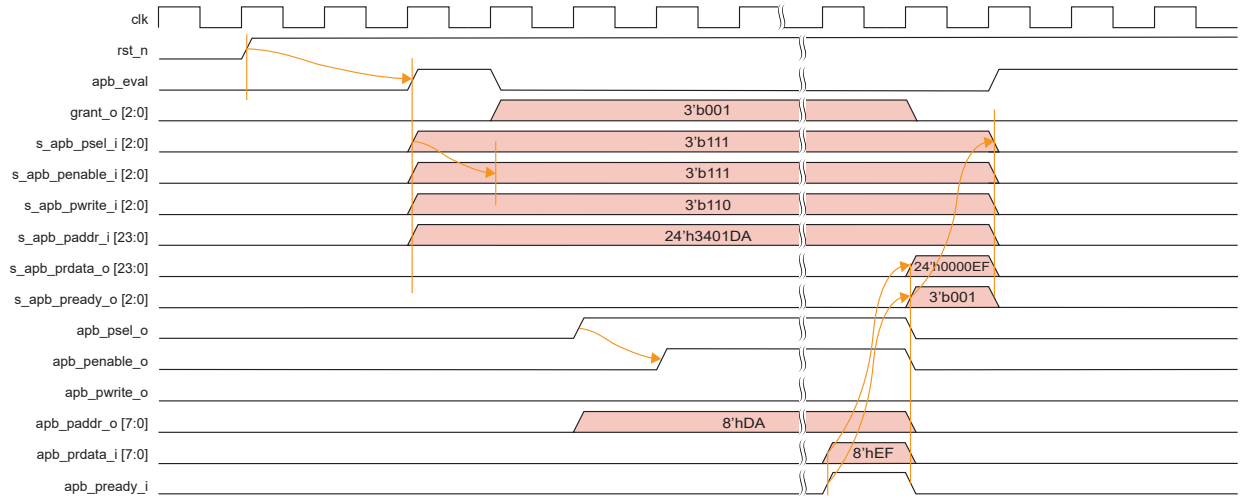
At the end of the read transfer, the granted master needs to de-assert `s_apb_penable_i` and may change the signal state of `s_apb_psel_i`, `s_apb_pwrite_i`, `s_apb_paddr_i`, `s_apb_pwdata_i`, `s_apb_pwdata_par_i`, `s_apb_pstrb_i`, and `s_apb_pstrb_par_i` for the next APB request (or no request).

Figure 4: Read Operation with No Wait State



The read operation with no wait state illustrates a situation where the slave responds immediately to the granted master's request, i.e., the slave asserts `apb_pready_i` at the same clock cycle as the assertion of `apb_penable_o`.

Figure 5: Read Operation with Wait State



The read operation with `wait` state is a typical situation, where upon receiving the granted master's request, the slave may take ≥ 1 clock cycle to response.

Optional Pipeline Register

The APB Interconnect core allows user to optimize latency and/or performance with the optional pipeline registers.

- *Input Register*—Enable this register to optimize timing but adds one clock latency at the master interface input paths that includes s_apb_psel_i, s_apb_penable_i, s_apb_pwrite_i, s_apb_paddr_i, s_apb_pdata_i, s_apb_pdata_par_i, s_apb_pstrb_i, and s_apb_pstrb_par_i.
- *Output Register*—Enable this register to optimize timing but adds one clock latency at the master interface output paths that includes s_apb_pready_o, s_apb_pslverr_o, s_apb_prdata_o, and s_apb_prdata_par_o.
- *Optimize Latency*—Enable this parameter to optimize latency at the slave interface that includes apb_sel_o, apb_penable_o, apb_pwrite_o, apb_paddr_o, apb_pdata_o, apb_pdata_par_o, apb_pstrb_o, and apb_pstrb_par_o.



Note: When both the input and output register are enabled, there are an additional 2 latencies.

Figure 6: Read Operation Waveform with Input Register Enabled

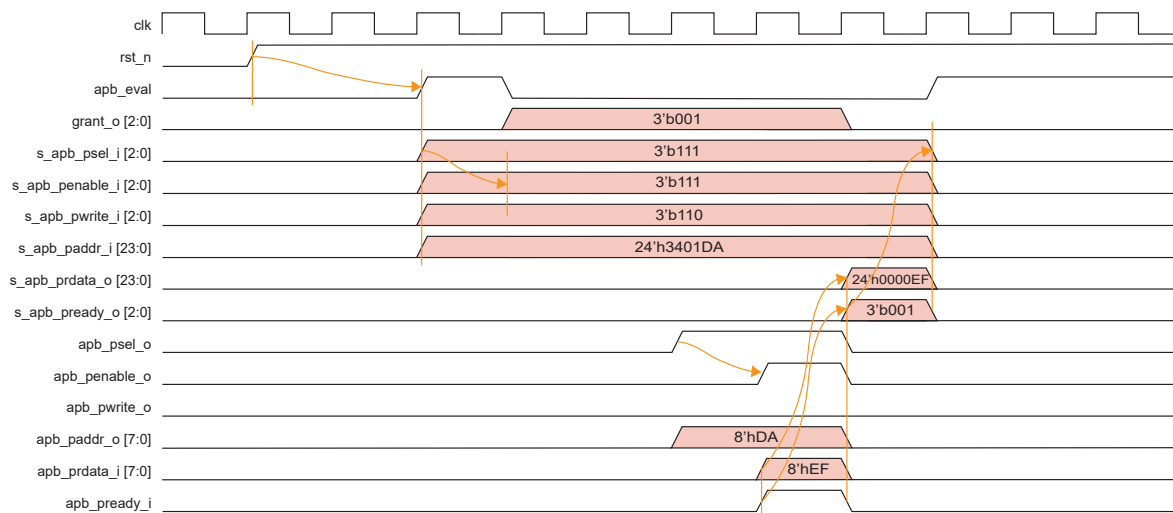


Figure 7: Read Operation Waveform with Output Register Enabled.

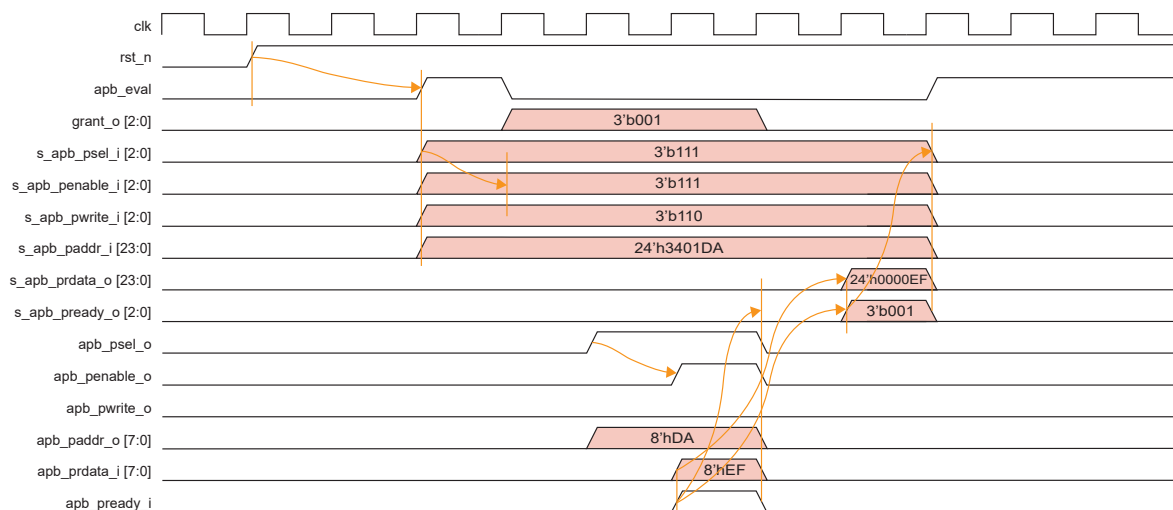


Figure 8: Read Operation Waveform with Input Register and Output Register Enabled

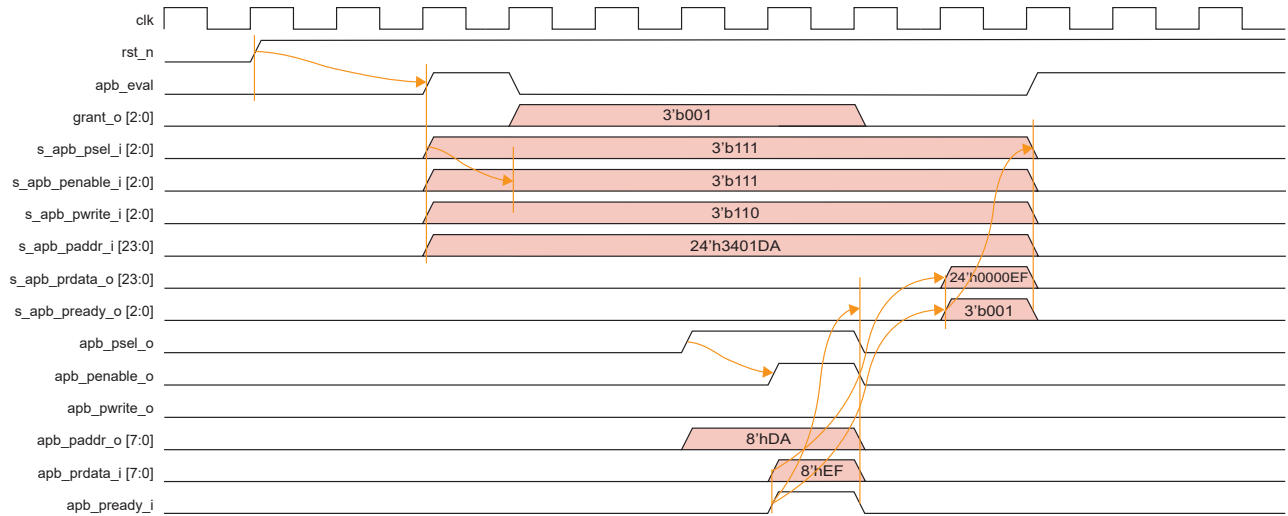
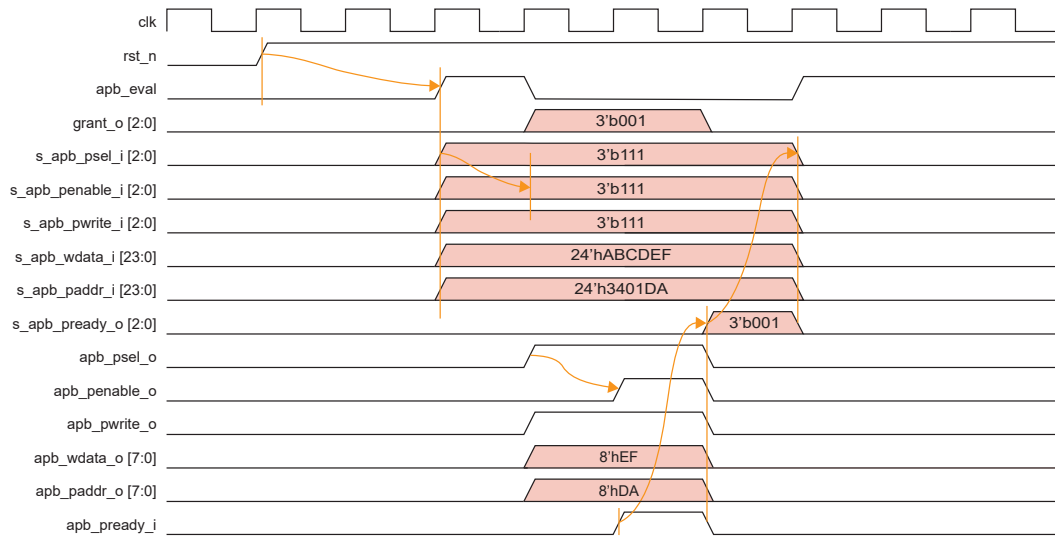


Figure 9: Write Operation Waveform with Optimize Latency Enabled



Clock and Reset

The APB Interconnect core has only 1 clock source (`clk`) and 1 reset signal (`rst_n`), and operates in a single clock domain. Efinix recommends the same clock domain for both the APB Interconnect core and the slave.

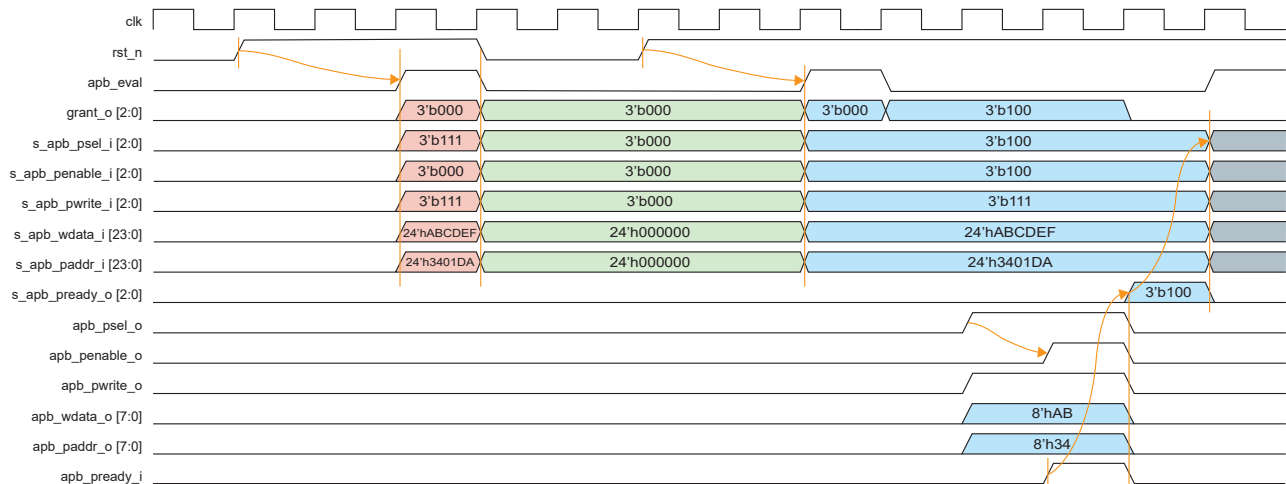
The reset signal, `rst_n` is an asynchronous reset. However, the de-assertion of the reset signal is synchronous. An internal reset synchronizer is available to synchronize the de-assertion of `rst_n` to `clk` domain. The reset synchronizer has 1 to 2 clock cycles latency.

The `rst_n` is an active low signal that needs to be initialized to 0 before the start of any operation.

Upon assertion of the reset signal, i.e., `rst_n = 0`, the entire APB Interconnect core resets. In the **Round-Robin** mode, the grant access resets to Master 0. In **Fixed-Priority** mode, the grant access remains unchanged, i.e., it is based on the masters' priority and the assertion of `s_apb_psel_i`.

Upon de-assertion of the reset signal, i.e., `rst_n = 1`, the APB Interconnect core becomes operational after 1 to 2 clock cycles of reset synchronization. The `apb_eval` signal becomes high, indicating that the APB Interconnect core is in an EVALUATION state, where the evaluation of all the masters' requests takes place to decide which master is to be granted access.

Figure 10: Single Reset Domain on All Modules (Slave, APB Interconnect Core, and Masters)



Warning: Efinix recommends that the APB Interconnect core shares the same reset domain as the slave. For cases where the reset is different between the slave and the APB Interconnect core, there may be a situation of potential deadlock where the operational APB Interconnect core is actively waiting for the assertion of `apb_pready_i` from the slave, which may not occur if the slave is in a reset.

The APB Interconnect core has a built-in latch mechanism to cater for a situation where the granted master does not comply with the APB protocol of maintaining the state of signals: `s_apb_sel_i`, `s_apb_penable_i`, `s_apb_pwrite_i`, `s_apb_paddr_i`, `s_apb_pwdata_i`, `s_apb_pwdata_par_i`, `s_apb_pstrb_i`, and `s_apb_pstrb_par_i` until the return of `s_apb_pready_o`.

This non-compliance situation may be caused by the following:

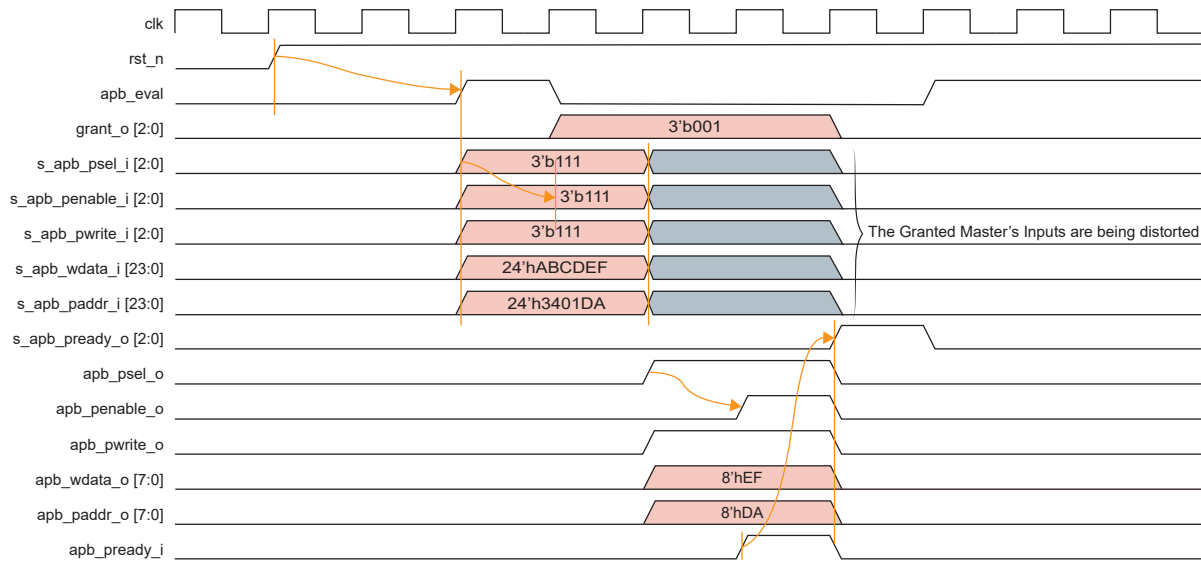
- Different reset domains between the granted master and the APB Interconnect core, where the granted master may encounter reset in the midst of an APB transfer.
- Signal corruption in the granted master during the APB transfer.

Figure 11: Reset Master on Different Domains with APB Interconnect on page 18 illustrates a situation where the `Master_0` is requesting an APB write transfer, but an independent reset event distorts `Master_0`'s `s_apb_psel_i`, `s_apb_penable_i`, `s_apb_pwrite_i`, `s_apb_paddr_i`, `s_apb_pwdata_i` before the return of `s_apb_pready_o`. As indicated by `grant_o`, the APB Interconnect core has successfully granted `Master_0`'s request and propagates the granted request to the slave. When the slave responds with the assertion of `apb_pready_i`, the `Master_0`'s `s_apb_pready_o` asserts too, marking the completion of the write transfer, despite the distorted `s_apb_sel_i` in the granted `Master_0`.

Using this same illustration, if the signal `grant_o` does not update to `3'b001`, it indicates that the APB Interconnect core is not successful in capturing the `Master_0`'s request. Here, there is no grant request for `Master_0`.

Figure 11: Reset Master on Different Domains with APB Interconnect

Only master modules are reset.

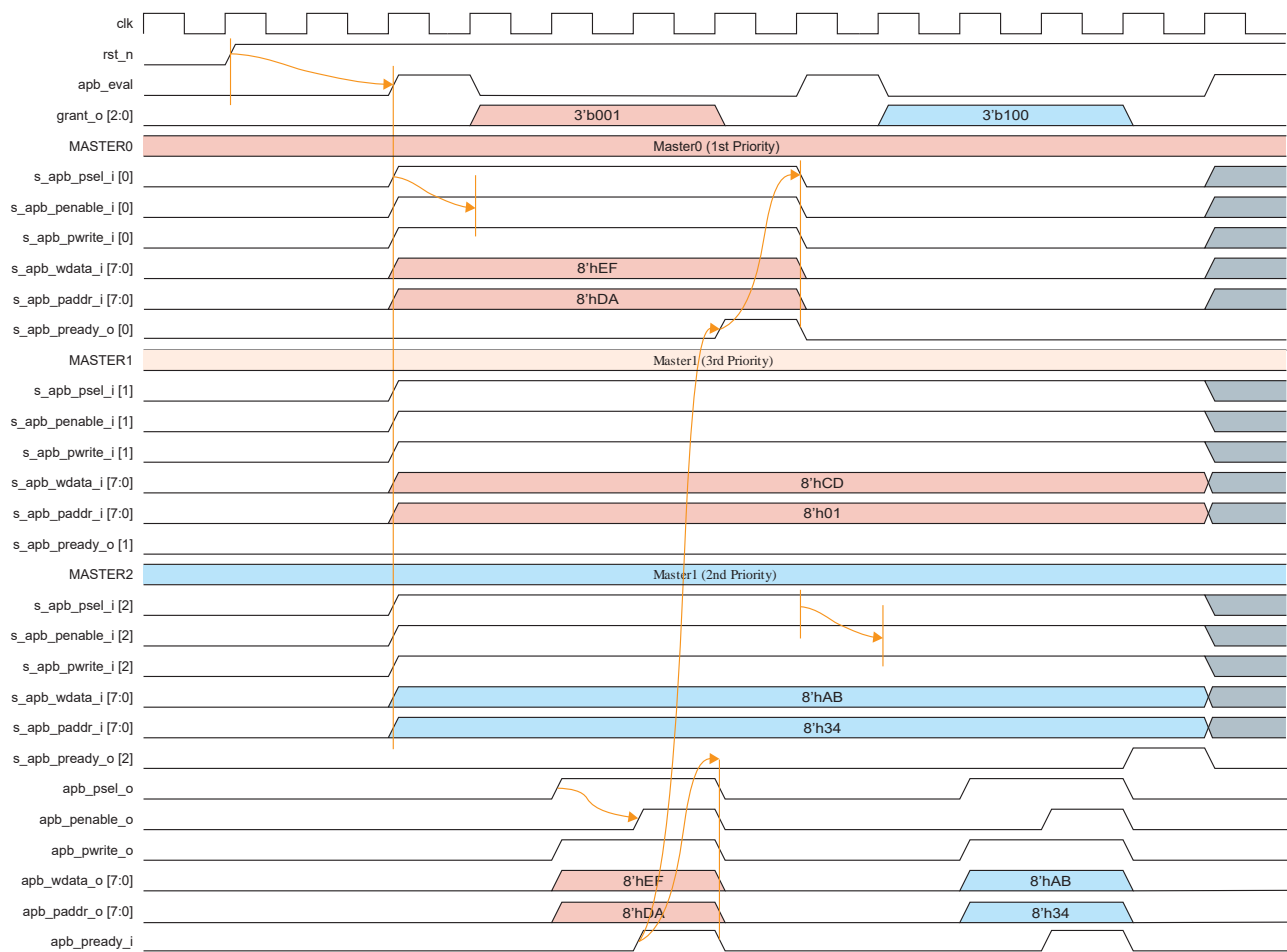


Arbitration Mode

The APB Interconnect core supports two arbitration mode:

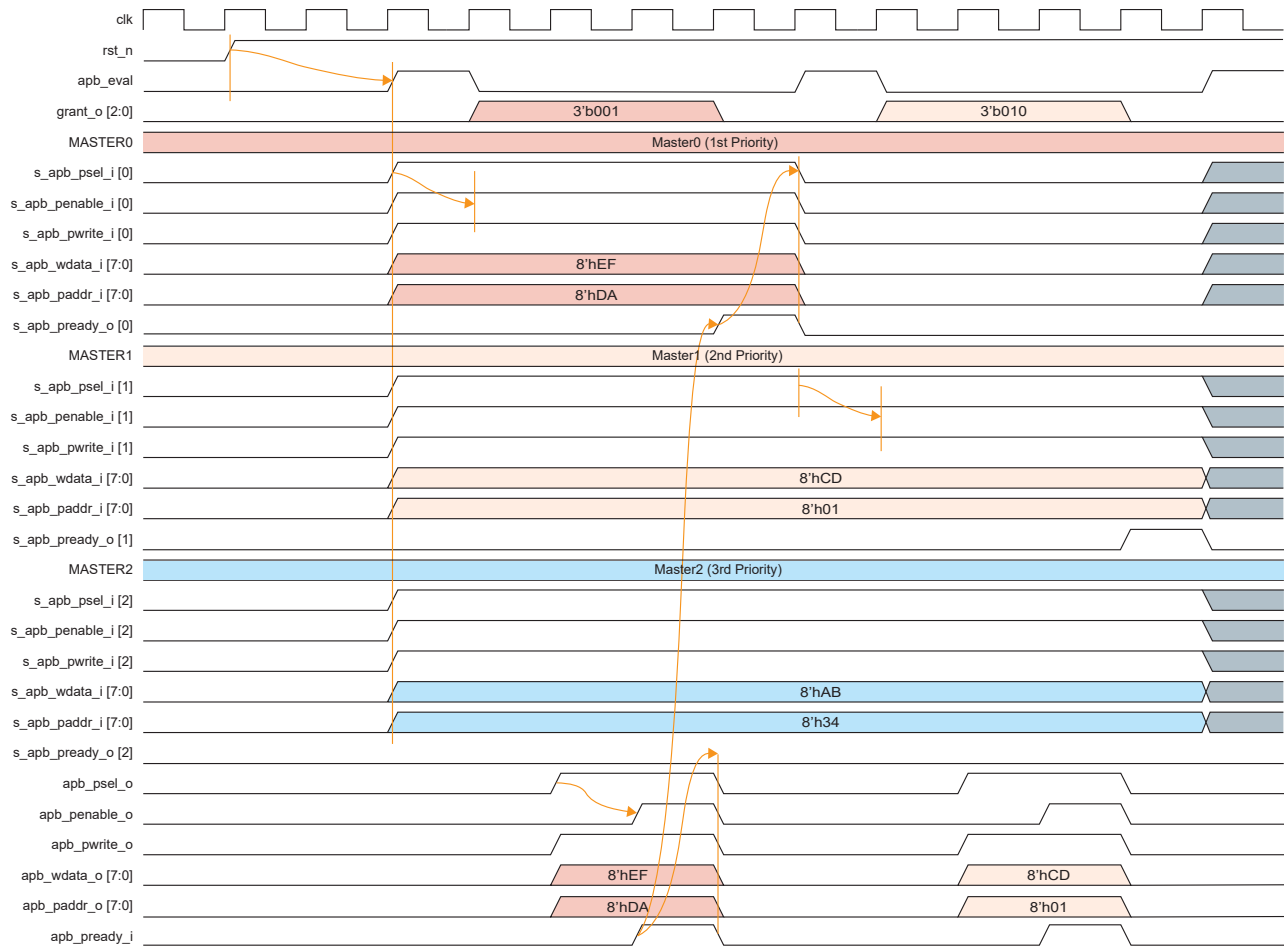
- *Fixed-Priority*—All enabled masters are granted the request based on the user-defined master priority in descending order. During the EVALUATION state, i.e., when `apb_eval = 1`, the APB Interconnect core evaluates all enabled masters' PSEL to decide which master is to be granted access. The master with **Priority 1** gets the highest priority. Upon completing the requested APB transfer, if the same master with **Priority 1** maintains its assertion of PSEL, it retains the priority. It is granted request every time it asserts PENABLE at one clock cycle after PSEL.
- *Round-Robin*—Each master has the same rational order and takes turns to be granted requests in ascending order of their master numbering. For example, in the event of 3 masters, `Master_0` is granted the request first, followed by `Master_1` and `Master_2`. Then, the sequence restarts with `Master_0`.

Figure 12: Fixed-Priority Mode



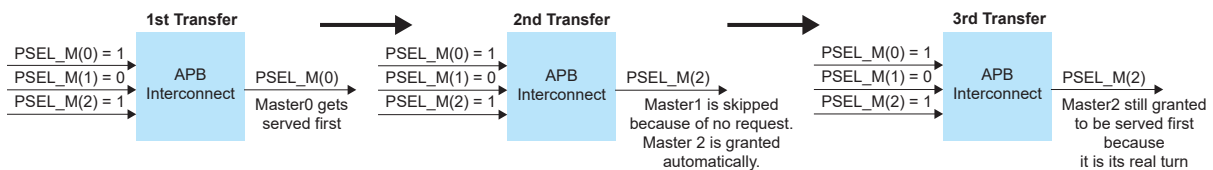
In **Fixed-Priority** mode, transfer access is granted to the master with the highest priority and request (PSEL) signal.

Figure 13: Round-Robin Mode



For Round-Robin mode, see **Figure 13: Round-Robin Mode** on page 20. When all enabled masters are requesting, i.e., all `s_apb_psel_i` are asserted, every master is granted request one by one starting with Master_0, followed by Master_1, and Master_2.

Figure 14: Round-Robin with Skipped Master



If the master is not requesting upon its turn, the APB Interconnect core skips that master and grants access to the next requested master. However, the next requested master is served twice since it takes the previous master's turn and serves its real turn.

IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate Efinity® IP cores. The IP Manager performs validation checks on the parameters you set to ensure that your selections are valid. When you generate the IP core, you can optionally generate an example design targeting an Efinity development board and/or a testbench. This wizard is helpful in situations in which you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core for different projects.



Note: Not all Efinity IP cores include an example design or a testbench.

Generating the APB Interconnect Core with the IP Manager

The following steps explain how to customize an IP core with the IP Configuration wizard.

1. Open the IP Catalog.
2. Choose **Bridge and Adaptors > APB Interconnect** core and click **Next**. The **IP Configuration** wizard opens.
3. Enter the module name in the **Module Name** box.



Note: You cannot generate the core without a module name.

4. Customize the IP core using the options shown in the wizard. For detailed information on the options, refer to the [Customizing the APB Interconnect Core](#) on page 22 section.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinity® development board and/or testbench. These options are turned on by default.
6. (Optional) In the **Summary** tab, review your selections.
7. Click **Generate** to generate the IP core and other selected deliverables.
8. In the **Review configuration generation** dialog box, click **Generate**. The Console in the **Summary** tab shows the generation status.



Note: You can disable the **Review configuration generation** dialog box by turning off the **Show Confirmation Box** option in the wizard.

9. When generation finishes, the wizard displays the **Generation Success** dialog box. Click **OK** to close the wizard.

The wizard adds the IP to your project and displays it under **IP** in the Project pane.

Generated Files

The IP Manager generates these files and directories:

- **<module name>_define.svh**—Contains the customized parameters.
- **<module name>_tpl.sv**—Verilog HDL instantiation template.
- **<module name>_tpl.vhd**—VHDL instantiation template.
- **<module name>.sv**—IP source code.
- **settings.json**—Configuration file.
- **Testbench**—Contains generated RTL and testbench files.

Customizing the APB Interconnect Core

The core has parameters so you can customize its function. You set the parameters in the **General** tab of the core's IP Configuration window.

Table 7: APB Interconnect Core Parameters (General Tab)

Port	Options	Description
Master Arbitration Mode	Fixed-Priority, Round-Robin	Defines the arbitration mode on master interface. Default: Fixed-Priority
Number of Master	2 - 32	Defines the number of masters to be enabled. Default: 32
Data Width	1 - 32	Defines the data width of each Master. The N-to-1 relationship of the number of masters and slaves where the slave is the common destination for all the masters. Hence, the data width of each master should match the data width of the slave. Default: 32
Address Width	1 - 32	Defines the address width of each master. The N-to-1 relationship of the number of masters and slaves where the slave is the common destination for all the masters. Hence, the address width of each master should match the address width of the slave. Default: 32
Optional APB Signals	On, Off	Defines the additional APB Signals for non-APB3. The signals are applicable when the data width is in multiple of 8-bit. Select On to enable Optional APB Signals , The Optional APB Signals are as follows: <ul style="list-style-type: none"> • APB_PRDATA_PAR • APB_PWDATA_PAR • APB_PSTRB_PAR • APB_PSTRB Default: Off
Register Input	On, Off	Defines register on the master input signals. Select On to enable Pipeline Registers at the master interface input paths that includes s_apb_psel_i, s_apb_penable_i, s_apb_pwrite_i, s_apb_paddr_i, s_apb_pwdata_i, s_apb_pwdata_par_i, s_apb_pstrb_i and s_apb_pstrb_par_i. Default: On
Register Output	On, Off	Defines register on the output-to-master signals. Select On to enable Pipeline Registers at the master interface output paths, namely s_apb_pready_o, s_apb_pslverr_o, s_apb_prdata_o and s_apb_prdata_par_o. Default: On
Optimize Latency	On, Off	Defines register on the output-to-slave signals. Select On to optimize latency by disabling Pipeline Registers at the slave interface, namely apb_sel_o, apb_penable_o, apb_pwrite_o, apb_paddr_o, apb_pwdata_o, apb_pwdata_par_o, apb_pstrb_o and apb_pstrb_par_o. Default: Off

Port	Options	Description
Master <i>n</i> Priority	Where number of master, <i>n</i> = 0, 1, 2, ..., 31	<p>Defines the priority of each master in descending order, i.e., master with priority 1 has the highest priority, while a master with priority 32 has the lowest priority.</p> <p>This setting is only applicable when the Master Arbitration mode = Fixed-Priority.</p> <p>This parameter depends on the number of master. For example, if the number of master = 3, the following 3 parameters appear:</p> <ul style="list-style-type: none"> • Master 0 Priority • Master 1 Priority • Master 2 Priority <p>Default Setting: Each Master is assigned with priority in descending order, e.g.,</p> <p>Master 0 Priority = 1 (Highest Priority) Master 1 Priority = 2 Master 2 Priority = 3 (Lowest Priority)</p>

APB Interconnect Core Example Design

You can access the example design from our portal on Ti375 PCIe[®] early access board. Compile the example design project and download the **.hex** or **.bit** file to your board.



Important: Efinix tested the example design generated with customized parameter options.

For this example design, the APB Interconnect core arbitrates among 3 masters to 1 APB interface (slave) to configure the PCIe Configuration Registers using the Titanium Ti375 N1156 Development Board. The example design contains a Virtual I/O debugger to issue read and write requests.

Furthermore, a logic analyzer is available to check for errors and compare the PRDATA with the PWDATA. The APB Interconnect core is set to **Round-Robin** mode, operates at 100 MHz, and enables **Pipeline Registers** at the masters' input paths (**Register Input = Yes**) and slave interface (**Optimize Latency = No**).

The APB Interconnect core configuration settings used in the example design are as follows:

Table 8: APB Interconnect Configuration Settings in Example Design

Parameter	Value
Master Arbitration Mode	Round-Robin
Number of Master	3
Address Width	24
Data Width	32
Register Input	Yes
Register Output	No
Optimize Latency	No

Table 9: Titanium Ti375 N1156 Development Board Example Design Implementation

FPGA	Logic Elements (Logic, Adders, Flipflops, etc.)	Memory Blocks	DSP Blocks	f _{MAX} (MHz) ⁽³⁾	Efinity [®] Version
Ti375 N1156 C4	41,677/370,137 (11.25%)	290/2,688 (10.79%)	0/1344 (0%)	132.05	2024.1

⁽³⁾ Using parameter settings in [Table 8: APB Interconnect Configuration Settings in Example Design](#) on page 24.

⁽⁴⁾ Using System Verilog 2005.

APB Interconnect Testbench

You can choose to generate the testbench when generating the core in the IP Manager Configuration window.



Note: You must include all `.v` files generated in the `/testbench` directory in your simulation.



Important: Efinix tested the testbench generated with the default parameter options only.

Efinix provides a simulation script for you to run the testbench using the Modelsim software. To run the Modelsim testbench script, run `vsim -do modelsim.do` or `sh run_modelsim.sh` in a terminal application. You must have Modelsim installed on your computer to use these scripts.

Revision History

Table 10: Revision History

Date	Document Version	IP Version	Description
December 2024	1.0	1.0	Initial release. (DOC-2200)