- DISCONTINUED PRODUCT -

FlexRay v1.1

Introduction

The LogiCORE™ FlexRay™ controller implements the FlexRay communication protocol as defined in the *FlexRay Protocol Specification v2.1 Rev A*. The FlexRay controller implementation supports a single communication channel. This document defines the architecture and features of the FlexRay controller. This document also defines the functionality of the modules in the design, in addition to defining the register addressing of the FlexRay controller. The scope of this document does not extend to describing the FlexRay protocol.

Features

- Designed to the *FlexRay Protocol Specification v 2.1 Rev A*
- Data rate of up to 10 Mbps
- Single Communication Channel
- Industrial (I), Extended Temperature Range (Q) and Automotive grade device support
- Scalable synchronous and asynchronous data transmission
- Configurable payload length up to a maximum of 256 bytes
- Configurable Tx buffers for storage of up to 128 messages
- Configurable Rx buffers for storage of up to 128 messages
- Configurable Rx FIFO buffer for storage of up to 128 messages.
- Frame ID, cycle counter and message ID based receive filtering
- 32 bit OPB interface with variable OPB interface clock that allows use in Xilinx EDK.

LogiCORE Facts				
Core Specifics				
Supported Device Family	Spartan™-3/3XA, Spartan-3A/3A DSP, Spartan-3E/XA Virtex™-II Pro, Virtex-4/4XA			
Resources Used	**I/O**	**LUTs**	**FFs**	**BRAMs**
	5	6,068 to 8,650	3,608 to 4,541	9 to 18
Provided with Core				
Documentation	Product Specification User Guide Getting Started Guide			
Design File Formats	VHDL			
Constraints File	.ucf (user constraints file)			
Verification	VHDL, Verilog Test bench			
Instantiation Template	VHDL, Verilog			
Design Tool Requirements				
Xilinx Implementation Tools	ISE v9.1i			
Verification	ModelSim SE/PE v6.1e			
Simulation	ModelSim SE/PE v6.1e Cadence™ IUS v5.5			
Synthesis	XST™			
Support				
www.xilinx.com/support				

Features Summary

Single Communication Channel

The FlexRay controller supports a single communication channel, and can be configured for either Channel A or Channel B. The Cyclic Redundancy Check (CRC) seed used to generate the Header CRC is different for Channel A and for Channel B. Therefore, connecting the FlexRay controller that is configured for Channel A to Channel B in the cluster (or vice-versa) is not allowed.

Configurable Payload Length

The Max Payload that can be supported by the controller during synchronous or aysnchronous transmission is user-configurable, depending upon specific design requirements.

Configurable number of Transmit Buffers

The number of transmit buffers in the FlexRay controller can be user-configured from a minimum of two to a maximum of 128 in powers of two. The block RAM resource utilization varies, based on the number of Transmit buffers and the configurable payload length.

Configurable number of Receive Buffers

The number of Receive buffers in the FlexRay controller can be user-configured from a minimum of two to a maximum of 128 in powers of two. The block RAM resource utilization varies, based on the number of Receive buffers and the configurable payload length.

Configurable Receive FIFO depth

The depth of the Receive FIFO in the FlexRay controller can be user-configured from a minimum of two to a maximum of 128 in powers of two. The block RAM resource utilization varies, based on the depth of the Receive FIFO and the configurable payload length.

Frame ID, Cycle Counter and Message ID based Receive Filtering

The FlexRay controller supports filtering of receive messages based on Frame ID, cycle counter and message ID. Any or all of these combinations can be used for filtering the receive messages.

Applications

The FlexRay controller is typically used in automotive networked applications, offering high fault tolerant synchronous and asynchronous transfer capabilities. It can function as a stand-alone FlexRay controller or it can be integrated with other Xilinx LogiCORE and EDK cores to build a variety of embedded systems.

The FlexRay solution allows for flexible partioning between software and hardware functions by providing a customized, scalable controller that off loads network specific overhead from the host processor. This is ideal for applications demanding maximum host application performance.

Integrated FlexRay Controller

This section provides an example application to illustrate the use of additional Xilinx IP cores. Figure 1 shows a typical application for the FlexRay controller, using the FlexRay and MicroBlaze cores. The FlexRay device driver stack and the host application run on the MicroBlaze.

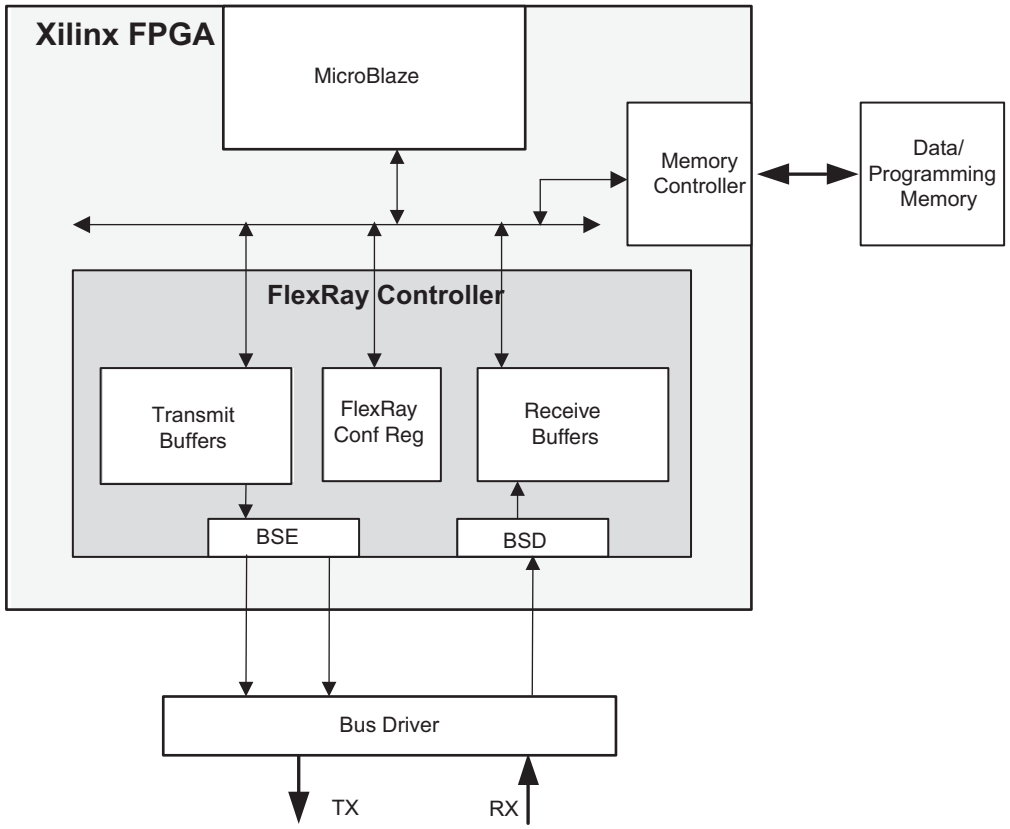

Figure 1: **Integrated FlexRay Controller**

FlexRay Cluster

Figure 2 illustrates the FlexRay controller as a node in a FlexRay cluster. The architecture of Node A on the bus is elaborated. Node A consists of a FlexRay Transceiver (Phy), FlexRay controller, MicroBlaze processor and application software.

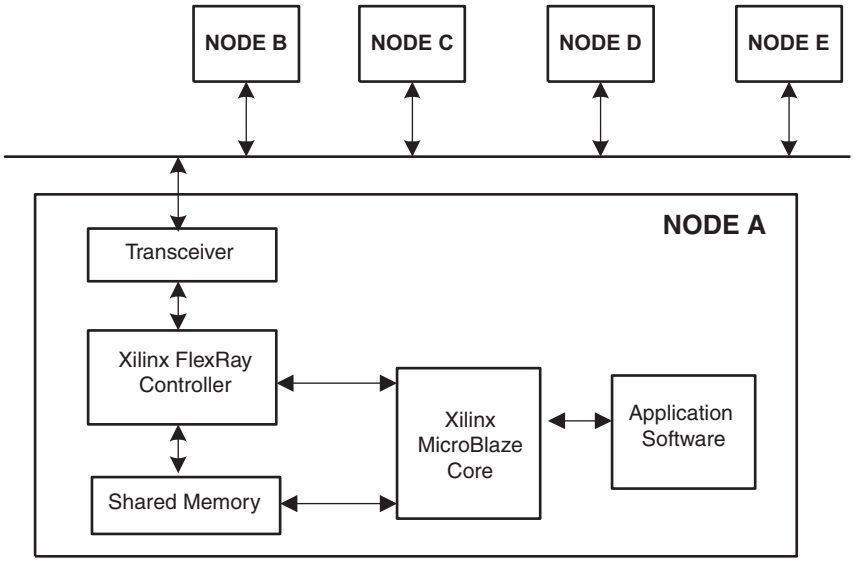

Figure 2: **FlexRay Controller in a Cluster**

Functional Overview

Figure 3 shows the high level architecture of the FlexRay controller.

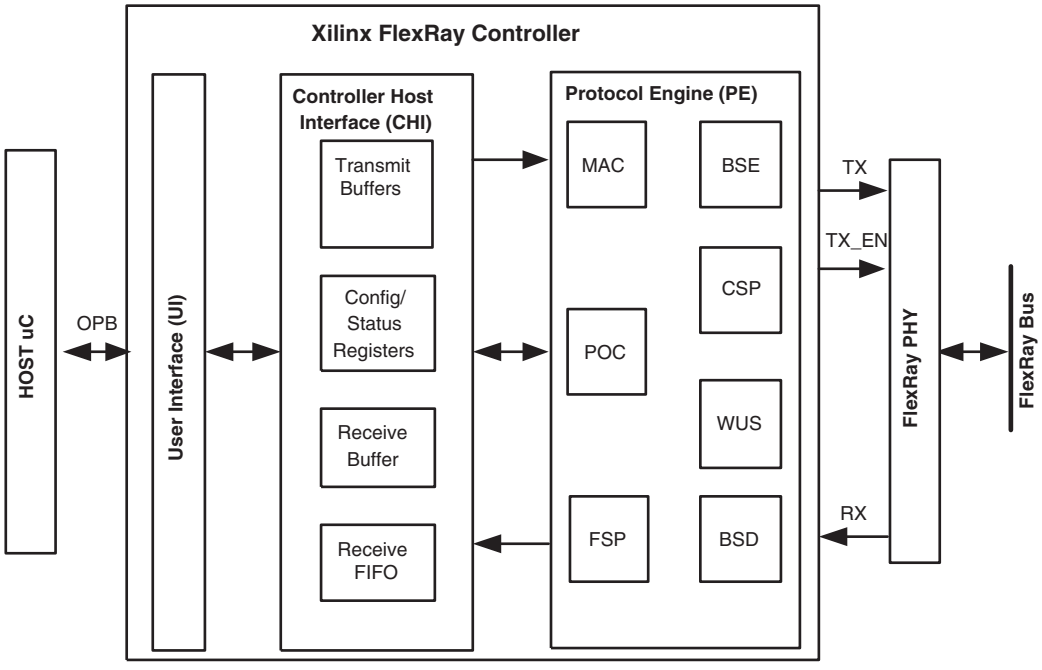

Figure 3: **FlexRay Controller Architecture**

The FlexRay controller contains three primary functional units/modules:

- User Interface (UI)

- Controller Host Interface (CHI)
- Protocol Engine (PE)

User Interface

This primary function of UI module is to provide OPB bus connectivity to the FlexRay controller. The UI module performs the following tasks:

- OPB Read/Write transactions
- Interrupt management

OPB Read/Write Transactions

For information about OPB Read/Write transactions, please see the section, "Single Read/Write" on page 23.

Interrupts Management

The controller provides four active-high interrupt lines to alert the host of the occurrence of interrupt events. The following are the active-high interrupt lines:

- **`Flex_Cirpt`**: Indicates interrupt conditions due to error conditions, timers and changes in operational state of the FlexRay controller.
- **`Flex_Txbirpt`**: Indicates interrupt events associated with the Tx Buffers.
- **`Flex_Rxbirpt`**: Indicates interrupt events associated with the Rx Buffers.
- **`Flex_Rxfirpt`**: Indicates interrupt events associated with the Rx FIFO.

Controller Host Interface

The Controller Host Interface (CHI) manages message, control, and configuration data flow between the host processor and the Protocol Engine (PE). The CHI contains 4 major blocks:

- Memory Map and Registers
- Transmit Buffers
- Receive Buffers
- Receive FIFO

Memory Map and Registers

This module contains the control, status, and configuration memory map and registers. It allows read and write access to control the following register sets:

- Protocol configuration , control and status
- General CHI configuration, control and status

Transmit Buffers

Tx buffers provide storage and control of message data to be transmitted over the FlexRay physical interface. The FlexRay controller provides up to 128 user- configurable Tx buffers, each of which can store one FlexRay frame of a variable payload length. The number of Tx buffers in the controller can be configured by the `Number of Tx Buffers` parameter. The maximum `Number of Tx Buffers` in the controller depends on the `Max Payload Size` parameter.

Receive Buffers

Receive buffers provide storage for received message data, and perform the following functions.

- Frame storage
- Frame filtering

The `Number of Rx Buffers` is user-configurable up to 128, each of which can store one FlexRay frame of variable payload length. The number of Receive buffers in the controller is configured using the `Number of Rx Buffers` parameter.

Filtering

Each Receive Buffer is associated with a filter that can be programmed to accept any combination of Frame ID, Cycle counter and Message ID values for storage in the buffer.

Receive FIFO

In addition to the Receive buffers described above, the FlexRay controller provides a flexible, variable length FIFO buffer for storage of the received FlexRay frames. The message storage space allocated to the Receive FIFO is user-configurable using the `Depth of Rx FIFO` parameter.

Filtering

Four acceptance filter pairs are provided to screen incoming messages from the Protocol Engine for storage into the Rx FIFO. Each filter pair contains a mask and data pair, each of which in turn contains fields for Frame ID, Cycle counter and Message ID.

Protocol Engine

The Protocol Engine (PE) module contains all functional blocks that perform low-level FlexRay protocol related tasks described in the FlexRay specification. These sub-blocks are listed and described below.

- Media Access Control
- Bit Stream Encoder
- Bit Stream Decoder
- Frame and Symbol Processing
- Protocol Operation Control
- Wakeup and Startup
- Clock Synchronization Process

Media Access Control

The Media Access Control (MAC) is used to maintain the timing within each communication cycle. It manages the timing of static and dynamic slot duration. It also manages the transmit operation of the node. The MAC also handles frame transmission by assembling the frames. The MAC maintains timing consistency within a communication cycle for the following segments:

- Static
- Dynamic
- Symbol Window
- Network Idle Time

The MAC asserts signals to other modules to indicate the static segment start, slot boundary, dynamic

segment start, symbol window start and network idle start time.

Bit Stream Encoder

The Bit Stream Encoder (BSE) module handles encoding frames and symbols.

Frame Encoding

The encoder takes the incoming bytes from the MAC and assembles the FlexRay bit stream as follows:

- Appending a Transmit Start Sequence (TSS) to start of bit stream
- Appending a Frame Start Sequence (FSS) immediately after the TSS
- Adding a Byte Start Sequence (BSS) at the beginning of each byte
- Transmitting each byte serially
- Calculating Frame CRC and appending it to the end of the Frame
- Appending a Frame End Sequence (FES)
- Appending a Dynamic Trailing Sequence (DTS) if the frame is transmitted in dynamic segment

Symbol Encoding

Three types of symbols can be transmitted – two of which (CAS and MTS) are identical in terms of bit pattern.

- CAS – Collision Avoidance Symbol
- MTS – Media Test Symbol
- WUS – Wakeup Symbol

Bit Stream Decoder

The Bit Stream Decoding block (BSD) performs the following functions:

- Sampling And Majority Voting
- Bit Clock Alignments (BCA) and Bit Strobing
- Frame and Symbol Decoding
- Channel Idle Detection
- Decoding Error Detection

This Frame and Symbol Decoding modules also deal with the detection of various signaling patterns. Those patterns can be described as follows:

- Communication Element (CE)—indicates the active period of a symbol/frame
- Channel idle detection—operates closely with CE generation
- TSS detection (common to frames and symbols)
- Symbol detection
- Frame detection—consists of checking all BSS indicators during frame transmission, FES detection, control of the header CRC check, and control of the trailer segment CRC check.
- Time Reference point generation
- Decoding Error detection

Frame and Symbol Processing

Frame and symbol processing (FSP) verifies the correct timing of frames and symbols with respect to the TDMA scheme, applies further syntactical tests to received frames, and checks the semantics for correctness of received frames. The result of these checks are then signaled to the host. FSP provides the following status indicators to the host.

- Valid Frame
- Valid Symbol
- Syntax Error
- Content Error
- Boundary Violation
- Tx Conflict

Protocol Operation Control

The Protocol Operation Control (POC) acts as an interface between Host and FlexRay PE sub-modules. The POC determines the operational state of the controller.

POC state transitions are controlled synchronously and occur as a consequence of a host command or an error condition occurring. The state transitions cause a change in operation of the protocol engine. See Figure 4 on page 12 for an illustration of the POC state machine.

Wakeup and Startup

The wakeup procedure describes the transition of a node from sleep mode to operational mode. The startup procedure describes the integration of nodes into the cluster. The Controller Host Interface triggers both Wakeup and Startup (WUS) procedures.

The wakeup pattern transmitted consists of a programmable number of wakeup symbols. If more than one node tries to wake a channel, the wakeup procedure resolves the situation by allowing only one node to transmit the pattern.

Wakeup Block

The wakeup procedure is triggered from a host command. The state machine enters a wakeup listen state and remains there for the amount of time specified in the Listen Timeout register. If no communication is observed during this time frame, the state machine enters a send state and the wakeup pattern is transmitted. If communication is observed while the wakeup pattern is being sent, then the wakeup detect state is entered and the cause of the collision is investigated.

Startup Block

The process by which startup occurs is different for leading coldstart nodes, integrating coldstart nodes and integrating nodes. The following are the major startup process types handled by the startup module.

- Leading coldstart node startup
- Integrating coldstart node startup
- Integrating non coldstart node startup

Clock Synchronization Process

The Clock Synchronization Process (CSP) performs the synchronization of all nodes in a cluster to

ensure that they share the same global view of time. Synchronization is achieved by using sync frames sent by nodes on the network. The two components of time adjustment that are handled by the CSP are Rate correction and Offset correction.

Two major sub-processes of the CSP that perform the synchronization functions are:

- Clock Sync Processing
- Clock Sync Startup

Protocol Operational States

The FlexRay controller supports the following protocol operational states. Each of these states is described in the following sections.

- Reset state
- Default Config state
- Config state
- Ready state
- Wakeup state
- Startup state
- Normal Active state
- Normal Passive state
- Halt state

Operational state transitions can occur with reset conditions, host commands, and internal protocol error conditions. Resets are asserted by writing to the SRR or by asserting the `OPB_Rst` line. The host can issue commands by writing to the CHI command register.

Reset

In the Reset state:

- The CHI and PE modules are reset. When the `OPB_Rst` signal is asserted the UI module is reset. Upon deassertion of `OPB_Rst`, the UI module moves out of reset.
- The FlexRay controller registers are reset to their default values. However, registers and memory locations that are stored in Block RAMs are not reset.
- Read access to the FlexRay controller Memory map is permitted.

The FlexRay controller enters the Reset state when the following occurs.

- The system reset is asserted by driving a '1' on the `OPB_Rst` input. The controller continues to stay in Reset state as long as the `OPB_Rst` input is '1'.
- The software reset is asserted by writing a '1' to the SRST bit in the SRR.

The FlexRay controller transitions to the Default Config state from the Reset state when the Reset sequence is completed.

Default Config

In the Default Config state the PE module is disabled.

The FlexRay controller enters a Default Config state when the following conditions occur.

- The controller receives the `CHI_DEFAULT_CONFIG` command (from the Halt state).

- The Reset sequence is completed (from the Reset state).

The FlexRay controller transitions from the Default Config state as follows:

- The controller receives the CHI_CONFIG command (to the Config state)
- The assertion of Software reset or Hardware reset (to the Reset state).

Config

In the Config state, the configuration of the Protocol Registers, Transmit Buffers, and Receive Buffers is performed, and the PE is disabled.

The FlexRay controller enters the Config state when the following conditions occur.

- The controller receives the CHI_CONFIG command (from the Default Config state).
- The controller receives the CHI_CONFIG command (from the Ready state).

The FlexRay controller transitions from the Config state as follows:

- The controller receives the CHI_CONFIG_COMPLETE command (to the Ready state).
- The assertion of Software reset or Hardware reset (to the Reset state).

Ready

In the Ready state the PE module is enabled. Protocol-specific configuration registers should not be written to.

The FlexRay controller enters Ready state when any of the following conditions occur.

- The CONFIG_COMPLETE command from the host (from the Config state) is received.
- The READY command from the host (from the Normal Active state, Normal Passive state, Wakeup state, Startup state) is received.

The FlexRay controller exits the Ready state when any of the following conditions occur.

- The controller receives the WAKEUP command from the host (to the Wakeup state).
- The controller receives the CHI_RUN command (to the Startup state).
- The controller receives the CHI_CONFIG command (to the Config state).

Wakeup

In the Wakeup state the node performs the wakeup operation, as described in "Wakeup and Startup" on page 8.

The FlexRay controller enters the Wakeup state when the controller receives the CHI_WAKEUP command (from the Ready state).

The FlexRay controller transitions from the Wakeup when any of the following occur.

- The controller receives the CHI_FREEZE command (to the Halt state).
- The controller receives the CHI_READY command (to the Ready state).
- The completion of the wakeup operation (to the Ready state).
- The assertion of Software reset or Hardware reset (to the Reset state).

Startup

In the Startup state the FlexRay controller performs the startup operation based on the mode of the node. The mode for a particular node can be coldstart or integrating node.

The FlexRay controller enters the Startup state when the controller receives the CHI_RUN command (from the Ready state).

The FlexRay controller transitions from the Startup state under any of the following conditions:

- The controller receives the CHI_FREEZE command (to the Halt state).
- The controller receives the CHI_READY command (to the Ready state).
- The completion of the startup operation (to the Normal Active state).
- The assertion of Software reset or Hardware reset (to the Reset state).

Normal Active

In Normal Active state, the FlexRay node performs normal frame and symbol transmission and reception. The clock synchronization process performs offset and rate correction.

The FlexRay controller enters the Normal Active state upon completion of the startup operation (from the Startup state).

The FlexRay controller transitions from the Normal Active state as follows:

- The controller receives the CHI_HALT or CHI_FREEZE command (to the Halt state).
- The controller receives the CHI_READY command (to the Ready state).
- The controller observes internal protocol errors (to the Normal Passive state or to the Halt state).
- The assertion of Software reset or Hardware reset (to the Reset state).

Normal Passive

The following conditions describe Normal Passive state.

- There is proper frame reception.
- Clock synchronization process performs offset and rate correction.
- There is no transmission of frames.

The FlexRay controller enters the Normal Passive state whenever the controller observes internal protocol errors (from the Normal Active state).

The FlexRay controller transitions from the Normal Passive whenever the controller:

- Receives the CHI_HALT or CHI_FREEZE command (to the Halt state).
- Observes internal protocol errors (to the Halt state).
- Receives the CHI_READY command (to the Ready state)
- Observes error free communication (to the Normal Active state)
- Asserts a Software reset or Hardware reset (to the Reset state).

Halt

The FlexRay controller enters the Halt state when the controller:

- Observes internal protocol errors (from the Normal Passive state or the Normal Active state).
- Receives the CHI_HALT or CHI_FREEZE command (from the Normal Passive state, Normal Active state, Wakeup state, or Startup state).

The FlexRay controller transitions from the Halt when the controller receives the CHI_DEFAULT_CONFIG command (to the Default Config state).

Figure 4 shows the operational state transition diagram for the FlexRay controller.

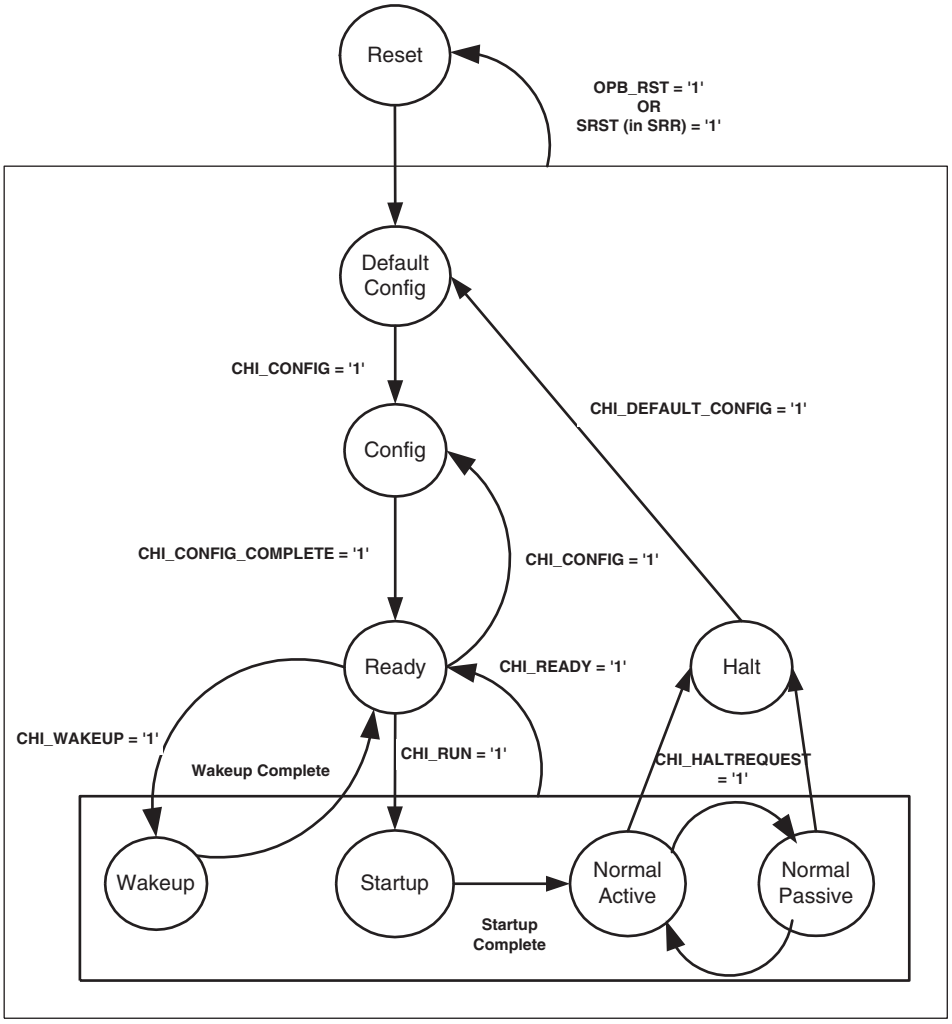

Figure 4: **FlexRay Controller State Diagram**

Design Parameters

To obtain a FlexRay controller tailored to meet minimum system requirements, certain features are parameterized. This results in a design that uses only the required resources; providing the best possible performance. Table 1 contains the FlexRay controller features that can be parameterized.

Table 1: **FlexRay Controller Design Parameters**

Parameter Name	Parameter Description	Allowable Values	Default Value
Channel Selection	Channel Selection A or B	A, B	A
Max Payload Size	Max Payload Size (bytes)	4,8,12,16.....252,256	4
Number of Tx Buffers	Number of Transmit buffers	2,4,8,16,32,64,128	2
Number of Rx Buffers	Number of Receive buffers	2,4,8,16,32,64,128	2
Rx FIFO Depth	Depth of Receive FIFO	2,4,8,16,32,64,128	2

Channel Selection

The FlexRay controller can be configured to support either Channel A or Channel B.

Max Payload Size

The payload size can be configured from 4 to 256 bytes in multiples of 4 bytes. The 32-bit OPB bus interface supports only word access; therefore, even though the `Max Payload Size` allowed by the *FlexRay Protocol Specification V2.1 Rev A* is 254 bytes, the physical storage allocated to store the maximum payload is fixed to 256 bytes. The value chosen for the `Max Payload Size` must be greater than or equal to the protocol parameters `gPayloadLengthStatic` and `pPayloadLengthDynMax`.

Number of Transmit Buffers

The number of the Tx Buffers can be configured from 2 to 128 in incremental powers of two. Each Tx Buffer consists of three dedicated header registers. A single block RAM stores the header registers for all Tx Buffers.

The payload associated with each Tx Buffer is stored in a Payload Buffer. Depending on the `Number of Tx Buffers` and the `Max Payload Size` chosen, the size of the Payload Buffer can range from 2 KBytes to 8 KBytes, with an increment step size of 2 KBytes.

Number of Receive Buffers

The user can configure the number of the Rx Buffers from 2 to 128 in incremental powers of 2. The maximum memory allocated to store Rx Buffers is 8 KBytes.

The upper limit of 8 KBytes for the storage of Rx Buffers places certain restrictions on the `Number of Rx Buffers` and the `Max Payload Size` that can be configured. The product of `Max Payload Size` + 8 and `Number of Rx Buffers` should be less than 8 KBytes.

Depth of the Receive FIFO

The depth of the Rx FIFO can be configured from 1 to 128 in incremental powers of two. The maximum memory allocated for the Rx FIFO is 8 KBytes.

The upper limit of 8 KBytes for the Rx FIFO places certain restrictions on the `Depth of Rx FIFO` and the `Max Payload Size` that can be configured. The product of `Max Payload Size` + 8 and `Rx FIFO Depth` should be less than 8 KBytes.

Configuration Map

Table 2 lists the FlexRay controller registers. Each of these registers is 32-bits wide and is represented in big-endian format. Read operations to reserved bits or bits that are not used return invalid data. Zeros should be written to reserved bits and bit fields not used. Writes to reserved locations are ignored. The FlexRay controller registers are grouped based on the sub-modules in which they are used.

Values for the protocol specific registers and the formulae to compute them can be obtained from "Appendix B" of the *FlexRay Protocol Specification v2.1 Rev A*. See the *FlexRay User Guide* for detailed descriptions of the Controller Registers.

Table 2: **FlexRay Controller Configuration**

Register Name	Address	Access
Protocol Engine Registers		
Software Reset Register (SRR)	0x00000000	Read/Write
Static Slot Duration Register (SSDR)	0x00000004	Read/Write
Number of Static Slots Register (NSSR)	0x00000008	Read/Write
Static Segment Payload Length Register (SSPLR)	0x0000000C	Read/Write
Action Point Offset Register (APOR)	0x00000010	Read/Write
Startup Frame Configuration Register (STFCR)	0x00000014	Read/Write
Sync Frame Configuration Register (SYFCR)	0x00000018	Read/Write
Network Management Vector Length Register (NMVLR)	0x0000001C	Read/Write
Reserved	0x00000020 - 0x0000003C	Read/Write
POC Module Registers		
CHI Command Register (CR)	0x00000040	Read/Write
CHI Status Register (CSR)	0x00000044	Read
Allow Halt Dueto Clock Register (AHCR)	0x00000048	Read/Write
Slot Mode Register (SMR)	0x0000004C	Read/Write
Maximum without Clock Correction Fatal Register(MCCFR)	0x00000050	Read/Write

Table 2: **FlexRay Controller Configuration *(Continued)***

Register Name	Address	Access
Maximum without Clock Correction Passive Register(MCCPR)	0x00000054	Read/Write
PAllow Passive to Active Register (PAPAR)	0x00000058	Read/Write
VAllow Passive to Active Status Register (VAPASR)	0x0000005C	Read
Clock Correction Failed Status Register (CCFSR)	0x00000060	Read
Reserved	0x00000064 - 0x00000078	Read/Write
Test Mode Register (TMR)	0x0000007C	Write
Reserved	0x00000080 - 0x000000FC	Read/Write
CODEC Module Registers		
TSS Register (TSSR)	0x00000100	Read/Write
WUP Receive Idle Register (WRIR)	0x00000104	Read/Write
WUP Receive Low Register (WRLR)	0x00000108	Read/Write
WUP Window Register(WRWR)	0x0000010C	Read/Write
WUP Transmit Idle Register (WTIR)	0x00000110	Read/Write
WUP Transmit Low Register (WTLR)	0x00000114	Read/Write
CAS Receive Max Register (CRMXR)	0x00000118	Read/Write
Decoding Correction Register (DCR)	0x0000011C	Read/Write
Wakeup Pattern Sequence Register (WPSR)	0x00000120	Read/Write
Reserved	0x00000124 - 0x0000013C	Read/Write
MAC Module Registers		
Sync Slot Register (SYSR)	0x00000140	Read/Write
Minislot Duration Register (MSDR)	0x00000144	Read/Write
Minislot Action Point Offset Register (MAPR)	0x00000148	Read/Write
Dynamic Slot Idle Phase Register (DSIPR)	0x0000014C	Read/Write
Number of Minislots Register (NMSR)	0x00000150	Read/Write
Latest Tx Register (LTR)	0x00000154	Read/Write
Symbol Window Duration Register (SWDR)	0x00000158	Read/Write
MTS Transmit Register (MTR)	0x0000015C	Read/Write
Slot Counter Status Register (SCSR)	0x00000160	Read

Table 2: **FlexRay Controller Configuration *(Continued)***

Register Name	Address	Access
Last Dynamic Frame Transmit Status Register (LDFTSR)	0x00000164	Read
Reserved	0x00000168 - 0x000001FC	Read/Write
Clock Sync Module Registers		
Extern Offset Correction Register (EOCR)	0x00000200	Read/Write
Extern Rate Correction Register (ERCR)	0x00000204	Read/Write
Apply Extern Correction Register(AECR)	0x00000208	Read/Write
Offset Correction Limit Register (OCLR)	0x0000020C	Read/Write
Rate Correction Limit Register (RCLR)	0x00000210	Read/Write
Number Of Microticks Per Cycle Register (NMICR)	0x00000214	Read/Write
Number Of Macroticks Per Cycle Register (NMACR)	0x00000218	Read/Write
Maximum Drift Register (MDR)	0x0000021C	Read/Write
Cluster Drift Damping Register (CDDR)	0x00000220	Read/Write
Macro Initial Offset Register (MAOR)	0x00000224	Read/Write
Offset Correction Start Register (OCSR)	0x00000228	Read/Write
Micro Initial Offset Register (MIOR)	0x0000022C	Read/Write
Sync Node Max Register (SNMR)	0x00000230	Read/Write
Accepted Startup Range Register (ASRR)	0x00000234	Read/Write
Reserved	0x00000238 - 0x0000023C	Read/Write
Rate Correction Status Register (RCSR)	0x00000240	Read
Offset Correction Status Register (OFCSR)	0x00000244	Read
Valid Sync Frames Status Register (VSFSR)	0x00000248	Read
Cycle Count Status Register (CCSR)	0x0000024C	Read
Macrotick Status Register (MSR)	0x00000250	Read
Reserved	0x00000254 - 0x000002FC	Read/Write
Sync Frames Status Register[1..15] (SFSR[n])	0x00000300 - 0x00000338	Read
Reserved	0x0000033C	Read
Sync Frames Status Register[16..30] (SFSR[n])	0x00000340 - 0x0000037C	Read
Reserved	0x00000380 - 0x000003FC	Read/Write

Table 2: **FlexRay Controller Configuration *(Continued)***

Register Name	Address	Access
WakeUp Module Registers		
ListenTimeout Register (LTOR)	0x00000400	Read/Write
Listen Noise Register (LNR)	0x00000404	Read/Write
Number of ColdStart Attempts Register (NCAR)	0x00000408	Read/Write
Wakeup Status Register (WSR)	0x0000040C	Read
ColdStart Noise Status Register (CNSR)	0x00000410	Read
Remaining Coldstart Attempts Status Register (RCASR)	0x00000414	Read
Startup Status Register (STSR)	0x00000418	Read
Reserved	0x0000041C - 0x0000043C	Read/Write
Timer Registers		
Absolute Timer Control Register(ATCR)	0x00000440	Read/Write
Absolute Timer Cycle Count Register(ATCCR)	0x00000444	Read/Write
Absolute Timer Macrotick Offset Register(ATMOR)	0x00000448	Read/Write
Relative Timer Control Register(RTCR)	0x0000044C	Read/Write
Relative Timer Macrotick Offset Register (RTMOR)	0x00000450	Read/Write
Reserved	0x00000454 - 0x0000045C	Read/Write
Controller Status Registers		
Aggregated Status Register (ASR)	0x00000460	Read
NIT Status Register (NSR)	0x00000464	Read
Symbol Window Status Register (SWSR)	0x00000468	Read
Network Management Vector Registers[1..3] (NMVR[n])	0x0000046C - 0x00000474	Read
Reserved	0x00000478 - 0x000004FC	Read/Write
Receive Slot Status Registers[1..512] (RSSR[n])	0x00000500-0x00000CFC	Read
Reserved	0x00000D00 - 0x00000EFC	Read/Write
Interrupt Control and Status Registers		
Controller Interrupt Status Register (CISR)	0x000000F00	Read
Controller Interrupt Enable Register(CIER)	0x000000F04	Read/Write

Table 2: **FlexRay Controller Configuration *(Continued)***

Register Name	Address	Access
Controller Interrupt Clear Register (CICR)	0x000000F08	Write
Reserved	0x00000F0C	Read/Write
Transmit Buffer Interrupt Status Register[1..4] (TXBISR[n])	0x000000F10 - 0x000000F1C	Read
Transmit Buffer Interrupt Enable Register[1..4] (TXBIEN[n])	0x000000F20 - 0x000000F2C	Read/Write
Transmit Buffer Interrupt Clear Register[1..4] (TXBICR[n])	0x000000F30 - 0x000000F3C	Write
Receive Buffer Interrupt Status Register[1..8] (RXBISR[n])	0x000000F40 - 0x000000F5C	Read
Receive Buffer Interrupt Enable Register[1..8] (RXBIER[n])	0x000000F60 - 0x000000F7C	Read/Write
Receive Buffer Interrupt Clear Register[1..8] (RXBICR[n])	0x000000F80 - 0x000000F9C	Write
Receive FIFO Interrupt Status Register (RXFISR)	0x000000FA0	Read
Receive FIFO Interrupt Enable Register (RXFIER)	0x000000FA4	Read/Write
Receive FIFO Interrupt Clear Register (RXFICR)	0x000000FA8	Write
Reserved	0x00000FAC - 0x00000FFC	Read/Write
Transmit Buffers		
Transmit Buffer Payload Storage	0x00001000-0x00002FFC	Read/Write
Transmit Buffer Header 1 Registers [1..128] (TXBH1R[n])	0x00003000-0x00003FFC	Read/Write
Transmit Buffer Header 2 Registers [1..128] (TXBH2R[n])	0x00003000-0x00003FFC	Read/Write
Transmit Buffer Header 3 Registers [1..128] (TXBH3R[n])	0x00003000-0x00003FFC	Read/Write
Receive Buffers		
Receive Buffer[1..66] Registers[1..128] (RXB[y]R[n])	0x00004000-0x00005FFC	Read
Receive FIFO		
Receive FIFO [1..66] Registers (RXF[y]R)	0x00006000-0x00006108	Read
Reserved	0x0000610C - 0x000061FC	Read/Write
Receive FIFO Threshold Register (RFTR)	0x00006200	Read/Write
Reserved	0x00006204 - 0x000062FC	Read/Write

Table 2: **FlexRay Controller Configuration *(Continued)***

Register Name	Address	Access
Acceptance Filters		
Receive Buffer Acceptance Filter 1 Registers [1..128] (RBAF1R[n])	0x00006300-0x000066FC	Read/Write
Receive Buffer Acceptance Filter 2 Registers [1..128] (RBAF2R[n])	0x00006300-0x000066FC	Read/Write
Receive FIFO Acceptance Filter ID 1 Register [1..4] (RFAFI1R[n])	0x00006700-0x0000673C	Read/Write
Receive FIFO Acceptance Filter Mask 1 Register [1..4] (RFAFM1R[n])	0x00006700-0x0000673C	Read/Write
Receive FIFO Acceptance Filter ID 2 Register [1..4] (RFAFI2R[n])	0x00006700-0x0000673C	Read/Write
Receive FIFO Acceptance Filter Mask 2 Register [1..4] (RFAFM2R[n])	0x00006700-0x0000673C	Read/Write

Clocking and Reset

This section describes the clocking and reset functions of the FlexRay controller.

Clocking

The FlexRay controller uses the following clocks.

System clock. The FlexRay controller uses a system clock of 40 MHz. The system clock must be generated by a DCM. The input to DCM can be from an external oscillator that meets the DCM input clock requirements. The PE module runs off the System clock.

Sample clock. The sample clock has a frequency of 80 MHz. The sample clock is used to perform the oversampling of the incoming receive data. Both the sample clock and the system clock must be phase aligned.

Interface clock. The interface clock is the OPB bus clock. You can specify the operating frequency of OPB clock. Using a DCM to generate the Interface clock is optional. Both the UI and the CHI modules use the interface clock.

Reset

Two different reset mechanisms are provided for the FlexRay controller. The `OPB_Rst` input acts as the System reset. Apart from the System reset, a Software reset is provided through the SRST bit in the SRR.

Software Reset

A software reset is asserted by writing a 1 to the SRST bit in the Software Reset Register (SRR). The assertion of software reset places the FlexRay controller in Reset state. Upon completion of the Reset sequence, the FlexRay controller transitions to the Default Config state. In the Reset state, reads to the SRST bit return a 1. A software reset during normal operation of the FlexRay controller immediately places the node into Reset state and current frame transmission or reception is abruptly stopped. The Software Reset does not clear the contents of the block RAMs used in the FlexRay controller. The Software reset resets the CHI and PE modules.

System Reset

The System reset can be enabled by driving a 1 on the OPB_Rst input. Upon deassertion of the system reset, the controller enters the Reset state. When the Reset sequence is complete, the FlexRay controller transitions to the Default Config state. As long as the System reset is asserted, reads and writes to the FlexRay controller memory map are prohibited. A System reset during normal operation of the FlexRay controller immediately places the controller into Reset state and current frame transmission or reception is abruptly stopped. The System Reset does not clear the contents of the block RAMs used in the FlexRay controller. When a System reset is asserted the UI, CHI, and the PE modules are reset. Upon deassertion of the System reset, the UI module is no longer in reset.

Interrupts

The FlexRay IP Core uses a hard-vector interrupt mechanism. Each of the four interrupt lines (Flex_Cirpt, Flex_Txbirpt, Flex_Rxbirpt, Flex_Rxfirpt) indicates an interrupt on transition from a logic '0' to a logic '1.' During power up, the Interrupt line is driven low.

The Interrupt Status Registers (CISR, TXBISR[n], RXBISR[n], RXFISR) indicate the interrupt status bits. These bits are set and cleared regardless of the status of the corresponding bit in the Interrupt Enable Registers (CIER, TXBIER[n], RXBIER[n], RXFIER). The Interrupt Enable Registers handle the interrupt-enable functionality. Clearing a status bit in the Interrupt Enable Registers is handled by writing a '1' to the corresponding bit in the Interrupt Clear Registers (CICR, TXBICR[n], RXBICR[n], RXFICR).

The following conditions cause the interrupt lines to be asserted:

- If a bit in the Interrupt Status Register is '1' and the corresponding bit in the Interrupt Enable Register is '1.'
- Changing an Interrupt Enable Register bit from a '0' to '1'; when the corresponding bit in the Interrupt Status Register is already '1.'

Two conditions cause the interrupt lines to be deasserted:

- Writing a '1' to a bit in the Interrupt Clear Register and the corresponding bit in the Interrupt Enable Register is '1.'
- Changing an Interrupt Enable Register bit from '1' to '0'; when the corresponding bit in the Interrupt Status Register is '1.'

When both deassertion and assertion conditions occur simultaneously, the interrupt line is deasserted first, and is reasserted if the assertion condition remains true.

Core Interface

Table 3 contains the external I/O interface of the FlexRay controller. The external host interface of the FlexRay controller is a subset of the OPB interface. See the *IBM OPB Architecture Specification version 2.1*

for more information about the OPB interface.

Table 3: **FlexRay Controller External I/O Signals**

Signal Name	Direction	Default Value	Description
Reset and Clocks			
OPB_Clk	input	n/a	**Interface clock:** All host interface signals are synchronous to this clock.
OPB_Rst	input	n/a	**Reset:** The FlexRay controller is reset to the default state upon assertion of this signal.
Flex_Clk	input	n/a	**System clock**: The PE module is clocked by this clock.
Sample_Clk	input	n/a	**Sample clock**: The PE module uses this clock for oversampling the input bit stream.
Interrupt Lines			
Flex_Cirpt	output	0	**FlexRay Controller interrupt**: Asserted to indicate a FlexRay controller Status interrupt condition to the microprocessor or interrupt controller.
Flex_Txbirpt	output	0	**Transmit Buffer interrupt**: Asserted to indicate a Transmit Buffer interrupt condition to the microprocessor or interrupt controller.
Flex_Rxbirpt	output	0	**Receive Buffer Interrupt**: Asserted to indicate a Receive Buffer interrupt condition to the microprocessor or interrupt controller.
Flex_Rxfirpt	output	0	**Receive FIFO Interrupt**: Asserted to indicate a Receive FIFO interrupt condition to the microprocessor or interrupt controller.
OPB Slave Signals			
OPB_select	input	n/a	**Select:** Indicates an active read or write access. This signal qualifies all bus inputs from the OPB master.
OPB_RNW	input	n/a	**Read not Write:** A logic '1' indicates a read access to the location address by OPB_Abus. A logic '0' indicates a write access to the location addressed by OPB_Abus.
OPB_seqAddr	input	n/a	**Sequential Transfer:** Indicates that the transfer being performed will be followed with a transfer to the next sequential address in the same direction, read or write. (**Reserved for future use**.)
OPB_ABus(0:31)	input	n/a	**Address:** Bus used to specify the address being accessed either for a read or write.
OPB_BE(0:3)	input	n/a	**Byte Enable:** Selects which byte lane of the data bus is being accessed. The FlexRay controller does not support individual byte accesses. The OPB_BE must always be driven to "1111".
OPB_DBus(0:31)	input	n/a	**Write Data Bus:** Data to be written to the address specified by OPB_Abus. The write is acknowledged by Sln_xferAck when complete.

Table 3: **FlexRay Controller External I/O Signals *(Continued)***

Signal Name	Direction	Default Value	Description
Sln_DBus(0:31)	output	X"00000000"	**Read_Data:** Data read from the address OPB_ABus. Data is valid when a read request followed by an acknowledge (Sln_xfer Ack) is asserted
Sln_xferAck	output	0	**Acknowledge:** Acknowledgement of a completed read or write transfer. Following a read request, indicates that the data on Sln_Dbus is valid. Follow a write request, indicates that the data on OPB_DBus was accepted.
Sln_Retry	output	0	**Retry:** Asserted to indicate that the FlexRay controller is unable to perform the transfer requested at this time. This signal will be asserted instead of Sln_xferAck when required. **(Reserved for Future Use)**.
Sln_ToutSup	output	0	**Time Out Suppress:** Asserted to indicate to the OPB arbiter that the bus operation will be delayed for an extended period of time. **(Reserved for Future Use).**
Sln_ErrAck	output	0	**Error Acknowledge:** Asserted to indicate that an error was encountered during the requested transfer. Asserted coincident with Sln_xferAck if asserted. **(Reserved for Future Use).**
PHY Interface			
TxD	Output	1	**Transmit Phy line:** The Transmit Phy line connected to the FlexRay PHY chip.
TxD_En	Output	1	**Transmit Enable Phy line:** The Transmit Enable signal connected to the FlexRay PHY chip. When '0' indicates an active transmission on the TxD line.
RxD	Input	1	**Receive Phy line:** The Receive Phy line connected to the FlexRay controller. The source of RxD is the FlexRay PHY chip.

The external host interface of the FlexRay node is a 32-bit OPB bus. Byte-access is not supported by the FlexRay controller.

Single Read/Write

The FlexRay controller supports the Single Read/Write mode of data transfer.

Single Read

When the Read transfer is enabled (`OPB_select` = '1' and `OPB_RNW` = '1'), the controller samples the address on the `OPB_ABus` and returns the corresponding read data on the `Sl_DBus`. Read data is returned on a successive clock rising edge. `Sl_xferAck` is asserted when the data is ready on the `Sl_DBus` pins. The address indicated by the `OPB_ABus` is assumed to be valid for the entire duration during which `OPB_select` is asserted.

`Sl_xferAck` is asserted for all read transactions. Successive single read operations require that the `OPB_select` be deasserted and reasserted. The timing diagram for a single read transaction is shown in Figure 5.

For single read transactions:

- Reads from address locations defined as reserved will return all 0s on the `Sln_DBus` bus.
- Reads from write only address locations will return all 0s on the `Sln_DBus` bus.

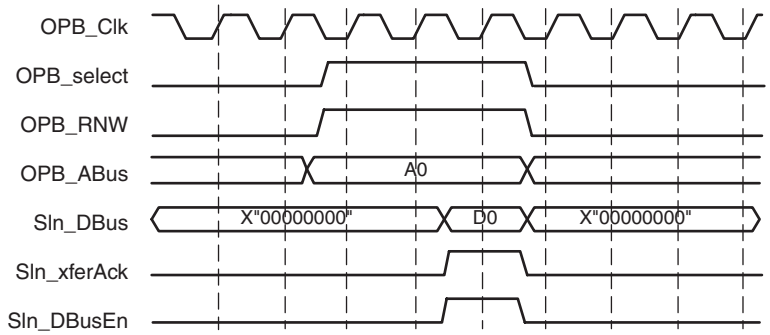

Figure 5: **Single Read Transaction**

Single Write

When the Write transfer is enabled (`OPB_select` = '1' and `OPB_RNW` = '0'), the controller samples both the address on the `OPB_ABus` and the data on the `OPB_DBus`. The write operation is completed on a successive clock rising edge. The completion of the write operation is indicated by the assertion of the `Sl_xferAck`. The address indicated by the `OPB_ABus` and the data indicated by the `OPB_Dbus` is assumed to be valid for the entire duration during which `OPB_select` is asserted.

The `Sl_xferAck` signal is asserted for all write transactions. Successive write operations require that `OPB_select` be deasserted and reasserted. The timing diagram for a single write transaction is shown in Figure 6.

For single write transactions:

- Writes to address locations and bits defined as reserved will not have any effect.
- Writes to address locations defined as read only will not have any effect.

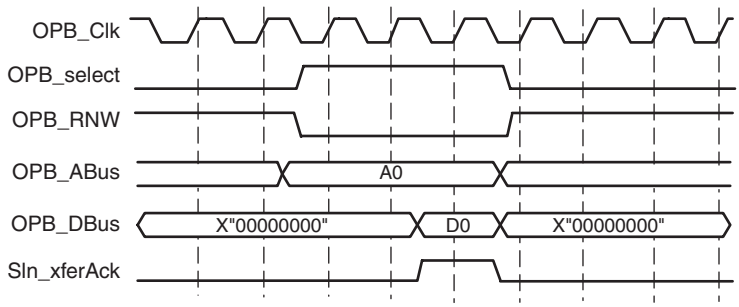

Figure 6: **Single Write Transaction**

Resource Utilization

The FlexRay core can be generated in numerous configurations to provide maximum flexibility for specific user applications. For this reason, resource utilization heavily depends on the core configuration selected. Table 4 provides sample resource utilization values. All cores are generated with minimum and maximum values of parameters `Max Payload Size`, `Number of Tx Buffers`, `Number of Rx Buffers`, and Depth of FIFO.

Table 4: **Slice Count for Different FPGA Devices**

Configuration	LUTs	FFs	Block RAMs
Minimum Configuration	6068	3608	9
Maximum Configuration	8650	4541	18

Minimum Configuration. `Number of Tx Buffers`: 2, `Number of Rx Buffers`: 2, `Rx FIFO Depth`: 4, `Max Payload size`: 4 bytes.

Maximum Configuration. `Number of Tx Buffers`: 128, `Number of Rx Buffers`: 128, `Rx FIFO Depth`: 128, `Max Payload Size`: 56 bytes

The most effective method for determining the resource requirement for the FlexRay core is to generate the core with the required parameters for the user application, and then run the provided implementation script. See the *FlexRay v1.1 Getting Started Guide* for information.

References

1. *FlexRay Communication System Protocol Specification Version 2.1 Revision A.*
2. *IBM OPB Architecture Specification version 2.1.*

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled.

Ordering Information

The FlexRay core is provided under the SignOnce IP Site License and can be generated using the Xilinx CORE Generator system v9.1i or higher. The CORE Generator system is shipped with Xilinx ISE Foun-

dation Series Development software.

A simulation evaluation license for the core is shipped with the CORE Generator system. To access the full functionality of the core, including FPGA bitstream generation, a full license must be obtained from Xilinx. For more information, please visit the product lounge at:

www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=DO-DI-FLEXRAY.

Please contact your local Xilinx sales representative for pricing and availability of additional Xilinx LogiCORE modules and software. Information about additional Xilinx LogiCORE modules is available on the Xilinx IP Center.

Revision History

Date	Version	Revision
05/17/07	1.1	Initial Xilinx release

09/29/09 - This is the final publication. No content was changed.