TM FAST Library

Manual

Table of Contents

1 Overview

This document describes the TM FAST library available to the user. This library is available in VHDL and it contains Quartus Block Symbol files (BSF) for creating Quartus Schematic.

The purpose of this library is primarily to aid in creation of TM FAST user application logic, however, the library can be used for migration of existing projects containing FM352-5 Boolean module(s). The library components can be found in their original form in chapters 6.9 and 6.10 of the **FM 325-5 high-speed Boolean processor** operating manual. The described symbols are from the instruction set for LAD programming of the FB library of operations for the FM 352-5 module.

1.1 TM FAST library block

In order to use a TM FAST library block connect the ports of the VHDL entity to signals in the user application.

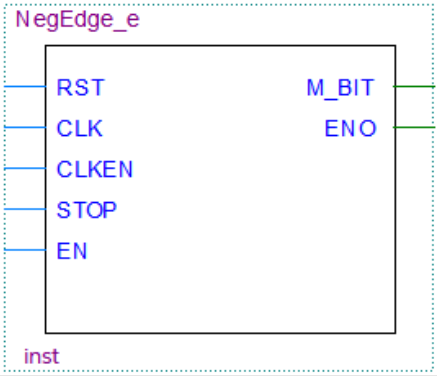
Example: Entity of NegEdge logic:

```
entity NegEdge_e is
    Port (
        RST   : in    STD_LOGIC;
        CLK   : in    STD_LOGIC;
        CLKEN: in    STD_LOGIC;
        STOP : in    STD_LOGIC;
        EN    : in    STD_LOGIC;
        M_BIT: inout STD_LOGIC;
        ENO   : inout STD_LOGIC
    );
end NegEdge_e;
```

- RST: master reset held by the system logic until the FPGA is activated.
- CLK: is connected to F_CLK_USER selected by the user in the Quartus project provided by Siemens.
- CLKEN: is connected to the clock-enable signal or to one of the PHASE clocks (if Phases will be used) selected by the user in the Quartus project provided by Siemens.
- STOP: is the present state of the RUN/STOP status of the CPU program.
- EN: is the input signal connection
- ENO: is the output signal connection
- M_BIT: is the bit address parameter which stores the results of the logic operation(RLO).

In order to use the TM FAST library Quartus Schematic Block Symbol (BSF) of a logic, connect the ports of the Block Symbol to your Quartus Schematic user application.

Example: The Quartus Schematic Block Symbol (BSF) symbol for the NegEdge component in Quartus Schematic contains all required signals.



1.2 Instantiation of logic blocks in the application

Example of an instantiation of a 16-bit counter logic block (FB121 CTU16):

In order to use a logic block in the user application, you need to instantiate the logic block in your user application. In this example we call the 16-bit up counter *FB121_CTU16.*

Entity signals of the FB121 CTU16 logic block:

```
entity FB121_CTU16_e is
    port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        CU   : in  STD_LOGIC;
        R    : in  STD_LOGIC;
        PV   : in  STD_LOGIC_VECTOR (15 downto 0);
        Q    : out STD_LOGIC;
        CV   : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB121_CTU16_e;
```

Instantiation of a 16-bit counter logic block (FB121 CTU16) in the VHDL application logic:

Each signal of the entity FB121_CTU16_e of the instantiated logic block must have values.

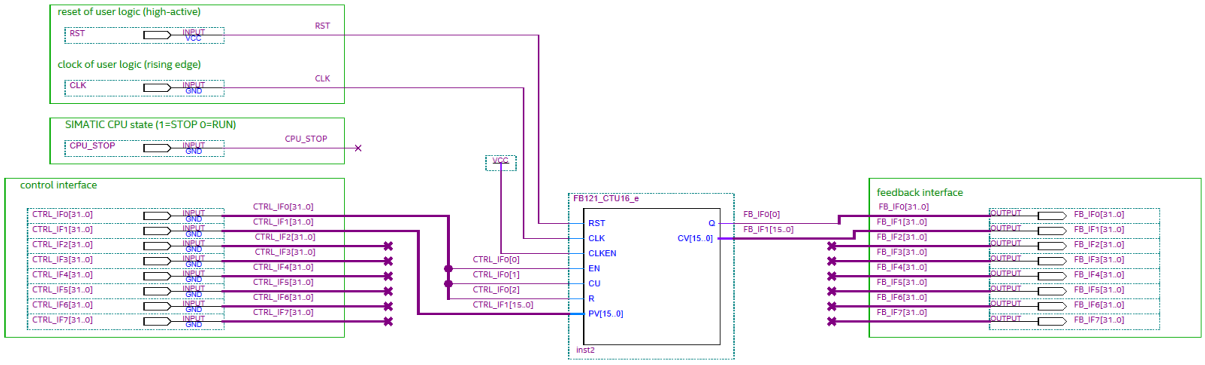
On the left side you find the ports of entity of the logic block and on the right side the connected signal in the user application.

```
CTU_16_EXAMPLE: entity work.FB121_CTU16_e(FB121_CTU16_a)
PORT MAP (
    RST      => RST,                    -- Reset signal of the application logic
    CLK      => CLK,                    -- Clock signal
    CLKEN    => ‘1’,                    -- Clock-enable signal
    EN       => '1',                    -- Enable signal,
```

```
    CU          => DI(0),                   -- Counter signal for counting up
    R           => CTRL_IF(0),              -- Reset input of counter
    PV          => PRESET_VAL,              -- Preset value
    Q           => STATUS,                  -- Status of the counter
    CV          => COUNT_VAL                -- Counter value
  );
```

Instantiation of a 16-bit counter logic block (FB121 CTU16) in the Quartus Schematic application logic:

Connect the inputs and outputs of the logic block to other logic blocks or to the ports of the user application.



1.3 TM FAST use of phase clocks

While it is not necessary to use any Phase clocks in the user design, the FM352-5 used phase clocks to help the user avoid race/timing issues without the use of constraints files.

The FM352-5 ran on a master clock of 12 MHz with 12 Phase clocks (each at 1 MHz). The TM FAST allows the user to select the clock frequency for the user application, F_CLK_USER. This clock can be selected to be 5 MHz, 15 MHz, 25 MHz, 50 MHz and 75 MHz.

To limit the logic required, a limit of 15 Phases is allowed in generation of Phase clocks. This means that using a 15 MHz F_CLK_USER, the user can select to use 15 Phase clocks (PHASE_MAX_QUANTITY := 14, 0-14) to generate the same 1-MHz Phase clock rate.

Phase 0 is considered the IO scan external to the user application. Therefore, Phase(1) is the first phase to be utilized in the user logic. A signal is propagated through the user program from function to function. If the result shall be available after 1 system cycle, then the second function should use Phase(2), third function Phase(3) and so on. If the overall application has more than 14 clocked functions that it must propagate through, then it will require more than one program cycle to complete, and the user must take this into account, related to his expected results.

2 TM FAST library: Instruction Set

The following table lists the symbolic names and description for each TM FAST library component.

Table 1: Instruction Set

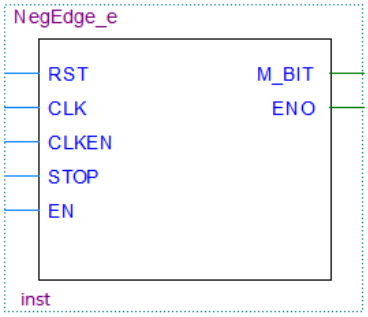
TM FAST Symbolic Name	FM 352-5 Symbolic Name	Description
NegEdge	-(N)-	Detect negative RLO edge
NegSignalEdge	NEG	Negative edge detection
PosEdge	-(P)-	Detect positive RLO edge
PosSignalEdge	POS	Positive edge detection
RSFF	RS	Reset/set flip-flop
SRFF	SR	Set/reset flip-flop
Compare_W	CMP	Comparison function (16-bit)
Compare_DW	CMP	Comparison function (32-bit)
INV_I	INV_I	Generate one's complement (16-bit)
INV_I_U		Generate one's complement (16-bit), unlatched logic
INV_DI	INV_DI	Generate one's complement (32-bit)
INV_DI_U		Generate one's complement (32-bit), unlatched logic
I_DI	I_DI	Convert Integer (16-bit) to double integer (32-bit)
I_DI_U		Convert Integer (16-bit) to double integer (32-bit), unlatched logic
Move_B	MOVE	Move a specified value (8-bit)
Move_B_U		Move a specified value (8-bit), unlatched logic
Move_W	MOVE	Move a specified value (16-bit)
Move_W_U		Move a specified value (16-bit), unlatched logic
Move_DW	MOVE	Move a specified value (32-bit)
Move_DW_U		Move a specified value (32-bit), unlatched logic
ShiftRotate_W	SHL_W, SHR_W, SHR_I	Rotate and Shift operations (16 bit)
ShiftRotate_W_U		Rotate and Shift operations (16 bit), unlatched logic
ShiftRotate_DW	ROL_DW, ROR_DW, SHL_DW, SHR_DW, SHR_DI	Rotate and Shift operations (32 bit)
ShiftRotate_DW_U		Rotate and Shift operations (32 bit), unlatched logic
WordLogic_W	WAND_W, WOR_W, WXOR_W	Word operations (16 bit)
WordLogic_W_U		Word operations (16 bit), unlatched logic
WordLogic_DW	WAND_DW, WOR_DW, WXOR_DW	Word operations (32 bit)
WordLogic_DW_U		Word operations (32 bit), unlatched logic

2.1 Detect negative edge

Description

The block NegEdge detects a signal change in the M_BIT from 1 to 0 and returns RLO equal to true after the operation. The current signal state of the Result of the Logic Operation (RLO) is compared to the signal state of the M_BIT. If the signal state of the M_BIT is 1 and the RLO was 0 before the operation, the RLO will be 1 (pulse) after the operation, and 0 in all other cases. The RLO prior to the operation is stored in M_BIT.

Table 2: TM FAST Detect negative RLO edge:

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	M_BIT	STD_LOGIC	Bit storage, clocked tag	Edge memory bit that stores the previous signal state of RLO

```
entity NegEdge_e is
    Port (
        RST   : in    STD_LOGIC;
        CLK   : in    STD_LOGIC;
        CLKEN: in    STD_LOGIC;
        STOP : in    STD_LOGIC;
        EN    : in    STD_LOGIC;
        M_BIT: inout STD_LOGIC;
        ENO   : inout STD_LOGIC
    );
end NegEdge_e;
```

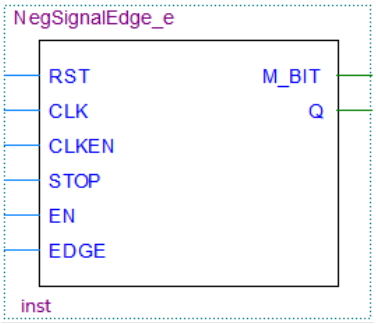
2.2 Detect negative signal edge

Description

Detect negative signal edge compares the signal state of EDGE with the signal state from the previous scan cycle that is stored in M_BIT. If the current RLO state is 1 before the operation and the state of the EDGE bit is 0 and the previous state of that bit was 1 (detect negative edge), the RLO bit will be 1 after this operation. The RLO then becomes the Q output.

You must label the M_BIT input with a unique element that is bit storage or a clocked signal.

Table 3: TM FAST Detect negative signal edge:

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	Q	STD_LOGIC	Output	One-shot output
	EDGE	STD_LOGIC	Input	Scanned signal
	MBIT	STD_LOGIC	Bit storage, clocked tag	Edge memory bit that stores the previous signal state of EDGE

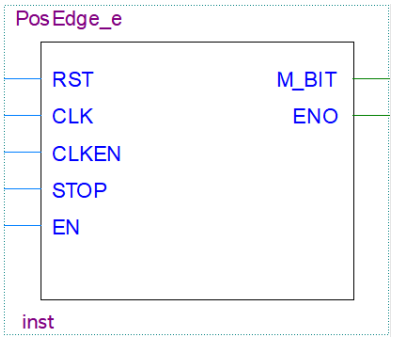
```
entity NegSignalEdge_e is
    Port (
        RST   : in    STD_LOGIC;
        CLK   : in    STD_LOGIC;
        CLKEN: in    STD_LOGIC;
        STOP  : in    STD_LOGIC;
        EN    : in    STD_LOGIC;
        EDGE  : in    STD_LOGIC;
        M_BIT: inout STD_LOGIC;
        Q     : inout STD_LOGIC
    );
end NegSignalEdge_e;
```

2.3 Detect positive edge

Description

Detect positive edge detects a signal change in the M_BIT from 0 to 1 and returns RLO equal to true after the operation. The current signal state of the RLO is compared to the signal state of the M_BIT. If the signal state of the M_BIT is 0 and the RLO was 1 before the operation, the RLO will be 1 (pulse) after the operation, and 0 in all other cases. The RLO prior to the operation is stored in M_BIT.

Table 4: TM FAST Detect positive RLO edge:

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	M_BIT	STD_LOGIC	Bit storage, clocked tag	Edge memory bit that stores the previous signal state of RLO

```
entity PosEdge_e is
    Port (
        RST   : in    STD_LOGIC;
        CLK   : in    STD_LOGIC;
        CLKEN: in    STD_LOGIC;
        STOP : in    STD_LOGIC;
        EN    : in    STD_LOGIC;
        M_BIT: inout STD_LOGIC;
        ENO   : inout STD_LOGIC
    );
end PosEdge_e;
```

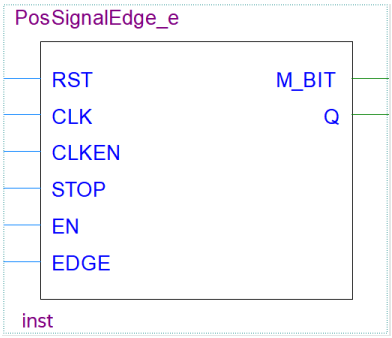
2.4 Detect positive signal edge

Description

Detect positive signal edge compares the signal state of EDGE with the signal state from the previous scan cycle that is stored in M_BIT. If the current RLO state before the operation is 0 and the state of the EDGE bit is 1 and the previous state of that bit was 0 (detect positive edge), the RLO bit will be 1 after this operation. The RLO then becomes the Q output.

You must label the M_BIT input with a unique element that is bit storage or a clocked signal.

Table 5: TM FAST Detect positive signal edge:

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	Q	STD_LOGIC	Output	One-shot output
	EDGE	STD_LOGIC	Input	Scanned signal
	MBIT	STD_LOGIC	Bit storage, clocked tag	Edge memory bit that stores the previous signal state of EDGE

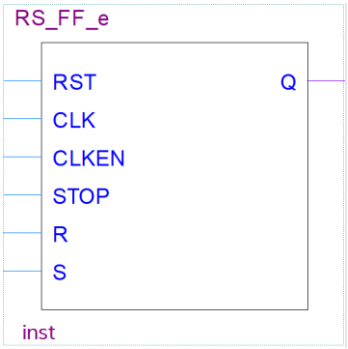
```
entity PosSignalEdge_e is
    Port (
        RST   : in    STD_LOGIC;
        CLK   : in    STD_LOGIC;
        CLKEN : in    STD_LOGIC;
        STOP  : in    STD_LOGIC;
        EN    : in    STD_LOGIC;
        EDGE  : in    STD_LOGIC;
        M_BIT : inout STD_LOGIC;
        Q     : inout STD_LOGIC
    );
end PosSignalEdge_e;
```

2.5 Reset/set flip-flop (RS_FF)

Description

RS_FF (reset/set flip-flop) is reset if the signal state is 1 at the R input and 0 at the S input. RS_FF is set if the signal state is 0 at the R input and 1 at the S input. If the RLO is 1 at both inputs, RS_FF is set.

Table 6: TM FAST Set/reset flip-flop (SR_FF):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	S	STD_LOGIC	Input	Enables set operation
	R	STD_LOGIC	Input	Enables reset operation
	Q	STD_LOGIC	Output	Signal state of output

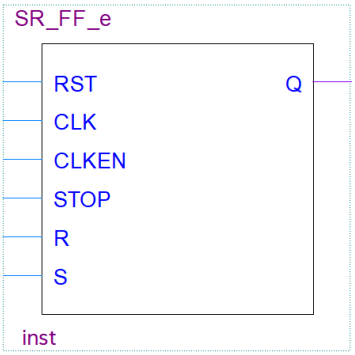
```
entity RS_FF_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        R     : in  STD_LOGIC;
        S     : in  STD_LOGIC;
        Q     : out STD_LOGIC
    );
end RS_FF_e;
```

2.6 Set/reset flip-flop (SR_FF)

Description

SR_FF (set/reset flip-flop) is set if the signal state is 1 at the S input and 0 at the R input. SR_FF is reset if the signal state is 0 at the S input and 1 at the R input. If the RLO is 1 at both inputs, SR_FF is reset.

Table 7: TM FAST Set/reset flip-flop (SR_FF):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	S	STD_LOGIC	Input	Enables set operation
	R	STD_LOGIC	Input	Enables reset operation
	Q	STD_LOGIC	Output	Signal state of output

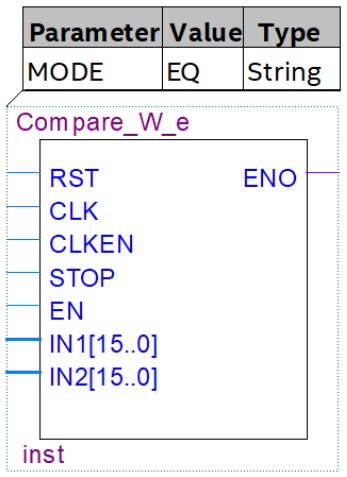
```
entity SR_FF_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        R    : in  STD_LOGIC;
        S    : in  STD_LOGIC;
        Q    : out STD_LOGIC
    );
end SR_FF_e;
```

2.7 Comparison functions (Compare_W)

Description

Compare operation with 16-bit values. Inputs IN1 and IN2 are compared according to the type of comparison you choose. If the comparison is true, the RLO of the function is 1.

Table 8: TM FAST Comparison function Compare_W

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC _VECTOR	Input, constant	First comparison value
	IN2	STD_LOGIC _VECTOR	Input, constant	Second comparison value
	MODE	STRING	Generic	“EQ”: IN1 is equal to IN2 (=) “NE”: IN1 is not equal to IN2 (/=) “GT”: IN1 is greater than IN2 (>) “LT”: IN1 is less than IN2 (<) “GE”: IN1 is greater than or equal to IN2 (>=) “LE”: IN1 is less than or equal to IN2 (<=)

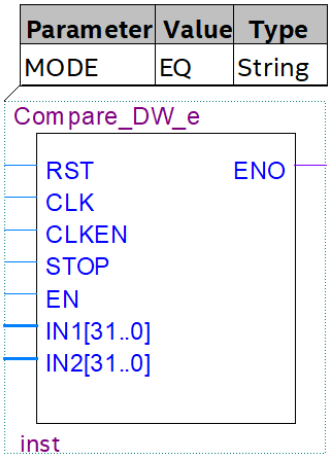
```
entity Compare_W_e is
    generic (
        MODE      : string := "EQ"
    );

    port (
            RST   : in  STD_LOGIC;
            CLK   : in  STD_LOGIC;
            CLKEN: in  STD_LOGIC;
            STOP  : in  STD_LOGIC;
            EN    : in  STD_LOGIC;
            IN1   : in  STD_LOGIC_VECTOR ( 15 downto 0 );
            IN2   : in  STD_LOGIC_VECTOR ( 15 downto 0 );
            ENO   : out STD_LOGIC := '0'
        );
end Compare_W_e;
```

2.8 Comparison functions (Compare_DW)

Description

Compare operation with 32-bit values. Inputs IN1 and IN2 are compared according to the type of comparison you choose. If the comparison is true, the RLO of the function is 1.

Table 9: TM FAST Comparison function Compare_DW

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC _VECTOR	Input, constant	First comparison value
	IN2	STD_LOGIC _VECTOR	Input, constant	Second comparison value
	MODE	STRING	Generic	“EQ”: IN1 is equal to IN2 (=) “NE”: IN1 is not equal to IN2 (/=) “GT”: IN1 is greater than IN2 (>) “LT”: IN1 is less than IN2 (<) “GE”: IN1 is greater than or equal to IN2 (>=) “LE”: IN1 is less than or equal to IN2 (<=)

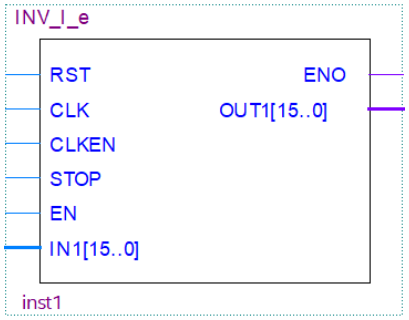
```
entity Compare_DW_e is
    generic (
        MODE      : string := "EQ"
    );

    port (
            RST   : in  STD_LOGIC;
            CLK   : in  STD_LOGIC;
            CLKEN: in  STD_LOGIC;
            STOP  : in  STD_LOGIC;
            EN    : in  STD_LOGIC;
            IN1   : in  STD_LOGIC_VECTOR ( 31 downto 0 );
            IN2   : in  STD_LOGIC_VECTOR ( 31 downto 0 );
            ENO   : out STD_LOGIC := '0'
        );
end Compare_DW_e;
```

2.9 Generate one's complement for 16-bit integer (INV_I)

Description

The INV_I operation reads the content of the IN1 parameter and performs an EXCLUSIVE OR function with the hexadecimal mask W#16#FFFF. This operation changes every bit to its opposite state. ENO always has the same signal state as EN.

Table 10: TM FAST generate one's complement for 16-bit integer (INV_I):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC _VECTOR	Input	Integer input value (16-bits)
	OUT1	STD_LOGIC _VECTOR	Output	One's complement of the 16-bit integer IN1

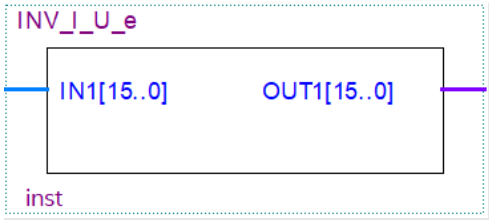
```
entity INV_I_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR (15 downto 0);
        ENO   : out STD_LOGIC;
        OUT1  : out STD_LOGIC_VECTOR (15 downto 0)
    );
end INV_I_e;
```

2.10 Generate unlatched one’s complement for 16-bit integer (INV_I_U)

Description

The INV_I_U operation reads the content of the IN1 parameter and performs an EXCLUSIVE OR function with the hexadecimal mask W#16#FFFF. This operation changes every bit to its opposite state. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 11: TM FAST generate unlatched one’s complement for 16-bit integer (INV_I_U):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC _VECTOR	Input	Integer input value (16-bits)
	OUT1	STD_LOGIC _VECTOR	Output	One’s complement of the 16-bit integer IN1

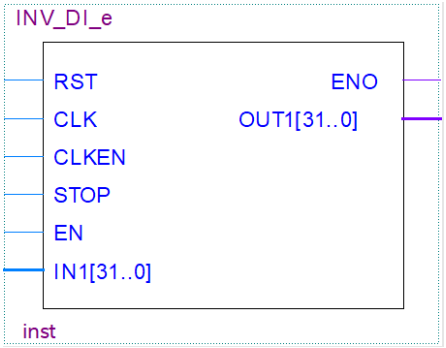
```
entity INV_I_U_e is
    Port (
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0)
    );
end INV_I_U_e;
```

2.11 Generate one's complement for 32-bit integer (INV_DI)

Description

The INV_DI operation reads the content of the IN1 parameter and performs an EXCLUSIVE OR function with the hexadecimal mask W#16#FFFF FFFF. This operation changes every bit to its opposite state. ENO always has the same signal state as EN.

Table 12: TM FAST generate one's complement for 32-bit integer (INV_DI):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC_ VECTOR	Input	Integer input value (32-bits)
	OUT1	STD_LOGIC_ VECTOR	Output	One's complement of the 32-bit integer IN1

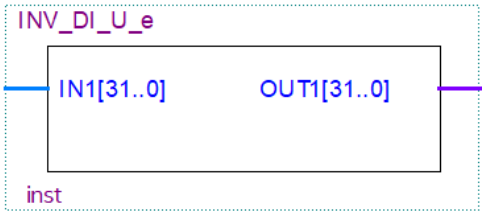
```
entity INV_DI_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end INV_DI_e;
```

2.12 Generate unlatched one's complement for 32-bit integer (INV_DI_U)

Description

The INV_DI_U operation reads the content of the IN1 parameter and performs an EXCLUSIVE OR function with the hexadecimal mask W#16#FFFF FFFF. This operation changes every bit to its opposite state. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 13: TM FAST generate unlatched one's complement for 32-bit integer (INV_DI_U):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Integer input value (32-bits)
	OUT1	STD_LOGIC_ VECTOR	Output	One's complement of the 32-bit integer IN1

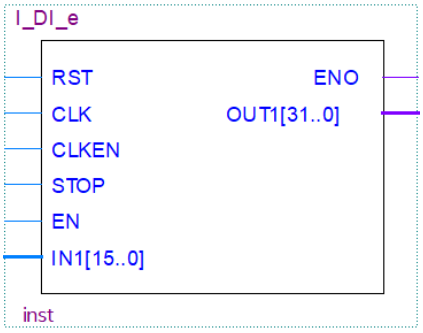
```
entity INV_DI_U_e is
    Port (
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end INV_DI_U_e;

architecture INV_DI_U_a of INV_DI_U_e is
```

2.13 Convert integer (16-bits) to double (32-bits) integer (I_DI)

Description

The I_DI operation reads the content of the IN1 parameter as an integer (16-bits) and converts it to a double integer (32-bits). The result is output by the OUT1 parameter. ENO always has the same signal state as EN.

Table 14: TM FAST convert integer (16-bits) to double (32-bits) integer (I_DI):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC_ VECTOR	Input	Integer input value (16-bits)
	OUT1	STD_LOGIC_ VECTOR	Output	Result: Double integer value (32-bits)

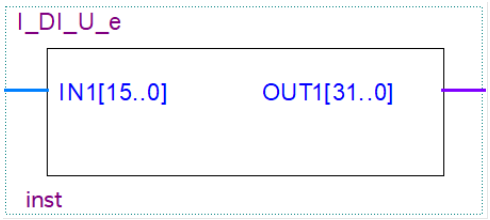
```
entity I_DI_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end I_DI_e;
```

2.14 Convert integer (16-bits) to double (32-bits) integer, unlatched logic (I_DI_U)

Description

The I_DI_U operation reads the content of the IN1 parameter as an integer (16-bits) and converts it to a double integer (32-bits). The result is output by the OUT1 parameter. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 15: TM FAST convert integer (16-bits) to double (32-bits) integer, unlatched logic (I_DI_U):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Integer input value (16-bits)
	OUT1	STD_LOGIC_ VECTOR	Output	Result: Double integer value (32-bits)

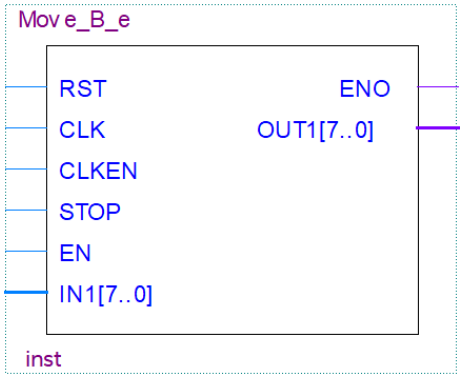
```
entity I_DI_U_e is
    Port (
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
        );
end I_DI_U_e;
```

2.15 Move (Move_B, Move_W, Move_DW)

Description

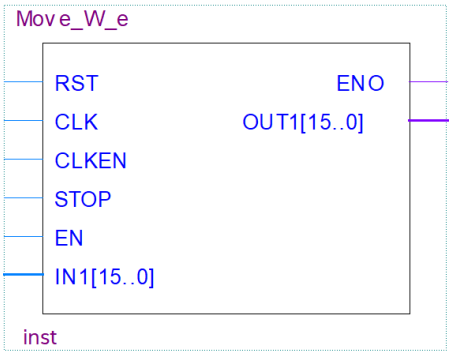
The value specified in the IN1 input is copied to the address specified at the OUT1 output. ENO always has the same signal state as EN.

Table 16: TM FAST Move_B:

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

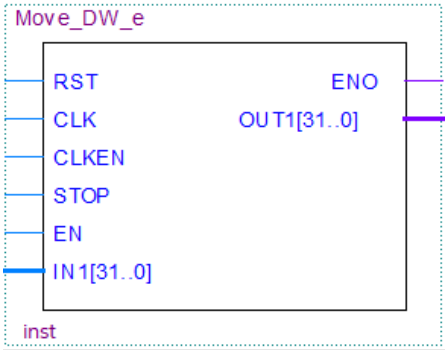
```
entity Move_B_e is
    port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR ( 7 downto 0 );
        ENO   : out STD_LOGIC;
        OUT1  : out STD_LOGIC_VECTOR ( 7 downto 0 )
    );
end Move_B_e;
```

Table 17: TM FAST Move_W

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

```
entity Move_W_e is
    port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0 );
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0 )
    );
end Move_W_e;
```

Table 18: TM FAST Move_DW

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

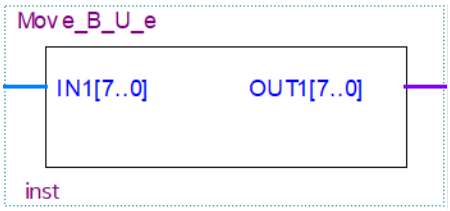
```
entity Move_DW_e is
    port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0 );
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0 )
    );
end Move_DW_e;
```

2.16 Move unlatched (Move_B_U, Move_W_U, Move_DW_U)

Description

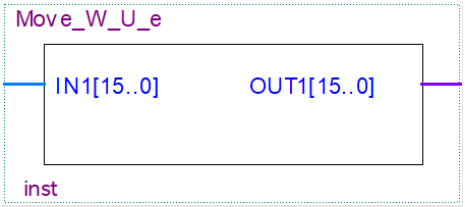
The value specified in the IN1 input is copied to the address specified at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 19: TM FAST Move_B_U unlatched:

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

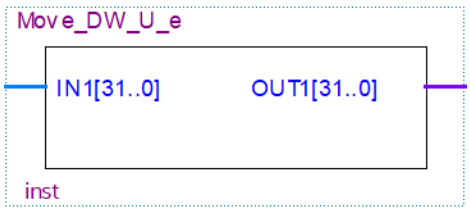
```
entity Move_B_U_e is
    port (
        IN1  : in  STD_LOGIC_VECTOR ( 7 downto 0 );
        OUT1 : out STD_LOGIC_VECTOR ( 7 downto 0 )
    );
end Move_B_U_e;
```

Table 20: TM FAST Move_W_U unlatched

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

```
entity Move_W_U_e is
    port (
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0 );
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0 )
    );
end Move_W_U_e;
```

Table 21: TM FAST Move_DW_U unlatched

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

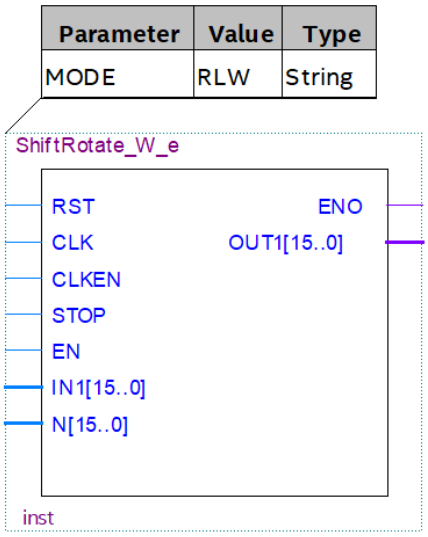
```
entity Move_DW_U_e is
    port (
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0 );
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0 )
    );
end Move_DW_U_e;
```

2.17 Rotate left word (ShiftRotate_W)

Description

The rotate left double word operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to rotate the entire contents of input IN1 bit by bit to the left. Input N specifies the number of bit positions for the rotation. If N is greater than 16, the word IN1 is rotated by (N modulo 16) bit positions. The bit positions coming from the right are occupied with the signal state of the bits which have been rotated to the left (left rotation). The result of the rotation operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 22:TM FAST rotate left word (ShiftRotate_W)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“RLW”**: rotate left
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	N	STD_LOGIC_ VECTOR	Input	Number of bit positions to be rotated.
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

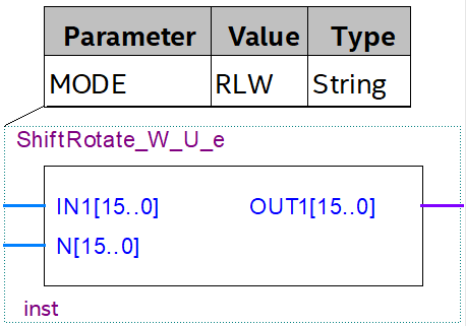
```
entity ShiftRotate_W_e is
    generic (
        MODE      : string := "RLW"
    );
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N     : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO   : out STD_LOGIC;
        OUT1  : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_e;
```

2.18 Rotate left word, unlatched logic (ShiftRotate_W_U)

Description

The operation is used to rotate the entire contents of input IN1 bit by bit to the left. Input N specifies the number of bit positions for the rotation. If N is greater than 16, the word IN1 is rotated by (N modulo 16) bit positions. The bit positions coming from the right are occupied with the signal state of the bits which have been rotated to the left (left rotation). The result of the rotation operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 23:TM FAST rotate left word, unlatched logic (ShiftRotate_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“RLW”**: rotate left
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	N	STD_LOGIC_ VECTOR	Input	Number of bit positions to be rotated.
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

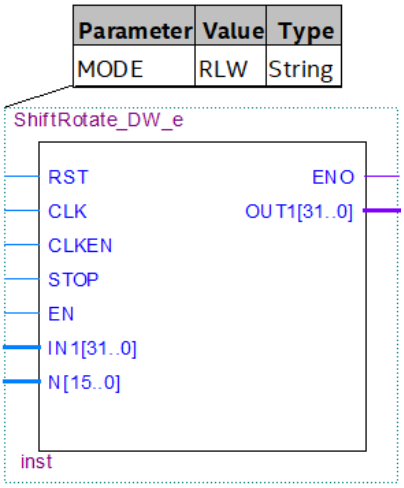
```
entity ShiftRotate_W_U_e is
    generic (
        MODE      : string := "RLW"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_U_e;
```

2.19 Rotate left double word (ShiftRotate_DW)

Description

The rotate left double word operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to rotate the entire contents of input IN1 bit by bit to the left. Input N specifies the number of bit positions for the rotation. If N is greater than 32, the double word IN1 is rotated by (N modulo 32) bit positions. The bit positions coming from the right are occupied with the signal state of the bits which have been rotated to the left (left rotation). The result of the rotation operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 24:TM FAST rotate left double word (ShiftRotate_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“RLW”**: rotate left
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	N	STD_LOGIC_ VECTOR	Input	Number of bit positions to be rotated.
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

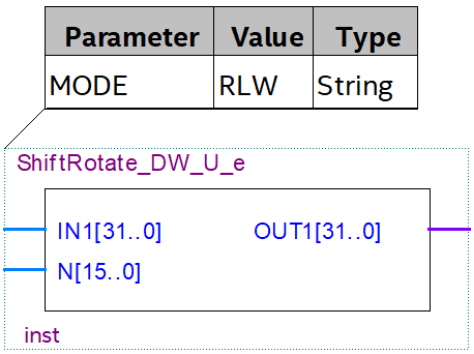
```
entity ShiftRotate_DW_e is
    generic (
        MODE : string := "RLW"
    );
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end ShiftRotate_DW_e;
```

2.20 Rotate left double word, unlatched logic (ShiftRotate_DW_U)

Description

The operation is used to rotate the entire contents of input IN1 bit by bit to the left. Input N specifies the number of bit positions for the rotation. If N is greater than 32, the double word IN1 is rotated by (N modulo 32) bit positions. The bit positions coming from the right are occupied with the signal state of the bits which have been rotated to the left (left rotation). The result of the rotation operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 25:TM FAST rotate left double word, unlatched logic (ShiftRotate_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“RLW”**: rotate left
	IN1	STD_LOGIC_ VECTOR	Input	Source value
	N	STD_LOGIC_ VECTOR	Input	Number of bit positions to be rotated.
	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.

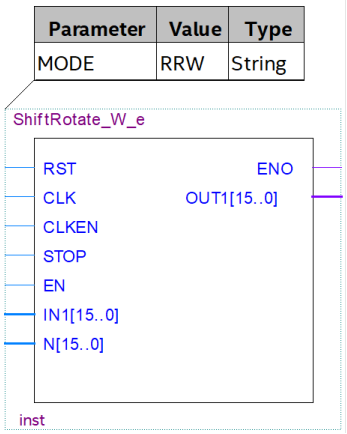
```
entity ShiftRotate_DW_U_e is
    generic (
        MODE : string := "RLW"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end ShiftRotate_DW_U_e;
```

2.21 Rotate right word (ShiftRotate_W)

Description

The rotate right double word operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to rotate the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the rotation. If N is greater than 16, the word IN1 is rotated by (N modulo 16) bit positions. The bit positions coming from the left are occupied with the signal state of the bits which have been rotated to the right (right rotation). The result of the rotation operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 26: TM FAST rotate right word (ShiftRotate_W)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“RRW”**: rotate right
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	DWORD	Input	Source value
	N	WORD	Input	Number of bit positions to be rotated.
	OUT1	DWORD	Output	Destination address of the value specified at the IN1 input.

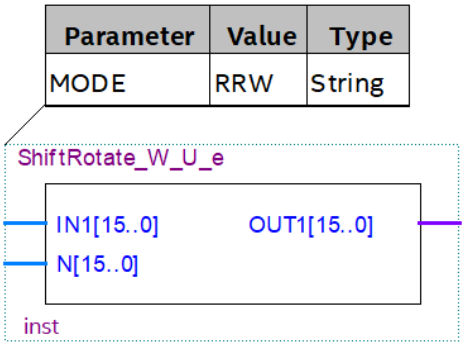
```
entity ShiftRotate_W_e is
    generic (
        MODE      : string := "RRW"
    );
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N     : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO   : out STD_LOGIC;
        OUT1  : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_e;
```

2.22 Rotate right word, unlatched logic (ShiftRotate_W_U)

Description

The operation is used to rotate the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the rotation. If N is greater than 16, the word IN1 is rotated by (N modulo 16) bit positions. The bit positions coming from the left are occupied with the signal state of the bits which have been rotated to the right (right rotation). The result of the rotation operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 27: TM FAST rotate right word, unlatched logic (ShiftRotate_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“RRW”**: rotate right
	IN1	DWORD	Input	Source value
	N	WORD	Input	Number of bit positions to be rotated.
	OUT1	DWORD	Output	Destination address of the value specified at the IN1 input.

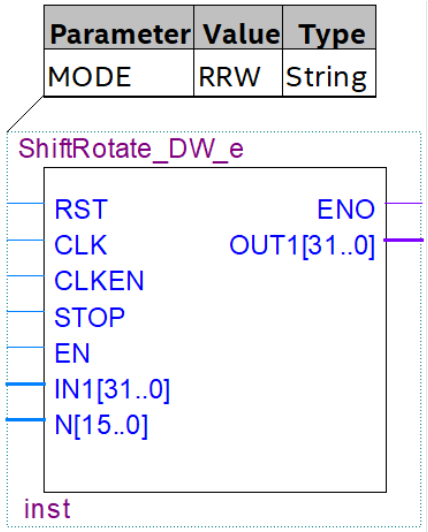
```
entity ShiftRotate_W_U_e is
    generic (
        MODE      : string := "RRW"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_U_e;
```

2.23 Rotate right double word (ShiftRotate_DW)

Description

The rotate right double word operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to rotate the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the rotation. If N is greater than 32, the double word IN1 is rotated by (N modulo 32) bit positions. The bit positions coming from the left are occupied with the signal state of the bits which have been rotated to the right (right rotation). The result of the rotation operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 28: TM FAST rotate right double word (ShiftRotate_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“RRW”**: rotate right
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	DWORD	Input	Source value
	N	WORD	Input	Number of bit positions to be rotated.
	OUT1	DWORD	Output	Destination address of the value specified at the IN1 input.

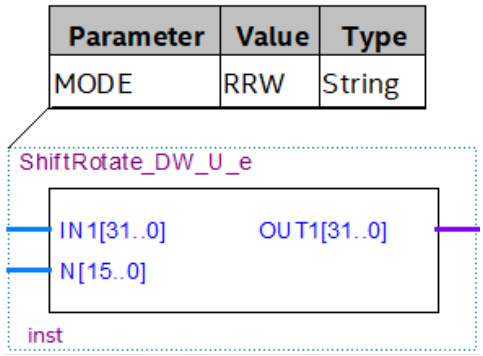
```
entity ShiftRotate_DW_e is
    generic (
        MODE : string := "RRW"
    );
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end ShiftRotate_DW_e;
```

2.24 Rotate right double word, unlatched logic (ShiftRotate_DW_U)

Description

The operation is used to rotate the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the rotation. If N is greater than 32, the double word IN1 is rotated by (N modulo 32) bit positions. The bit positions coming from the left are occupied with the signal state of the bits which have been rotated to the right (right rotation). The result of the rotation operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 29: TM FAST rotate right double word, unlatched logic (ShiftRotate_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“RRW”**: rotate right
	IN1	DWORD	Input	Source value
	N	WORD	Input	Number of bit positions to be rotated.
	OUT1	DWORD	Output	Destination address of the value specified at the IN1 input.

```
entity ShiftRotate_DW_U_e is
    generic (
        MODE : string := "RRW"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end ShiftRotate_DW_U_e;
```

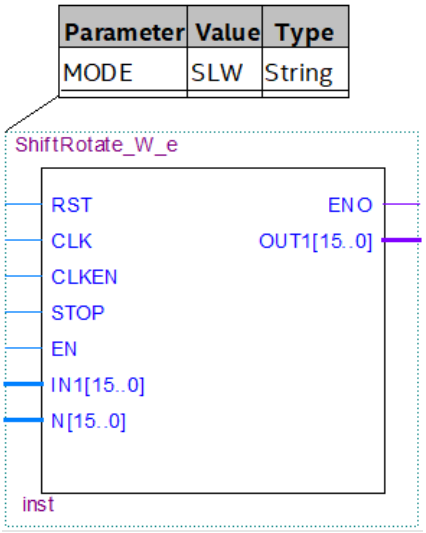
2.25 Shift left word (ShiftRotate_W)

Description

The shift left word operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to shift the entire contents of input IN1 bit by bit to the left. Input N specifies the number of bit positions for the shift. If N is greater than 16, result is 0. The bit positions coming from the right are 0. The result of the shift operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 30: TM FAST shift left word (ShiftRotate_W)

Quartus Schematic Block Symbol



Parameter	Data type	Direction	Description
MODE	STRING	Generic	Defines Mode: **“SLW”**: shift left
EN	STD_LOGIC	Input	Enable input
ENO	STD_LOGIC	Output	Enable output
IN1	WORD	Input	Source value
N	WORD	Input	Number of bit positions to be shifted.
OUT1	WORD	Output	Destination address of the value specified at the IN1 input.

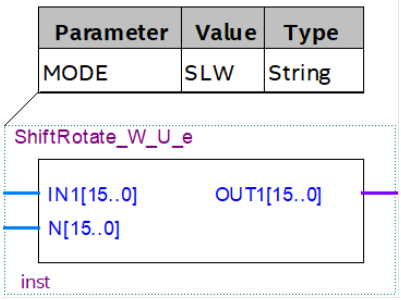
```
entity ShiftRotate_W_e is
    generic (
        MODE : string := "SLW"
    );
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N     : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO   : out STD_LOGIC;
        OUT1  : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_e;
```

2.26 Shift left word, unlatched logic (ShiftRotate_W_U)

Description

It is used to shift the entire contents of input IN1 bit by bit to the left. Input N specifies the number of bit positions for the shift. If N is greater than 16, result is 0. The bit positions coming from the right are 0. The result of the shift operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 31: TM FAST shift left word, unlatched logic (ShiftRotate_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“SLW”**: shift left
	IN1	WORD	Input	Source value
	N	WORD	Input	Number of bit positions to be shifted.
	OUT1	WORD	Output	Destination address of the value specified at the IN1 input.

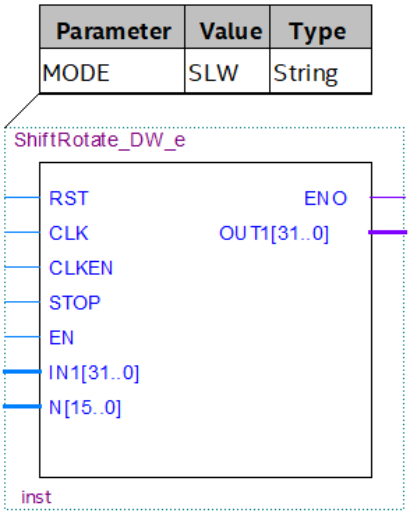
```
entity ShiftRotate_W_U_e is
    generic (
        MODE      : string := "SLW"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_U_e;
```

2.27 Shift left double word (ShiftRotate_DW)

Description

The shift left double word operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to shift the entire contents of input IN1 bit by bit to the left. Input N specifies the number of bit positions for the shift. If N is greater than 32, result is 0. The bit positions coming from the right are 0. The result of the shift operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 32: TM FAST shift left double word (ShiftRotate_DW)

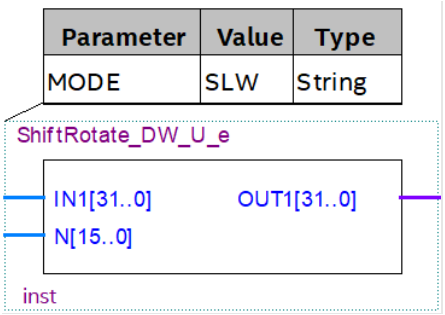
Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“SLW”**: shift left
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	DWORD	Input	Source value
	N	WORD	Input	Number of bit positions to be shifted.
	OUT1	DWORD	Output	Destination address of the value specified at the IN1 input.

2.28 Shift left double word, unlatched logic (ShiftRotate_DW_U)

Description

It is used to shift the entire contents of input IN1 bit by bit to the left. Input N specifies the number of bit positions for the shift. If N is greater than 32, result is 0. The bit positions coming from the right are 0. The result of the shift operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 33: TM FAST shift left double word, unlatched logic (ShiftRotate_DW_U)

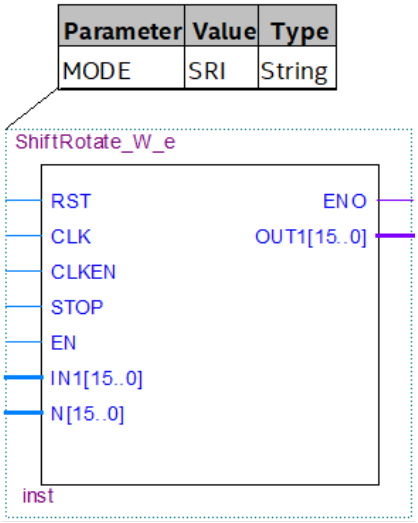
<table>
<tr><th>Quartus Schematic Block Symbol</th><th>Parameter</th><th>Data type</th><th>Direction</th><th>Description</th></tr>
<tr><td rowspan="4"> 
</td><td>MODE</td><td>STRING</td><td>Generic</td><td>Defines Mode: “SLW”: shift left</td></tr>
<tr><td>IN1</td><td>DWORD</td><td>Input</td><td>Source value</td></tr>
<tr><td>N</td><td>WORD</td><td>Input</td><td>Number of bit positions to be shifted.</td></tr>
<tr><td>OUT1</td><td>DWORD</td><td>Output</td><td>Destination address of the value specified at the IN1 input.</td></tr>
<tr><td colspan="5"><pre>
entity ShiftRotate_DW_U_e is
    generic (
        MODE : string := "SLW"
    );
    Port (
        IN1 : in STD_LOGIC_VECTOR (31 downto 0);
        N   : in STD_LOGIC_VECTOR (15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end ShiftRotate_DW_U_e;
</pre></td></tr>
</table>

2.29 Shift right integer (ShiftRotate_W)

Description

The shift right integer operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to shift the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the shift. If N is greater than 16, the command operates as if N = 16 were set. The bit positions coming from the left are filled by the sign bit at bit position 15 of IN1. The result of the shift operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 34: TM FAST shift right integer (ShiftRotate_W)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“SRI”**: shift right
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	INT	Input	Source value
	N	WORD	Input	Number of bit positions to be shifted.
	OUT1	INT	Output	Destination address of the value specified at the IN1 input.

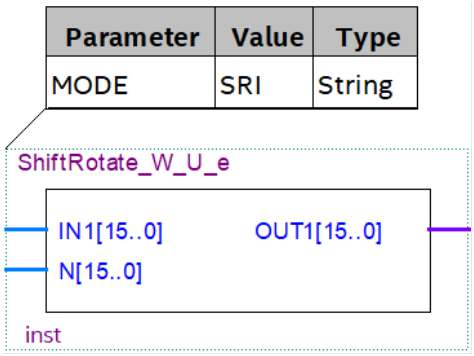
```
entity ShiftRotate_W_e is
    generic (
        MODE     : string := "SRI"
    );
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_e;
```

2.30 Shift right integer, unlatched logic (ShiftRotate_W_U)

Description

It is used to shift the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the shift. If N is greater than 16, the command operates as if N = 16 were set. The bit positions coming from the left are filled by the sign bit at bit position 15 of IN1. The result of the shift operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 35: TM FAST shift right integer, unlatched logic (ShiftRotate_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“SRI”**: shift right
	IN1	INT	Input	Source value
	N	WORD	Input	Number of bit positions to be shifted.
	OUT1	INT	Output	Destination address of the value specified at the IN1 input.

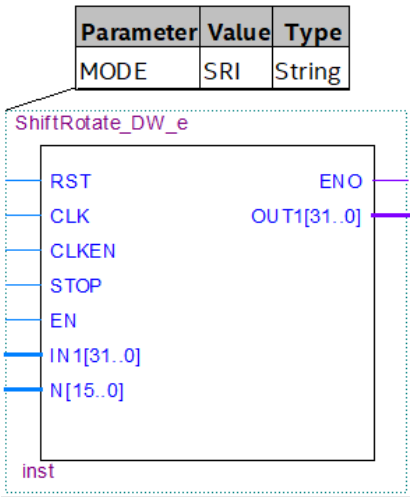
```
entity ShiftRotate_W_U_e is
    generic (
        MODE      : string := "SRI"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_U_e;
```

2.31 Shift right double integer (ShiftRotate_DW)

Description

The shift right double integer operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to shift the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the shift. If N is greater than 32, the command operates as if N = 32 were set. The bit positions coming from the left are filled by the sign bit at bit position 31 of IN1. The result of the shift operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 36: TM FAST shift right double integer (ShiftRotate_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“SRI”**: shift right
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	DINT	Input	Source value
	N	WORD	Input	Number of bit positions to be shifted.
	OUT1	DINT	Output	Destination address of the value specified at the IN1 input.

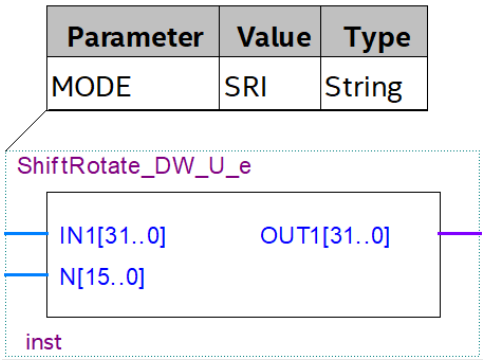
```
entity ShiftRotate_DW_e is
    generic (
        MODE : string := "SRI"
    );
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end ShiftRotate_DW_e;
```

2.32 Shift right double integer, unlatched logic (ShiftRotate_DW_U)

Description

The operation is used to shift the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the shift. If N is greater than 32, the command operates as if N = 32 were set. The bit positions coming from the left are filled by the sign bit at bit position 31 of IN1. The result of the shift operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 37: TM FAST shift right double integer, unlatched logic (ShiftRotate_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“SRI”**: shift right
	IN1	DINT	Input	Source value
	N	WORD	Input	Number of bit positions to be shifted.
	OUT1	DINT	Output	Destination address of the value specified at the IN1 input.

```
entity ShiftRotate_DW_U_e is
    generic (
        MODE : string := "SRI"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end ShiftRotate_DW_U_e;
```

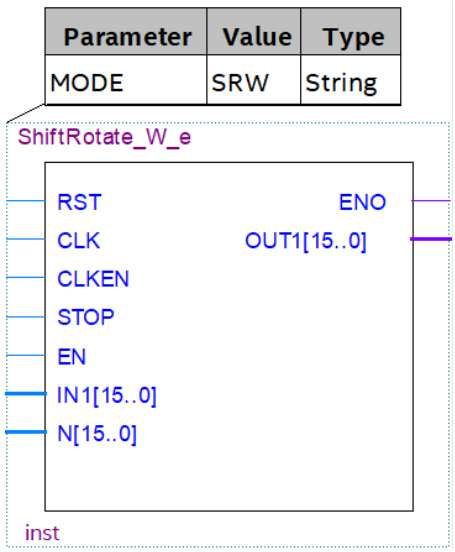
2.33 Shift right word (ShiftRotate_W)

Description

The shift right word operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to shift the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the shift. If N is greater than 16, result is 0. The bit positions coming from the left are 0. The result of the shift operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 38: TM FAST shift right word (ShiftRotate_W)

Quartus Schematic Block Symbol



Parameter	Data type	Direction	Description
MODE	STRING	Generic	Defines Mode: **“SRW”**: shift right
EN	STD_LOGIC	Input	Enable input
ENO	STD_LOGIC	Output	Enable output
IN1	WORD	Input	Source value
N	WORD	Input	Number of bit positions to be shifted.
OUT1	WORD	Output	Destination address of the value specified at the IN1 input.

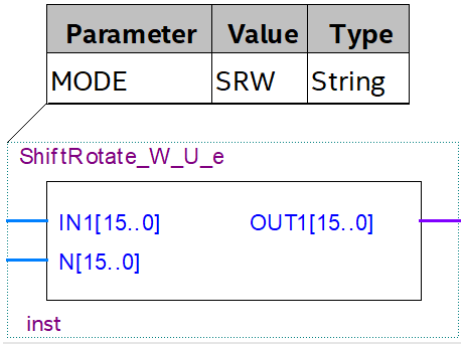
```
entity ShiftRotate_W_e is
    generic (
        MODE     : string := "SRW"
    );
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_e;
```

2.34 Shift right word, unlatched logic (ShiftRotate_W_U)

Description

It is used to shift the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the shift. If N is greater than 16, result is 0. The bit positions coming from the left are 0. The result of the shift operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 39: TM FAST shift right word, unlatched logic (ShiftRotate_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“SRW”**: shift right
	IN1	WORD	Input	Source value
	N	WORD	Input	Number of bit positions to be shifted.
	OUT1	WORD	Output	Destination address of the value specified at the IN1 input.

```
entity ShiftRotate_W_U_e is
    generic (
        MODE     : string := "SRW"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end ShiftRotate_W_U_e;
```

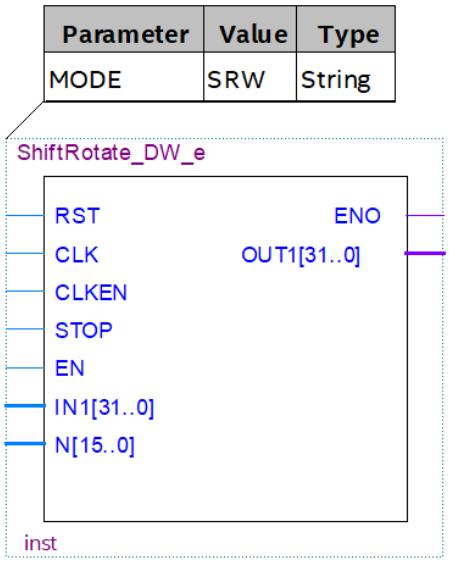
2.35 Shift right double word (ShiftRotate_DW)

Description

The shift right word operation is enabled by signal state “1” at the Enable (EN) input. The operation is used to shift the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the shift. If N is greater than 16, result is 0. The bit positions coming from the left are 0. The result of the shift operation can be queried at the OUT1 output. ENO has the same signal state as EN.

Table 40: TM FAST shift right double word (ShiftRotate_DW)

Quartus Schematic Block Symbol



Parameter	Data type	Direction	Description
MODE	STRING	Generic	Defines Mode: **“SRW”**: shift right
EN	STD_LOGIC	Input	Enable input
ENO	STD_LOGIC	Output	Enable output
IN1	WORD	Input	Source value
N	WORD	Input	Number of bit positions to be shifted.
OUT1	WORD	Output	Destination address of the value specified at the IN1 input.

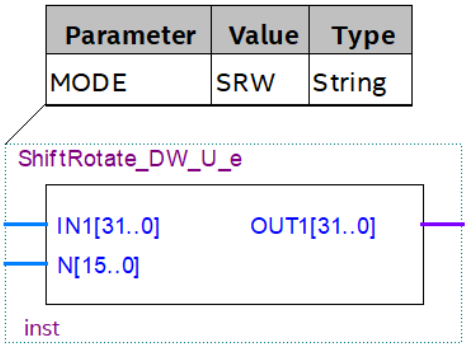
```
entity ShiftRotate_DW_e is
    generic (
        MODE : string := "SLW"
    );
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end ShiftRotate_DW_e;
```

2.36 Shift right double word, unlatched logic (ShiftRotate_DW_U)

Description

It is used to shift the entire contents of input IN1 bit by bit to the right. Input N specifies the number of bit positions for the shift. If N is greater than 16, result is 0. The bit positions coming from the left are 0. The result of the shift operation can be queried at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 41: TM FAST shift right double word, unlatched logic (ShiftRotate_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“SRW”**: shift right
	IN1	WORD	Input	Source value
	N	WORD	Input	Number of bit positions to be shifted.
	OUT1	WORD	Output	Destination address of the value specified at the IN1 input.

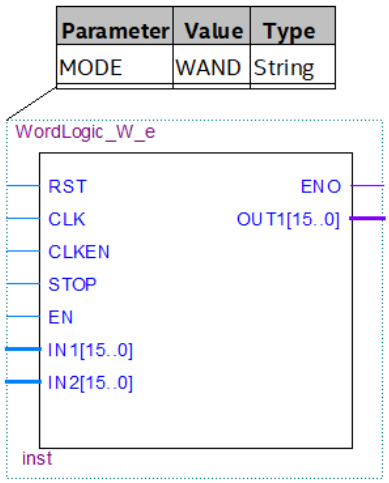
```
entity ShiftRotate_DW_U_e is
    generic (
        MODE : string := "SRW"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        N    : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end ShiftRotate_DW_U_e;
```

2.37 Word AND word (WAND_W)

Description

The word AND word operation is enabled by signal state “1” at the Enable (EN) input and ANDs the two word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. ENO has the same signal state as EN.

Table 42: TM FAST word AND word (WordLogic_W)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WAND”**: AND operation
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	WORD	Input	First value of logic operation
	IN2	WORD	Input	Second value of logic operation
	OUT1	WORD	Output	Result word of the logic operation

```
entity WordLogic_W_e is
    generic (
        MODE : string := "WAND"
    );
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC := '1';
        STOP : in  STD_LOGIC := '0';
        EN    : in  STD_LOGIC := '1';
        IN1   : in  STD_LOGIC_VECTOR ( 15 downto 0);
        IN2   : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO   : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end WordLogic_W_e;
```

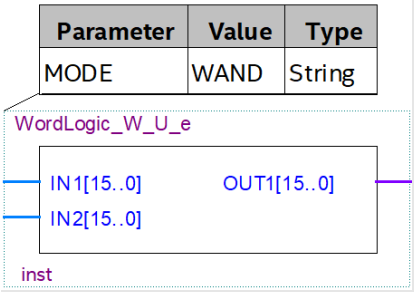
2.38 Word AND word, unlatched logic (WAND_W_U)

Description

The word AND word operation ANDs the two word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output.

The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 43: TM FAST word AND word, unlatched logic (WordLogic_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WAND”**: AND operation
	IN1	WORD	Input	First value of logic operation
	IN2	WORD	Input	Second value of logic operation
	OUT1	WORD	Output	Result word of the logic operation

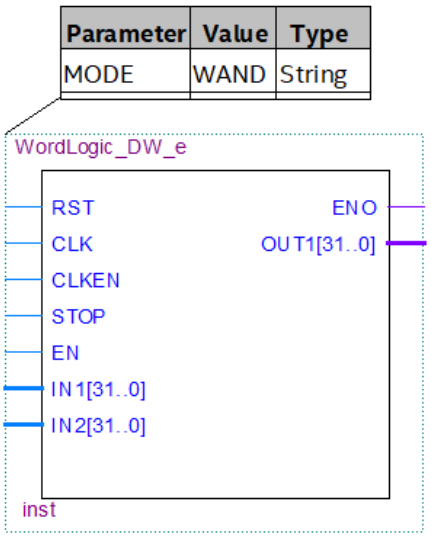
```
entity WordLogic_W_U_e is
    generic (
        MODE      : string := "WAND"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end WordLogic_W_U_e;
```

2.39 Word AND double word (WordLogic_DW)

Description

The word AND double word operation is enabled by signal state “1” at the Enable (EN) input and ANDs the two double word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. ENO has the same signal state as EN.

Table 44: TM FAST word AND double word (WordLogic_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WAND”**: AND operation
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	DWORD	Input	First value of logic operation
	IN2	DWORD	Input	Second value of logic operation
	OUT1	DWORD	Output	Result double word of the logic operation

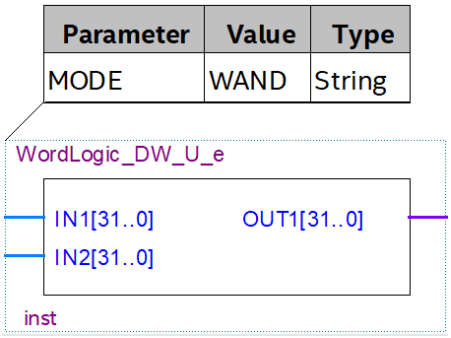
```
entity WordLogic_DW_e is
    generic (
        MODE : string := "WAND"
    );
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC := '1';
        STOP : in  STD_LOGIC := '0';
        EN    : in  STD_LOGIC := '1';
        IN1   : in  STD_LOGIC_VECTOR ( 31 downto 0);
        IN2   : in  STD_LOGIC_VECTOR ( 31 downto 0);
        ENO   : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end WordLogic_DW_e;
```

2.40 Word AND double word, unlatched logic (WordLogic_DW_U)

Description

The word AND double word operation ANDs the two double word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 45: TM FAST word AND double word (WordLogic_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WAND”**: AND operation
	IN1	DWORD	Input	First value of logic operation
	IN2	DWORD	Input	Second value of logic operation
	OUT1	DWORD	Output	Result double word of the logic operation

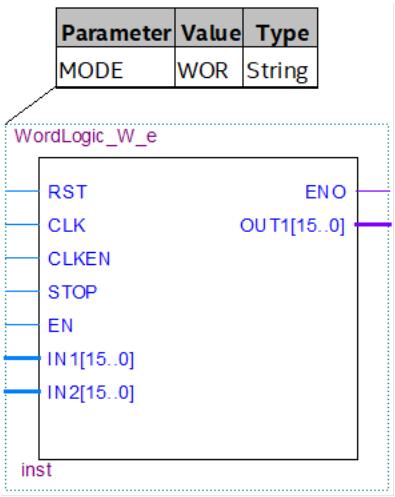
```
entity WordLogic_DW_U_e is
    generic (
        MODE      : string := "WAND"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        IN2  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end WordLogic_DW_U_e;
```

2.41 Word OR word (WordLogic_W)

Description

The word OR word operation is enabled by signal state “1” at the Enable (EN) input and logical ORs the two word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns, a binary OR is executed. The result can be scanned at the OUT1 output. ENO has the same signal state as EN.

Table 46: TM FAST word OR word (WordLogic_W)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WOR”**: OR operation
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	WORD	Input	First value of logic operation
	IN2	WORD	Input	Second value of logic operation
	OUT1	WORD	Output	Result word of the logic operation

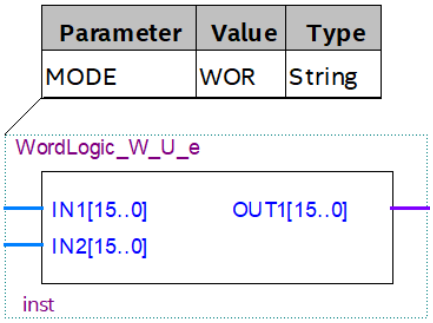
```
entity WordLogic_W_e is
    generic (
        MODE : string := "WAND"
    );
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC := '1';
        STOP : in  STD_LOGIC := '0';
        EN   : in  STD_LOGIC := '1';
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end WordLogic_W_e;
```

2.42 Word OR word, unlatched logic (WordLogic_W_U)

Description

The word OR word operation ORs the two word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 47: TM FAST word OR word, unlatched logic (WordLogic_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WOR”**: OR operation
	IN1	WORD	Input	First value of logic operation
	IN2	WORD	Input	Second value of logic operation
	OUT1	WORD	Output	Result word of the logic operation

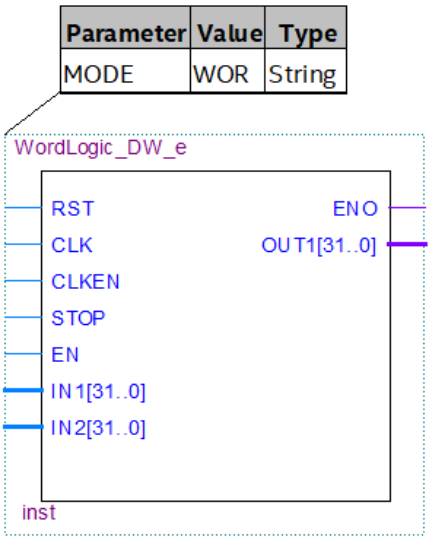
```
entity WordLogic_W_U_e is
    generic (
        MODE      : string := "WOR"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end WordLogic_W_U_e;
```

2.43 Word OR double word (WordLogic_DW)

Description

The word OR double word operation is enabled by signal state “1” at the Enable (EN) input and ORs the two double word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. ENO has the same signal state as EN.

Table 48: TM FAST word OR double word (WordLogic_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WOR”**: OR operation
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	DWORD	Input	First value of logic operation
	IN2	DWORD	Input	Second value of logic operation
	OUT1	DWORD	Output	Result double word of the logic operation

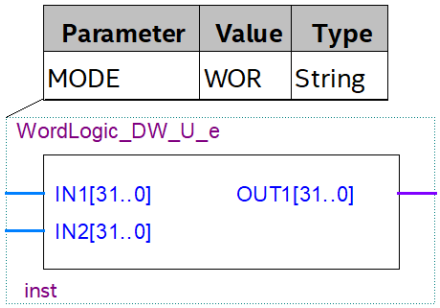
```
entity WordLogic_DW_e is
    generic (
        MODE : string := "WAND"
    );
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC := '1';
        STOP : in  STD_LOGIC := '0';
        EN    : in  STD_LOGIC := '1';
        IN1   : in  STD_LOGIC_VECTOR ( 31 downto 0);
        IN2   : in  STD_LOGIC_VECTOR ( 31 downto 0);
        ENO   : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end WordLogic_DW_e;
```

2.44 Word OR double word, unlatched logic (WordLogic_DW_U)

Description

The word OR double word operation ORs the two double word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 49: TM FAST word OR double word, unlatched logic (WordLogic_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WOR”**: OR operation
	IN1	DWORD	Input	First value of logic operation
	IN2	DWORD	Input	Second value of logic operation
	OUT1	DWORD	Output	Result double word of the logic operation

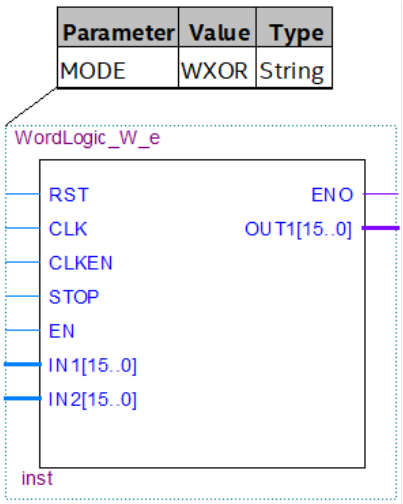
```
entity WordLogic_DW_U_e is
    generic (
        MODE      : string := "WOR"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        IN2  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end WordLogic_DW_U_e;
```

2.45 Word XOR word (WordLogic_W)

Description

The word XOR word operation is enabled by signal state “1” at the Enable (EN) input and XORs the two word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. ENO has the same signal state as EN.

Table 50: TM FAST word XOR word (WordLogic_W)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WXOR”**: XOR operation
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	WORD	Input	First value of logic operation
	IN2	WORD	Input	Second value of logic operation
	OUT1	WORD	Output	Result word of the logic operation

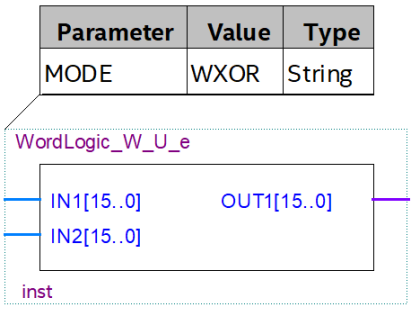
```
entity WordLogic_W_e is
    generic (
        MODE : string := "WAND"
    );
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC := '1';
        STOP : in  STD_LOGIC := '0';
        EN   : in  STD_LOGIC := '1';
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end WordLogic_W_e;
```

2.46 Word XOR word, unlatched logic (WordLogic_W_U)

Description

The word XOR word operation XORs the two word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 51: TM FAST word XOR word, unlatched logic (WordLogic_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WXOR”**: XOR operation
	IN1	WORD	Input	First value of logic operation
	IN2	WORD	Input	Second value of logic operation
	OUT1	WORD	Output	Result word of the logic operation

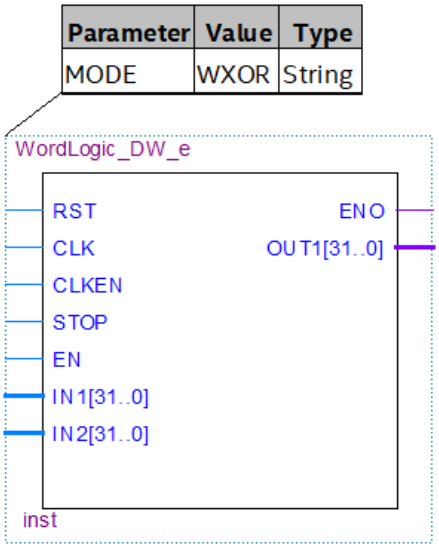
```
entity WordLogic_W_U_e is
    generic (
        MODE      : string := "WXOR"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR ( 15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)
    );
end WordLogic_W_U_e;
```

2.47 Word XOR double word (WordLogic_DW)

Description

The word XOR double word operation is enabled by signal state “1” at the Enable (EN) input and XORs the two double word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. ENO has the same signal state as EN.

Table 52: TM FAST word XOR double word (WordLogic_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WXOR”**: XOR operation
	EN	STD_LOGIC	Input	Enable input
	ENO	STD_LOGIC	Output	Enable output
	IN1	DWORD	Input	First value of logic operation
	IN2	DWORD	Input	Second value of logic operation
	OUT1	DWORD	Output	Result double word of the logic operation

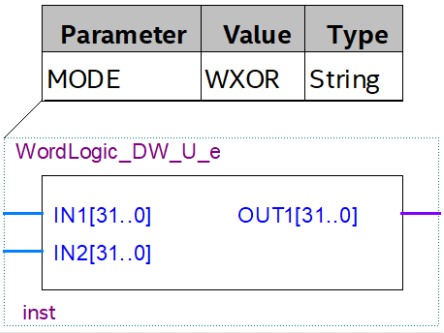
```
entity WordLogic_DW_e is
    generic (
        MODE : string := "WAND"
    );
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC := '1';
        STOP : in  STD_LOGIC := '0';
        EN   : in  STD_LOGIC := '1';
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        IN2  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        ENO  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end WordLogic_DW_e;
```

2.48 Word XOR double word, latched logic (WordLogic_DW_U)

Description

The word XOR double word XORs the two double word values at IN1 and IN2 bit by bit. The values are interpreted as pure bit patterns. The result can be scanned at the OUT1 output. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 53: TM FAST word XOR double word, latched logic (WordLogic_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	MODE	STRING	Generic	Defines Mode: **“WXOR”**: XOR operation
	IN1	DWORD	Input	First value of logic operation
	IN2	DWORD	Input	Second value of logic operation
	OUT1	DWORD	Output	Result double word of the logic operation

```
entity WordLogic_DW_U_e is
    generic (
        MODE      : string := "WXOR"
    );
    Port (
        IN1  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        IN2  : in  STD_LOGIC_VECTOR ( 31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR ( 31 downto 0)
    );
end WordLogic_DW_U_e;
```

3 TM FAST library: Operation function blocks (FBs)

The following table lists the symbolic names and description for each TM FAST function block library component.

Table 54: TM FAST function blocks (FBs)

FB Number	TM FAST Symbolic Name	FM 352-5 Symbolic Name	Description
FB 76	FB76_WordPack	WORDPACK	Concatenates 2 WORDs into 1 DWORD
	FB76_WordPack_U		Concatenates 2 WORDs into 1 DWORD, unlatched logic
FB 77	FB77_WordCast	WORDCAST	Converts 1DWORD into 2 WORDs
	FB77_WordCast_U		Converts 1DWORD into 2 WORDs, unlatched logic
FB 78	FB78_BitSum	BITSUM	Counts the bits set in a DWORD
	FB78_BitSum_U		Counts the bits set in a DWORD, unlatched logic
FB 79	FB79_Encode	ENCODE	Locates the most significant bit set in a DWORD
	FB79_Encode_U		Locates the most significant bit set in a DWORD, unlatched logic
FB 80	FB80_Period32	PERIOD32	Period measurement (32-bits)
FB 81	FB81_Period16	PERIOD16	Period measurement (16-bits)
FB 82	FB82_Freq32	FREQ32	Frequency measurement (32-bits)
FB 83	FB83_Freq16	FREQ16	Frequency measurement (16-bits)
FB 84	FB84_Shift32	SHIFT32	DINT shift register, max length 256.
FB 85	FB85_Shift16	SHIFT16	INT shift register, max length 256.
FB 86	FB86_BitPick_DW	BITPICK_DW	Selects a bit from a DWORD
	FB86_BitPick_DW_U		Selects a bit from a DWORD, unlatched logic
FB 87	FB87_BitPick_W	BITPICK_W	Selects a bit from a WORD
	FB87_BitPick_W_U		Selects a bit from a WORD, unlatched logic
FB 88	FB88_BitShift_DW	BITSHIFT_DW	Bit shift register, length 32 bits
	FB88_BitShift_DW_U		Bit shift register, length 32 bits, unlatched logic
FB 89	FB89_BitShift_W	BITSHIFT_W	Bit shift register, length 16 bits
	FB89_BitShift_W_U		Bit shift register, length 16 bits, unlatched logic
FB 90	FB90_BitCast_DW	BITCAST_DW	Converts a DWORD into 32 digital bits
	FB90_BitCast_DW_U		Converts a DWORD into 32 digital bits, unlatched logic
FB 91	FB91_BitCast_W	BITCAST_W	Converts a WORD into 16 digital bits
	FB91_BitCast_W_U		Converts a DWORD into 16 digital bits, unlatched logic
FB 92	FB92_BitPack_DW	BITPACK_DW	Packs 32 digital bits into a DWORD
	FB92_BitPack_DW_U		Packs 32 digital bits into a DWORD, unlatched logic
FB 93	FB93_BitPack_W	BITPACK_W	Packs 16 digital bits into a WORD
	FB93_BitPack_W_U		Packs 16 digital bits into a DWORD, unlatched logic
FB 94	FB94_BitInsert32	BITINSERT32	Inserts a bit into a DINT, 32-Bits
	FB94_BitInsert32_U		Inserts a bit into a DINT, 32-Bits, unlatched logic
FB 95	FB95_BitInsert16	BITINSERT16	Inserts a bit into an INT, 16-Bits
	FB95_BitInsert16_U		Inserts a bit into an INT, 16-Bits, unlatched logic
FB 96	FB96_FIFO32	FIFO32	First-In-First-Out memory, 32-bits, length 256

FB 97	FB97_FIFO16	FIFO16	First-In-First-Out memory, 16-bits, length 256
FB 98	FB98_LIFO32	LIFO32	Last-In-First-Out memory, 32-bits, length 256
FB 99	FB99_LIFO16	LIFO16	Last-In-First-Out memory, 16-bits, length 256
FB 100	FB100_FMMul32	FMMUL32	Multiply, 32-Bits
FB 101	FB101_FMMul16	FMMUL16	Multiply, 16-Bits
FB 102	FB102_FMDiv32	FMDIV32	Divide, 32-Bits
FB 103	FB103_FMDiv16	FMDIV16	Divide, 16-Bits
FB 104	FB104_FMAbs32	FMABS32	Absolute value, 32-Bits
FB 105	FB105_FMAbs16	FMABS16	Absolute value, 16-Bits
FB 106	FB106_FMAdd32	FMADD32	Add, 32-Bits
FB 107	FB107_FMAdd16	FMADD16	Add, 16-Bits
FB 108	FB108_FMSub32	FMSUB32	Subtract, 32-Bits
FB 109	FB109_FMSub16	FMSUB16	Subtract, 16-Bits
FB 110	FB110_DatSel32	DATSEL32	Data selector, 32-Bits
FB 111	FB111_DatSel16	DATSEL16	Data selector, 16-Bits
FB 112	FB112_BiScale	BISCALE	Binary Scaler
FB 113	FB113_TP32	TP32	Timer, 32-Bit Pulse
FB 114	FB114_TOn32	TON32	Timer, 32-Bit On delay
FB 115	FB115_TOf32	TOF32	Timer, 32-Bit Off delay
FB 116	FB116_TP16	TP16	Timer, 16-Bit Pulse
FB 117	FB117_TOn16	TON16	Timer, 16-Bit On delay
FB 118	FB118_TOf16	TOF16	Timer, 16-Bit Off delay
FB 119	FB119_CP_Gen	CP_GEN	Clock pulse generator
FB 120	FB120_CTUD32	CTUD32	Counter, 32-Bit up/down
FB 121	FB121_CTU16	CTU16	Counter, 16-Bit up
FB 122	FB122_CTD16	CTD16	Counter, 16-Bit down
FB 123	FB123_CTUD16	CTUD16	Counter, 16-Bit up/down
FB 124	FB124_Shift	SHIFT	Bit shift register, 1-bit, max length 4096
FB 125	FB125_Shift2	SHIFT2	Bit shift register, 2-bit, max length 2048
FB 126	FB126_Shift4	SHIFT4	Bit shift register, 4-bit, max length 1024
FB 127	FB127_Shift8	SHIFT8	Bit shift register, 8-bit, max length 512
-	ClkTick100k	-	Generates a 100kHz clock tic signal

3.1 WordPack (FB76_WordPack)

Description

When the function is enabled (EN = ‘1’) the input WORDs are concatenated into one DWORD. IN1 is the most significant word and IN2 is the least significant word.

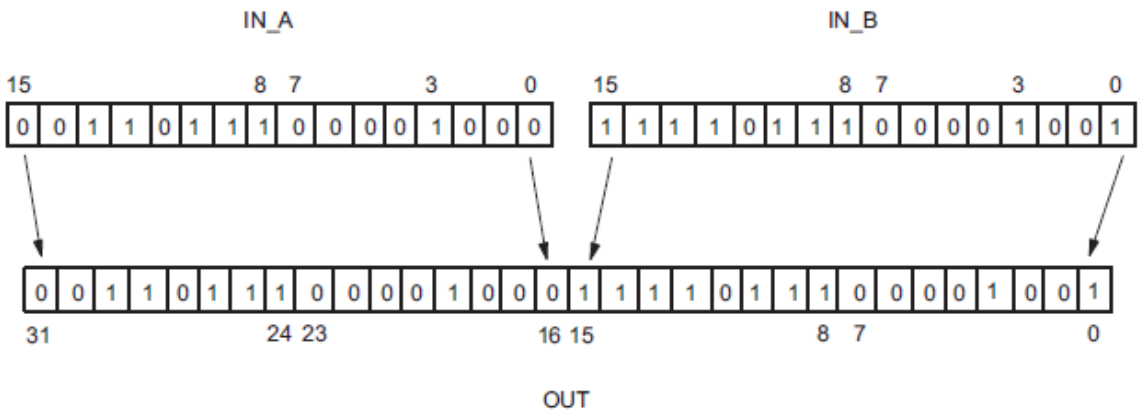
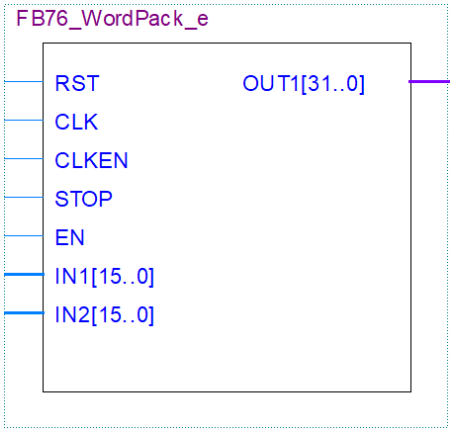


Figure 1: Example of WordPack

Table 55: TM FAST function WordPack (FB76_WordPack)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	WORD	Input	Input with the most significant word
	IN2	WORD	Input	Input with the least significant word
	OUT1	DWORD	Output	Output of function

```
entity FB76_WordPack_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB76_WordPack_e;
```

3.2 WordPack, unlatched logic (FB76_WordPack_U)

Description

When the function is enabled (EN = ‘1’) the input WORDs are concatenated into one DWORD. IN1 is the most significant word and IN2 is the least significant word. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

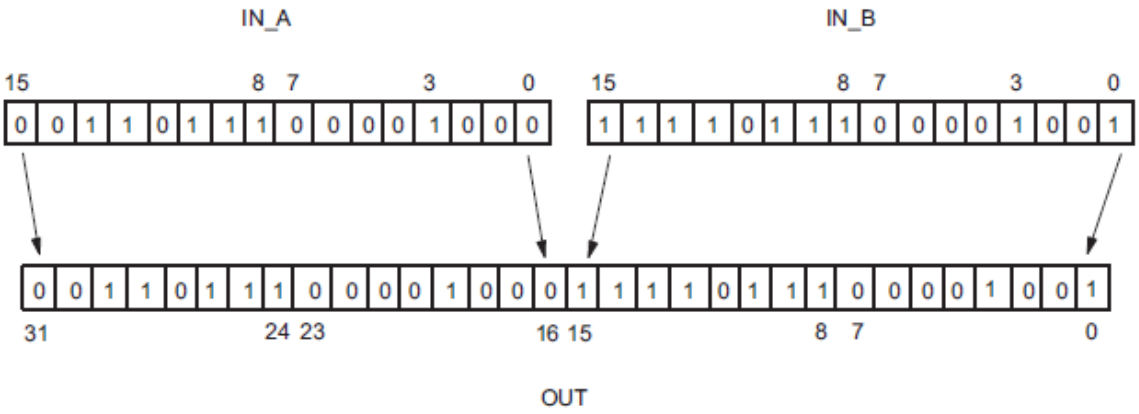
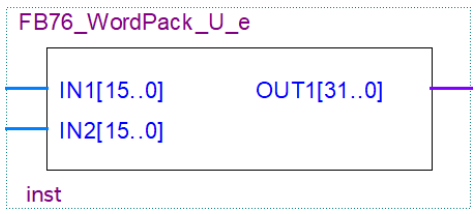


Figure 2: Example of WordPack

Table 56: TM FAST function WordPack, unlatched logic (FB76_WordPack_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	WORD	Input	Input with the most significant word
	IN2	WORD	Input	Input with the least significant word
	OUT1	DWORD	Output	Output of function

```
entity FB76_WordPack_U_e is
    Port (
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB76_WordPack_U_e;
```

3.3 WordCast (FB77_WordCast)

Description

When the function is enabled (EN = ‘1’) the input DWORD is converted into two WORDs. OUT1 is the most significant word and OUT2 is the least significant word.

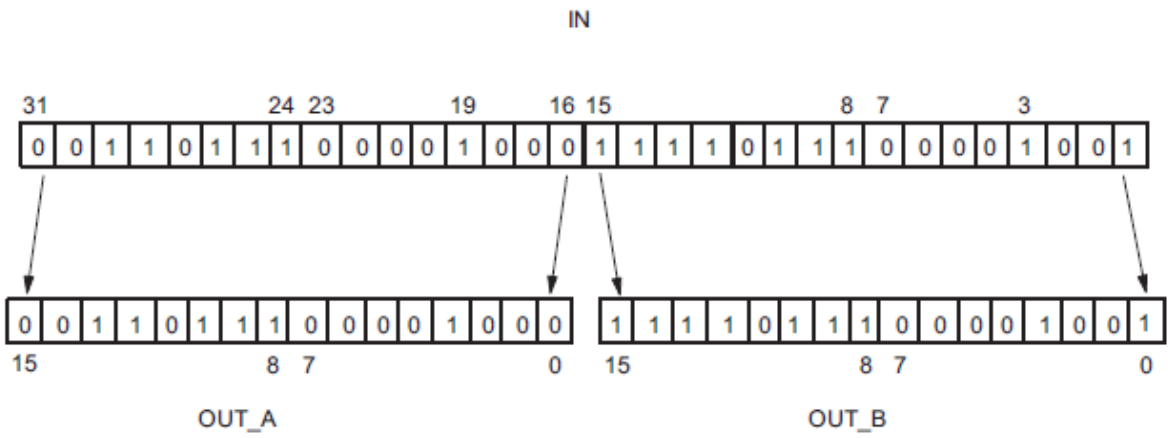
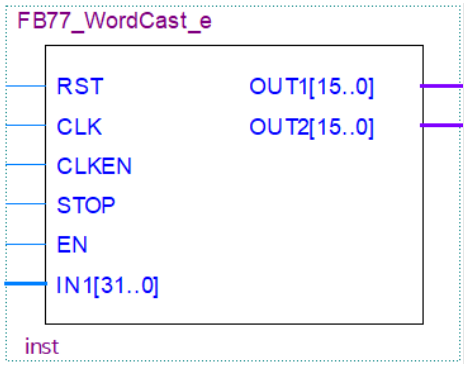


Figure 3: Example of WordCast

Table 57: TM FAST function WordCast (FB77_WordCast)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC _VECTOR	Input	Input of function
	OUT1	STD_LOGIC _VECTOR	Output	Output with the most significant word
	OUT2	STD_LOGIC _VECTOR	Output	Output with the least significant word

```
entity FB77_WordCast_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0);
        OUT2 : out STD_LOGIC_VECTOR (15 downto 0) );
end FB77_WordCast_e;
```

3.4 WordCast, unlatched logic (FB77_WordCast_U)

Description

The input DWORD is converted into two WORDs. OUT1 is the most significant word and OUT2 is the least significant word. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

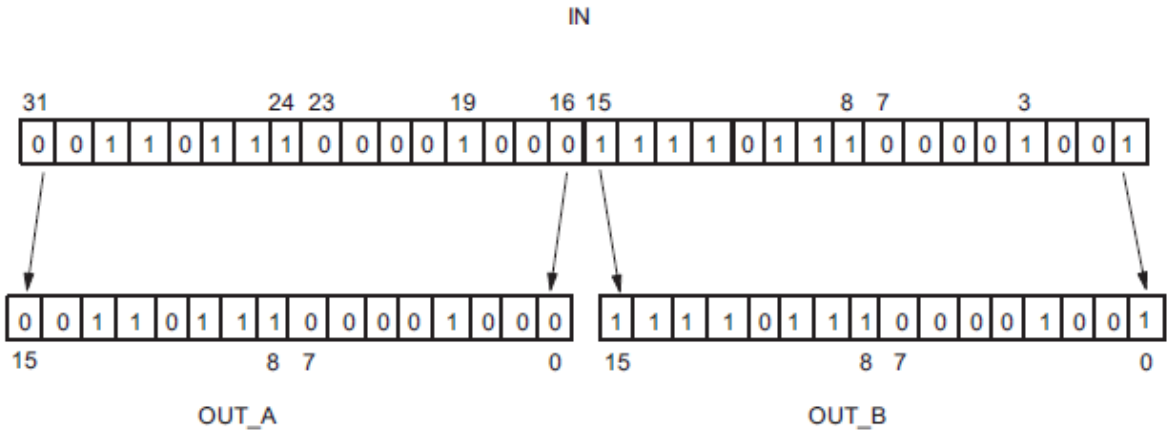
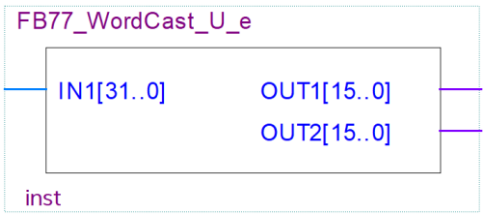


Figure 4: Example of WordCast_U

Table 58: TM FAST function WordCast, unlatched logic (FB77_WordCast_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC _VECTOR	Input	Input of function
	OUT1	STD_LOGIC _VECTOR	Output	Output with the most significant word
	OUT2	STD_LOGIC _VECTOR	Output	Output with the least significant word

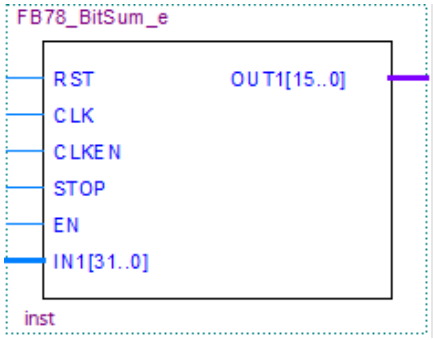
```
entity FB77_WordCast_U_e is
    Port (
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0);
        OUT2 : out STD_LOGIC_VECTOR (15 downto 0) );
end FB77_WordCast_U_e;
```

3.5 BitSum (FB78_BitSum)

Description

When the function is enabled (EN = ‘1’) the function counts the number of bits that are set to a value of 1 in the IN1 input and returns this as the function’s value.

Table 59: TM FAST function BitSum (FB78_BitSum)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC_ VECTOR	Input	Variable in which bits are counted
	OUT1	STD_LOGIC_ VECTOR	Output	Value output

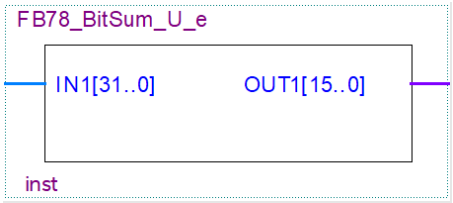
```
entity FB78_BitSum_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1  : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB78_BitSum_e;
```

3.6 BitSum, unlatched logic (FB78_BitSum_U)

Description

The function counts the number of bits that are set to a value of 1 in the IN1 input and returns this as the function's value. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

Table 60: TM FAST function BitSum, unlatched logic (FB78_BitSum_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Variable in which bits are counted
	OUT1	STD_LOGIC_ VECTOR	Output	Value output

```
entity FB78_BitSum_U_e is
    Port (
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB78_BitSum_U_e;
```

3.7 Encode (FB79_Encode)

Description

When the function is enabled (EN = ‘1’) the function converts the contents of IN1 to a binary number corresponding to the bit position of the leftmost bit set in IN1, and then returns the result as the function’s value. If IN1 is either DW#16#00000001 or DW#16#00000000, a value of 0 is returned.

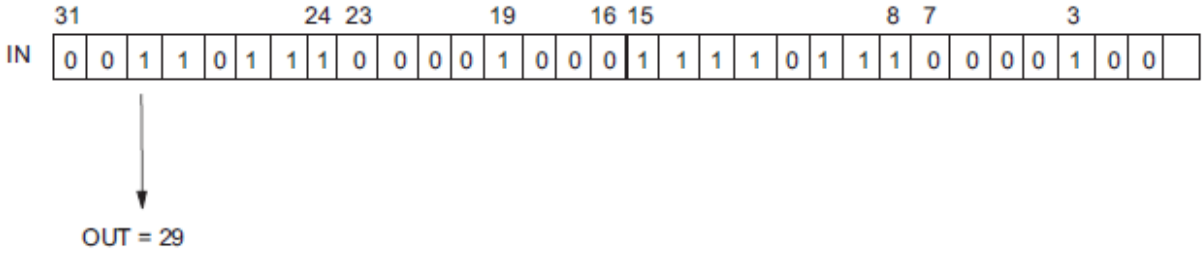
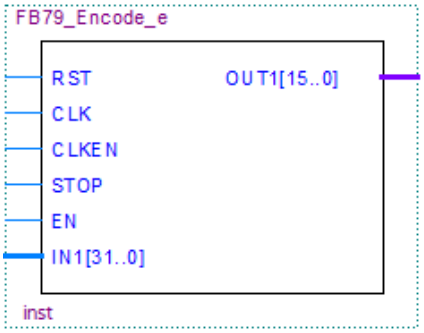


Figure 5: Example of Encode

Table 61: TM FAST function Encode (FB79_Encode)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC_VECTOR	Input	Variable in which bits are counted
	OUT1	STD_LOGIC_VECTOR	Output	Value output

```
entity FB79_Encode_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1  : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB79_Encode_e;
```

3.8 Encode, unlatched logic (FB79_Encode_U)

Description

The function converts the contents of IN1 to a binary number corresponding to the bit position of the leftmost bit set in IN1, and then returns the result as the function's value. If IN1 is either DW#16#00000001 or DW#16#00000000, a value of 0 is returned. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

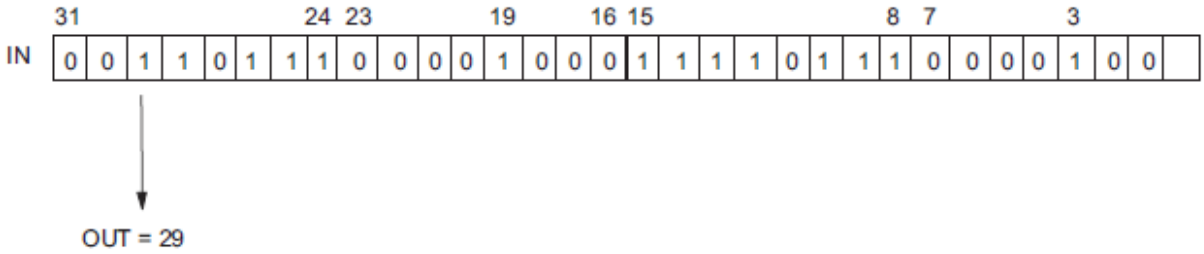
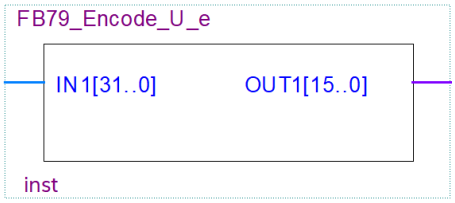


Figure 6: Example of Encode

Table 62: TM FAST function Encode, unlatched logic (FB79_Encode_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_VECTOR	Input	Variable in which bits are counted
	OUT1	STD_LOGIC_VECTOR	Output	Value output

```
entity FB79_Encode_U_e is
    Port (
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB79_Encode_U_e;
```

3.9 Period measurement (FB80_Period32, FB81_Period16)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

While EN is active, OUT1 is updated on every rising edge at IN1. VALID is true when OUT1 has valid data. VALID is false if OUT1 cannot represent the count (rollover occurs) and it is false until the initial period has been measured. If the module changes to STOP or if EN is inactive, the operation is reset. Two rising edges must be present at IN1 before OUT1 can be represented.

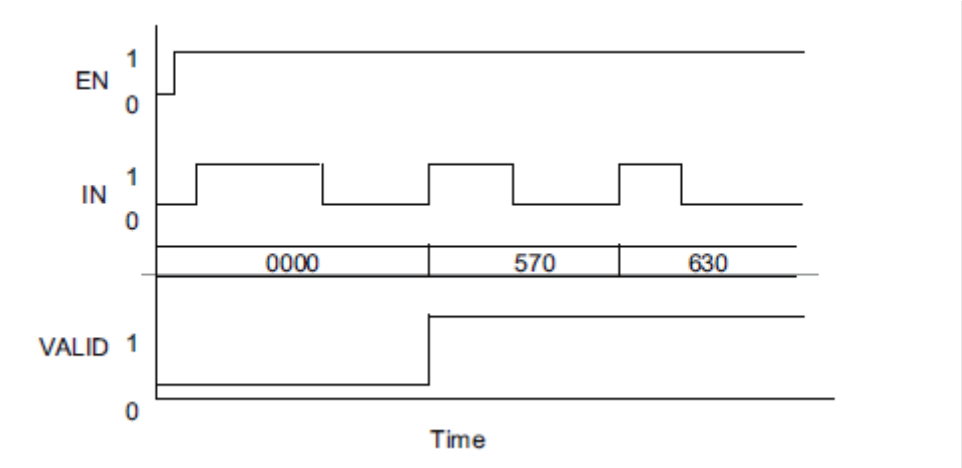


Figure 7: Example of FB80_Period32, FB81_Period16

<u>If Phases were used:</u>

(PHASE_QUANTITY = 14 and F_CLK_USER = 15_000_000 in TFL_FAST_USER_IP_CONF_PUBLIC_MP_FAST_1_p.vhd) :

The measured periode can be calculated:

T = 1.000.000 Hz * [OUT1 / periode]

FB80_Period32 is used to measure periods of 2 to 4,294,967,295 (2^{32}-1) microseconds. Periods greater than 2,147,483,647 (2^{31}-1) microseconds will appear negative. VALID will be 0 if the period exceeds 4,294,967,295 microseconds.

FB81_Period16 is used to measure periods of 2 to 65535 (2^{16}-1) microseconds. Periods greater than 32767 (2^{15}-1) microseconds will appear negative. VALID will be 0 if the period exceeds 65535 microseconds.

This operation outputs OUT1 in Hz if the period is set to 1,000,000 (1 second). If period is set to 10,000,000 (10 seconds) then OUT1 is in units of 0.1Hz (in other words, if OUT1 = 600, then the frequency is 60.0 Hz). The output value is retentive and uses one clock phase.

If Phases were not used:

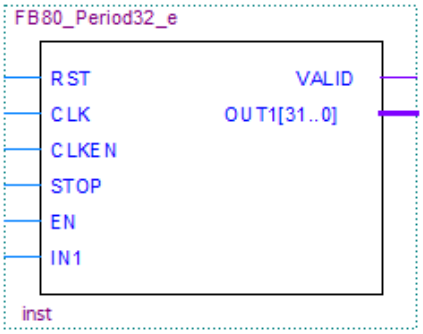
(PHASE_QUANTITY = 0 in TFL_FAST_USER_IP_CONF_PUBLIC_MP_FAST_1_p.vhd) :

The measured frequency can be calculated:

F = F_CLK_USER * [(OUT1) / (periode * PHASES)]

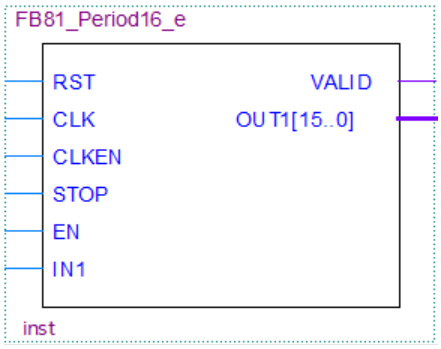
- F_CLK_USER: user clock frequency (value from TFL_FAST_USER_IP_CONF_PUBLIC_MP_FAST_1_p.vhd)
- PHASES: set to PHASE_QUANTITY + 1 (value from TFL_FAST_USER_IP_CONF_PUBLIC_MP_FAST_1_p.vhd)

Table 63: TM FAST function Period32 (FB80_Period32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC	Input	Input signal whose period is measured.
	VALID	STD_LOGIC	Output	Indicates PERIOD is valid
	OUT1	STD_LOGIC_ VECTOR	Output	Output of function

```
entity FB80_Period32_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC;
        VALID: out STD_LOGIC;
        OUT1  : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB80_Period32_e;
```

Table 64: TM FAST function Period16 (FB81_Period16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC	Input	Input signal whose period is measured
	VALID	STD_LOGIC	Output	Indicates PERIOD is valid
	OUT1	STD_LOGIC_ VECTOR	Output	Output of function

```
entity FB81_Period16_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC;
        VALID: out STD_LOGIC;
        OUT1  : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB81_Period16_e;
```

3.10 Frequency measurement (FB82_Freq32, FB83_Freq16)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

While EN is active, the operation counts the number of rising edges at IN1 during the number of microseconds defined in PERIOD. OUT1 is updated at an interval of PERIOD microseconds. VALID is true when OUT1 has valid data. VALID is false if OUT1 cannot represent the count (rollover occurs) and it is false if the initial period has not elapsed. If the module changes to STOP or if EN is inactive, the operation is reset. The number of microseconds defined in period must elapse before OUT1 can be represented.

<u>If Phases were used:</u>

(PHASE_QUANTITY = 14 and F_CLK_USER = 15_000_000 in TFL_FAST_USER_IP_CONF_PUBLIC_MP_FAST_1_p.vhd) :

The measured frequency can be calculated:

F = 1.000.000 * (OUT1 / periode) [Hz]

FB82_Freq32 is used to measure frequencies of $4{,}66*10^{-4}$ Hz to 500 kHz (periode durations of 2 to 4,294,967,295 (2^{32}-1) microseconds). Frequencies lower than $4{,}66*10^{-4}$ Hz will appear negative. VALID will be 0 if the periode duration exceeds 4,294,967,295 (2^{31}-1) microseconds.

FB83_Freq16 is used to measure frequencies of 15,26 Hz to 500 kHz (periode durations of 2 to 65535 (2^{16}-1) microseconds). Frequencies greater than 32767 (2^{15}-1) microseconds will appear negative. VALID will be 0 if the frequency exceeds 65535 microseconds.

This operation outputs OUT1 in Hz if the period is set to 1,000,000 (1 second). If period is set to 10,000,000 (10 seconds) then OUT1 is in units of 0.1Hz (in other words, if OUT1 = 600, then the frequency is 60.0 Hz). The output value is retentive and uses one clock phase.

<u>If Phases were not used:</u>

(PHASE_QUANTITY = 0 in TFL_FAST_USER_IP_CONF_PUBLIC_MP_FAST_1_p.vhd) :

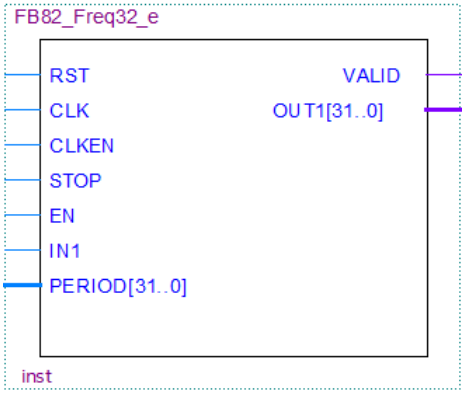
The measured frequency can be calculated:

F = F_CLK_USER * [(OUT1) / (periode * 15)] [Hz]

- F_CLK_USER: user clock frequency (value from TFL_FAST_USER_IP_CONF_PUBLIC_MP_FAST_1_p.vhd)

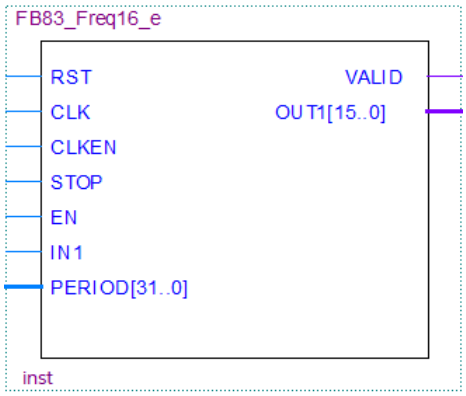
	PHASES used Phases=15 (PHASE_QUANTITY =14)	**Phases not used PHASES =1 (PHASE_QUANTITY = 0)**				
	F_CLK_USER =15MHz	**F_CLK_USER = 5MHz**	**F_CLK_USER= 15MHz**	**F_CLK_USER= 25MHz**	**F_CLK_USER= 50MHz**	**F_CLK_USER= 75MHz**
Min. Frequency (FB82_Freq 32)	$4{,}66*10^{-4}$ Hz [1000000/ (2^{31}-1) Hz]	$2{,}33*10^{-3}$ Hz [5000000/ (2^{31}-1) Hz]	$6{,}98*10^{-3}$ Hz [15000000/ (2^{31}-1) Hz]	$1{,}16*10^{-2}$ Hz [25000000/ (2^{31}-1) Hz]	$2{,}33*10^{-2}$ Hz [50000000/ (2^{31}-1) Hz]	$3{,}49*10^{-2}$ Hz [75000000/ (2^{31}-1) Hz]
Max. Frequency (FB82_Freq 32)	500kHz	2,5MHz	7,5MHz	12,5MHz	25MHz	37,5MHz
Min. Frequency (FB83_Freq 16)	15,26 Hz [1000000/ (2^{15}-1) Hz]	76,3 Hz [5000000/ (2^{15}-1) Hz]	228,9 Hz [15000000/ (2^{15}-1) Hz]	381,5 Hz [25000000/ (2^{15}-1) Hz]	763 Hz [50000000/ (2^{15}-1) Hz]	1144,4 Hz [75000000/ (2^{15}-1) Hz]
Max. Frequency (FB83_Freq 16)	500kHz	2,5MHz	7,5MHz	12,5MHz	25MHz	37,5MHz

Table 65: TM FAST function Freq32 (FB82_Freq32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC	Input	Input signal whose frequency is measured.
	PERIOD	STD_LOGIC_VECTOR	Input	Time period for frequency measurement (in microseconds)
	VALID	STD_LOGIC	Output	Indicates that frequency data is valid
	OUT1	STD_LOGIC_VECTOR	Output	Output of function

```
entity FB82_Freq32_e is
    Port (
        RST    : in  STD_LOGIC;
        CLK    : in  STD_LOGIC;
        CLKEN  : in  STD_LOGIC;
        STOP   : in  STD_LOGIC;
        EN     : in  STD_LOGIC;
        IN1    : in  STD_LOGIC;
        PERIOD: in  STD_LOGIC_VECTOR (31 downto 0);
        VALID  : out STD_LOGIC;
        OUT1   : out STD_LOGIC_VECTOR (31 downto 0)
         );
end FB82_Freq32_e;
```

Table 66: TM FAST function Freq16 (FB83_Freq16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Input signal whose frequency is measured.
	IN1	STD_LOGIC	Input	Time period for frequency measurement (in microseconds)
	PERIOD	STD_LOGIC_ VECTOR	Input	Indicates that frequency data is valid
	VALID	STD_LOGIC	Output	Output of function
	OUT1	STD_LOGIC_ VECTOR	Output	Input signal whose frequency is measured.

```
entity FB83_Freq16_e is
    Port (
        RST    : in  STD_LOGIC;
        CLK    : in  STD_LOGIC;
        CLKEN  : in  STD_LOGIC;
        STOP   : in  STD_LOGIC;
        EN     : in  STD_LOGIC;
        IN1    : in  STD_LOGIC;
        PERIOD: in  STD_LOGIC_VECTOR (31 downto 0);
        VALID  : out STD_LOGIC;
        OUT1   : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB83_Freq16_e;
```

3.11 Bit shift registers (FB124_Shift, FB125_Shift2, FB126_Shift4, FB127_Shift8, FB85_Shift16, FB84_Shift32)

Description

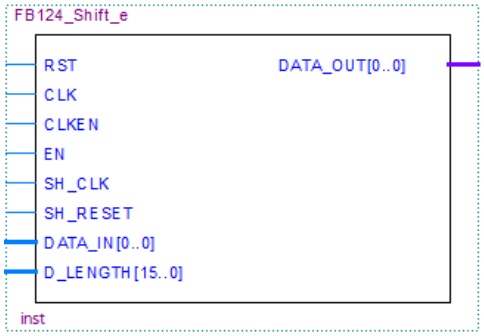
This operation is available in six versions defined by the number of simultaneously shifted bits.

When the SH_CLK input changes from 0 to 1, the value at the DATA_IN is shifted into the first stage of the shift register and is shifted again on each subsequent SH_CLK rising edge. The DATA_OUT is set by the last stage in the shift register. When the EN = ‘1’ and SH_RESET = ‘1’, all of the stages of the shift register are reset to 0.

D_LENGTH sets the depth of the shift register, before the first value clocked in reaches DATA_OUT.

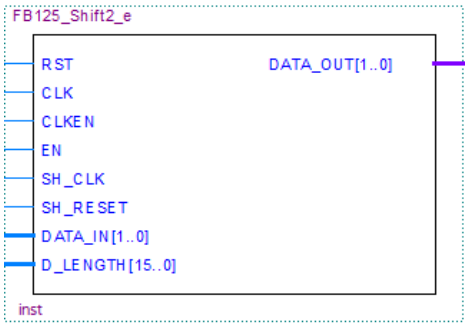
FB124_Shift	2 to 4096
FB125_Shift2	2 to 2048
FB126_Shift4	2 to 1024
FB127_Shift8	2 to 512
FB85_Shift16	2 to 256
FB84_Shift32	2 to 256

Table 67: TM FAST function Shift (FB124_Shift):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	SH_CLK	STD_LOGIC	Input	Rising edge that shifts the data through the shift register
	SH_RESET	STD_LOGIC	Input	Resets all stages of the shift register.
	DATA_IN	STD_LOGIC _VECTOR	Input	Data input for the shift register
	D_LENGTH	STD_LOGIC _VECTOR	Input	Length of the shift register
	DATA_OUT	STD_LOGIC _VECTOR	Output	Output of the shift register

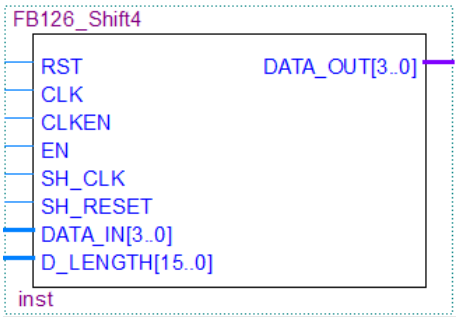
```
entity FB124_Shift_e is
    Port (
        RST      : in  STD_LOGIC;
        CLK      : in  STD_LOGIC;
        CLKEN    : in  STD_LOGIC;
        EN       : in  STD_LOGIC;
        SH_CLK   : in  STD_LOGIC;
        SH_RESET: in  STD_LOGIC;
        DATA_IN : in  STD_LOGIC_VECTOR (0 downto 0);
        D_LENGTH: in  STD_LOGIC_VECTOR (15 downto 0);
        DATA_OUT: out STD_LOGIC_VECTOR (0 downto 0)
    );
end FB124_Shift_e;
```

Table 68: TM FAST function Shift2 (FB125_Shift2)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	SH_CLK	STD_LOGIC	Input	Rising edge that shifts the data through the shift register
	SH_RESET	STD_LOGIC	Input	Resets all stages of the shift register.
	DATA_IN	STD_LOGIC _VECTOR	Input	Data input for the shift register
	D_LENGTH	STD_LOGIC _VECTOR	Input	Length of the shift register
	DATA_OUT	STD_LOGIC _VECTOR	Output	Output of the shift register

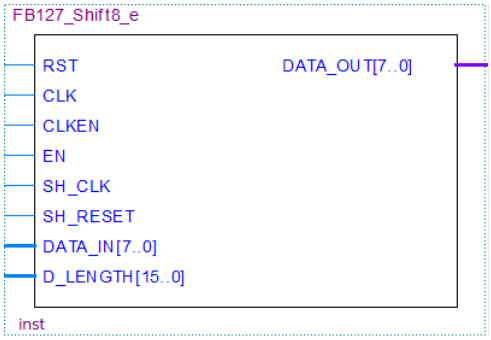
```
entity FB125_Shift2_e is
    Port (
        RST      : in  STD_LOGIC;
        CLK      : in  STD_LOGIC;
        CLKEN    : in  STD_LOGIC;
        EN       : in  STD_LOGIC;
        SH_CLK   : in  STD_LOGIC;
        SH_RESET: in  STD_LOGIC;
        DATA_IN : in  STD_LOGIC_VECTOR (1 downto 0);
        D_LENGTH: in  STD_LOGIC_VECTOR (15 downto 0);
        DATA_OUT: out STD_LOGIC_VECTOR (1 downto 0)
    );
end FB125_Shift2_e;
```

Table 69: TM FAST function Shift4 (FB126_Shift4):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	SH_CLK	STD_LOGIC	Input	Rising edge that shifts the data through the shift register
	SH_RESET	STD_LOGIC	Input	Resets all stages of the shift register.
	DATA_IN	STD_LOGIC _VECTOR	Input	Data input for the shift register
	D_LENGTH	STD_LOGIC _VECTOR	Input	Length of the shift register
	DATA_OUT	STD_LOGIC _VECTOR	Output	Output of the shift register

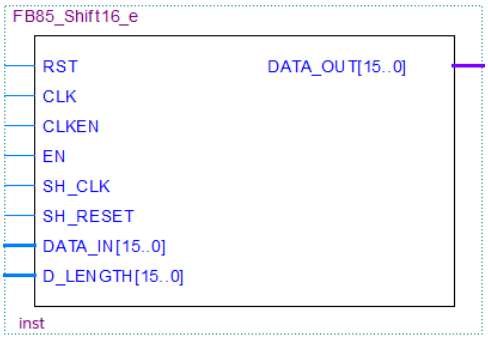
```
entity FB126_Shift4 is
    Port (
        RST      : in  STD_LOGIC;
        CLK      : in  STD_LOGIC;
        CLKEN    : in  STD_LOGIC;
        EN       : in  STD_LOGIC;
        SH_CLK   : in  STD_LOGIC;
        SH_RESET: in  STD_LOGIC;
        DATA_IN : in  STD_LOGIC_VECTOR (3 downto 0);
        D_LENGTH: in  STD_LOGIC_VECTOR (15 downto 0);
        DATA_OUT: out STD_LOGIC_VECTOR (3 downto 0)
    );
end FB126_Shift4;
```

Table 70: TM FAST function Shift8 (FB127_Shift8):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	SH_CLK	STD_LOGIC	Input	Rising edge that shifts the data through the shift register
	SH_RESET	STD_LOGIC	Input	Resets all stages of the shift register.
	DATA_IN	STD_LOGIC _VECTOR	Input	Data input for the shift register
	D_LENGTH	STD_LOGIC _VECTOR	Input	Length of the shift register
	DATA_OUT	STD_LOGIC _VECTOR	Output	Output of the shift register

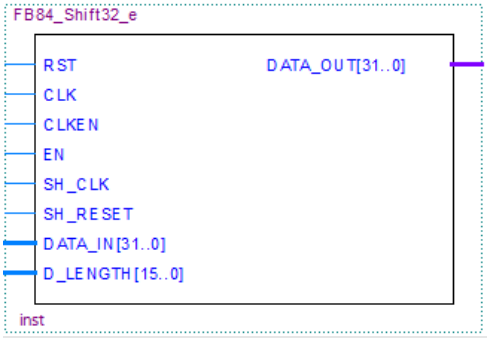
```
entity FB127_Shift8_e is
    Port (
        RST      : in  STD_LOGIC;
        CLK      : in  STD_LOGIC;
        CLKEN    : in  STD_LOGIC;
        EN       : in  STD_LOGIC;
        SH_CLK   : in  STD_LOGIC;
        SH_RESET: in  STD_LOGIC;
        DATA_IN : in  STD_LOGIC_VECTOR (7 downto 0);
        D_LENGTH: in  STD_LOGIC_VECTOR (15 downto 0);
        DATA_OUT: out STD_LOGIC_VECTOR (7 downto 0)
    );
end FB127_Shift8_e;
```

Table 71: TM FAST function Shift16 (FB85_Shift16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	SH_CLK	STD_LOGIC	Input	Rising edge that shifts the data through the shift register
	SH_RESET	STD_LOGIC	Input	Resets all stages of the shift register.
	DATA_IN	STD_LOGIC _VECTOR	Input	Data input for the shift register
	D_LENGTH	STD_LOGIC _VECTOR	Input	Length of the shift register
	DATA_OUT	STD_LOGIC _VECTOR	Output	Output of the shift register

```
entity FB85_Shift16_e is
    Port (
        RST      : in  STD_LOGIC;
        CLK      : in  STD_LOGIC;
        CLKEN    : in  STD_LOGIC;
        EN       : in  STD_LOGIC;
        SH_CLK   : in  STD_LOGIC;
        SH_RESET: in  STD_LOGIC;
        DATA_IN : in  STD_LOGIC_VECTOR (15 downto 0);
        D_LENGTH: in  STD_LOGIC_VECTOR (15 downto 0);
        DATA_OUT: out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB85_Shift16_e;
```

Table 72: TM FAST function Shif32t (FB84_Shift32):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	SH_CLK	STD_LOGIC	Input	Rising edge that shifts the data through the shift register
	SH_RESET	STD_LOGIC	Input	Resets all stages of the shift register.
	DATA_IN	STD_LOGIC _VECTOR	Input	Data input for the shift register
	D_LENGTH	STD_LOGIC _VECTOR	Input	Length of the shift register
	DATA_OUT	STD_LOGIC _VECTOR	Output	Output of the shift register

```
entity FB84_Shift32_e is
    Port (
        RST      : in  STD_LOGIC;
        CLK      : in  STD_LOGIC;
        CLKEN    : in  STD_LOGIC;
        EN       : in  STD_LOGIC;
        SH_CLK   : in  STD_LOGIC;
        SH_RESET: in  STD_LOGIC;
        DATA_IN : in  STD_LOGIC_VECTOR (31 downto 0);
        D_LENGTH: in  STD_LOGIC_VECTOR (15 downto 0);
        DATA_OUT: out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB84_Shift32_e;
```

3.12 BitPick (FB86_BitPick_DW, FB87_BitPick_W)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

While EN is active, the bit selected within the input Word or DWord is transferred to OUT1. The bit position selected within IN1 is determined by the integer provided at BITSELECT. If BITSELECT is 0, then the LSB of the input WORD or DWORD is transferred to OUT1. If BITSELECT is 15 (or 31) the MSB of the input WORD (DWORD) is transferred to OUT1.

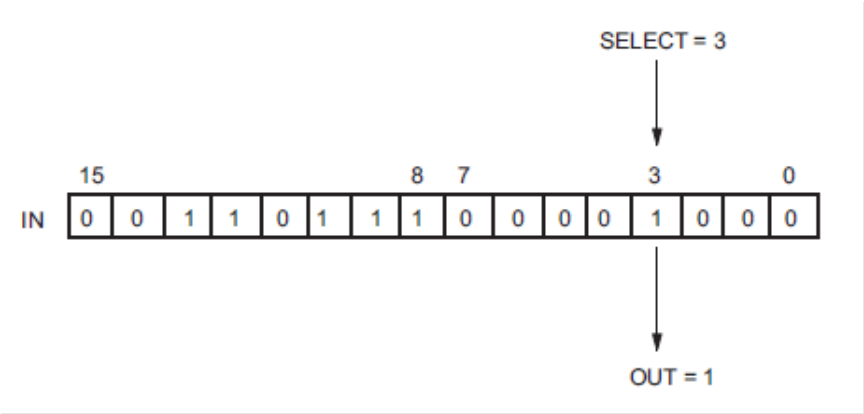
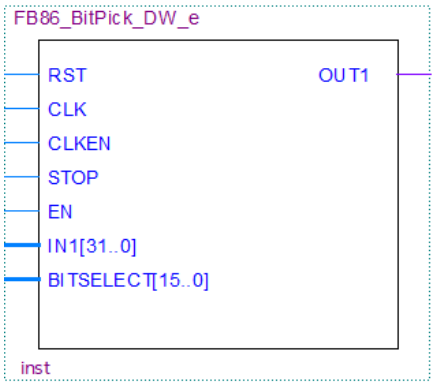


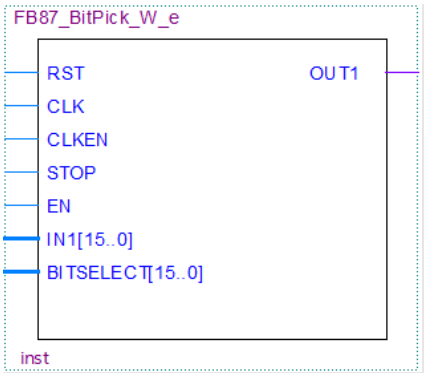
Figure 8: Example of FB86_BitPick_DW, FB87_BitPick_W

Table 73: TM FAST function BitPick_DW (FB86_BitPick_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
	BITSELECT	STD_LOGIC_ VECTOR	Input	Bit position to be selected within IN1
	OUT1	STD_LOGIC	Output	Output of function

```
entity FB86_BitPick_DW_e is
    Port (
        RST       : in  STD_LOGIC;
        CLK       : in  STD_LOGIC;
        CLKEN     : in  STD_LOGIC;
        STOP      : in  STD_LOGIC;
        EN        : in  STD_LOGIC;
        IN1       : in  STD_LOGIC_VECTOR (31 downto 0);
        BITSELECT: in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1      : out STD_LOGIC
    );
end FB86_BitPick_DW_e;
```

Table 74: TM FAST function BitPick_W (FB87_BitPick_W)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
	BITSELECT	STD_LOGIC_ VECTOR	Input	Bit position to be selected within IN1
	OUT1	STD_LOGIC	Output	Output of function

```
entity FB87_BitPick_W_e is
    Port (
        RST       : in  STD_LOGIC;
        CLK       : in  STD_LOGIC;
        CLKEN     : in  STD_LOGIC;
        STOP      : in  STD_LOGIC;
        EN        : in  STD_LOGIC;
        IN1       : in  STD_LOGIC_VECTOR (15 downto 0);
        BITSELECT: in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1      : out STD_LOGIC
    );
end FB87_BitPick_W_e;
```

3.13 BitPick, unlatched logic (FB86_BitPick_DW_U, FB87_BitPick_W_U)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

The bit selected within the input Word or DWord is transferred to OUT1. The bit position selected within IN1 is determined by the integer provided at BITSELECT. If BITSELECT is 0, then the LSB of the input WORD or DWORD is transferred to OUT1. If BITSELECT is 15 (or 31) the MSB of the input WORD (DWORD) is transferred to OUT1.

The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

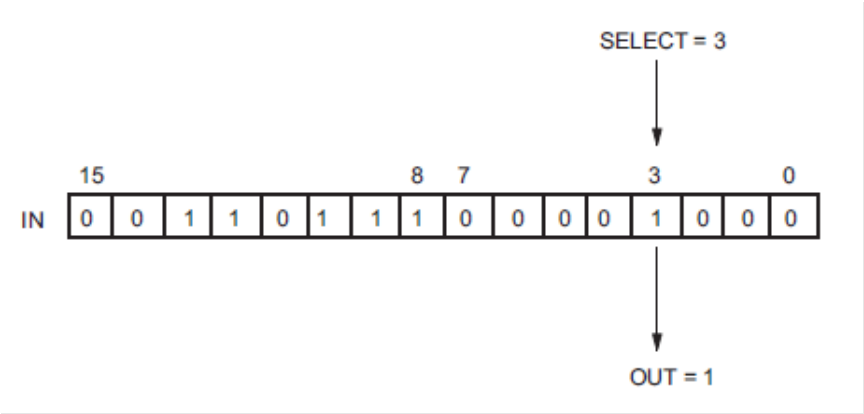
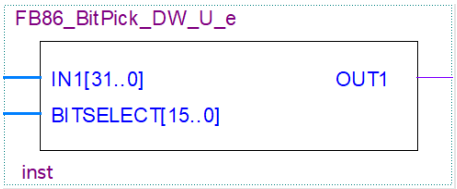


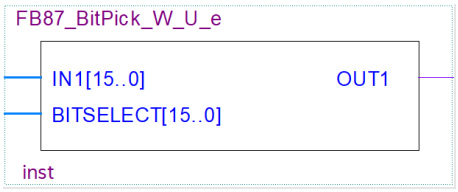
Figure 9: Example of FB86_BitPick_DW_U, FB87_BitPick_W_U

Table 75: TM FAST function BitPick_DW_U (FB86_BitPick_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
	BITSELECT	STD_LOGIC_ VECTOR	Input	Bit position to be selected within IN1
	OUT1	STD_LOGIC	Output	Output of function

```
entity FB86_BitPick_DW_U_e is
    Port (
        IN1       : in  STD_LOGIC_VECTOR (31 downto 0);
        BITSELECT: in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1      : out STD_LOGIC
    );
end FB86_BitPick_DW_U_e;
```

Table 76: TM FAST function BitPick_W_U (FB87_BitPick_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
	BITSELECT	STD_LOGIC_ VECTOR	Input	Bit position to be selected within IN1
	OUT1	STD_LOGIC	Output	Output of function

```
entity FB87_BitPick_W_U_e is
    Port (
        IN1       : in  STD_LOGIC_VECTOR (15 downto 0);
        BITSELECT: in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1      : out STD_LOGIC
    );
end FB87_BitPick_W_U_e;
```

3.14 BitShift (FB88_BitShift_DW, FB89_BitShift_W)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

While EN is active and SHIFT transitions from 0 to 1, the IN1 BOOL is left-shifted into OUT1. The MSB of OUT1 is discarded. The LSB is replaced with IN1. IF EN = 1 and SH_RESET = 1 then OUT1 is reset to 0.

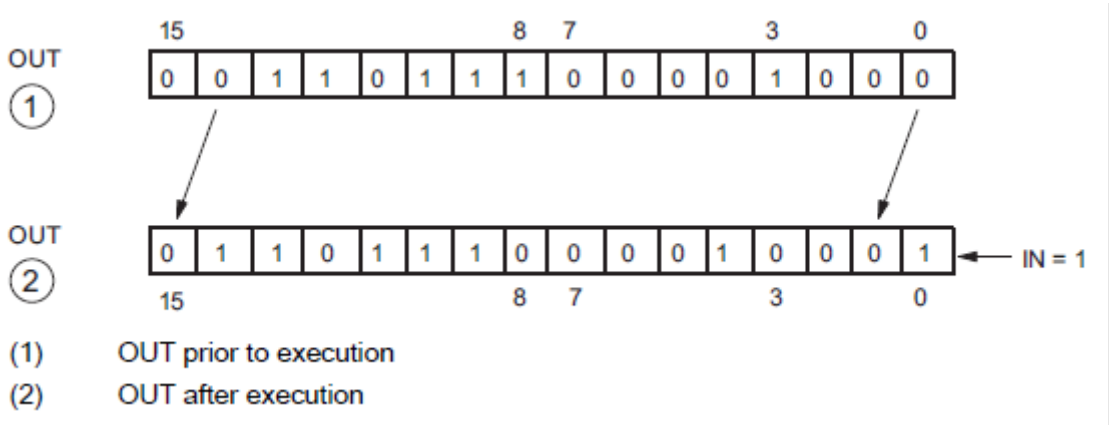
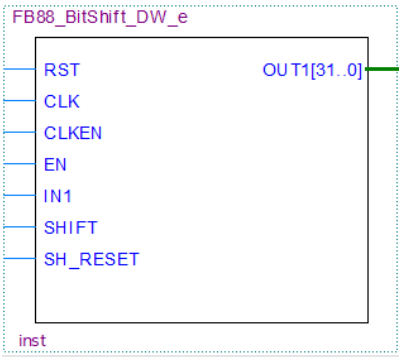


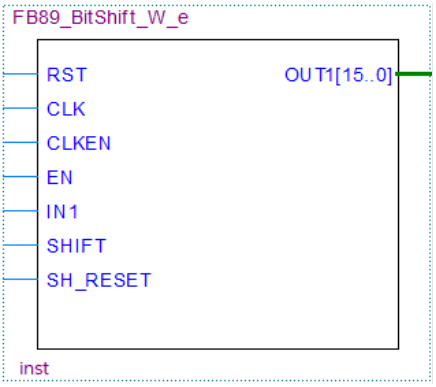
Figure 10: Example of FB88_BitShift_DW, FB89_BitShift_W

Table 77: TM FAST function BitShift_DW (FB88_BitShift_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC	Input	Input bit to be shifted into LSB of OUT1
	SHIFT	STD_LOGIC	Input	If 1 and EN is active, shift is enabled.
	SH_RESET	STD_LOGIC	Input	IF 1 and EN is active, OUT1 is reset to 0.
	OUT1	STD_LOGIC _VECTOR	Output	Output of function

```
entity FB88_BitShift_DW_e is
    Port (
        RST      : in    STD_LOGIC;
        CLK      : in    STD_LOGIC;
        CLKEN    : in    STD_LOGIC;
        EN       : in    STD_LOGIC;
        IN1      : in    STD_LOGIC;
        SHIFT    : in    STD_LOGIC;
        SH_RESET: in    STD_LOGIC;
        OUT1     : inout STD_LOGIC_VECTOR (31 downto 0)
    );
end FB88_BitShift_DW_e;
```

Table 78: TM FAST function BitShift_W (FB89_BitShift_W):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC	Input	Input bit to be shifted into LSB of OUT1
	SHIFT	STD_LOGIC	Input	If 1 and EN is active, shift is enabled.
	SH_RESET	STD_LOGIC	Input	IF 1 and EN is active, OUT1 is reset to 0.
	OUT1	STD_LOGIC_VECTOR	Output	Output of function

```
entity FB89_BitShift_W_e is
    Port (
        RST      : in    STD_LOGIC;
        CLK      : in    STD_LOGIC;
        CLKEN    : in    STD_LOGIC;
        EN       : in    STD_LOGIC;
        IN1      : in    STD_LOGIC;
        SHIFT    : in    STD_LOGIC;
        SH_RESET: in    STD_LOGIC;
        OUT1     : inout STD_LOGIC_VECTOR (15 downto 0)
    );
end FB89_BitShift_W_e;
```

3.15 BitCast (FB90_BitCast_DW, FB91_BitCast_W)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

While EN is active, the WORD or DWORD is converted into individual bits.

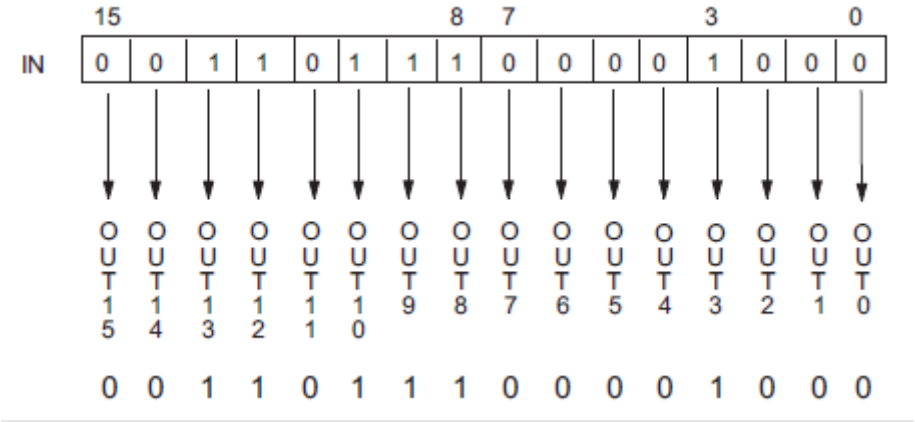
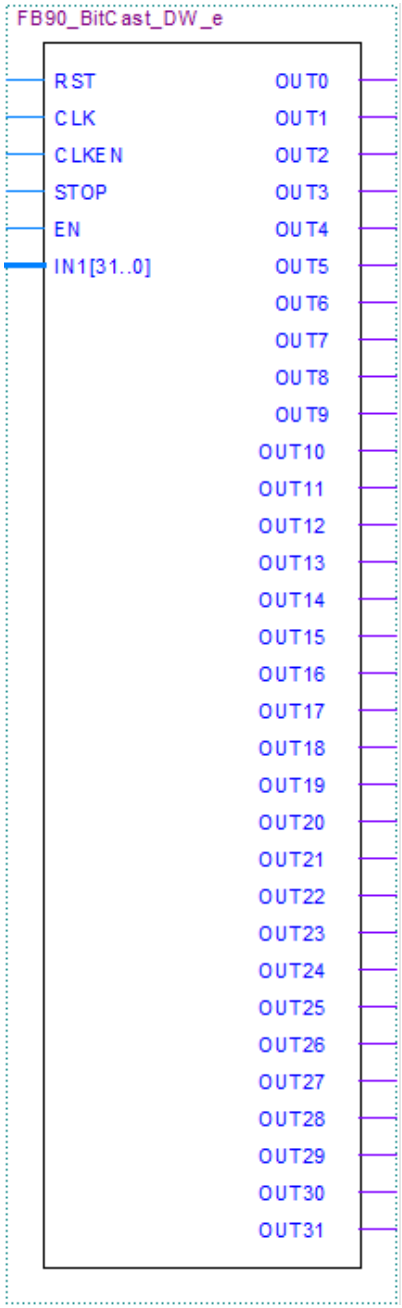


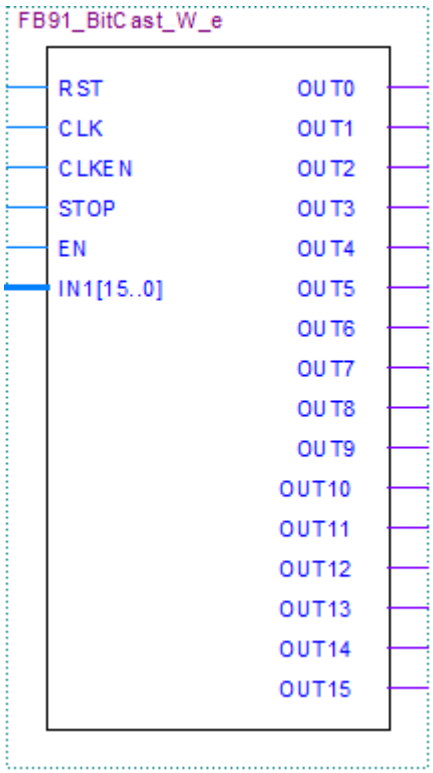
Figure 11: Example of FB90_BitCast_DW, FB91_BitCast_W

Table 79: TM FAST function BitCast_DW (FB90_BitCast_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC_ VECTOR	Input	Input to be converted
	OUTn	STD_LOGIC	Output	Output of function

```
entity FB90_BitCast_DW_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN : in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT0  : out STD_LOGIC;
        OUT1  : out STD_LOGIC;
        OUT2  : out STD_LOGIC;
        OUT3  : out STD_LOGIC;
        OUT4  : out STD_LOGIC;
        OUT5  : out STD_LOGIC;
        OUT6  : out STD_LOGIC;
        OUT7  : out STD_LOGIC;
        OUT8  : out STD_LOGIC;
        OUT9  : out STD_LOGIC;
        OUT10 : out STD_LOGIC;
        OUT11 : out STD_LOGIC;
        OUT12 : out STD_LOGIC;
        OUT13 : out STD_LOGIC;
        OUT14 : out STD_LOGIC;
        OUT15 : out STD_LOGIC;
        OUT16 : out STD_LOGIC;
        OUT17 : out STD_LOGIC;
        OUT18 : out STD_LOGIC;
        OUT19 : out STD_LOGIC;
        OUT20 : out STD_LOGIC;
        OUT21 : out STD_LOGIC;
        OUT22 : out STD_LOGIC;
        OUT23 : out STD_LOGIC;
        OUT24 : out STD_LOGIC;
        OUT25 : out STD_LOGIC;
        OUT26 : out STD_LOGIC;
        OUT27 : out STD_LOGIC;
        OUT28 : out STD_LOGIC;
        OUT29 : out STD_LOGIC;
        OUT30 : out STD_LOGIC;
        OUT31 : out STD_LOGIC
    );
end FB90_BitCast_DW_e;
```

Table 80: TM FAST function BitCast_W (FB91_BitCast_W):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC_ VECTOR	Input	Input to be converted
	OUTn	STD_LOGIC	Output	Output of function

```
entity FB91_BitCast_W_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT0  : out STD_LOGIC;
        OUT1  : out STD_LOGIC;
        OUT2  : out STD_LOGIC;
        OUT3  : out STD_LOGIC;
        OUT4  : out STD_LOGIC;
        OUT5  : out STD_LOGIC;
        OUT6  : out STD_LOGIC;
        OUT7  : out STD_LOGIC;
        OUT8  : out STD_LOGIC;
        OUT9  : out STD_LOGIC;
        OUT10: out STD_LOGIC;
        OUT11: out STD_LOGIC;
        OUT12: out STD_LOGIC;
        OUT13: out STD_LOGIC;
        OUT14: out STD_LOGIC;
        OUT15: out STD_LOGIC
    );
end FB91_BitCast_W_e;
```

3.16 BitCast, unlatched logic (FB90_BitCast_DW_U, FB91_BitCast_W_U)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

The WORD or DWORD is converted into individual bits. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

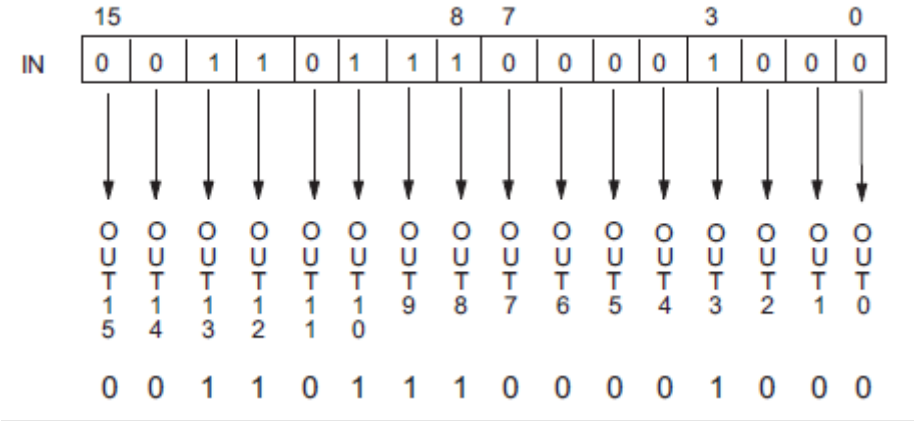
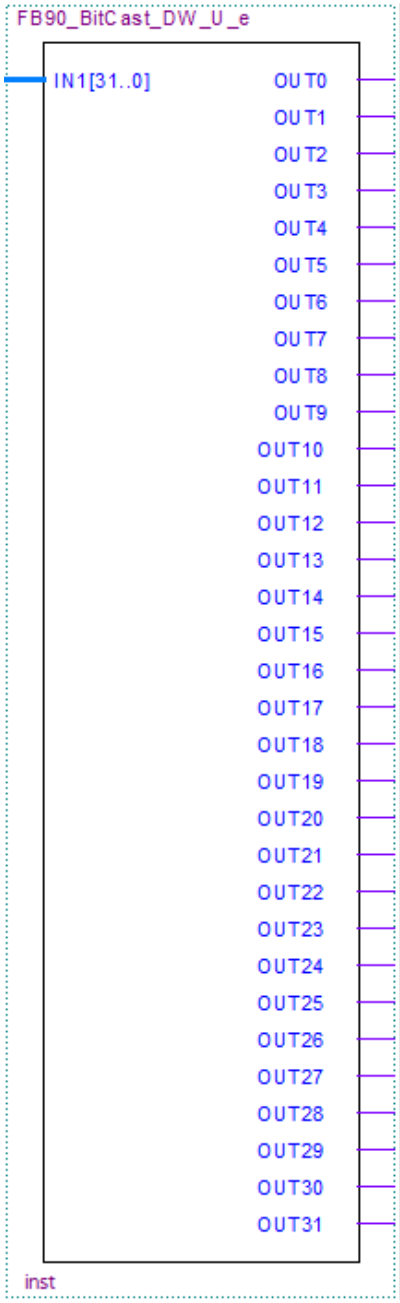


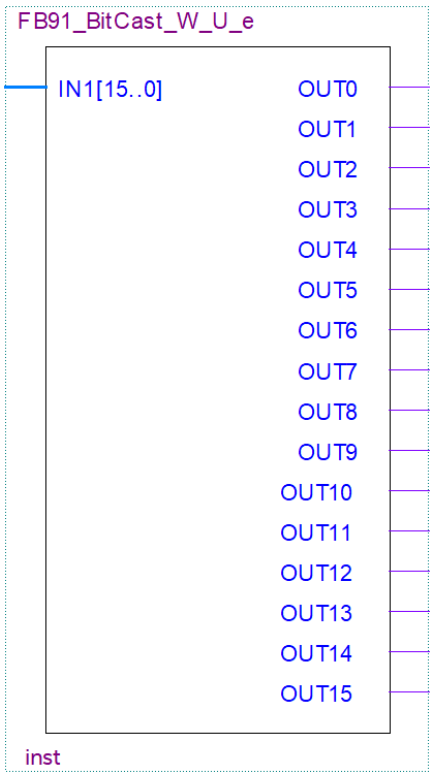
Figure 12: Example of FB90_BitCast_DW, FB91_BitCast_W

Table 81: TM FAST function BitCast_DW, unlatched logic (FB90_BitCast_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_VECTOR	Input	Input to be converted
	OUTn	STD_LOGIC	Output	Output of function

```
entity FB90_BitCast_DW_U_e is
    Port (
        IN1    : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT0   : out STD_LOGIC;
        OUT1   : out STD_LOGIC;
        OUT2   : out STD_LOGIC;
        OUT3   : out STD_LOGIC;
        OUT4   : out STD_LOGIC;
        OUT5   : out STD_LOGIC;
        OUT6   : out STD_LOGIC;
        OUT7   : out STD_LOGIC;
        OUT8   : out STD_LOGIC;
        OUT9   : out STD_LOGIC;
        OUT10  : out STD_LOGIC;
        OUT11  : out STD_LOGIC;
        OUT12  : out STD_LOGIC;
        OUT13  : out STD_LOGIC;
        OUT14  : out STD_LOGIC;
        OUT15  : out STD_LOGIC;
        OUT16  : out STD_LOGIC;
        OUT17  : out STD_LOGIC;
        OUT18  : out STD_LOGIC;
        OUT19  : out STD_LOGIC;
        OUT20  : out STD_LOGIC;
        OUT21  : out STD_LOGIC;
        OUT22  : out STD_LOGIC;
        OUT23  : out STD_LOGIC;
        OUT24  : out STD_LOGIC;
        OUT25  : out STD_LOGIC;
        OUT26  : out STD_LOGIC;
        OUT27  : out STD_LOGIC;
        OUT28  : out STD_LOGIC;
        OUT29  : out STD_LOGIC;
        OUT30  : out STD_LOGIC;
        OUT31  : out STD_LOGIC
    );
end FB90_BitCast_DW_U_e;
```

Table 82: TM FAST function BitCast_W_U, unlatched logic (FB91_BitCast_W_U):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Input to be converted
	OUTn	STD_LOGIC	Output	Output of function

```
entity FB91_BitCast_W_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT0  : out STD_LOGIC;
        OUT1  : out STD_LOGIC;
        OUT2  : out STD_LOGIC;
        OUT3  : out STD_LOGIC;
        OUT4  : out STD_LOGIC;
        OUT5  : out STD_LOGIC;
        OUT6  : out STD_LOGIC;
        OUT7  : out STD_LOGIC;
        OUT8  : out STD_LOGIC;
        OUT9  : out STD_LOGIC;
        OUT10: out STD_LOGIC;
        OUT11: out STD_LOGIC;
        OUT12: out STD_LOGIC;
        OUT13: out STD_LOGIC;
        OUT14: out STD_LOGIC;
        OUT15: out STD_LOGIC
    );
end FB91_BitCast_W_e;
```

3.17 BitPack (FB92_BitPack_DW, FB93_BitPack_W)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

While EN is active, the BOOL inputs at INn are packed to form a WORD or DWORD.

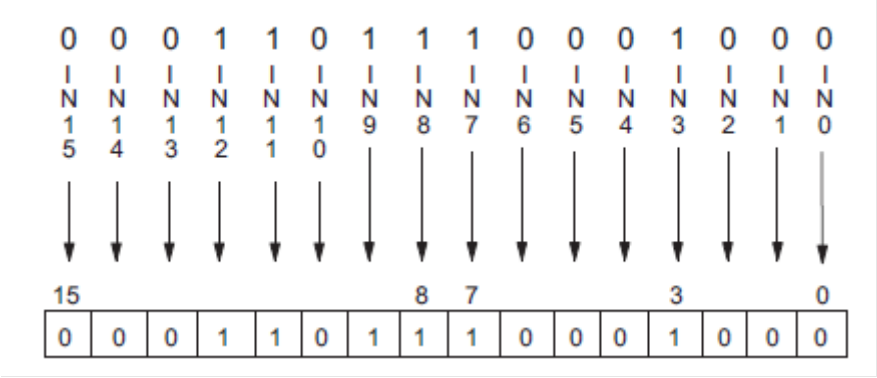
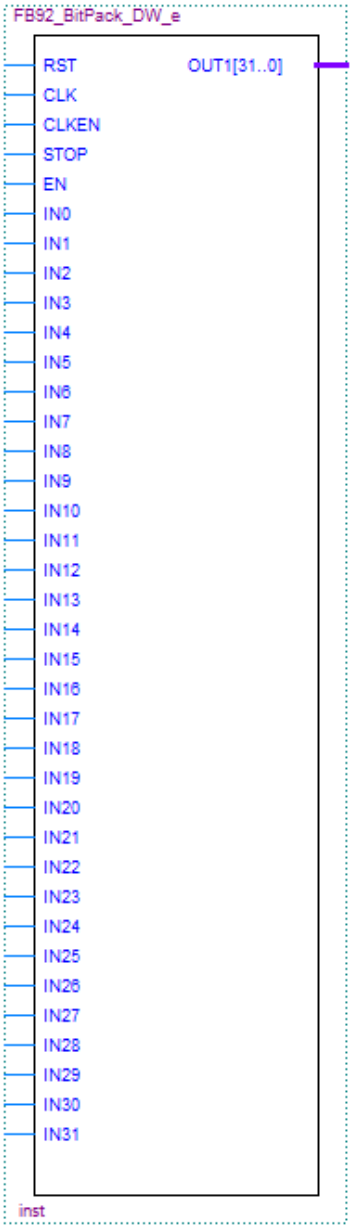


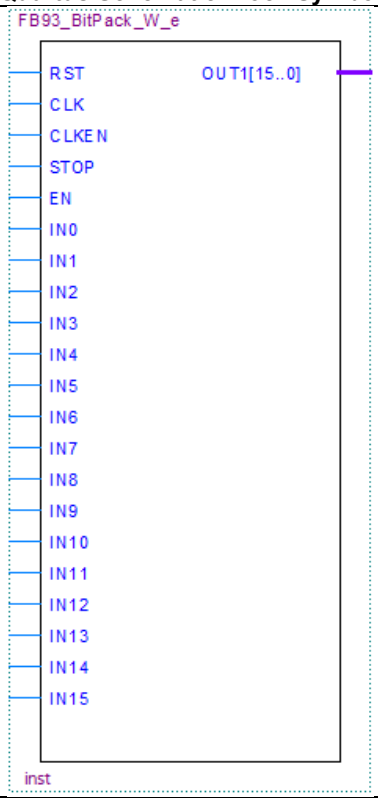
Figure 13: Example of FB92_BitPack_DW, FB93_BitPack_W

Table 83: TM FAST function BitPack_DW (FB92_BitPack_DW)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	INn	STD_LOGIC	Input	Inputs to be packed
	OUT1	STD_LOGIC_VECTOR	Output	Output of function

```
entity FB92_BitPack_DW_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN0   : in  STD_LOGIC;
        IN1   : in  STD_LOGIC;
        IN2   : in  STD_LOGIC;
        IN3   : in  STD_LOGIC;
        IN4   : in  STD_LOGIC;
        IN5   : in  STD_LOGIC;
        IN6   : in  STD_LOGIC;
        IN7   : in  STD_LOGIC;
        IN8   : in  STD_LOGIC;
        IN9   : in  STD_LOGIC;
        IN10 : in  STD_LOGIC;
        IN11 : in  STD_LOGIC;
        IN12 : in  STD_LOGIC;
        IN13 : in  STD_LOGIC;
        IN14 : in  STD_LOGIC;
        IN15 : in  STD_LOGIC;
        IN16 : in  STD_LOGIC;
        IN17 : in  STD_LOGIC;
        IN18 : in  STD_LOGIC;
        IN19 : in  STD_LOGIC;
        IN20 : in  STD_LOGIC;
        IN21 : in  STD_LOGIC;
        IN22 : in  STD_LOGIC;
        IN23 : in  STD_LOGIC;
        IN24 : in  STD_LOGIC;
        IN25 : in  STD_LOGIC;
        IN26 : in  STD_LOGIC;
        IN27 : in  STD_LOGIC;
        IN28 : in  STD_LOGIC;
        IN29 : in  STD_LOGIC;
        IN30 : in  STD_LOGIC;
        IN31 : in  STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB92_BitPack_DW_e;
```

Table 84: TM FAST function BitPack_W (FB93_BitPack_W)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	INn	STD_LOGIC	Input	Inputs to be packed
	OUT1	STD_LOGIC_VECTOR	Output	Output of function

```
entity FB93_BitPack_W_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN0   : in  STD_LOGIC;
        IN1   : in  STD_LOGIC;
        IN2   : in  STD_LOGIC;
        IN3   : in  STD_LOGIC;
        IN4   : in  STD_LOGIC;
        IN5   : in  STD_LOGIC;
        IN6   : in  STD_LOGIC;
        IN7   : in  STD_LOGIC;
        IN8   : in  STD_LOGIC;
        IN9   : in  STD_LOGIC;
        IN10  : in  STD_LOGIC;
        IN11  : in  STD_LOGIC;
        IN12  : in  STD_LOGIC;
        IN13  : in  STD_LOGIC;
        IN14  : in  STD_LOGIC;
        IN15  : in  STD_LOGIC;
        OUT1  : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB93_BitPack_W_e;
```

3.18 BitPack, unlatched logic (FB92_BitPack_DW_U, FB93_BitPack_W_U)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

The BOOL inputs at INn are packed to form a WORD or DWORD. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

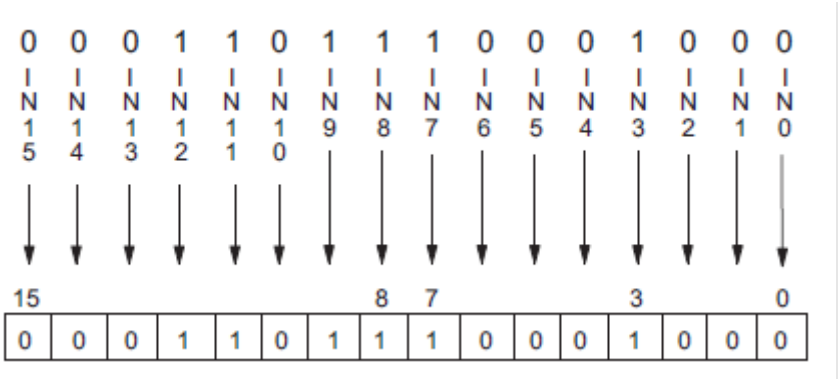
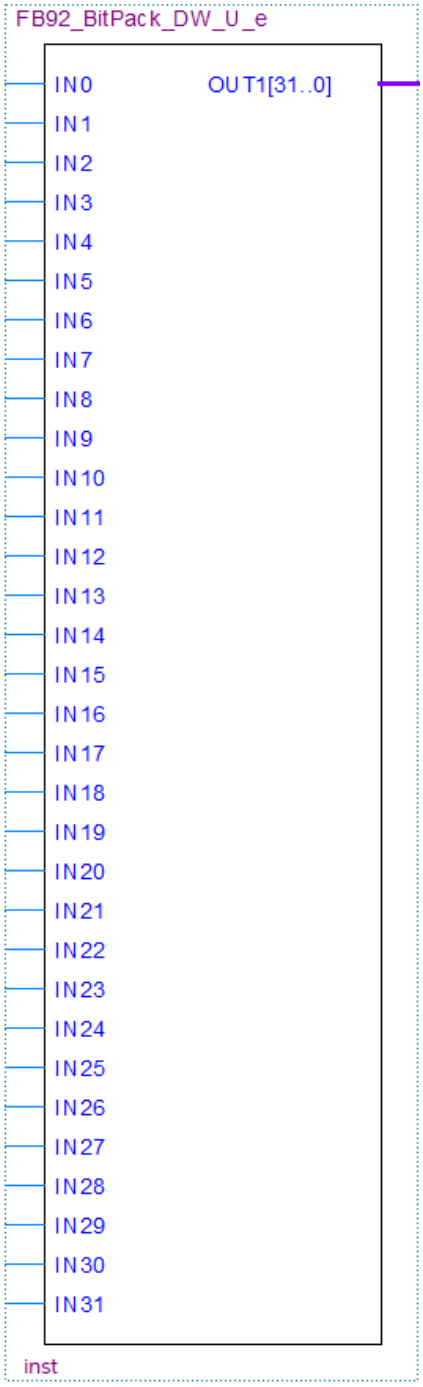


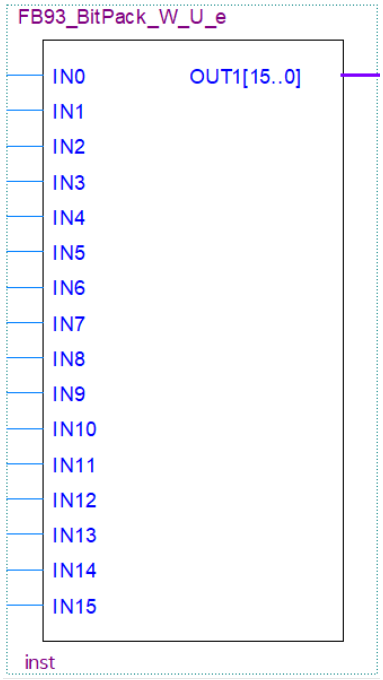
Figure 14: Example of FB92_BitPack_DW_U, FB93_BitPack_W_U

Table 85: TM FAST function BitPack_DW_U, unlatched logic (FB92_BitPack_DW_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	INn	STD_LOGIC	Input	Inputs to be packed
	OUT1	STD_LOGIC_VECTOR	Output	Output of function

```
entity FB92_BitPack_DW_U_e is
    Port (
        IN0  : in  STD_LOGIC;
        IN1  : in  STD_LOGIC;
        IN2  : in  STD_LOGIC;
        IN3  : in  STD_LOGIC;
        IN4  : in  STD_LOGIC;
        IN5  : in  STD_LOGIC;
        IN6  : in  STD_LOGIC;
        IN7  : in  STD_LOGIC;
        IN8  : in  STD_LOGIC;
        IN9  : in  STD_LOGIC;
        IN10 : in  STD_LOGIC;
        IN11 : in  STD_LOGIC;
        IN12 : in  STD_LOGIC;
        IN13 : in  STD_LOGIC;
        IN14 : in  STD_LOGIC;
        IN15 : in  STD_LOGIC;
        IN16 : in  STD_LOGIC;
        IN17 : in  STD_LOGIC;
        IN18 : in  STD_LOGIC;
        IN19 : in  STD_LOGIC;
        IN20 : in  STD_LOGIC;
        IN21 : in  STD_LOGIC;
        IN22 : in  STD_LOGIC;
        IN23 : in  STD_LOGIC;
        IN24 : in  STD_LOGIC;
        IN25 : in  STD_LOGIC;
        IN26 : in  STD_LOGIC;
        IN27 : in  STD_LOGIC;
        IN28 : in  STD_LOGIC;
        IN29 : in  STD_LOGIC;
        IN30 : in  STD_LOGIC;
        IN31 : in  STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB92_BitPack_DW_U_e;
```

Table 86: TM FAST function BitPack_W_U, unlatched logic (FB93_BitPack_W_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	INn	STD_LOGIC	Input	Inputs to be packed
	OUT1	STD_LOGIC_VECTOR	Output	Output of function

```
entity FB93_BitPack_W_U_e is
    Port (
        IN0  : in  STD_LOGIC;
        IN1  : in  STD_LOGIC;
        IN2  : in  STD_LOGIC;
        IN3  : in  STD_LOGIC;
        IN4  : in  STD_LOGIC;
        IN5  : in  STD_LOGIC;
        IN6  : in  STD_LOGIC;
        IN7  : in  STD_LOGIC;
        IN8  : in  STD_LOGIC;
        IN9  : in  STD_LOGIC;
        IN10 : in  STD_LOGIC;
        IN11 : in  STD_LOGIC;
        IN12 : in  STD_LOGIC;
        IN13 : in  STD_LOGIC;
        IN14 : in  STD_LOGIC;
        IN15 : in  STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB93_BitPack_W_U_e;
```

3.19 BitInsert (FB94_BitInsert32, FB95_BitInsert32)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

While EN is active, the bit selected within the input Word or DWORD is replaced. All other bits are transferred with no change.

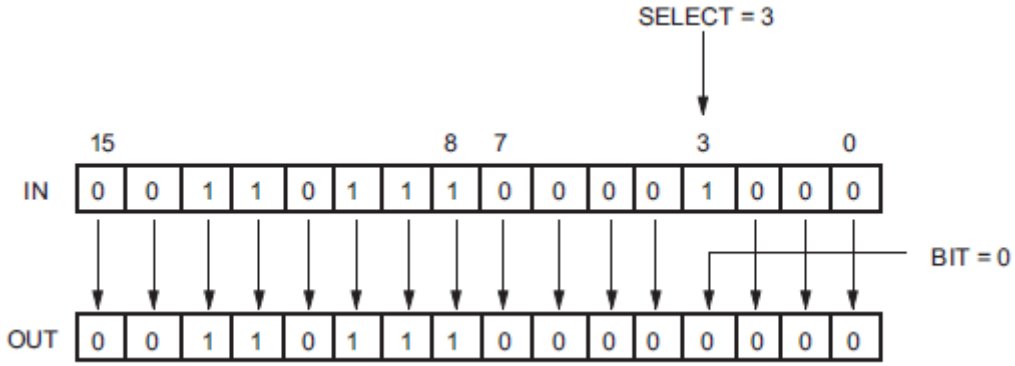
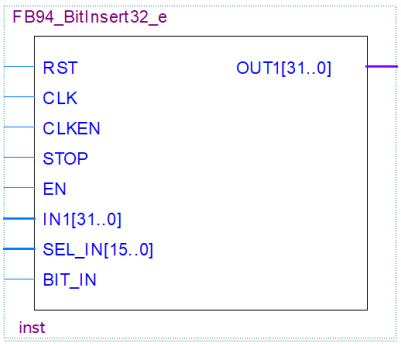


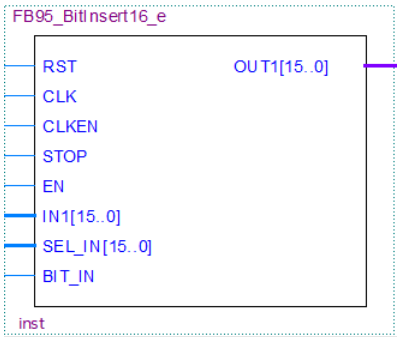
Figure 15: Example of FB94_BitInsert32, FB95_BitInsert16

Table 87: TM FAST function BitInsert32 (FB94_BitInsert32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
	SEL_IN	STD_LOGIC_ VECTOR	Input	Bit position to be replaced
	BIT_IN	STD_LOGIC	Input	Bit to be inserted
	OUT1	STD_LOGIC_ VECTOR	Output	Output of function

```
entity FB94_BitInsert32_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN : in  STD_LOGIC;
        STOP  : in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        IN1   : in  STD_LOGIC_VECTOR (31 downto 0);
        SEL_IN: in  STD_LOGIC_VECTOR (15 downto 0);
        BIT_IN: in  STD_LOGIC;
        OUT1  : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB94_BitInsert32_e;
```

Table 88: TM FAST function BitInsert16 (FB95_BitInsert16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	IN1	STD_LOGIC _VECTOR	Input	Input from which the bit is selected.
	SEL_IN	STD_LOGIC _VECTOR	Input	Bit position to be replaced
	BIT_IN	STD_LOGIC	Input	Bit to be inserted
	OUT1	STD_LOGIC _VECTOR	Output	Output of function

```
entity FB95_BitInsert16_e is
    Port (
        RST    : in  STD_LOGIC;
        CLK    : in  STD_LOGIC;
        CLKEN  : in  STD_LOGIC;
        STOP   : in  STD_LOGIC;
        EN     : in  STD_LOGIC;
        IN1    : in  STD_LOGIC_VECTOR (15 downto 0);
        SEL_IN: in  STD_LOGIC_VECTOR (15 downto 0);
        BIT_IN: in  STD_LOGIC;
        OUT1   : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB95_BitInsert16_e;
```

3.20 BitInsert, unlatched logic (FB94_BitInsert32_U, FB95_BitInsert32_U)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the output WORD or DWORD.

The bit selected within the input Word or DWORD is replaced. All other bits are transferred with no change. The unlatched (combinatorical) logic works without a clock, clock-enable and reset.

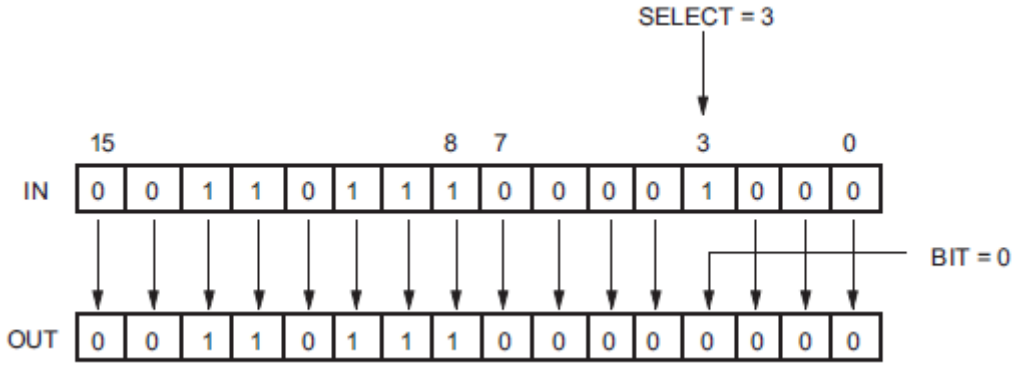
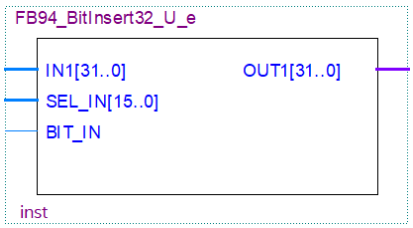


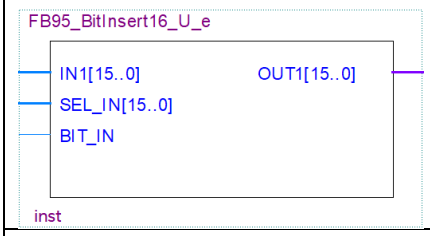
Figure 16: Example of FB94_BitInsert32_U, FB95_BitInsert16_U

Table 89: TM FAST function BitInsert32_U, unlatched logic (FB94_BitInsert32_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
	SEL_IN	STD_LOGIC_ VECTOR	Input	Bit position to be replaced
	BIT_IN	STD_LOGIC	Input	Bit to be inserted
	OUT1	STD_LOGIC_ VECTOR	Output	Output of function

```
entity FB94_BitInsert32_U_e is
    Port (
        IN1    : in  STD_LOGIC_VECTOR (31 downto 0);
        SEL_IN: in  STD_LOGIC_VECTOR (15 downto 0);
        BIT_IN: in  STD_LOGIC;
        OUT1   : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB94_BitInsert32_U_e;
```

Table 90: TM FAST function BitInsert16, unlatched logic (FB95_BitInsert16_U):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
	SEL_IN	STD_LOGIC_ VECTOR	Input	Bit position to be replaced
	BIT_IN	STD_LOGIC	Input	Bit to be inserted
	OUT1	STD_LOGIC_ VECTOR	Output	Output of function

```
entity FB95_BitInsert16_U_e is
    Port (
        IN1    : in  STD_LOGIC_VECTOR (15 downto 0);
        SEL_IN: in  STD_LOGIC_VECTOR (15 downto 0);
        BIT_IN: in  STD_LOGIC;
        OUT1   : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB95_BitInsert16_U_e;
```

3.21 FIFO (FB96_FIFO32, FB97_FIFO16)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the data width.

While EN is active, FIFO shift register store entries that are written into the FIFO box and represents the stored data upon request. When WR and EN are active, the data present at DATA_IN is written into the FIFO Box. The oldest entry in the FIFO box is available at DATA_OUT until it is discarded by activating RD_NEXT. The next to oldest entry then becomes the oldest entry. If the FIFO box is full (256 entries) then FULL becomes active. Any write operation that occurs while FULL is active will be discarded. EMPTY signals that the FIFO is empty (0 entries). DATA_OUT is indeterminate while EMPTY is active. ENTRIES indicates the number of entries contained in the FIFO box.

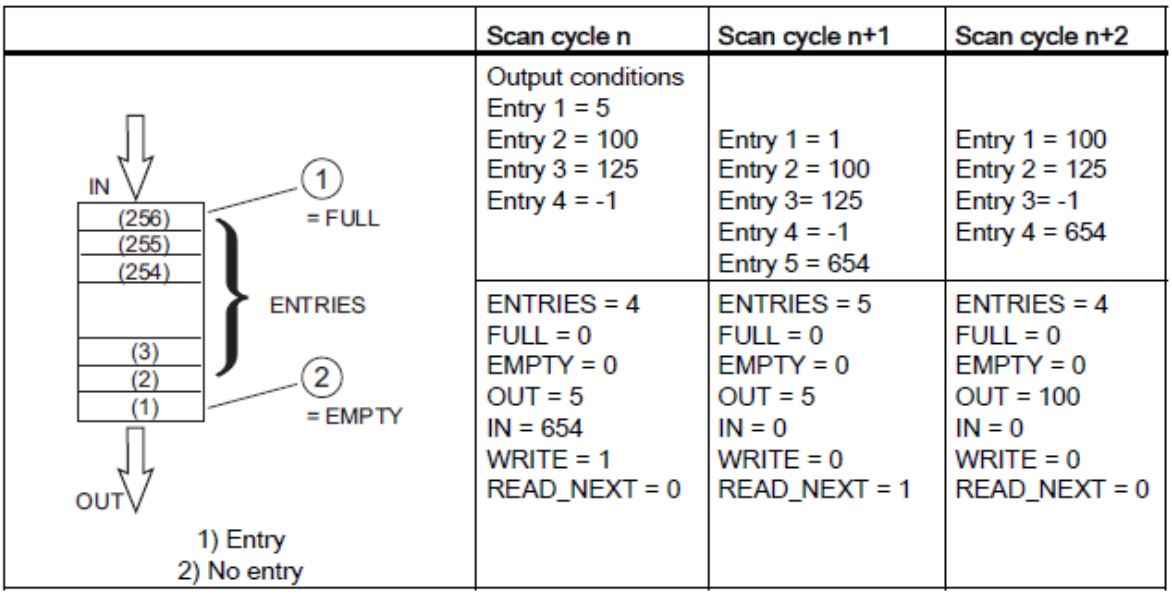
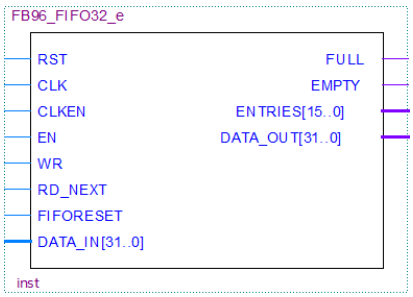


Figure 17: Example of FB96_FIFO32, FB97_FIFO16

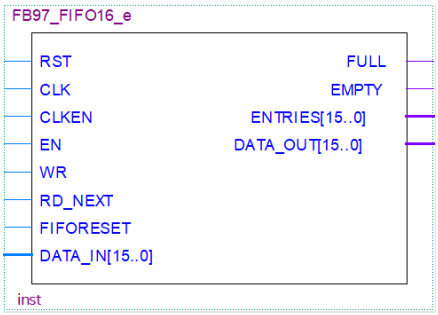
Table 91: TM FAST function FIFO32 (FB96_FIFO32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	WR	STD_LOGIC	Input	If active and FULL = 0, DATA_IN is written into the FIFO
	RD_NEXT	STD_LOGIC	Input	If active and EMPTY = 0, next entry is placed in DATA_OUT
	FIFORESET	STD_LOGIC	Input	FIFO entries are reset to 0
	DATA_IN	STD_LOGIC_ VECTOR	Input	Data input to FIFO
	FULL	STD_LOGIC	Output	1 indicates FIFO is full.

	EMPTY	STD_LOGIC	Output	1 indicate that FIFO is empty.
	ENTRIES	STD_LOGIC_ VECTOR	Output	Indicates the number of entries stored in the FIFO.
	DATA_OUT	STD_LOGIC_ VECTOR	Output	Data out from FIFO

```
ENTITY FB96_FIFO32_e IS
    PORT (
        RST       : IN  STD_LOGIC;
        CLK       : IN  STD_LOGIC;
        CLKEN     : IN  STD_LOGIC;
        EN        : IN  STD_LOGIC;
        WR        : IN  STD_LOGIC;
        RD_NEXT   : IN  STD_LOGIC;
        FIFORESET: IN  STD_LOGIC;
        DATA_IN   : IN  STD_LOGIC_VECTOR (31 DOWNTO 0);
        FULL      : OUT STD_LOGIC;
        EMPTY     : OUT STD_LOGIC;
        ENTRIES   : OUT STD_LOGIC_VECTOR (15 DOWNTO 0);
        DATA_OUT  : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
    );
END FB96_FIFO32_e;
```

Table 92: TM FAST function FIFO16 (FB97_FIFO16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	WR	STD_LOGIC	Input	If active and FULL = 0, DATA_IN is written into the FIFO
	RD_NEXT	STD_LOGIC	Input	If active and EMPTY = 0, next entry is placed in DATA_OUT
	FIFORESET	STD_LOGIC	Input	FIFO entries are reset to 0
	DATA_IN	STD_LOGIC_ VECTOR	Input	Data input to FIFO
	FULL	STD_LOGIC	Output	1 indicates FIFO is full.
	EMPTY	STD_LOGIC	Output	1 indicate that FIFO is empty.
	ENTRIES	STD_LOGIC_ VECTOR	Output	Indicates the number of entries stored in the FIFO.
	DATA_OUT	STD_LOGIC_ VECTOR	Output	Data out from FIFO

```
entity FB97_FIFO16_e is
    Port (
        RST       : in  STD_LOGIC;
        CLK       : in  STD_LOGIC;
        CLKEN     : in  STD_LOGIC;
        EN        : in  STD_LOGIC;
        WR        : in  STD_LOGIC;
        RD_NEXT   : in  STD_LOGIC;
        FIFORESET: in  STD_LOGIC;
        DATA_IN   : in  STD_LOGIC_VECTOR (15 downto 0);
        FULL      : out STD_LOGIC;
        EMPTY     : out STD_LOGIC;
        ENTRIES   : out STD_LOGIC_VECTOR (15 downto 0);
        DATA_OUT  : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB97_FIFO16_e;
```

3.22 LIFO (FB98_LIFO32, FB99_LIFO16)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version defined by the data width.

While EN is active, LIFO shift register store entries that are written into the LIFO box and represents the stored data upon request. When WR and EN are active, the data present at DATA_IN is written into the LIFO Box. The newest entry in the LIFO box is available at DATA_OUT until it is discarded by activating RD_NEXT. The next to newest entry then becomes the newest entry. If the LIFO box is full (256 entries) then FULL becomes active. Any write operation that occurs while FULL is active will be discarded. EMPTY signals that the LIFO is empty (0 entries). DATA_OUT is indeterminate while EMPTY is active. ENTRIES indicates the number of entries contained in the LIFO box.

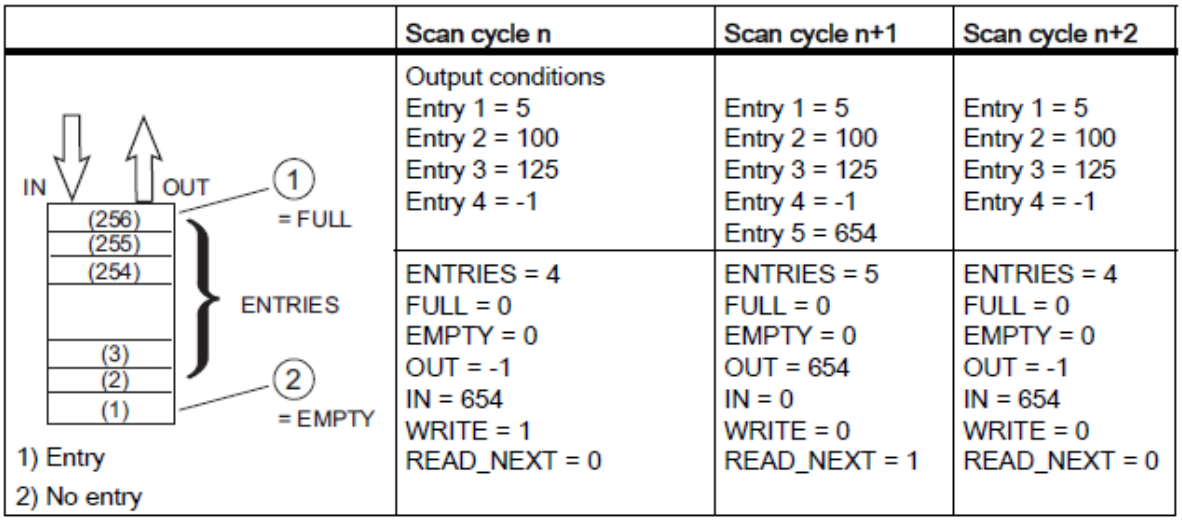
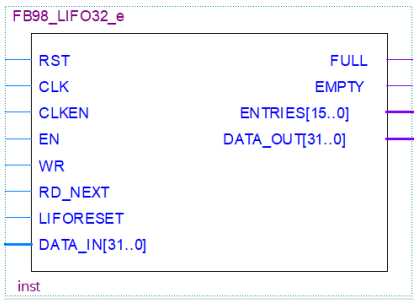


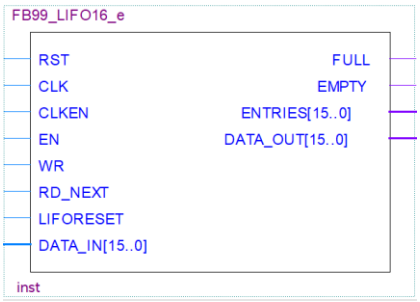
Figure 18: Example of FB98_LIFO32, FB99_LIFO16

Table 93: TM FAST function LIFO32 (FB98_LIFO32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	WR	STD_LOGIC	Input	If active and FULL = 0, DATA_IN is written into the LIFO
	RD_NEXT	STD_LOGIC	Input	If active and EMPTY = 0, next entry is placed in DATA_OUT
	LIFORESET	STD_LOGIC	Input	LIFO entries are reset to 0
	DATA_IN	STD_LOGIC_ VECTOR	Input	Data input to LIFO
	FULL	STD_LOGIC	Output	1 indicates LIFO is full.
	EMPTY	STD_LOGIC	Output	1 indicate that LIFO is empty.
	ENTRIES	STD_LOGIC_ VECTOR	Output	Indicates the number of entries stored in the LIFO.
	DATA_OUT	STD_LOGIC_ VECTOR	Output	Data out from LIFO

```
entity FB98_LIFO32_e is
    Port (
        RST       : in  STD_LOGIC;
        CLK       : in  STD_LOGIC;
        CLKEN     : in  STD_LOGIC;
        EN        : in  STD_LOGIC;
        WR        : in  STD_LOGIC;
        RD_NEXT   : in  STD_LOGIC;
        LIFORESET: in  STD_LOGIC;
        DATA_IN   : in  STD_LOGIC_VECTOR (31 downto 0);
        FULL      : out STD_LOGIC;
        EMPTY     : out STD_LOGIC;
        ENTRIES   : out STD_LOGIC_VECTOR (15 downto 0);
        DATA_OUT : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB98_LIFO32_e;
```

Table 94: TM FAST function LIFO16 (FB99_LIFO16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	WR	STD_LOGIC	Input	If active and FULL = 0, DATA_IN is written into the LIFO
	RD_NEXT	STD_LOGIC	Input	If active and EMPTY = 0, next entry is placed in DATA_OUT
	LIFORESET	STD_LOGIC	Input	LIFO entries are reset to 0
	DATA_IN	STD_LOGIC_ VECTOR	Input	Data input to LIFO
	FULL	STD_LOGIC	Output	1 indicates LIFO is full.
	EMPTY	STD_LOGIC	Output	1 indicate that LIFO is empty.
	ENTRIES	STD_LOGIC_ VECTOR	Output	Indicates the number of entries stored in the LIFO.
	DATA_OUT	STD_LOGIC_ VECTOR	Output	Data out from LIFO

```
entity FB99_LIFO16_e is
    Port (
        RST       : in  STD_LOGIC;
        CLK       : in  STD_LOGIC;
        CLKEN     : in  STD_LOGIC;
        EN        : in  STD_LOGIC;
        WR        : in  STD_LOGIC;
        RD_NEXT   : in  STD_LOGIC;
        LIFORESET: in  STD_LOGIC;
        DATA_IN   : in  STD_LOGIC_VECTOR (15 downto 0);
        FULL      : out STD_LOGIC;
        EMPTY     : out STD_LOGIC;
        ENTRIES   : out STD_LOGIC_VECTOR (15 downto 0);
        DATA_OUT  : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB99_LIFO16_e;
```

3.23 Multiply (FB100_FMMul32 and FB101_FMMul16)

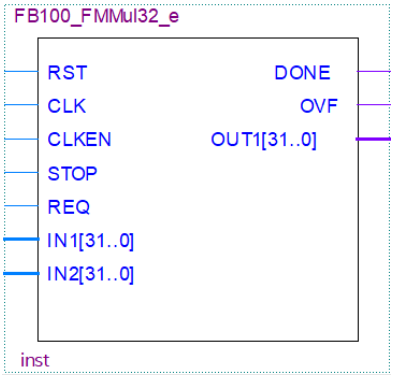
Description

This operation multiplies the value in IN1 by the value in IN2 and writes the result to the OUT1 output. The DONE output signals that the result is available.

For the 32-bit multiply, the valid range for inputs IN1, IN2 and output OUT1 is -2,147,483,648 to +2,147,483,647. The OVF output is set to logic 1 if an overflow occurs: otherwise, it is logic 0.

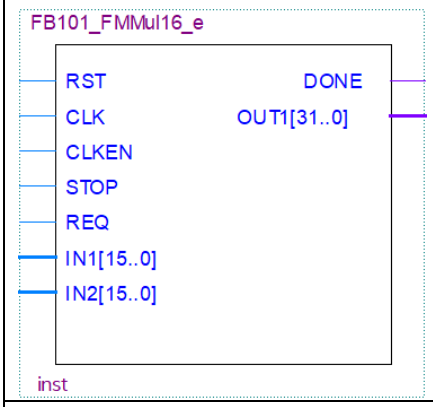
For the 16-bit multiply, the valid range for inputs IN1, IN2 and output OUT1 is -32768 to +32767. Since OUT1 is a DINT, there can be no overflow.

Table 95: TM FAST function FMMul32 (FB100_FMMul32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	REQ	STD_LOGIC	Input	Enable the multiply operation on a 0 to 1 change. It must remain 1 until DONE = 1; otherwise, the multiply terminates.
	IN1	STD_LOGIC_ VECTOR	Input	Input multiplicand
	IN2	STD_LOGIC_ VECTOR	Input	Input multiplier
	DONE	STD_LOGIC	Output	1 = result is available
	OVF	STD_LOGIC	Output	1, if multiplication results in overflow.
	OUT1	STD_LOGIC_ VECTOR	Output	Output value = IN1 x IN2

```
entity FB100_FMMul32_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        REQ  : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (31 downto 0);
        DONE : out STD_LOGIC;
        OVF  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0));
end FB100_FMMul32_e;
```

Table 96: TM FAST function FMMul16 (FB101_FMMul16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	REQ	STD_LOGIC	Input	Enable the multiply operation on a 0 to 1 change. It must remain 1 until DONE = 1; otherwise, the multiply terminates.
	IN1	STD_LOGIC_ VECTOR	Input	Input multiplicand
	IN2	STD_LOGIC_ VECTOR	Input	Input multiplier
	DONE	STD_LOGIC	Output	1 = result is available
	OUT1	STD_LOGIC_ VECTOR	Output	Output value = IN1 x IN2

```
entity FB101_FMMul16_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        STOP : in  STD_LOGIC;
        REQ  : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (15 downto 0);
        DONE : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0));
end FB101_FMMul16_e;
```

3.24 Divide (FB102_FMDiv32 and FB103_FMDiv16)

Description

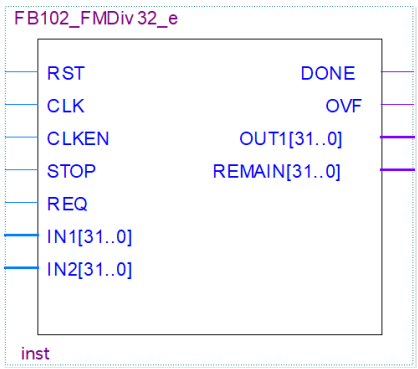
This operation divides the double integer value in IN1 by the value in IN2 (32-bit, 16-bit) and writes the result to the OUT1 output and the remainder to the REMAIN output. The DONE output signals that the result is available.

For the 32-bit multiply, the valid range for inputs IN1, IN2 and outputs OUT1 and REMAIN are -2,147,483,648 to +2,147,483,647.

For the 16-bit multiply, the valid range for input IN1 is -2,147,483,648 to +2,147,483,647, IN2 and outputs OUT1 and REMAIN is -32768 to +32767.

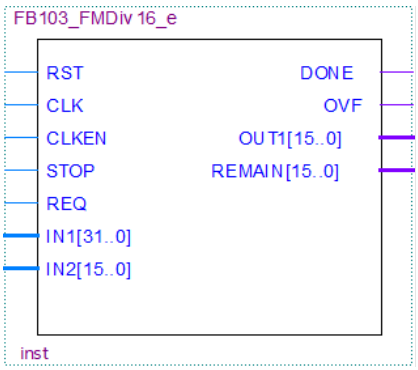
The OVF output is set to logic 1 if an overflow occurs: otherwise, it is logic 0. When OVF is 1, the OUT1 and REMAIN outputs are set to 0.

Table 97: TM FAST function FMDiv32 (FB102_FMDiv32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	REQ	STD_LOGIC	Input	Enable the divide operation on a 0 to 1 change. It must remain 1 until DONE = 1; otherwise, the divide terminates.
	IN1	STD_LOGIC_ VECTOR	Input	Input dividend
	IN2	STD_LOGIC_ VECTOR	Input	Input divisor
	DONE	STD_LOGIC	Output	1 = result is available
	OVF	STD_LOGIC	Output	1, if division results in overflow.
	OUT1	STD_LOGIC_ VECTOR	Output	Output value = IN1 ÷ IN2
	REMAIN	STD_LOGIC_ VECTOR	Output	Remainder of the division.

```
entity FB102_FMDiv32_e is
    Port (
          RST    : in  STD_LOGIC;
        CLK    : in  STD_LOGIC;
        CLKEN  : in  STD_LOGIC;
        STOP   : in  STD_LOGIC;
        REQ    : in  STD_LOGIC;
        IN1    : in  STD_LOGIC_VECTOR (31 downto 0);
        IN2    : in  STD_LOGIC_VECTOR (31 downto 0);
        DONE   : out STD_LOGIC;
        OVF    : out STD_LOGIC;
        OUT1   : out STD_LOGIC_VECTOR (31 downto 0);
        REMAIN: out STD_LOGIC_VECTOR (31 downto 0));
end FB102_FMDiv32_e;
```

Table 98: TM FAST function FMDiv16 (FB103_FMDiv16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	REQ	STD_LOGIC	Input	Enable the divide operation on a 0 to 1 change. It must remain 1 until DONE = 1; otherwise, the divide terminates.
	IN1	STD_LOGIC_ VECTOR	Input	Input dividend
	IN2	STD_LOGIC_ VECTOR	Input	Input divisor
	DONE	STD_LOGIC	Output	1 = result is available
	OVF	STD_LOGIC	Output	1, if division results in overflow.
	OUT1	STD_LOGIC_ VECTOR	Output	Output value = IN1 ÷ IN2
	REMAIN	STD_LOGIC_ VECTOR	Output	Remainder of the division.

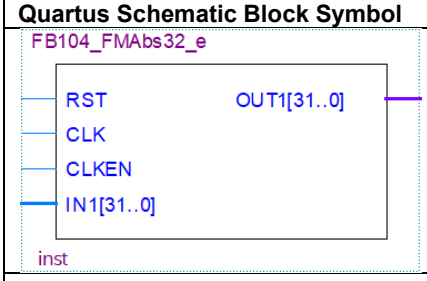
```
entity FB103_FMDiv16_e is
    Port (
        RST    : in  STD_LOGIC;
        CLK    : in  STD_LOGIC;
        CLKEN  : in  STD_LOGIC;
        STOP   : in  STD_LOGIC;
        REQ    : in  STD_LOGIC;
        IN1    : in  STD_LOGIC_VECTOR (31 downto 0);
        IN2    : in  STD_LOGIC_VECTOR (15 downto 0);
        DONE   : out STD_LOGIC;
        OVF    : out STD_LOGIC;
        OUT1   : out STD_LOGIC_VECTOR (15 downto 0);
        REMAIN: out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB103_FMDiv16_e;
```

3.25 Absolute value (FB104_FMAbs32 and FB105_FMAbs16)

Description

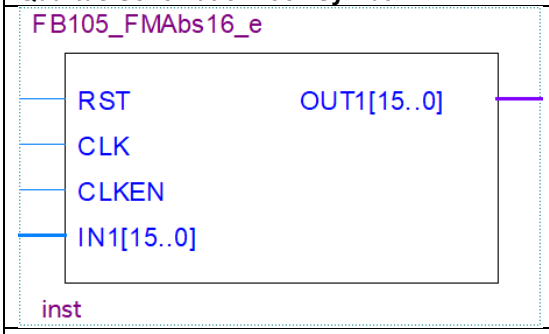
This operation writes the absolute value of the number supplied at the IN1 input to the OUT1 output. The absolute value of a number is the number without its sign.

Table 99: TM FAST function FMAbs32 (FB104_FMAbs32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_VECTOR	Input	Input value
	OUT1	STD_LOGIC_VECTOR	Output	Output value: Absolute value of the input value.

```
entity FB104_FMAbs32_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB104_FMAbs32_e;
```

Table 100: TM FAST function FMAbs16 (FB105_FMAbs16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC _VECTOR	Input	Input value
	OUT1	STD_LOGIC _VECTOR	Output	Output value: Absolute value of the input value.

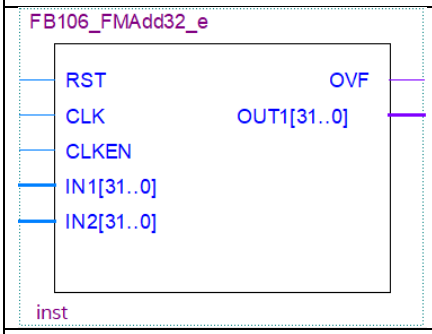
```
entity FB105_FMAbs16_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB105_FMAbs16_e;
```

3.26 Add (FB106_FMAdd32 and FB107_FMAdd16)

Description

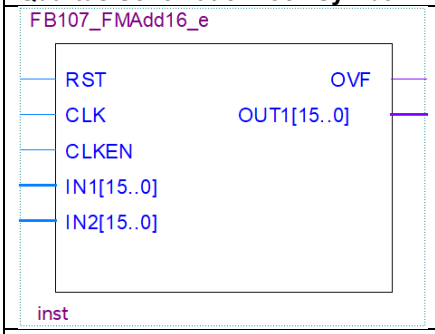
This operation adds the value in IN1 to the value in IN2 and writes the result to the OUT1 output. The OVF output is set to logic 1 if an overflow occurs: otherwise, it is logic 0.

Table 101: TM FAST function FMAdd32 (FB106_FMAdd32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_VECTOR	Input	Input value 1
	IN2	STD_LOGIC_VECTOR	Input	Input value 2
	OVF	STD_LOGIC	Output	1, if addition results in overflow.
	OUT1	STD_LOGIC_VECTOR	Output	Output value = IN1 + IN2

```
entity FB106_FMAdd32_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (31 downto 0);
        OVF  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB106_FMAdd32_e;
```

Table 102: TM FAST function FMAdd16 (FB107_FMAdd16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Input value 1
	IN2	STD_LOGIC_ VECTOR	Input	Input value 2
	OVF	STD_LOGIC	Output	1, if addition results in overflow.
	OUT1	STD_LOGIC_ VECTOR	Output	Output value = IN1 + IN2

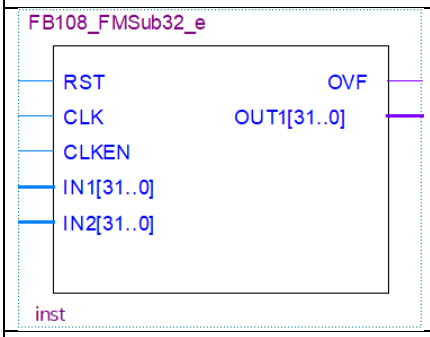
```
entity FB107_FMAdd16_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (15 downto 0);
        OVF  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB107_FMAdd16_e;
```

3.27 Subtract (FB108_FMSub32 and FB109_FMSub16)

Description

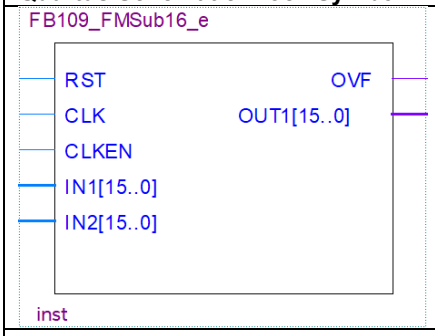
This operation subtracts the value in IN2 from the value in IN1 and writes the result to the OUT1 output. The OVF output is set to logic 1 if an overflow occurs: otherwise, it is logic 0.

Table 103: TM FAST function FMSub32 (FB108_FMSub32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_VECTOR	Input	Input value 1
	IN2	STD_LOGIC_VECTOR	Input	Input value 2
	OVF	STD_LOGIC	Output	1, if addition results in overflow.
	OUT1	STD_LOGIC_VECTOR	Output	Output value = IN1 - IN2

```
entity FB108_FMSub32_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (31 downto 0);
        OVF  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB108_FMSub32_e;
```

Table 104: TM FAST function FMSub16 (FB109_FMSub16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Input value 1
	IN2	STD_LOGIC_ VECTOR	Input	Input value 2
	OVF	STD_LOGIC	Output	1, if addition results in overflow.
	OUT1	STD_LOGIC_ VECTOR	Output	Output value = IN1 - IN2

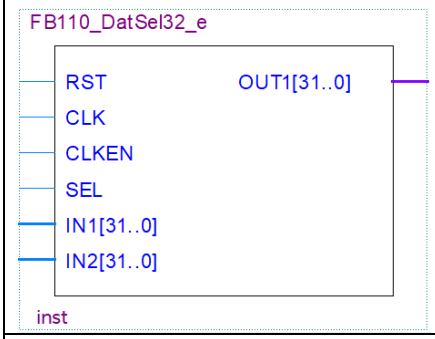
```
entity FB109_FMSub16_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (15 downto 0);
        OVF  : out STD_LOGIC;
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB109_FMSub16_e;
```

3.28 Data selector (FB110_DatSel32 and FB111_DatSel16)

Description

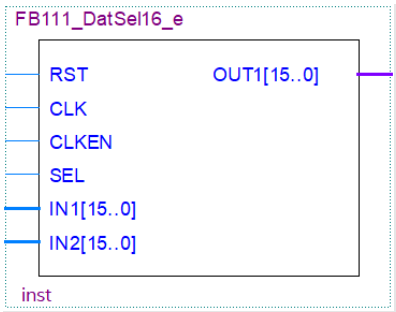
This operation provides the function of a 2-to-1 multiplexer by copying the value at the IN1 input to output OUT1 if input SEL is logic 0 or copying the value at the IN2 input to output OUT1 if input SEL is logic 1. An N-to-1 multiplexer can be created by cascading multiple DatSel operations.

Table 105: TM FAST function DatSel32 (FB110_DatSel32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Input value 1
	IN2	STD_LOGIC_ VECTOR	Input	Input value 2
	SEL	STD_LOGIC	Input	If 0, the value of IN1 is copied to the output. If 1, the value of IN2 is copied to the output.
	OUT1	STD_LOGIC_ VECTOR	Output	Output value: IN1 if SEL = 0 IN2 if SEL = 1

```
entity FB110_DatSel32_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        SEL  : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (31 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB110_DatSel32_e;
```

Table 106: TM FAST function DatSel16 (FB111_DatSel16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC_ VECTOR	Input	Input value 1
	IN2	STD_LOGIC_ VECTOR	Input	Input value 2
	SEL	STD_LOGIC	Input	If 0, the value of IN1 is copied to the output. If 1, the value of IN2 is copied to the output.
	OUT1	STD_LOGIC_ VECTOR	Output	Output value: IN1 if SEL = 0 IN2 if SEL = 1

```
entity FB111_DatSel16_e is
    Port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        SEL  : in  STD_LOGIC;
        IN1  : in  STD_LOGIC_VECTOR (15 downto 0);
        IN2  : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1 : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB111_DatSel16_e;
```

3.29 Binary scaler (FB112_BiScale)

Description

This operation provides a way of producing a series of output pulses at half the rate of the input pulses.

Each rising edge at input CIN inverts the output OUT1 effectively dividing the frequency of the input by two, as shown in the figure below.

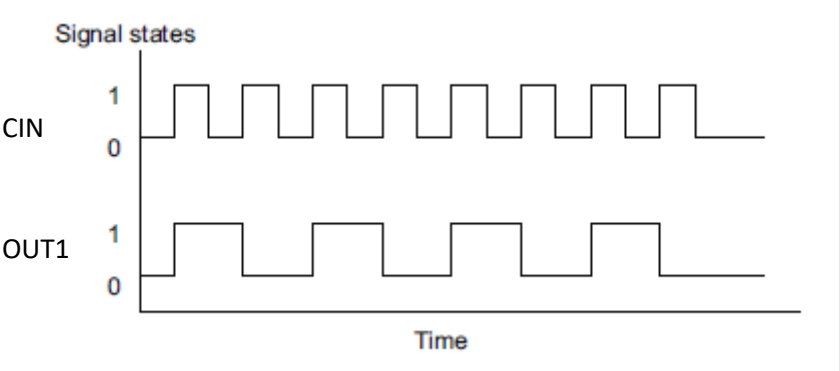
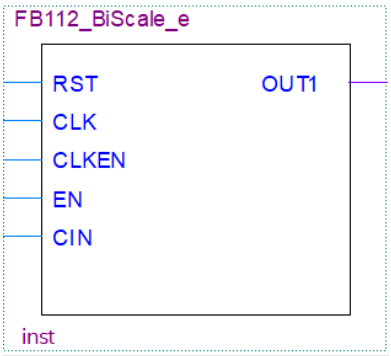


Figure 19: Timing Diagram for Binary Scaler (FB112_BiScale)

Table 107: TM FAST function Period32 (FB80_Period32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	CIN	STD_LOGIC	Input	Input to be converted.
	OUT1	STD_LOGIC	Output	Output of the function.

```
entity FB112_BiScale_e is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        CIN   : in  STD_LOGIC;
        OUT1  : out STD_LOGIC
    );
end FB112_BiScale_e;
```

3.30 Pulse timer (FB113_TP32, FB116_TP16)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version.

Pulse timers generate a pulse with the length PT.

Connect the CLK100k input to the **ClkTic100k** logic which generates the 100kHz reference signal.

A rising edge at input IN1 starts the pulse. Output OUT1 remains set for the time PT regardless of changes in the input signal (in other words even when the IN1 input changes back from 1 to 0 before the time PT has expired). The ET output provides the time for which output OUT1 has already been set. The maximum value of the ET output is the value of the PT input. Output ET is reset when input IN1 changes to 0; however, not before the time PT has expired.

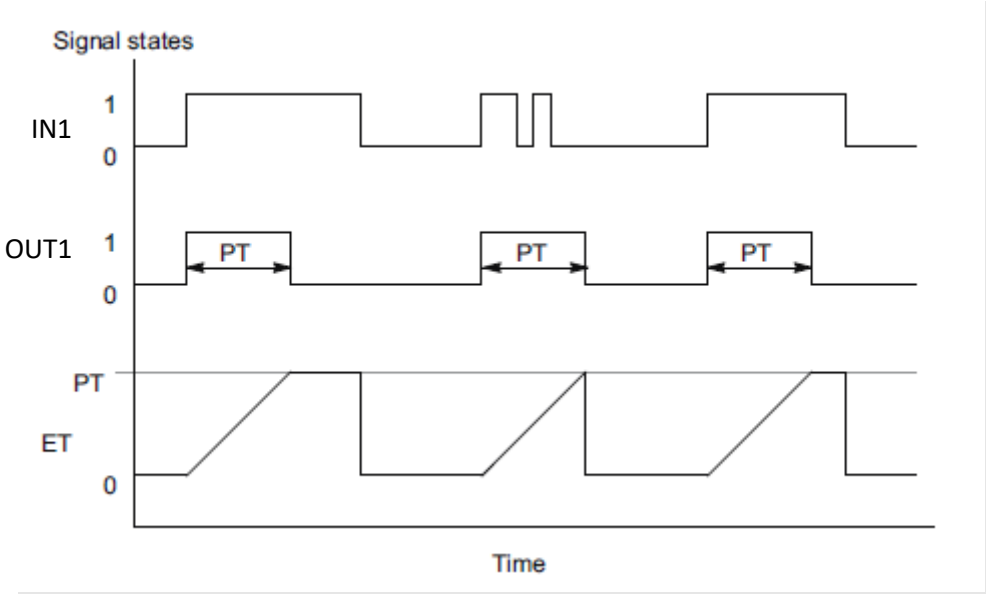
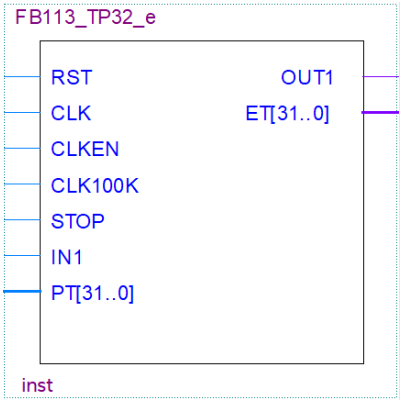


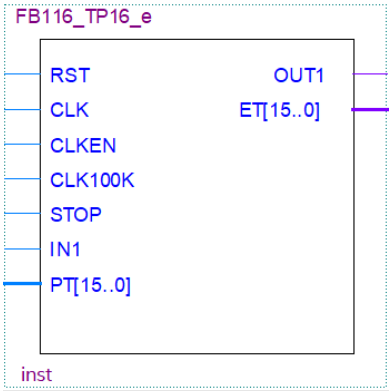
Figure 20: Timing Diagram for Pulse Timer (FB113_TP32, FB116_TP16)

Table 108: TM FAST function TP32 FB113_TP32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC	Input	Start input
	PT	STD_LOGIC_VECTOR	Input	Duration of the pulse in 10usec units. PT must be a positive constant.
	CLK100k	STD_LOGIC	Input	Connect to the **ClkTic100k** logic
	OUT1	STD_LOGIC	Output	Status of the time.
	ET	STD_LOGIC_VECTOR	Output	Elapsed time.

```
entity FB113_TP32_e is
    Port (
        RST     : in  STD_LOGIC;
        CLK     : in  STD_LOGIC;
        CLKEN   : in  STD_LOGIC;
        CLK100K: in  STD_LOGIC;
        STOP    : in  STD_LOGIC;
        IN1     : in  STD_LOGIC;
        PT      : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1    : out STD_LOGIC;
        ET      : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB113_TP32_e;
```

Table 109: TM FAST function TP16 (FB116_TP16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC	Input	Start input
	PT	STD_LOGIC _VECTOR	Input	Duration of the pulse in 10usec units. PT must be a positive constant.
	CLK100k	STD_LOGIC	Input	Connect to the **ClkTic100k** logic
	OUT1	STD_LOGIC	Output	Status of the time.
	ET	STD_LOGIC _VECTOR	Output	Elapsed time.

```
entity FB116_TP16_e is
    Port (
        RST     : in  STD_LOGIC;
        CLK     : in  STD_LOGIC;
        CLKEN   : in  STD_LOGIC;
        CLK100K: in  STD_LOGIC;
        STOP    : in  STD_LOGIC;
        IN1     : in  STD_LOGIC;
        PT      : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1    : out STD_LOGIC;
        ET      : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB116_TP16_e;
```

3.31 On-delay timer (FB114_TOn32, FB117_TOn16)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version.

On-delay timers delay a rising edge by the time PT.

Connect the CLK100k input to the **ClkTic100k** logic which generates the 100kHz reference signal.

A rising edge at input IN1 causes a rising edge at output OUT1 after the time PT has expired. OUT1 then remains set until the IN1 input changes to 0 again. If the IN1 input changes to 0 before the time PT expires, the output OUT1 remains 0.

The ET output provides the time that has passed since the last rising edge at the IN1 input. Its maximum value is the value of the PT input. ET is reset when the IN1 input changes to 0.

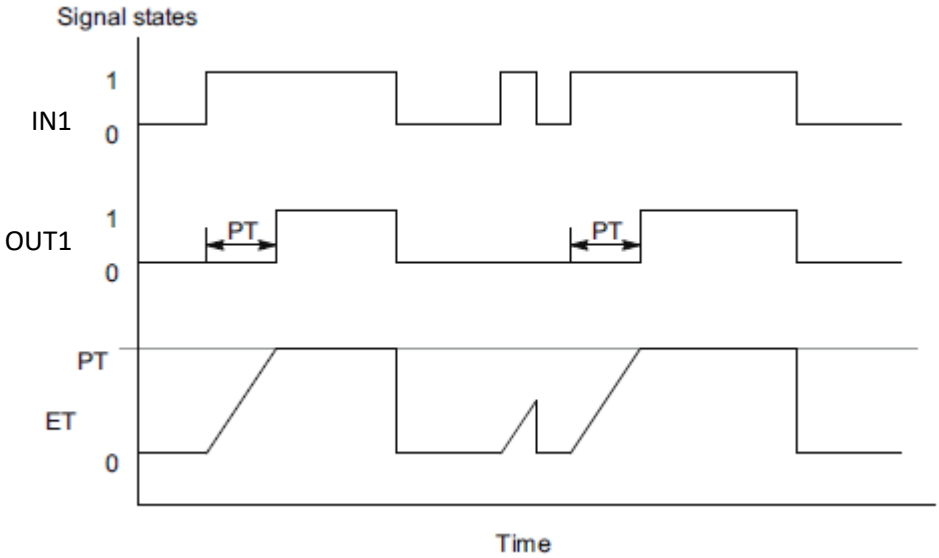
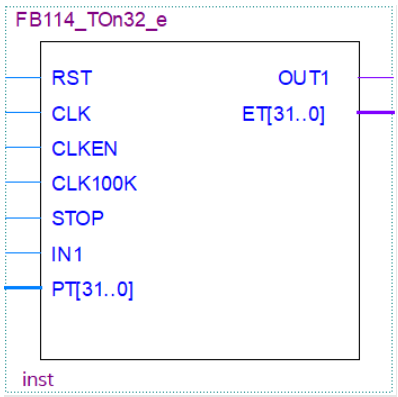


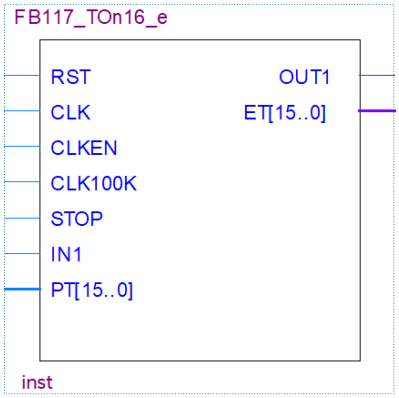
Figure 21: Timing Diagram for On-Delay Timer (FB114_TOn32, FB117_TOn16)

Table 110: TM FAST function TOn32 FB114_TOn32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC	Input	Start input
	PT	STD_LOGIC _VECTOR	Input	Duration of the pulse in 10usec units. PT must be a positive constant.
	CLK100k	STD_LOGIC	Input	Connect to the **ClkTic100k** logic
	OUT1	STD_LOGIC	Output	Status of the time.
	ET	STD_LOGIC _VECTOR	Output	Elapsed time.

```
entity FB114_TOn32_e is
    Port (
        RST     : in  STD_LOGIC;
        CLK     : in  STD_LOGIC;
        CLKEN   : in  STD_LOGIC;
        CLK100K: in  STD_LOGIC;
        STOP    : in  STD_LOGIC;
        IN1     : in  STD_LOGIC;
        PT      : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1    : out STD_LOGIC;
        ET      : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB114_TOn32_e;
```

Table 111: TM FAST function Ton16 (FB117_TOn16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC	Input	Start input
	PT	STD_LOGIC_ VECTOR	Input	Duration of the pulse in 10usec units. PT must be a positive constant.
	CLK100k	STD_LOGIC	Input	Connect to the **ClkTic100k** logic
	OUT1	STD_LOGIC	Output	Status of the time.
	ET	STD_LOGIC_ VECTOR	Output	Elapsed time.

```
entity FB117_TOn16_e is
    Port (
        RST     : in  STD_LOGIC;
        CLK     : in  STD_LOGIC;
        CLKEN   : in  STD_LOGIC;
        CLK100K: in  STD_LOGIC;
        STOP    : in  STD_LOGIC;
        IN1     : in  STD_LOGIC;
        PT      : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1    : out STD_LOGIC;
        ET      : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB117_TOn16_e;
```

3.32 Off-Delay timer (FB115_TOf32, FB118_TOf16)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version.

Off-delay timers delay a falling edge by the time PT.

Connect the CLK100k input to the **ClkTic100k** logic which generates the 100kHz reference signal.

A rising edge at input IN1 causes a rising edge at output OUT1. A falling edge at the IN1 input causes a falling edge at output OUT1 by the time PT. If the IN1 input changes back to 1 before the time PT has expired, output OUT1 remains set to 1. The ET output provides the time that has elapsed since the last falling edge at the IN1 input. Its maximum value is, however, the value at the PT input. ET is reset when the IN1 input changes to 1.

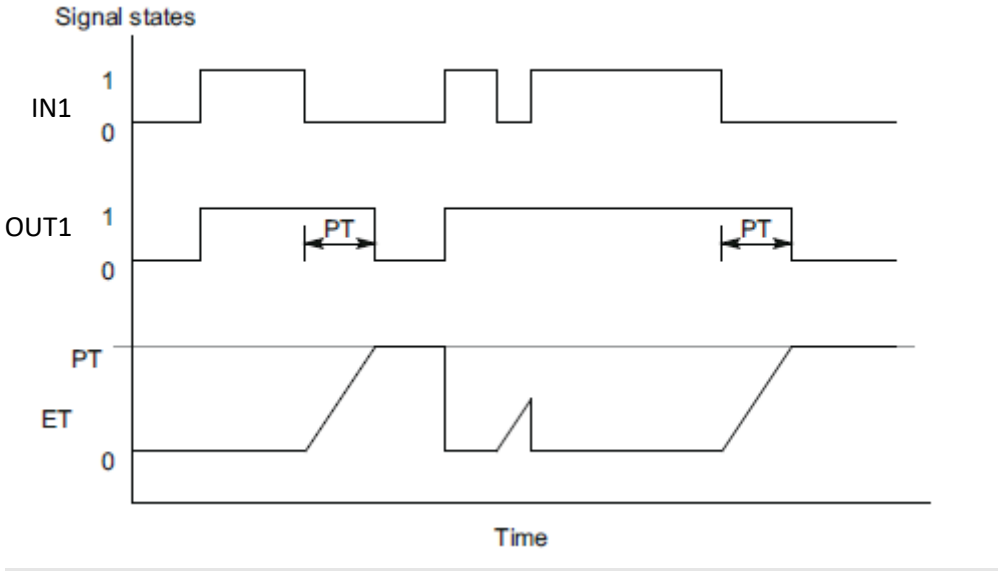
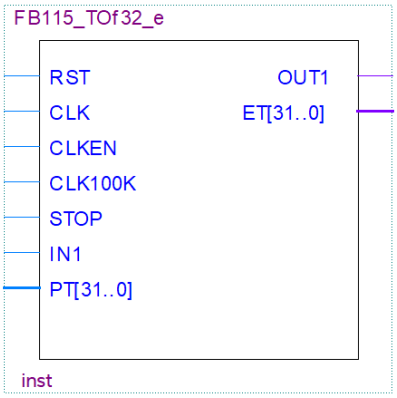


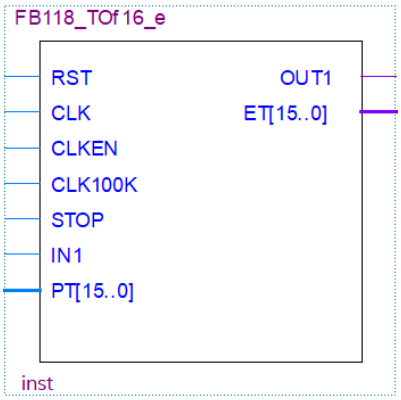
Figure 22: Timing Diagram for Off-Delay Timer (FB115_TOf32, FB118_TOf16)

Table 112: TM FAST function TOf32 FB115_TOf32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC	Input	Start input
	PT	STD_LOGIC_VECTOR	Input	Duration of the pulse in 10usec units. PT must be a positive constant.
	CLK100k	STD_LOGIC	Input	Connect to the **ClkTic100k** logic
	OUT1	STD_LOGIC	Output	Status of the time.
	ET	STD_LOGIC_VECTOR	Output	Elapsed time.

```
entity FB115_TOf32_e is
    Port (
        RST     : in  STD_LOGIC;
        CLK     : in  STD_LOGIC;
        CLKEN   : in  STD_LOGIC;
        CLK100K: in  STD_LOGIC;
        STOP    : in  STD_LOGIC;
        IN1     : in  STD_LOGIC;
        PT      : in  STD_LOGIC_VECTOR (31 downto 0);
        OUT1    : out STD_LOGIC;
        ET      : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB115_TOf32_e;
```

Table 113: TM FAST function TOf16 (FB118_TOf16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC	Input	Start input
	PT	STD_LOGIC_VECTOR	Input	Duration of the pulse in 10usec units. PT must be a positive constant.
	CLK100k	STD_LOGIC	Input	Connect to the **ClkTic100k** logic
	OUT1	STD_LOGIC	Output	Status of the time.
	ET	STD_LOGIC_VECTOR	Output	Elapsed time.

```
entity FB118_TOf16_e is
    Port (
        RST     : in  STD_LOGIC;
        CLK     : in  STD_LOGIC;
        CLKEN   : in  STD_LOGIC;
        CLK100K: in  STD_LOGIC;
        STOP    : in  STD_LOGIC;
        IN1     : in  STD_LOGIC;
        PT      : in  STD_LOGIC_VECTOR (15 downto 0);
        OUT1    : out STD_LOGIC;
        ET      : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB118_TOf16_e;
```

3.33 Clock pulse generator (FB119_CP_Gen)

Description

This operation allows you to output a pulse at a specified frequency from less than 1 Hz to a maximum of 50 kHz.

Connect the CLK100k input to the **ClkTic100k** logic which generates the 100kHz reference signal.

When the signal state at the ENABLE input is 1, a clock pulse is generated at the Q output, as shown in the figure below. The output frequency is specified by inverting the value of the PERIOD input that is an unsigned integer multiplied by 20 usec.

The frequency is equal to 50,000 ÷ PERIOD.

PERIOD is equal to 50,000 divided by the desired frequency.

Example:

- When PERIOD = w#16#C350, a frequency of 1 Hz is output.
- When PERIOD = W#16#1, a frequency of 50 kHz is output.

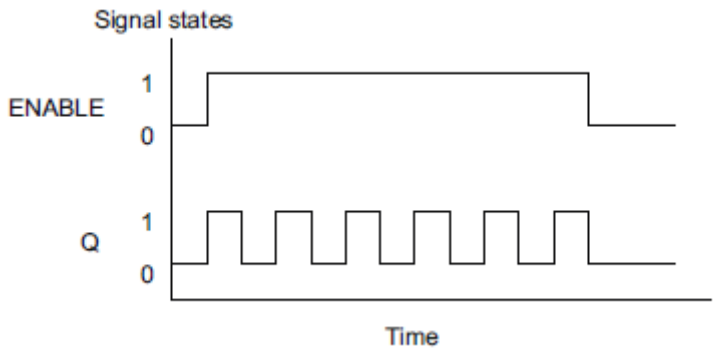
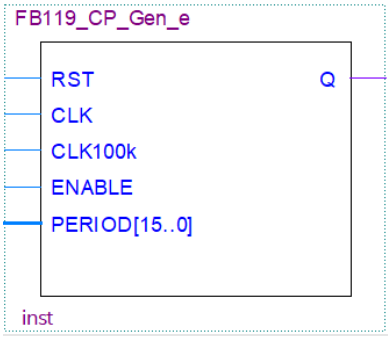


Figure 23: Timing Diagram for Clock Pulse Generator (FB119_CP_Gen)

Table 114: TM FAST function CP_Gen (FB119_CP_Gen)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	ENABLE	STD_LOGIC	Input	Enable input
	PERIOD	STD_LOGIC _VECTOR	Input	The number of 20usec steps in the period.
	CLK100k	STD_LOGIC	Input	Connect to the **ClkTic100k** logic
	Q	STD_LOGIC	Output	Status of the time

```
entity FB119_CP_Gen_e is
    port (
        RST     : in  STD_LOGIC;
        CLK     : in  STD_LOGIC;
        CLK100k: in  STD_LOGIC;
        ENABLE : in  STD_LOGIC;
        PERIOD : in  STD_LOGIC_VECTOR (15 downto 0);
        Q       : out STD_LOGIC
    );
end FB119_CP_Gen_e;
```

3.34 Up/down counter (FB120_CTUD32, FB123_CTUD16)

Description

This operation is available in two versions: As a 16-bit version and a 32-bit version.

The count value is changed by a rising edge as follows:

- The counted value is incremented by 1 on a rising edge at the CU input. If the count value reaches the upper limit, it is no longer incremented.
- The counted value is decremented by 1 on a rising edge of the CD input. If the count value reaches the lower limit, it is no longer decremented.

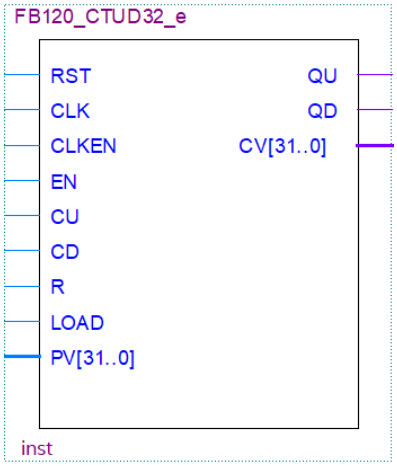
If there is a rising edge at both the CU and CD input in one cycle, the counter retains its current value.

A signal level 1 at the LOAD input presets the counter to the value PV regardless of the values at the CU and CD inputs.

A signal level 1 at the R input resets the counter to the value 0 regardless of the values at the CU, CD, and LOAD inputs.

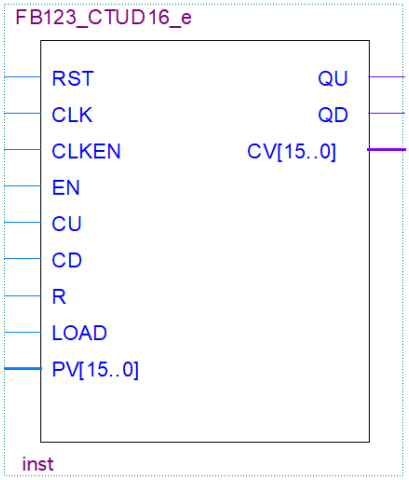
The QU outputs indicates whether the current counted value is greater than or equal to the preset value PV. The QD output indicates whether the value is less than or equal to 0.

Table 115: TM FAST function CTUD32 (FB120_CTUD32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	CU	STD_LOGIC	Input	Count up input.
	CD	STD_LOGIC	Input	Count down input
	R	STD_LOGIC	Input	Reset input. R is dominant over CU.
	LOAD	STD_LOGIC	Input	Load input. LOAD input is dominate over CD.
	PV	STD_LOGIC_ VECTOR	Input	Preset value. The counter is preset to PV when the signal level at the LOAD input is 1.
	QU	STD_LOGIC	Output	Status of the counter: QU has the following value: 1 if CV ≥ PV 0 in all other situations
	QD	STD_LOGIC	Output	Status of the counter: QD has the following value: 1 if CV ≤ 0 0 in all other situations
	CV	STD_LOGIC_ VECTOR	Output	Counter value

```
entity FB120_CTUD32_e is
    port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        CU    : in  STD_LOGIC;
        CD    : in  STD_LOGIC;
        R     : in  STD_LOGIC;
        LOAD  : in  STD_LOGIC;
        PV    : in  STD_LOGIC_VECTOR (31 downto 0);
        QU    : out STD_LOGIC;
        QD    : out STD_LOGIC;
        CV    : out STD_LOGIC_VECTOR (31 downto 0)
    );
end FB120_CTUD32_e;
```

Table 116: TM FAST function CTUD16 (FB123_CTUD16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	CU	STD_LOGIC	Input	Count up input.
	CD	STD_LOGIC	Input	Count down input
	R	STD_LOGIC	Input	Reset input. R is dominant over CU.
	LOAD	STD_LOGIC	Input	Load input. LOAD input is dominate over CD.
	PV	STD_LOGIC_VECTOR	Input	Preset value. The counter is preset to PV when the signal level at the LOAD input is 1.
	QU	STD_LOGIC	Output	Status of the counter: QU has the following value: 1 if CV ≥ PV 0 in all other situations
	QD	STD_LOGIC	Output	Status of the counter: QD has the following value: 1 if CV ≤ 0 0 in all other situations
	CV	STD_LOGIC_VECTOR	Output	Counter value

```
entity FB123_CTUD16_e is
    port (
        RST  : in  STD_LOGIC;
        CLK  : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        EN   : in  STD_LOGIC;
        CU   : in  STD_LOGIC;
        CD   : in  STD_LOGIC;
        R    : in  STD_LOGIC;
        LOAD : in  STD_LOGIC;
        PV   : in  STD_LOGIC_VECTOR (15 downto 0);
        QU   : out STD_LOGIC;
        QD   : out STD_LOGIC;
        CV   : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB123_CTUD16_e;
```

3.35 Up counter (FB121_CTU16)

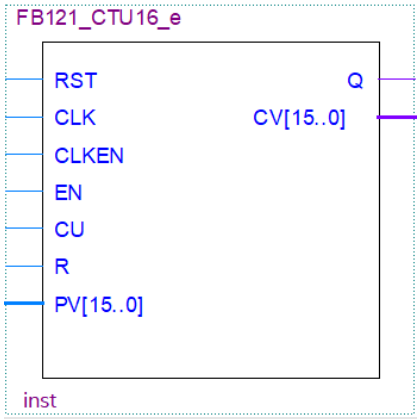
Description

The count value is incremented by 1 on a rising edge at the CU input. If the count value reaches the upper limit, it is no longer incremented. Each subsequent rising edge at the CU input no longer has an effect.

A signal level 1 at the R input resets the counter to the value 0 regardless of the value at the CU.

The Q outputs indicates whether the current counted value is greater than or equal to the preset value PV.

Table 117: TM FAST function CTU16 (FB121_CTU16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	CU	STD_LOGIC	Input	Count up input.
	R	STD_LOGIC	Input	Reset input. R is dominant over CU.
	PV	STD_LOGIC_ VECTOR	Input	Preset value. The counter is preset to PV when the signal level at the LOAD input is 1.
	Q	STD_LOGIC	Output	Status of the counter: QU has the following value: 1 if CV ≥ PV 0 in all other situations
	CV	STD_LOGIC_ VECTOR	Output	Counter value

```
entity FB121_CTU16_e is
    port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        CU    : in  STD_LOGIC;
        R     : in  STD_LOGIC;
        PV    : in  STD_LOGIC_VECTOR (15 downto 0);
        Q     : out STD_LOGIC;
        CV    : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB121_CTU16_e;
```

3.36 Down counter (FB122_CTD16)

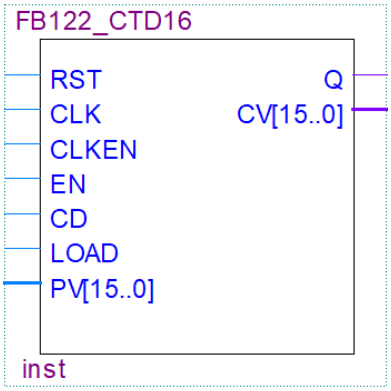
Description

The count value is decremented by 1 on a rising edge of the CD input. If the count value reaches the lower limit of -32768, it is no longer decremented. Any subsequent rising edge at the CD input no longer has an effect.

A signal level 1 at the LOAD input presets the counter to the value PV regardless of the value at the CD input.

The Q output indicates whether the value is less than or equal to 0

Table 118: TM FAST function CTD16 (FB122_CTD16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
	CD	STD_LOGIC	Input	Count down input
	LOAD	STD_LOGIC	Input	Load input. The LOAD input overrides the CD input.
	PV	STD_LOGIC_VECTOR	Input	Preset value. The counter is preset to PV when the signal level at the LOAD input is 1.
	Q	STD_LOGIC	Output	Status of the counter: QD has the following value: 1 if CV ≤ 0 0 in all other situations
	CV	STD_LOGIC_VECTOR	Output	Counter value

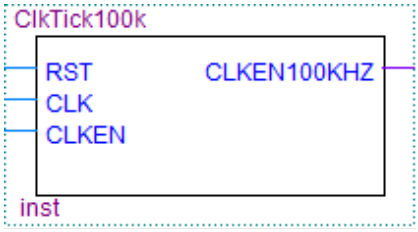
```
entity FB122_CTD16 is
    Port (
        RST   : in  STD_LOGIC;
        CLK   : in  STD_LOGIC;
        CLKEN: in  STD_LOGIC;
        EN    : in  STD_LOGIC;
        CD    : in  STD_LOGIC;
        LOAD  : in  STD_LOGIC;
        PV    : in  STD_LOGIC_VECTOR (15 downto 0);
        Q     : out STD_LOGIC;
        CV    : out STD_LOGIC_VECTOR (15 downto 0)
    );
end FB122_CTD16;
```

3.37 ClkTic100k

Description

The ClkTic100k generates the 100kHz Clock-tic signal which will be needed by some logic blocks.

Table 119: TM FAST ClkTic100k

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	CLKEN100KHZ	STD_LOGIC	Output	Clock-tic signal (100kHz) needed by other logic blocks.

```
entity ClkTick100k is
    port (
        RST         : in  STD_LOGIC;
        CLK         : in  STD_LOGIC;
        CLKEN       : in  STD_LOGIC;
        CLKEN100KHZ: OUT STD_LOGIC
    );
end ClkTick100k;
```

4 TM FAST library: Encoder blocks of the FM 352-5

The following table lists the symbolic names and description for each TM FAST encoder block library component.

Table 120: TM FAST encoder blocks

Symbolic Name	Description
Inc_Enc	Incremental Encoder interface, 10…64-Bit – 24V HTL, Pulse/Dir or RS422 ABN
SSI	SSI Encoder interface, 10…64-Bit
SSI_ListenIn	SSI ListenIn Encoder interface, 10…64-Bit

4.1 Incremental Encoder (Inc_Enc)

Description

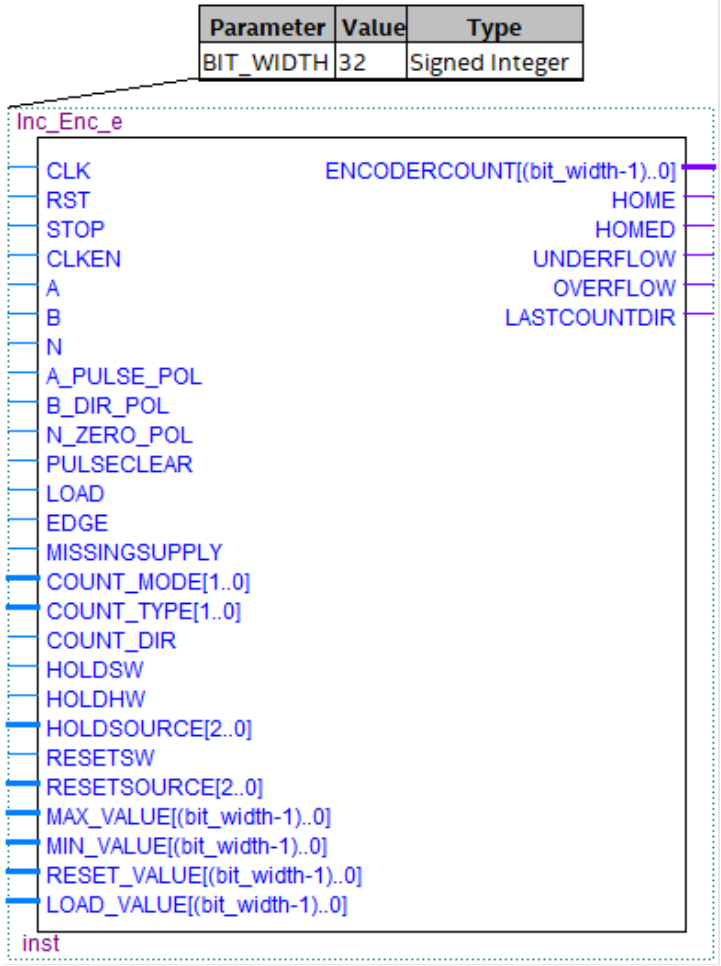
The incremental encoders inputs (A/B/N) can be driven by standard 24V inputs or by RS422/485 differential inputs for increased speed.

Table 121: TM FAST Incremental Encoder

Parameter	Data type	Direction	Description
BIT_WIDTH	Integer	Generic	Number of bits (range 10 to 64 Bits)
STOP	STD_LOGIC	Input	If active, resets HOMED output and sets Encoder counter value to zero
A	STD_LOGIC	Input	Encoder A / Pulse Input
B	STD_LOGIC	Input	Encoder B / Direction Input
N	STD_LOGIC	Input	Encoder N: null or zero pulse
A_PULSE_POL	STD_LOGIC	Input	Invert A Input
B_DIR_POL	STD_LOGIC	Input	Invert B Input
N_ZERO_POL	STD_LOGIC	Input	Invert N Input
PULSECLEAR	STD_LOGIC	Input	Reset Counter value to 0
LOAD	STD_LOGIC	Input	Load input
EDGE	STD_LOGIC	Input	Edge input – latch hold/reset signals 0 -> Reset is dominant. If Hold and Reset are activated simultaneously, the count is reset, and then held. 1 -> Hold is dominant. If Hold and Reset are activated simultaneously, no reset occurs. If Hold is removed first, the count is reset. If Reset is removed before Hold, no reset will occur.
MISSINGSUPPLY	STD_LOGIC	Input	If active, resets HOME/HOMED outputs
COUNT_MODE	STD_LOGIC_VECTOR	Input	Counting Mode • 0 -> single count • 1 -> double count • 2 -> quad count • 3 -> pulse/direction
COUNT_TYPE	STD_LOGIC_VECTOR	Input	Counting Type • 0 -> continuous count • 1 -> periodic count • 2 -> single count
COUNT_DIR	STD_LOGIC	Input	Main count direction inverted
HOLDSW	STD_LOGIC	Input	SW-Hold input
HOLDHW	STD_LOGIC	Input	HW-Hold input
HOLDSOURCE	STD_LOGIC_VECTOR	Input	Hold source • 1 -> HW Hold • 2 -> SW Hold • 3 -> HW and SW Hold • 4 -> HW or SW Hold • Other -> none
RESETSW	STD_LOGIC	Input	SW-Reset input
RESETSOURCE	STD_LOGIC_VECTOR	Input	Reset source • 1 -> HW Reset • 2 -> SW Reset • 3 -> HW and SW Reset • 4 -> HW or SW Reset • Other -> none
MAX_VALUE	STD_LOGIC_VECTOR	Input	Count range max value

MIN_VALUE	STD_LOGIC_VECTOR	Input	Count range min value
RESET_VALUE	STD_LOGIC_VECTOR	Input	Reset value
LOAD_VALUE	STD_LOGIC_VECTOR	Input	Load value
ENCODERCOUNT	STD_LOGIC_VECTOR	Output	Encoder counter value
HOME	STD_LOGIC	Output	Indicates that the encoder is currently at the home position, which is defined as a reset of the counterl
HOMED	STD_LOGIC	Output	Indicates that the encoder has reached its home position since the last power up, and that position data is accurate (the encoder is synchronized).
UNDERFLOW	STD_LOGIC	Output	Indicates that the counter has reached the minimum value and exceeded it (decremented by 1).
OVERFLOW	STD_LOGIC	Output	Indicates that the counter has reached the maximum value and exceeded it (incremented by 1).
LASTCOUNTDIR	STD_LOGIC	Output	Indicates the direction of the last count.

Quartus Schematic Block Symbol



```
entity Inc_Enc_e is
    Generic (
        -- bit width of the counter value
        BIT_WIDTH : natural := 32
    );

    Port (
        -- Logic-Clock and -Reset
        CLK           : in  std_logic;
        RST           : in  std_logic;
        STOP          : in  std_logic;
        CLKEN         : in  std_logic;
        -- Counter Signals A/B/N
        A             : in  std_logic;
        B             : in  std_logic;
        N             : in  std_logic;
        -- Control  Interface
        A_PULSE_POL   : in  std_logic;
        B_DIR_POL     : in  std_logic;
        N_ZERO_POL    : in  std_logic;
        PULSECLEAR    : in  std_logic;
        LOAD          : in  std_logic;
        EDGE          : in  std_logic;
        MISSINGSUPPLY: in  std_logic;
        COUNT_MODE    : in  std_logic_vector (  1 downto 0 );
        COUNT_TYPE    : in  std_logic_vector (  1 downto 0 );
        COUNT_DIR     : in  std_logic;
        HOLDSW        : in  std_logic;
        HOLDHW        : in  std_logic;
        HOLDSOURCE    : in  std_logic_vector (2 downto 0);
        RESETSW       : in  std_logic;
        RESETSOURCE   : in  std_logic_vector (2 downto 0);
        MAX_VALUE     : in  std_logic_vector ((BIT_WIDTH-1) downto 0);
        MIN_VALUE     : in  std_logic_vector ((BIT_WIDTH-1) downto 0);
        RESET_VALUE   : in  std_logic_vector ((BIT_WIDTH-1) downto 0);
        LOAD_VALUE    : in  std_logic_vector ((BIT_WIDTH-1) downto 0);
        ENCODERCOUNT : out std_logic_vector ((BIT_WIDTH-1) downto 0);
        HOME          : out std_logic;
        HOMED         : out std_logic;
        UNDERFLOW     : out std_logic;
        OVERFLOW      : out std_logic;
        LASTCOUNTDIR : out std_logic
    );
end Inc_Enc_e;
```

Counter Behavior Common to the Three Counting Modes

If the counter is loaded with a value outside the count range, then the counter counts in the requested direction, and rolls over at the upper limit. (This rollover is not reported in the overflow or underflow status bits.) Once the counter value is within the specified range, it remains within the range until a Load or Reset loads it outside the range.
The counting process can be started or stopped using the software Hold or Reset signals, but the counter is neither held nor reset when the module goes to STOP mode. Software controls (Reset, Hold, and Load) are cleared by module STOP. The counter continues to count based on hardware inputs. The counter is not affected when the PLC changes to STOP. The current count value can be loaded using the load signal.

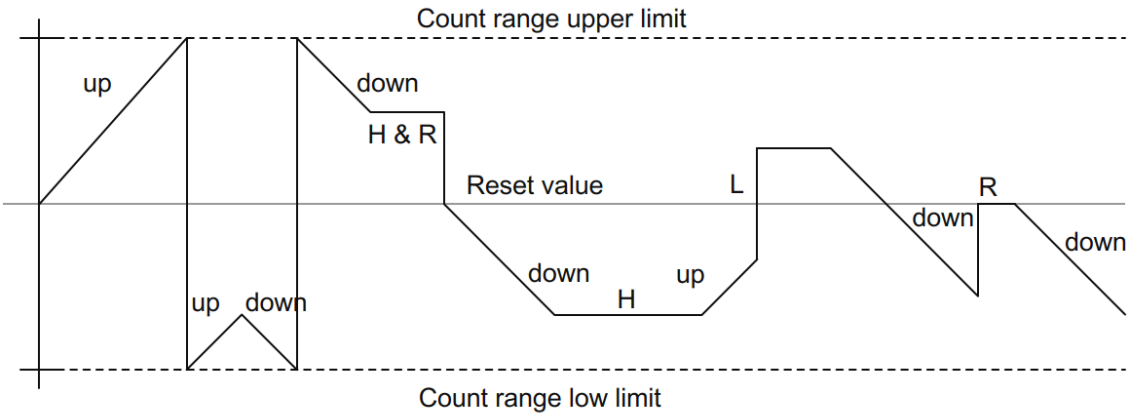
Continuous Counting Mode

In the continuous counting mode, the count ranges are variable and can be changed.

- Count range example of 16-bit counter: -32768 to 32767 up to
- Count range example of 32-bit counter: -2,147,483,648 to 2,147,483,647

At power-up, the counter has a start value of 0, until either the hardware configuration or the software program give it a different starting value. You must initialize the counter to a known value with a reset or load before you begin counting. You can program the reset signal to load the counter with 0, the minimum value, or the load value.
The "main count direction" parameter has no effect on this counter mode.
When counting up, the module increments to the maximum value, then rolls over to the minimum value and continues counting. (This rollover is reported in the overflow status bit.)
When counting down, the module decrements to the minimum value, then rolls over to the maximum value and continues counting. (This rollover is reported in the underflow status bit.)

The figure below illustrates how continuous counting functions.



up:	Up count
down:	Down count
H:	Hold is active
R:	Reset is active
L:	Loading is active

Single Counting

In the single counting mode, you can specify the count range of 10 bits up to 32 bits.

You must initialize the counter to a known value with a reset or load before you begin counting. You can program the reset signal to load the counter with 0, the minimum or maximum value, or the load value.

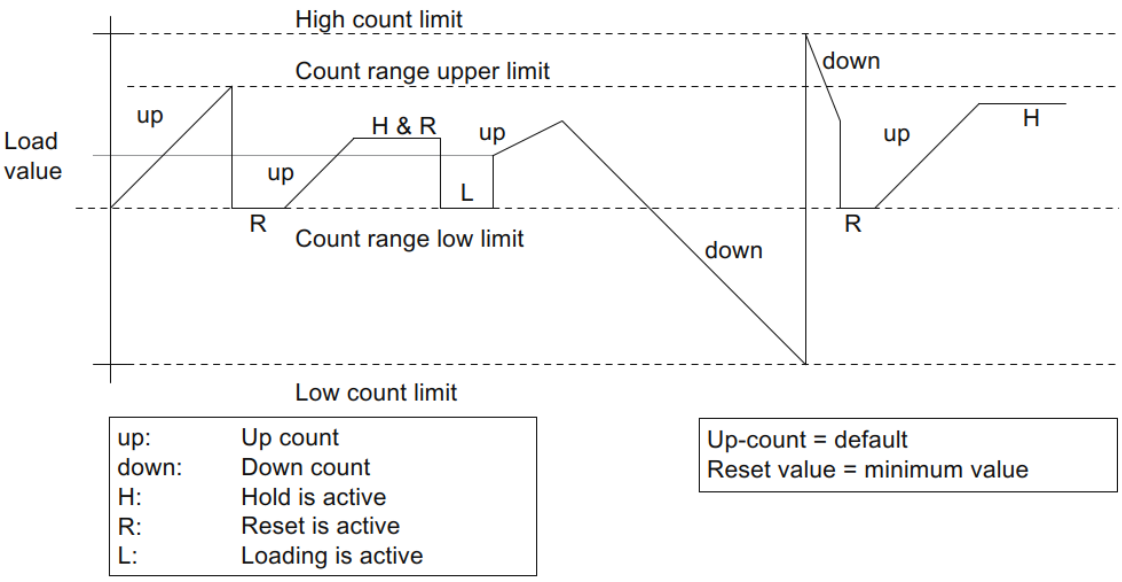
When the "main count direction" is set to Count Up, the counter behaves in the following ways:

- It increments to the maximum value, then rolls over to the minimum value and holds this value until reset or loaded. (This rollover is reported in the overflow status bit.)
- It decrements to the lower limit of the counter, rolls over to the upper limit, and continues counting. (This rollover is not reported in the overflow or underflow status bits.)

When the "main count direction" is set to Count Down, the counter behaves in the following ways:

- It decrements to the minimum value, then rolls over to the maximum value and holds this value until reset or loaded. (This rollover is reported in the underflow status bit.)
- It increments to the upper limit of the counter, rolls over to the lower limit, and continues counting. (This rollover is not reported in the overflow or underflow status bits.)

The figure below illustrates how single counting functions.



Periodic counting

In the periodic counting mode, you can specify the count range of 10 bits up to 32 bits. You must initialize the counter to a known value with a reset or load before you begin counting. You can program the reset signal to load the counter with 0, the minimum or maximum value, or the load value.

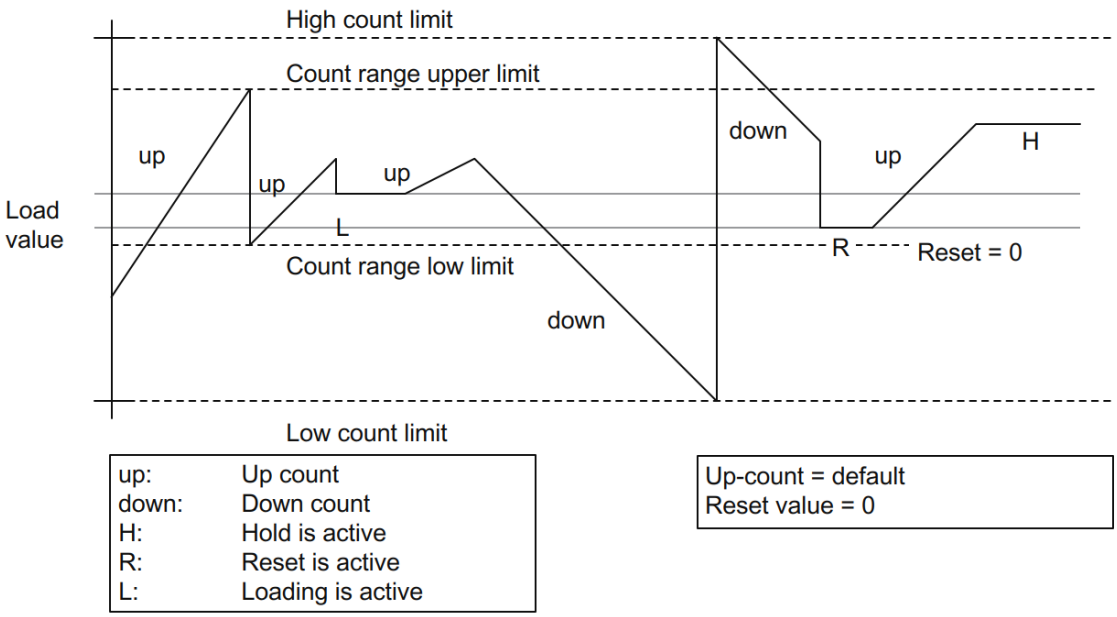
When the "main count direction" is set to Count Up, the counter behaves in the following ways:

- It increments to the maximum value, then rolls over to the minimum value and continues counting. (This rollover is reported in the overflow status bit.)
- It decrements to the lower limit of the counter, rolls over to the upper limit, and continues counting. (This rollover is not reported in the overflow or underflow status bits.)

When the "main count direction" is set to Count Down, the counter behaves in one of the following ways:

- It decrements to the minimum value, then rolls over to the maximum value and continues counting. (This rollover is reported in the underflow status bit.)
- It increments to the upper limit of the counter, rolls over to the lower limit, and continues counting. (This rollover is not reported in the overflow or underflow status bits.)

The figure below illustrates how periodic counting functions.



Pulse Evaluation

The Incremental Encoder logic counts the edges of the signals. Normally, the edge at A is evaluated for a single evaluation (x1). To achieve a higher resolution, you can assign the mode for the encoder signal evaluation to use double or quadruple (x2 or x4) evaluation of the signals. Use the **count_mode** to select the type of encoder signal evaluation. The A and B signals must be displaced by 90° to select single, double, or quadruple evaluation.

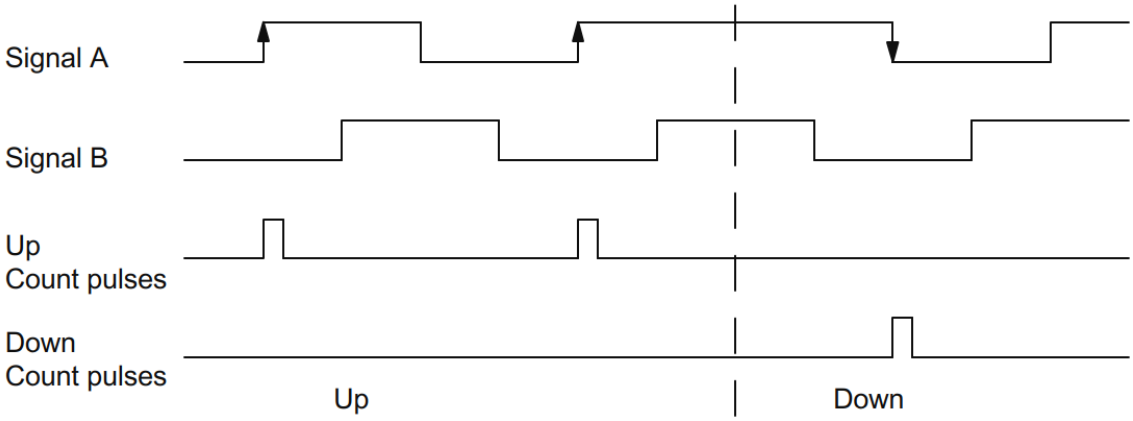
Counting Mode Selection

- Count_MODE = 0: single count
- Count_MODE = 1: double count
- Count_MODE = 2: quad count
- Count_MODE = 3: pulse/direction

Single Evaluation (single count)

Single evaluation (x1) means that only one edge of A is evaluated.
- The counter increments on a rising edge of A when B is low.
- The counter decrements on a falling edge of A when B is low.

The figure below illustrates single evaluation of the signals.

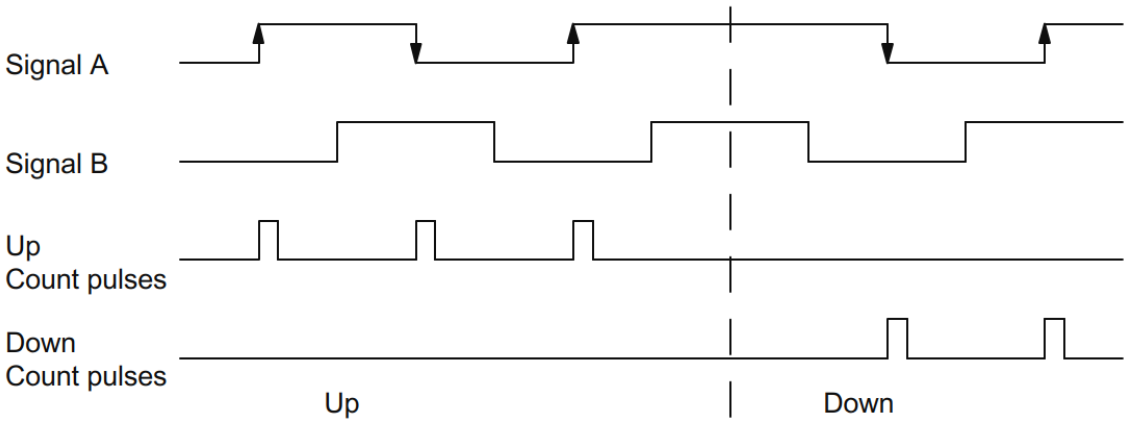


Double Evaluation (double count)

Double evaluation (x2) means that the rising and falling edges of signal A are evaluated. The level of signal B determines the direction of counting.

- The counter increments on the rising edge of A when B is low, and on the falling edge of A when B is high.
- The counter decrements on the rising edge of A when B is high, and on the falling edge of A when B is low.

The figure below illustrates double evaluation of the signals.

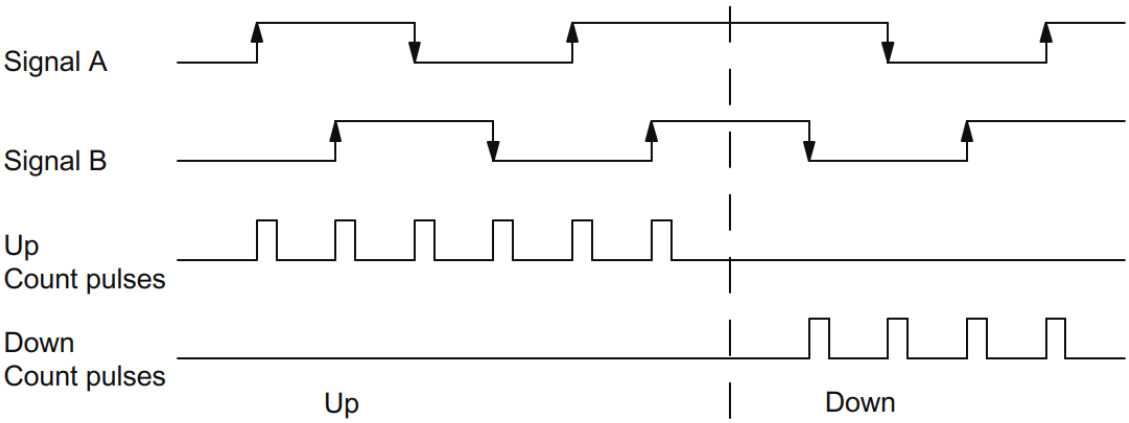


Quadruple Evaluation (quad count)

Quadruple evaluation (x4) means that the rising and falling edges of A and B are evaluated. The levels of signals A and B determine the direction of counting.

- The counter increments: on the rising edge of A when B is low, on the falling edge of A when B is high, on the rising edge of B when A is high, and on the falling edge of B when A is low.
- The counter decrements: on the falling edge of A when B is low, on the rising edge of A when B is high, on the falling edge of B when A is high, and on the rising edge of B when A is low.

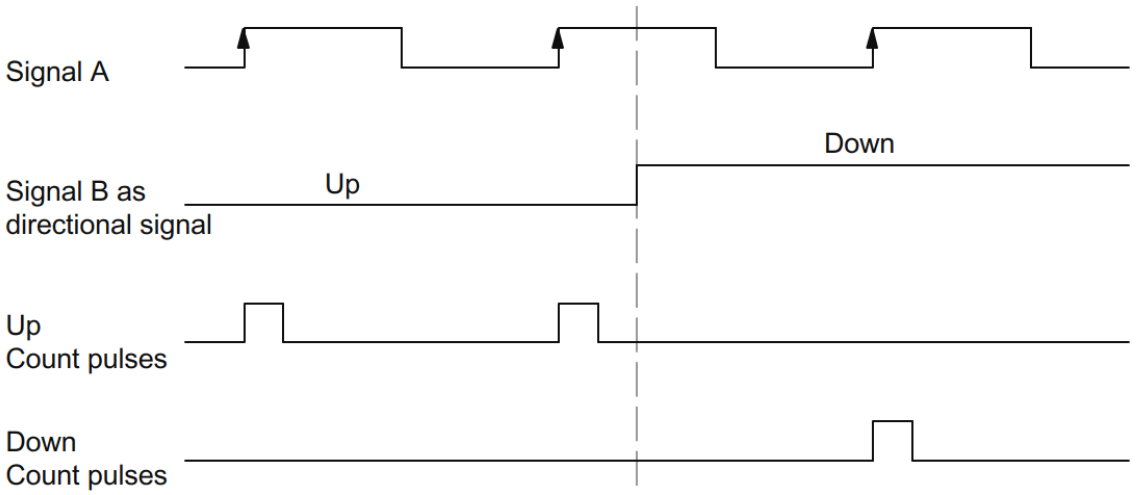
The figure below illustrates quadruple evaluation of the signals.



Pulse and direction

When you select Pulse & Direction for the encoder signal evaluation type, the module counts on the rising edge of each signal A pulse. If signal B is 0 (low), the counter is incremented. If signal B is 1 (high), the counter decrements.

The figure below illustrates pulse & direction counting.


Signal A
Down
Signal B as
directional signal
Up
Up
Count pulses
Down
Count pulses

4.2 SSI Encoder

Description

Absolute encoders with synchronous-serial interface (SSI) assign a fixed numeric value to each position. This value is permanently available and can be read out serially.

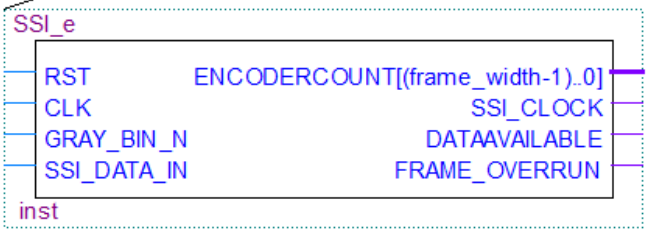
This library component generates a SSI Clock output and allows the reading of a numeric value for each absolute position.

In case of a misconfiguration of the frame or Monoflop time, FRAME_OVERRUN will be set to TRUE to signal an error.

Table 122: TM FAST SSI Encoder

Parameter	Data type	Direction	Description
BIT_WIDTH	Integer	Generic	Number of bits (range 10 to 64 bits)
MONOFLOPSEL	Integer	Generic	Monoflop time selection • 0 -> 16 µs • 1 -> 32 µs • 2 -> 48 µs • 3 -> 64 µs
CLOCKSEL	Integer	Generic	SSI clock speed selection • 0 -> 125 kHz • 1 -> 250 kHz • 2 -> 500 kHz • 3 -> 1 MHz
SHIFT_DIR	String	Generic	Alignment “Right” or “Left”
SHIFT_COUNT	Integer	Generic	Number of bits to shift (0 to 64 bits)
FRAME_WIDTH	Integer	Generic	Number of bits of the frame (13 to 64 bits)
GREY_BIN_N	STD_LOGIC	Input	Signal Encoding • 0 -> Absolute binary • 1 -> Grey code
SSI_DATA_IN	STD_LOGIC	Input	SSI data input, connect this signal to a RS422 input
ENCODERCOUNT	STD_LOGIC_VECTOR	Output	Encoder position value
FRAME_OVERRUN	STD_LOGIC	Output	Frame overrun error
DATA_AVAILABLE	STD_LOGIC	Output	Valid data is available
SSI_CLOCK	STD_LOGIC	Output	SSI clock output connect this signal to a RS422 output
Quartus Schematic Block Symbol			

Parameter	Value	Type
BIT_WIDTH	13	Signed Integer
MONOFLOPSEL	0	Signed Integer
CLOCKSEL	0	Signed Integer
SHIFT_DIR	right	String
SHIFT_COUNT	0	Signed Integer
FRAME_WIDTH	13	Signed Integer




```
entity SSI_e is
    Generic (
        BIT_WIDTH    : integer range 10 to 64 := 13;
        MONOFLOPSEL  : integer range  0 to 3  :=  0;
        CLOCKSEL     : integer range  0 to 3  :=  0;
        SHIFT_DIR    : string                 := "right";
        SHIFT_COUNT  : integer range  0 to 64 :=  0;
        FRAME_WIDTH  : integer range 10 to 64 := 13
    );
    Port (
        RST           : in  STD_LOGIC;
        CLK           : in  STD_LOGIC;
        GRAY_BIN_N    : in  STD_LOGIC;
        SSI_DATA_IN   : in  STD_LOGIC;
        ENCODERCOUNT  : out STD_LOGIC_VECTOR ( (BIT_WIDTH - 1) downto 0 );
        SSI_CLOCK     : out STD_LOGIC;
        DATAAVAILABLE: out STD_LOGIC;
        FRAME_OVERRUN: out STD_LOGIC
    );
end SSI_e;
```

Shift Register Frame Width

You can select a shift register frame width of 10 bits up to 64 bits, depending on the frame length of your SSI encoder.

Monoflop time

Available monoflop times for the SSI encoder are 16, 32, 48, or 64 µs.
You must select a monoflop time equal to or greater than the encoder's specified minimum time. If you do not know the specification for your encoder, select 64 µs.

Clock rate

You can select a clock rate of 125 kHz, 250 kHz, 500 kHz, or 1 MHz in the Parameters tab dialog, based on the ca
pabilities of the encoder, the update time required, and the length of
the cable. The maximum clock rate you can select is limited by the length of shielded encoder cable you use.

- CLOCKSEL = 0: CLK = 125 kHz
- CLOCKSEL = 1: CLK = 250 kHz
- CLOCKSEL = 2: CLK = 500 kHz
- CLOCKSEL = 3: CLK = 1 MHz

Data shift direction

You can select the direction of data to shift left or right.

Normalization Data Shift Length

You can specify the number of bit positions to be shifted within the range of 0 to 64. Normalization allows the SSI encoder data to be scaled to more convenient units used in the module program.

4.3 SSI ListenIn

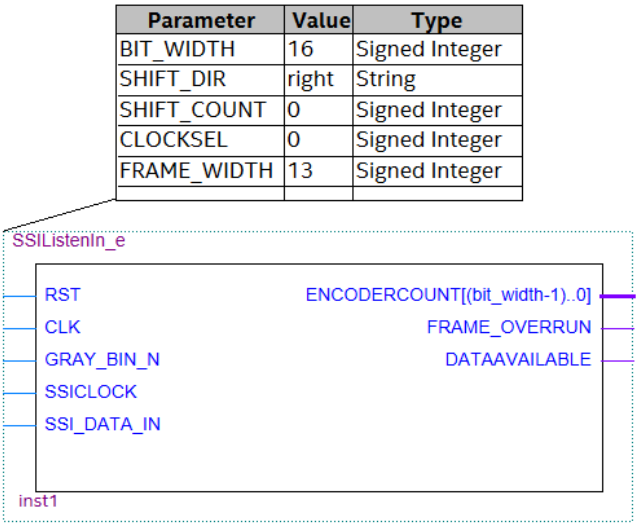
Description

Absolute encoders' data are transmitted using binary encoding. The SSI ListenIn logic allows the TM FAST module to connect to the same encoder for synchronized control. This library component reads-in the SSI clock of the SSI-Master and allows the reading of a numeric value for each absolute position of the encoder (SSI-Slave).

In case of a misconfiguration of the frame, FRAME_OVERRUN will be set to TRUE to signal an error.

Table 123: TM FAST SSI ListenIn

Parameter	Data type	Direction	Description
BIT_WIDTH	Integer	Generic	Number of bits (range 10 to 64 Bits)
CLOCKSEL	Integer	Generic	SSI clock speed selection • 0 -> 125 kHz • 1 -> 250 kHz • 2 -> 500 kHz • 3 -> 1 MHz
SHIFT_DIR	String	Generic	Alignment "Right" or "Left"
SHIFT_COUNT	Integer	Generic	Number of bits to shift. (0 to 32)
FRAME_WIDTH	Integer	Generic	Number of bits of the frame. (13 to 32)
GREY_BIN_N	STD_LOGIC	Input	Signal Encoding • 0 -> Absolute binary • 1 -> Grey code
SSI_DATA_IN	STD_LOGIC	Input	SSI data input, connect this signal to the RS422 input
SSICLOCK	STD_LOGIC	Input	SSI clock input, connect this signal to the RS422 input
ENCODERCOUNT	STD_LOGIC_VECTOR	Output	Encoder counter value
FRAME_OVERRUN	STD_LOGIC	Output	Frame overrun error
DATAAVAILABLE	STD_LOGIC	Output	Valid data is available
Quartus Schematic Block Symbol			



```
entity SSIListenIn_e is
        Generic (
        BIT_WIDTH   : integer :=      16;
        SHIFT_DIR   : string  := "right";
        SHIFT_COUNT : integer :=       0;
        CLOCKSEL    : integer :=       0;
        FRAME_WIDTH : integer :=      13
        );
    Port (
        RST           : in  STD_LOGIC;
        CLK           : in  STD_LOGIC;
        GRAY_BIN_N    : in  STD_LOGIC;
        SSICLOCK      : in  STD_LOGIC;
        SSI_DATA_IN   : in  STD_LOGIC;
        ENCODERCOUNT  : out STD_LOGIC_VECTOR ((BIT_WIDTH - 1) downto 0);
        FRAME_OVERRUN: out STD_LOGIC;
        DATAAVAILABLE: out STD_LOGIC
    );
end SSIListenIn_e;
```

Shift Register Frame Width

You can select a shift register frame width of 10 bits up to 32 bits, depending on the frame length of your SSI encoder.

Clock rate

You can select a clock rate of 125 kHz, 250 kHz, 500 kHz, or 1 MHz. For a SSI Listen application, you must select a clock rate equal to the master's clock rate.
The maximum clock rate you can select is limited by the length of shielded encoder cable you use.

- CLOCKSEL = 0: CLK = 125 kHz
- CLOCKSEL = 1: CLK = 250 kHz
- CLOCKSEL = 2: CLK = 500 kHz
- CLOCKSEL = 3: CLK = 1 MHz

Data shift direction

You can select the direction of data to shift left or right.

Normalization Data Shift Length

You can specify the number of bit positions to be shifted within the range of 0 to 12. Normalization allows the SSI encoder data to be scaled to more convenient units used in the module program.