36. Avalon Streaming Data Pattern Generator and Checker Cores

Core Overview

The data generation and monitoring solution for Avalon® Streaming (Avalon-ST) interfaces consists of two components: a data pattern generator core that generates data patterns and sends it out on an Avalon-ST interface, and a data pattern checker core that receives the same data and checks it for correctness.

Both cores are SOPC Builder-ready and integrate easily into any SOPC Builder-generated system.

This chapter contains the following sections:

Data Pattern Generator

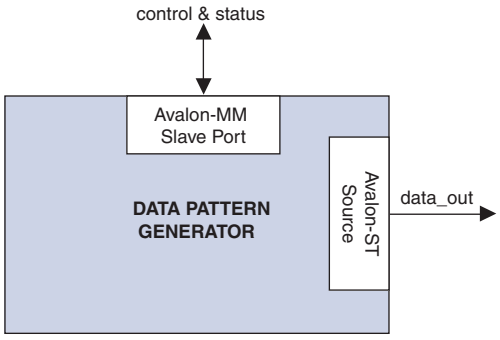
This section describes the hardware structure and functionality of the data pattern generator core.

Functional Description

The data pattern generator core accepts commands to generate and drive data onto an Avalon-ST source interface.

Figure 36–1 shows a block diagram of the data pattern generator core.

Figure 36–1. Data Pattern Generator Core Block Diagram



You can configure the width of the output data signal to either 32-bit or 40-bit when instantiating the core.

You can configure this core to output 8-bit or 10-bit wide symbols. By default, the core generates 4 symbols per beat, which outputs 32-bit or 40-bit wide data to the Avalon-ST interfaces, respectively. The core's data format endianness is the most significant symbol first within a beat and the most significant bit first within a symbol. For example, when you configure the output data to 32-bit, bit 31 is the first data bit, followed by bit 30, and so forth. This interface's endianness may change in future versions of the core.

For smaller data widths, you can use the Avalon-ST Data Format Adapter for data width adaptation. The Avalon-ST Data Format Adapter converts the output from 4 symbols per beat, to 2 or 1 symbol per beat. In this way, the 32-bit output of the core can be adapted to a 16-bit or 8-bit output and the 40-bit output can be adapted to a 20-bit or 10-bit output.

For more information about the Avalon-ST Data Format Adapter, refer to *SOPC Builder User Guide*.

Control and Status Register Interface

The control and status register interface is an Avalon-MM slave that allows you to enable or disable the data generation. This interface also provides the run-time ability to choose data pattern and inject an error into the data stream.

Output Interface

The output interface is an Avalon-ST interface. You can configure the data width at the output interface to suit your requirements.

Clock Interface

The data pattern generator core has separate clock domains for the control and status register interface (Avalon-MM) and the output interface (Avalon-ST) to allow the two interfaces to operate in different frequencies.

Supported Data Patterns

The following data patterns are supported in the following manner, per beat. When the core is disabled or is in idle state, the default pattern generated on the data output is 0×5555 (for 32-bit data width) or 0×55555 (for 40-bit data width). This core does not support custom data patterns.

Table 36–1 lists the supported data patterns for the data pattern generator core.

Table 36–1. Supported Data Patterns (Binary Encoding) ***(Note 1)***

Pattern	32-bit	40-bit
PRBS-7	PRBS in parallel	PRBS in parallel
PRBS-15	PRBS in parallel	PRBS in parallel
PRBS-23	PRBS in parallel	PRBS in parallel
PRBS-31	PRBS in parallel	PRBS in parallel
High Frequency	10101010 x 4	1010101010 x 4

Table 36–1. Supported Data Patterns (Binary Encoding) ***(Note 1)***

Pattern	32-bit	40-bit
Low Frequency	11110000 x 4	1111100000 x 4

Note to Table 36–1:

(1) All PRBS patterns are seeded with 11111111.

Inject Error

Errors can be injected into the data stream by controlling the `Inject Error` register bits in the register map (refer to Table 36–6 on page 36–7). When the inject error bit is set, one bit of error is produced by inverting the LSB of the next data beat.

If the inject error bit is set before the core starts generating the data pattern, the error bit is inserted in the first output cycle.

The `Inject Error` register bit is automatically reset after the error is introduced in the pipeline, so that the next error can be injected.

Preamble Mode

The preamble mode is used for synchronization or word alignment. When the preamble mode is set, the preamble control register sends the preamble character a specified number of times before the selected pattern is generated, so the word alignment block in the receiver can determine the word boundary in the bit stream.

The number of beats (`NumBeats`) determines the number of cycles to output the preamble character in the preamble mode. You can set the number of beats (`NumBeats`) in the preamble control register. The default setting is 0 and the maximum value is 255 beats. This mode can only be set when the data pattern generation core is disabled.

Configuration

The following section lists the available option in the MegaWizard™ interface.

Output Parameter

You can configure the output interface of the data pattern generator core using the following parameter:

- **ST_DATA_W** — The width of the output data signal that the data pattern generator core supports. Valid values are 32 and 40.

Data Pattern Checker

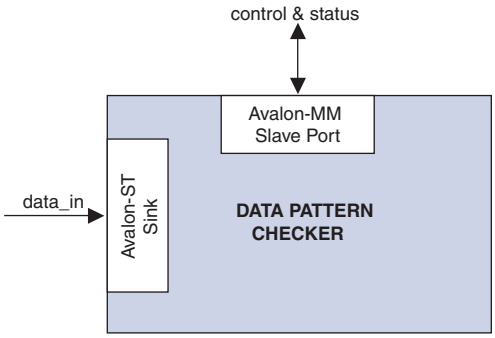
This section describes the hardware structure and functionality of the data pattern checker core.

Functional Description

The data pattern checker core accepts data via an Avalon-ST sink interface, checks it for correctness against the same predetermined pattern used by the data pattern generator core or other PRBS generators to produce the data, and reports any exceptions to the control interface.

Figure 36–2 shows a block diagram of the data pattern checker core.

Figure 36–2. Data Pattern Checker



You can configure the width of the output data signal to either 32-bit or 40-bit when instantiating the core. The chosen data width is not configurable during run time.

You can configure this core to output 8-bit or 10-bit wide symbols. By default, the core generates 4 symbols per beat, which outputs 32-bit or 40-bit wide data to the Avalon-ST interfaces, respectively. The core's data format endianness is the most significant symbol first within a beat and the most significant bit first within a symbol. For example, when you configure the output data to 32-bit, bit 31 is the first data bit, followed by bit 30, and so forth. This interface's endianness may change in future versions of the core.

If you configure the width of the output data to 32-bit, the core inputs four 8-bit wide symbols per beat. To achieve an 8-bit and 16-bit data width, you can use the Avalon-ST Data Format Adapter component to convert 4 symbols per beat to 1 or 2 symbols per beat.

Similarly, if you configure the width of the output data to 40-bit, the core inputs four 10-bit wide symbols per beat. The 10-bit and 20-bit input can be achieved by switching from 4 symbols per beat to 1 and 2 symbols per beat.

Control and Status Register Interface

The control and status interface is an Avalon-MM slave that allows you to enable or disable the pattern checking. This interface also provides the run-time ability to choose the data pattern and read the status signals.

Input Interface

The input interface is an Avalon-ST interface. You can configure the data width at this interface to suit your requirements.

Clock Interface

The data pattern checker core has separate clock domains for the control and status register interface (Avalon-MM) and the input interface (Avalon-ST) to allow the two interfaces to operate in different frequencies.

Supported Data Patterns

The following data patterns are supported in the following manner, per beat. When the core is disabled or in idle state, the default pattern generated on the data output is 0×5555 (for 32-bit data width) or 0×55555 (for 40-bit data width).

Table 36–2 lists the supported data patterns for the data pattern checker core.

Table 36–2. Supported Data Patterns (Binary Encoding)

Pattern	32-bit	40-bit
PRBS-7	PRBS in parallel	PRBS in parallel
PRBS-15	PRBS in parallel	PRBS in parallel
PRBS-23	PRBS in parallel	PRBS in parallel
PRBS-31	PRBS in parallel	PRBS in parallel
High Frequency	10101010 x 4	1010101010 x 4
Low Frequency	11110000 x 4	1111100000 x 4

Lock

The LOCKED bit in the status register is asserted when 40 consecutive beats of correct data are received. This bit is deasserted and the receiver loses the lock when 40 consecutive beats of incorrect data are received.

Bit and Error Counters

The core has two 64-bit internal counters to keep track of the number of bits and number of error bits received. A snapshot has to be executed to update the NumBits and NumErrors registers with the current value from the internal counters.

A counter reset can be executed to reset both the registers and internal counters. If the counters are not being reset and the core is enabled, the internal counters continues the increment base on their current value.

The internal counters only start to increment after a lock has been acquired.

Clock Sensor

The clock sensor register checks whether the clock is idle by detecting the postive edge of the clock. The CLOCK RUNNING bit in the clock sensor register shows the Avalon-ST clock status, where 1 indicates that the clock is running and 0 indicates that the clock is idle. This bit remains set until you manually reset it by writing to the RESET_CLOCK_RUNNING bit.

After a write to the RESET_CLOCK_RUNNING bit, you should wait for a reasonable amount of time to detect a positive edge of the clock before reading the CLOCK RUNNING bit again.

Configuration

The following section lists the available option in the MegaWizard™ interface.

Input Parameter

You can configure the input interface of the data pattern checker core using the following parameter:

- **ST_DATA_W** — The width of the input data signal that the data pattern checker core supports. Valid values are 32 and 40.

Hardware Simulation Considerations

The data pattern generator and checker cores do not provide a simulation testbench for simulating a stand-alone instance of the component. However, you can use the standard SOPC Builder simulation flow to simulate the component design files inside an SOPC Builder system.

Software Programming Model

This section describes the software programming model for the data pattern generator and checker cores.

Register Maps

This section describes the register maps for the data pattern generator and checker cores.

Data Pattern Generator Control Registers

Table 36–3 shows the register map for the data pattern generator core control registers. To access each register, add the *BASE* address to the offset value.

Table 36–3. Data Pattern Generator Core Register Map ***(Note 1)***

Offset	Register Name	R/W	Bit Description														
			31	...	16	15	...	9	8	7	6	5	4	3	2	1	0
0	Enable	RW	Reserved														ENABLE
1	Pattern Select	RW	Reserved									LF	HF	PRBS 31	PRBS 23	PRBS 15	PRBS 7
2	Inject Error	RW	Reserved														INJECT
3	Preamble Control	RW	Reserved			NUM BEATS				Reserved							ENABLE PREAMBLE
4	Preamble Character (Low Bits)	RW	PREAMBLE LO														

Table 36–3. Data Pattern Generator Core Register Map ***(Note 1)***

Offset	Register Name	R/W	Bit Description													
			31	...	16	15	...	9	8	7	6	5	4	3	2	1	0
5	Preamble Character (High Bits)	RW	Reserved							PREAMBLE HI							

Note to Table 36–3:

(1) Reserved fields—Read values are undefined. Write zero.

Table 36–4 describes the Enable register bits. This register enables or disables the pattern generation.

Table 36–4. Enable Field Descriptions ***(Note 1)***

Bit(s)	Name	Access	Description
[0]	ENABLE	RW	Setting this bit to 1 enables the data pattern generator core.
[31:1]	Reserved		

Note to Table 36–4:

(1) When the core is enabled, only the Enable register and the Inject Error register have write access. Write access to all other registers are ignored. When the core is disabled, the final output is observed at the next clock cycle.

Table 36–5 describes the Pattern Select register bits.

Table 36–5. Pattern Select Field Descriptions ***(Note 1)***

Bit(s)	Name	Access	Description
[0]	PRBS7	RW	Setting this bit to 1 outputs a PRBS 7 pattern with T [7, 6].
[1]	PRBS15	RW	Setting this bit to 1 outputs a PRBS 15 pattern with T [15, 14].
[2]	PRBS23	RW	Setting this bit to 1 outputs a PRBS 23 pattern with T [23, 18].
[3]	PRBS31	RW	Setting this bit to 1 outputs a PRBS 31 pattern with T [31, 28].
[4]	HF	RW	Setting this bit to 1 outputs a constant pattern of 0101010101… bits.
[5]	LF	RW	Setting this bit to 1 outputs a constant word pattern of 1111100000 for 10-bit words, or 11110000 for 8-bit words.
[31:6]	Reserved		

Note to Table 36–5:

(1) This register is one-hot encoded where only one of the pattern selector bits should be set to 1. For all other settings, the behaviors are undefined.

Table 36–6 describes the Inject Error register bits. This register allows you to set the error inject bit and insert one bit of error into the stream.

Table 36–6. Inject Error Field Descriptions ***(Note 1)***

Bit(s)	Name	Access	Description
[0]	INJECT	RW	Setting this bit to 1 injects error into the stream. The bit stays high until the error is injected. Setting this bit to 1 while the bit is high has no effect.
[31:1]	Reserved		

Note to Table 36–6:

(1) The data beat that is injected with error might not be observed from the source if the core is disabled while the error is injected into the stream.

Table 36–7 describes the Preamble Control register bits. This register enables preamble and set the number of cycles to output the preamble character.

Table 36–7. Preamble Control Field Descriptions

Bit(s)	Name	Access	Description
[0]	ENABLE PREAMBLE	RW	Setting this bit to 1 at the start of pattern generation, enables the preamble character to be sent for NUM BEATS cycles before switching over to the selected pattern.
[7:1]	Reserved		
[15:8]	NUM BEATS	RW	The number of beats to repeat the preamble character.
[31:16]	Reserved		

Table 36–8 describes the Preamble Character (Low Bits) register bits. This register is for the user-defined preamble character (bit 0-31).

Table 36–8. Preamble Character Low Bits Field Descriptions

Bit(s)	Name	Access	Description
[31:0]	PREAMBLE LO	RW	Sets bit 31-0 for the preamble character to output.

Table 36–9 describes the Preamble Character (High Bits) register bits. This register is for the user-defined preamble character (bit 32-39) but is ignored if the ST_DATA_W value is set to 32.

Table 36–9. Preamble Character High Bits Field Descriptions

Bit(s)	Name	Access	Description
[7:0]	PREAMBLE HI	RW	Sets bit 39-32 for the preamble character. This is ignored when the ST_DATA_W value is set to 32.
[31:8]	Reserved		

Data Pattern Checker Control and Status Registers

Table 36–10 shows the register map for the data pattern checker core control and status registers. To access each register, add the *BASE* address to the offset value.

Table 36–10. Data Pattern Checker Core Register Map *(Note 1)*

Offset	Register Name	R/W	Bit Description														
			31	...	16	15	...	9	8	7	6	5	4	3	2	1	0
0	Status	RW	Reserved													LOCKED	ENABLE
1	Pattern Set	RW	Reserved									LF	HF	PRBS 31	PRBS 23	PRBS 15	PRBS 7
2	Counter Control	RW	Reserved						VALID	Reserved						CLEAR	SNAP
3	NumBits (Lower Bits)	R	NUM BITS LO														

Table 36–10. Data Pattern Checker Core Register Map ***(Note 1)***

Offset	Register Name	R/W	Bit Description														
			31	...	16	15	...	9	8	7	6	5	4	3	2	1	0
4	NumBits (Higher Bits)	R	NUM BITS HI														
5	NumErrors (Lower Bits)	R	NUM ERROR LO														
6	NumErrors (Higher Bits)	R	NUM ERROR HI														
7	Clock Sensor	RW	Reserved													CLOCK RUNNING	RESET CLOCK RUNNING

Note to Table 36–3:

(1) Reserved fields—Read values are undefined. Write zero.

Table 36–11 describes the `Status` register bits.

Table 36–11. Status Field Descriptions ***(Note 1)***

Bit(s)	Name	Access	Description
[0]	ENABLE	RW	Setting this bit to 1 enables pattern checking.
[1]	LOCKED	R	Indicate lock status. Writing to this bit has no effect.
[31:2]	Reserved		

Note to Table 36–11:

(1) When the core is enabled, only the `Status` register's `ENABLE` bit and the `Counter Control` register's `SNAP` and `CLEAR` bits have write access. Write access to all other registers are ignored.

Table 36–12 describes the `Pattern Select` register bits. This register is one-hot encoded where only one of the pattern selector bits should be set to 1. For all other settings, the behaviors are undefined.

Table 36–12. Pattern Select Field Descriptions

Bit(s)	Name	Access	Description
[0]	PRBS7	RW	Setting this bit to 1 compares the data to a PRBS 7 pattern with T [7, 6].
[1]	PRBS15	RW	Setting this bit to 1 compares the data to a PRBS 15 pattern with T [15, 14].
[2]	PRBS23	RW	Setting this bit to 1 compares the data to a PRBS 23 pattern with T [23, 18].
[3]	PRBS31	RW	Setting this bit to 1 compares the data to a PRBS 31 pattern with T [31, 28].
[4]	HF	RW	Setting this bit to 1 compares the data to a constant pattern of 0101010101… bits.
[5]	LF	RW	Setting this bit to 1 compares the data to a constant word pattern of 1111100000 for 10-bit words, or 11110000 for 8-bit words.
[31:6]	Reserved		

Table 36–13 describes the Counter Control register bits. This register snapshots and resets the NumBits and NumErrors register, and also the internal counters.

Table 36–13. Counter Control Field Descriptions

Bit(s)	Name	Access	Description
[0]	SNAP *(1)*	W	Writing this bit to 1 captures the number of bits received and number of error bits received from the internal counters to the respective NumBits and NumErrors registers within the same clock cycle. Writing this bit to 1 after disabling the core will still capture the correct values from the internal counters to the NumBits and NumErrors registers.
[1]	CLEAR *(1)*	W	Writing this bit to 1 resets all internal counters and snapped registers. You can reset the counters and registers even when the Avalon-ST clock is idle. Re-enabling the core does not automatically reset the number of bits received and number of error bits received in the internal counter. This bit resets itself automatically after the reset process.
[7:2]	Reserved		
[8]	VALID	R	indicates the validity of the NumBits and NumErrors register values. Writing to this bit has no effect. This bit is driven low after the SNAP or CLEAR bit is set to 1.
[31:9]	Reserved		

Note to Table 36–13:

(1) The SNAP and CLEAR bits have no effect on the counters if the VALID bit is low.

Table 36–14 describes the NumBits (Lower Bits) register bits. This register is the lower word of the 64-bit bit counter snapshot value. The register resets when the component-reset is asserted or when the CLEAR bit is set to 1.

Table 36–14. NumBits (Lower Bits) Field Descriptions

Bit(s)	Name	Access	Description
[31:0]	NUM BITS LO	R	Sets bit 31-0 for the NumBits (number of bits received).

Table 36–15 describes the NumBits (Higher Bits) register bits. This register is the higher word of the 64-bit bit counter snapshot value. The register resets when the component-reset is asserted or when the CLEAR bit is set to 1.

Table 36–15. NumBits (Higher Bits) Field Descriptions

Bit(s)	Name	Access	Description
[31:0]	NUM BITS HI	R	Sets bit 63-32 for the NumBits (number of bits received).

Table 36–16 describes the NumErrors (Lower Bits) register bits. This register is the lower word of the 64-bit error counter snapshot value. The register resets when the component-reset is asserted or when the CLEAR bit is set to 1.

Table 36–16. NumErrors (Lower Bits) Field Descriptions

Bit(s)	Name	Access	Description
[31:0]	NUM ERROR LO	R	Sets bit 31-0 for the NumErrors (number of error bits received).

Table 36–17 describes the NumErrors (Higher Bits) register bits. This register is the higher word of the 64-bit error counter snapshot value. The register resets when the component-reset is asserted or when the CLEAR bit is set to 1.

Table 36–17. NumErrors (Higher Bits) Field Descriptions

Bit(s)	Name	Access	Description
[31:0]	NUM ERROR HI	R	Sets bit 63-32 for the NumErrors (number of error bits received).

Table 36–13 describes the Clock Sensor register bits. This register check whether the Avalon-ST clock is in idle state. Use this bit if the VALID bit does not go high in a reasonable amount of time—where a positive edge of the Avalon-ST clock can be detected.

Table 36–18. Clock Sensor Field Descriptions

Bit(s)	Name	Access	Description
[0]	RESET CLOCK RUNNING	W	Writing this bit to 1 clears the CLOCK RUNNING bit.
[1]	CLOCK RUNNING	R	Indicates the clock status. This bit goes high when the clock is running and goes low when the clock is idle. To clear this bit, reset the system or set the RESET CLOCK RUNNING bit to 1 and wait for a reasonable amount of time before reading this bit.
[31:2]	Reserved		

Document Revision History

The following table shows the revision history for this document.

Date	Version	Changes
June 2011	11.0	■ Updated the functional description of the data pattern generator and data pattern checker core. ■ Revised the core's register map in Table 36–3 and Table 36–10 for a better representation. ■ Updated the register bits in "Register Maps" on page 36–6. ■ Added a new register in Table 36–10. ■ Converted the document to new frame template version 2.0 and made textual and style changes.
December 2010	10.1	Removed the "Device Support", "Instantiating the Core in SOPC Builder", and "Referenced Documents" sections.
July 2010	10.0	No change from previous release.
January 2010	9.1 SP1	Initial release.

For previous versions of this chapter, refer to the Quartus II Handbook Archive.

Section V. Clock Control Peripherals

This section describes clock control peripherals provided by Altera for SOPC Builder systems.

This section includes the following chapter:

- Chapter 38, PLL Cores

For information about the revision history for chapters in this section, refer to each individual chapter for that chapter's revision history.