# TM FAST Library

Manual

# **Table of Contents**

1	Overview	4
1.1	TM FAST library block	4
1.2	Instantiation of logic blocks in the application	5
1.3	TM FAST use of phase clocks	7
2	TM FAST library: Instruction Set	8
2.1	Detect negative edge	9
2.2	Detect negative signal edge	10
2.3	Detect positive edge	11
2.4	Detect positive signal edge	12
2.5	Reset/set flip-flop (RS_FF)	13
2.6	Set/reset flip-flop (SR_FF)	14
2.7	Comparison functions (Compare_W)	15
2.8	Comparison functions (Compare_DW)	16
2.9	Generate one's complement for 16-bit integer (INV_I)	17
2.10	Generate unlatched one's complement for 16-bit integer (INV_I_U)	18
2.11	Generate one's complement for 32-bit integer (INV_DI)	19
2.12	Generate unlatched one's complement for 32-bit integer (INV_DI_U)	20
2.13	Convert integer (16-bits) to double (32-bits) integer (I_DI)	21
2.14	Convert integer (16-bits) to double (32-bits) integer, unlatched logic (I_DI_U)	22
2.15	Move (Move_B, Move_W, Move_DW)	23
2.16	Move unlatched (Move_B_U, Move_W_U, Move_DW_U)	26
2.17	Rotate left word (ShiftRotate_W)	28
2.18	Rotate left word, unlatched logic (ShiftRotate_W_U)	29
2.19	Rotate left double word (ShiftRotate_DW)	30
2.20	Rotate left double word, unlatched logic (ShiftRotate_DW_U)	31
2.21	Rotate right word (ShiftRotate_W)	32
2.22	Rotate right word, unlatched logic (ShiftRotate_W_U)	33
2.23	Rotate right double word (ShiftRotate_DW)	34
2.24	Rotate right double word, unlatched logic (ShiftRotate_DW_U)	34
2.25	Shift left word (ShiftRotate_W)	36
2.26	Shift left word, unlatched logic (ShiftRotate_W_U)	37
2.27	Shift left double word (ShiftRotate_DW)	38
2.28	Shift left double word, unlatched logic (ShiftRotate_DW_U)	39

1

3

2.29	Shift right integer (ShiftRotate_W)	40
2.30	Shift right integer, unlatched logic (ShiftRotate_W_U)	41
2.31	Shift right double integer (ShiftRotate_DW)	42
2.32	Shift right double integer, unlatched logic (ShiftRotate_DW_U)	43
2.33	Shift right word (ShiftRotate_W)	44
2.34	Shift right word, unlatched logic (ShiftRotate_W_U)	45
2.35	Shift right double word (ShiftRotate_DW)	46
2.36	Shift right double word, unlatched logic (ShiftRotate_DW_U)	47
2.37	Word AND word (WAND_W)	48
2.38	Word AND word, unlatched logic (WAND_W_U)	49
2.39	Word AND double word (WordLogic_DW)	50
2.40	Word AND double word, unlatched logic (WordLogic_DW_U)	51
2.41	Word OR word (WordLogic_W)	52
2.42	Word OR word, unlatched logic (WordLogic_W_U)	53
2.43	Word OR double word (WordLogic_DW)	54
2.44	Word OR double word, unlatched logic (WordLogic_DW_U)	55
2.45	Word XOR word (WordLogic_W)	56
2.46	Word XOR word, unlatched logic (WordLogic_W_U)	57
2.47	Word XOR double word (WordLogic_DW)	58
2.48	Word XOR double word, latched logic (WordLogic_DW_U)	59
	TM FAST library: Operation function blocks (FBs)	60
3.1	WordPack (FB76_WordPack)	62
3.2	WordPack, unlatched logic (FB76_WordPack_U)	62
3.3	WordCast (FB77_WordCast)	64
3.4	WordCast, unlatched logic (FB77_WordCast_U)	65
3.5	BitSum (FB78_BitSum)	66
3.6	BitSum, unlatched logic (FB78_BitSum_U)	67
3.7	Encode (FB79_Encode)	68
3.8	Encode, unlatched logic (FB79_Encode_U)	69
3.9	Period measurement (FB80_Period32, FB81_Period16)	70
3.10	Frequency measurement (FB82_Freq32, FB83_Freq16)	73
3.11	Bit shift registers (FB124_Shift, FB125_Shift2, FB126_Shift4, FB127_Shift8, FB85_Shift16, FB84_Shift32)	77
3.12	BitPick (FB86_BitPick_DW, FB87_BitPick_W)	83
3.13	BitPick, unlatched logic (FB86_BitPick_DW_U, FB87_BitPick_W_U)	85
3.14	BitShift (FB88 BitShift DW, FB89 BitShift W)	87

4

3.15	BitCast (FB90_BitCast_DW, FB91_BitCast_W)	89
3.16	BitCast, unlatched logic (FB90_BitCast_DW_U, FB91_BitCast_W_U)	93
3.17	BitPack (FB92_BitPack_DW, FB93_BitPack_W)	97
3.18	BitPack, unlatched logic (FB92_BitPack_DW_U, FB93_BitPack_W_U)	101
3.19	BitInsert (FB94_BitInsert32, FB95_BitInsert32)	105
3.20	BitInsert, unlatched logic (FB94_BitInsert32_U, FB95_BitInsert32_U)	107
3.21	FIFO (FB96_FIFO32, FB97_FIFO16)	109
3.22	LIFO (FB98_LIFO32, FB99_LIFO16)	112
3.23	Multiply (FB100_FMMul32 and FB101_FMMul16)	115
3.24	Divide (FB102_FMDiv32 and FB103_FMDiv16)	117
3.25	Absolute value (FB104_FMAbs32 and FB105_FMAbs16)	119
3.26	Add (FB106_FMAdd32 and FB107_FMAdd16)	121
3.27	Subtract (FB108_FMSub32 and FB109_FMSub16)	123
3.28	Data selector (FB110_DatSel32 and FB111_DatSel16)	125
3.29	Binary scaler (FB112_BiScale)	127
3.30	Pulse timer (FB113_TP32, FB116_TP16)	128
3.31	On-delay timer (FB114_TOn32, FB117_TOn16)	130
3.32	Off-Delay timer (FB115_TOf32, FB118_TOf16)	132
3.33	Clock pulse generator (FB119_CP_Gen)	134
3.34	Up/down counter (FB120_CTUD32, FB123_CTUD16)	136
3.35	Up counter (FB121_CTU16)	139
3.36	Down counter (FB122_CTD16)	140
3.37	ClkTic100k	141
	TM FAST library: Encoder blocks of the FM 352-5	142
4.1	Incremental Encoder (Inc_Enc)	143
4.2	SSI Encoder	152
4.3	SSI ListenIn	155

#### 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.

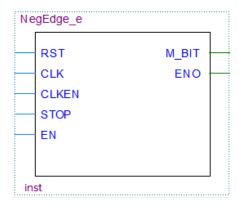
Example: Entity of NegEdge logic:

```
entity NegEdge e is
   Port (
       RST : in
                    STD LOGIC;
                    STD LOGIC;
       CLK : in
                    STD LOGIC:
       CLKEN: in
                    STD LOGIC;
       STOP : in
       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;
            : in STD LOGIC;
       R
       PV : in STD LOGIC VECTOR (15 downto 0);
            : out STD LOGIC;
       Q
            : out STD LOGIC VECTOR (15 downto 0)
       CV
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

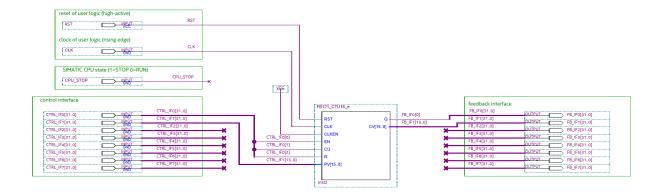
CLKEN => '1', -- Clock-enable signal

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
              => STATUS,
  Q
                                                -- Status of the counter
  \mathsf{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	FM 352-5 Symbolic	Description
Name	Name	
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	1	Generate one's complement (32-bit), unlatched logic
I_DI	I_DI	Convert Integer (16-bit) to double integer (32-bit)
I_DI_U	1 -	Convert Integer (16-bit) to double integer (32-bit), unlatched
		logic
Move_B	MOVE	Move a specified value (8-bit)
Move_B_U	1	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,	Rotate and Shift operations (16 bit)
ShiftRotate_W_U	SHR_I	Rotate and Shift operations (16 bit), unlatched logic
ShiftRotate_DW	ROL_DW, ROR_DW,	Rotate and Shift operations (32 bit)
ShiftRotate_DW_U	SHL_DW, SHR_DW,	Rotate and Shift operations (32 bit), unlatched logic
	SHR_DI	. , ,
WordLogic_W	WAND_W, WOR_W,	Word operations (16 bit)
WordLogic_W_U	WXOR_W	Word operations (16 bit), unlatched logic
WordLogic_DW	WAND_DW,	Word operations (32 bit)
WordLogic_DW_U	WOR_DW,	Word operations (32 bit), unlatched logic
-	WXOR_DW	

# 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,	Edge memory bit that
NegEdge_e			clocked tag	stores the previous signal state of RLO
RST M_BIT				
CLK ENO				
CLKEN				
STOP				
inst				
	CLK CLKEN STOP EN M_BIT	in STD inout STD	LOGIC; LOGIC; LOGIC; LOGIC; LOGIC;	

© Siemens 2025

# 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
NegSignalEdge_e	EDGE	STD_LOGIC	Input	Scanned signal
RST M_BIT CLK Q CLKEN STOP EN EDGE	MBIT	STD_LOGIC	Bit storage, clocked tag	Edge memory bit that stores the previous signal state of EDGE
	Port ( RST CLK CLKEN STOP EN EDGE M_BIT	in STD inout STD	LOGIC; LOGIC; LOGIC; LOGIC; LOGIC; LOGIC; LOGIC;	

# 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:

Parameter	Data type	Direction	Description
M_BIT	STD_LOGIC	Bit storage,	Edge memory bit that
		clocked tag	stores the previous signal
			state of RLO
DogEdge	o i.e		
	_6 18		
	_	-	
CLK :	_	_	
CLKEN:	in STD_	LOGIC;	
STOP :	in STD_	LOGIC;	
EN :	in STD	LOGIC;	
M BIT:	inout STD	LOGIC;	
ENO :	inout STD	LOGIC	
	_	_	
PosEdge e;			
	M_BIT  M_BIT  Port (  RST :  CLK :  CLKEN:  STOP :  EN :  M_BIT:	ty PosEdge_e is Port (  RST : in STD CLK : in STD CLKEN: in STD STOP : in STD EN : in STD M_BIT: inout STD ENO : inout STD	<pre>ty 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</pre>

# 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
PosSignalEdge_e	EDGE	STD_LOGIC	Input	Scanned signal
RST M_BIT CLK Q CLKEN STOP EN EDGE	MBIT	STD_LOGIC	Bit storage, clocked tag	Edge memory bit that stores the previous signal state of EDGE
		lEdge_e is		
Ро	rt (			
		in STD_LO		
		in STD_LO		
	CLKEN :	_		
		in STD_LO		
	EN :	in STD_LO		
	EDGE :	in STD_LO	GIC;	
	M_BIT :	inout STD_LO	GIC;	
	Q :	inout STD_LO	GIC	
);				
and Do	sSignalEd <sub>9</sub>	re 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	Parameter	Data type	Direction	Description
Symbol	S	STD LOGI	C Input	Enables set operation
RS_FF_e	R	STD LOGI		Enables reset operation
	Q	STD_LOGI		Signal state of output
RST Q		_		
CLK				
CLKEN				
STOP				
R				
s				
inst				
en	tity RS_F Port (	_		
	RST		STD_LOGIC;	
	CLK	: in	STD_LOGIC;	
	CLK	EN: in	STD_LOGIC;	
	STO	P : in	STD_LOGIC;	
	R	: in	STD_LOGIC;	
	S	: in	STD_LOGIC;	
	Q	: out	STD_LOGIC	
	);			
en	d 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
SR_FF_e	R	STD_LOGIC	Input	Enables reset operation
	Q	STD_LOGIC	Output	Signal state of output
RST Q				
CLK				
CLKEN				
STOP				
R				
S				
inst				
ent		: in S : in S EN: in S P: in S : in S	TD_LOGIC; TD_LOGIC; TD_LOGIC; TD_LOGIC; TD_LOGIC; TD_LOGIC;	
	Q		TD 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
Parameter Value Typ	IN1	STD_LOGIC _VECTOR	Input, constant	First comparison value
MODE EQ Strir	I IN2	STD_LOGIC _VECTOR	Input, constant	Second comparison value
Compare_W_e  RST ENO CLK CLKEN STOP EN IN1[150] IN2[150]	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 Comp generic MOI );	. (	.ng := "EQ"		
port (				
	RST : in CLK : in	STD_LOGIC;		
		STD_LOGIC; STD LOGIC;		
		STD LOGIC;		
		STD_LOGIC;		
	IN1 : in	STD_LOGIC_V	ECTOR ( 1	5 downto 0 );
	IN2 : in			5 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

uartus Schematic Block vmbol	Parameter	Data type	Direction	Description
Parameter Value Type	IN1	STD_LOGIC _VECTOR	Input, constant	First comparison value
MODE EQ String	IN2	STD_LOGIC _VECTOR	Input, constant	Second comparison value
Compare_DW_e  RST ENO CLK CLKEN STOP EN IN1[310] IN2[310]	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 Compa generic MODE );		ng := "EQ"		
generic MODE	( : stri			
generic MODE );	RST : in	STD_LOGIC;		
generic MODE );	RST : in CLK : in	STD_LOGIC; STD_LOGIC;		
generic MODE );	RST : in CLK : in CLKEN: in	STD_LOGIC; STD_LOGIC; STD_LOGIC;		
generic MODE );	RST : in CLK : in CLKEN: in STOP : in	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC;		
generic MODE );	RST : in CLK : in CLKEN: in STOP : in EN : in	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC;	VECTOR ( 3	31 downto 0 );
generic MODE );	RST : in CLK : in STOP : in EN : in IN1 : in	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC_V	-	31 downto 0 ); 31 downto 0 );
generic MODE );	RST : in CLK : in CLKEN: in STOP : in EN : in IN1 : in IN2 : in	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC_V	ECTOR ( 3	
generic MODE );	RST : in CLK : in CLKEN: in STOP : in EN : in IN1 : in IN2 : in	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC_V	ECTOR ( 3	

# 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
INV_I_e	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC	Input	Integer input
RST ENO		_VECTOR		value (16-bits)
CLK OUT1[150]	OUT1	STD_LOGIC	Output	One's
CLKEN		_VECTOR		complement of
STOP				the 16-bit integer IN1
EN				integer in i
IN1[150]				
IN I[150]				
inst1				
HISCI				
<pre>entity INV_I_e is Port (</pre>				
RST : in	STD_LOGIC;			
CLK : in	STD_LOGIC;			
CLKEN: in	STD_LOGIC;			
STOP : in	STD_LOGIC;			
EN : in	STD_LOGIC;			
IN1 : in	STD_LOGIC_V	JECTOR (15 d	ownto 0);	
ENO : out	STD_LOGIC;			
OUT1 : out	STD_LOGIC_V	JECTOR (15 d	ownto 0)	
);				
<pre>end INV_I_e;</pre>				

# 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#FFF. 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
INV_I_U_e	IN1	STD_LOGIC _VECTOR	Input	Integer input value (16-bits)
IN1[150] OUT1[150]	OUT1	STD_LOGIC _VECTOR	Output	One's complement of the 16-bit integer IN1
<pre>entity INV_I_U_e is Port (</pre>				
	STD_LOGIC_VE STD_LOGIC_VE			
end inv_i_o_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
INV_DI_e	ENO	STD_LOGIC	Output	Enable output
RST ENO	IN1	STD_LOGIC_ VECTOR	Input	Integer input value (32-bits)
CLK OUT1[310]	OUT1	STD_LOGIC_	Output	One's
		VECTOR		complement of
CLKEN				the 32-bit
STOP				integer IN1
EN EN				
IN1[310]				
1141[516]				
inst				
<pre>entity INV_DI_e is</pre>				
Port (				
RST : in	STD_LOGIC;			
CLK : in	STD_LOGIC;			
CLKEN: in	STD_LOGIC;			
STOP : in	STD_LOGIC;			
EN : in				
IN1 : in	STD_LOGIC_	VECTOR (31 do	wnto 0);	
ENO : out	STD_LOGIC;			
OUT1 : out	STD_LOGIC_	VECTOR (31 do	wnto 0)	
);				
<pre>end INV_DI_e;</pre>				

# 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#FFF 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			
INV_DI_U_e	IN1	STD_LOGIC_ VECTOR	Input	Integer input value (32-bits)			
IN1[310] OUT1[310]	OUT1	STD_LOGIC_ VECTOR	Output	One's complement of the 32-bit integer IN1			
<pre>entity INV_DI_U_e is     Port (</pre>							

# 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
I_DI_e	ENO	STD_LOGIC	Output	Enable output
	IN1	STD_LOGIC_	Input	Integer input
RST ENO		VECTOR		value (16-bits)
CLK OUT1[310]	OUT1	STD_LOGIC_	Output	Result: Double
CLKEN		VECTOR		integer value
STOP				(32-bits)
EN				
IN1[150]				
inst				
<pre>entity I_DI_e is    Port (</pre>				
	STD_LOGIC;			
CLK : in	STD_LOGIC;			
	STD_LOGIC;			
STOP : in	STD_LOGIC;			
	STD_LOGIC;			
IN1 : in	STD_LOGIC_	VECTOR (15 d	lownto 0);	
	STD_LOGIC;			
OUT1 : out	STD_LOGIC_	VECTOR (31 d	lownto 0)	
);				
<pre>end I_DI_e;</pre>				

# 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			
I_DI_U_e	IN1	STD_LOGIC_ VECTOR	Input	Integer input value (16-bits)			
IN1[150] OUT1[310] inst	OUT1	STD_LOGIC_ VECTOR	Output	Result: Double integer value (32-bits)			
<pre>entity I DI U e is     Port (</pre>							

#### 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:

```
Parameter
Quartus Schematic Block Symbol
                                                 Data type
                                                              Direction
                                                                         Description
                                                 STD_LOGIC
                                     ΕN
                                                              Input
                                                                         Enable input
                                     ENO
                                                 STD_LOGIC
                                                              Output
                                                                         Enable output
  Mov e_B_e
                                     IN1
                                                 STD_LOGIC_
                                                              Input
                                                                         Source value
                                                 VECTOR
     RST
                         ENO
                                                 STD LOGIC
                                     OUT1
                                                              Output
                                                                         Destination
     CLK
                   OUT1[7..0]
                                                 VECTOR
                                                                         address of the
                                                                         value specified
     CLKEN
                                                                         at the IN1 input.
     STOP
     ΕN
     IN1[7..0]
  inst
              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

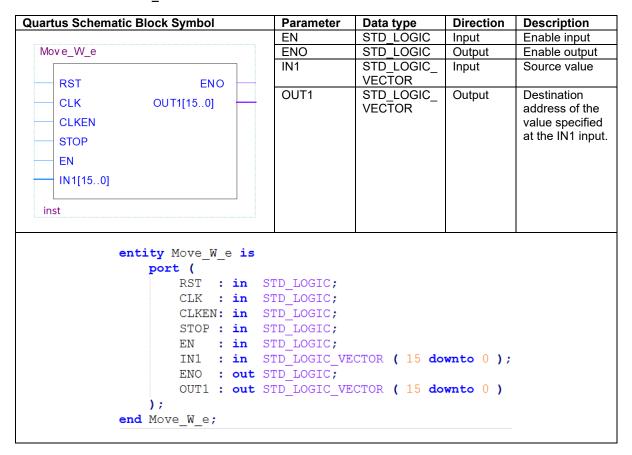
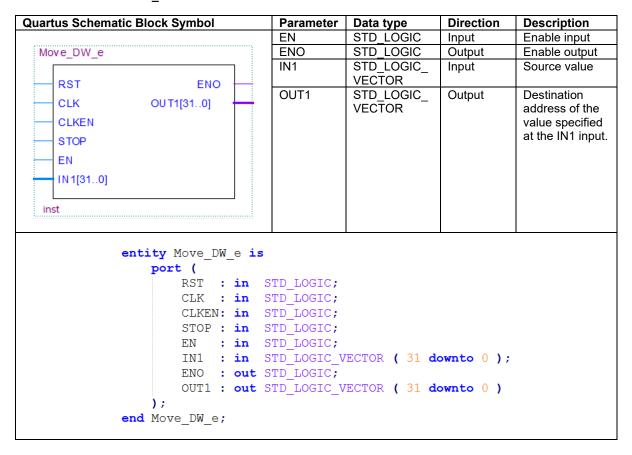


Table 18: TM FAST Move\_DW



# 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.

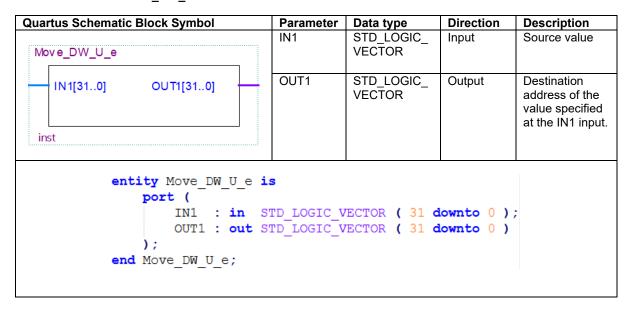
Table 19: TM FAST Move\_B\_U unlatched:

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
Mov e_B_U_e	IN1	STD_LOGIC_ VECTOR	Input	Source value
IN1[70] OUT1[70]	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.
<pre>entity Move_B_U_e is     port (</pre>	TD_LOGIC_VE			

Table 20: TM FAST Move\_W\_U unlatched

```
Quartus Schematic Block Symbol
                                       Parameter
                                                                 Direction
                                                                            Description
                                                   Data type
                                       IN1
                                                   STD_LOGIC_
                                                                            Source value
                                                                 Input
                                                   VECTOR
  Move_W_U_e
                                       OUT1
                                                   STD LOGIC
                                                                 Output
                                                                            Destination
     IN1[15..0]
                   OUT1[15..0]
                                                   VECTOR
                                                                            address of the
                                                                            value specified
                                                                            at the IN1 input.
  inst
             entity Move W U e is
                           : 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



### 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
Parameter Value Type	MODE	STRING	Generic	Defines Mode: "RLW": rotate left
MODE RLW String	EN	STD_LOGIC	Input	Enable input
8	ENO	STD_LOGIC	Output	Enable output
ShiftRotate_W_e	IN1	STD_LOGIC_ VECTOR	Input	Source value
RST ENO	N	STD_LOGIC_ VECTOR	Input	Number of bit positions to be rotated.
CLKEN STOP EN	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.
IN1[150] N[150]				
inst				
<pre>entity ShiftRotate_N</pre>	V_e is			
generic (				
MODE : S	string := "H	RLW"		
); Port (				
RST : in S	STD_LOGIC;			
CLK : in	STD_LOGIC;			
CLKEN: in	STD_LOGIC;			
STOP : in S	STD_LOGIC;			
EN : in S	STD_LOGIC;			
		ECTOR ( 15 d		
		ECTOR ( 15 d	ownto 0);	
	STD_LOGIC;			
OUT1 : out	STD_LOGIC_VE	ECTOR ( 15 d	ownto 0)	
);				
<pre>end ShiftRotate_W_e;</pre>	;			

#### 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)

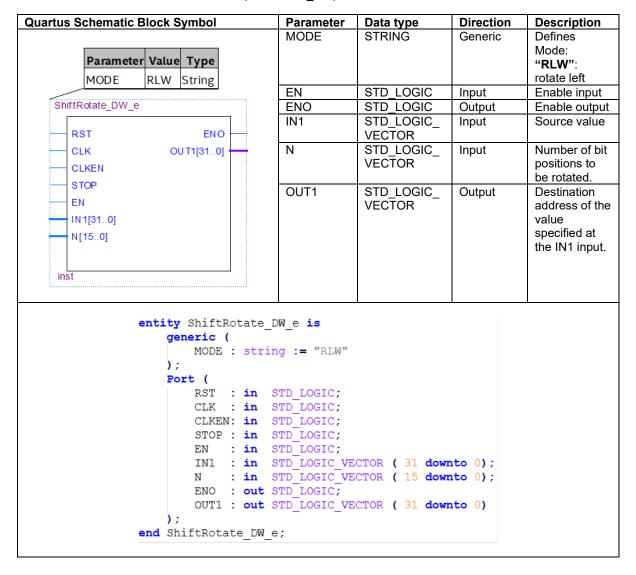
Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
Parameter Value Type	MODE	STRING	Generic	Defines Mode: "RLW": rotate left
MODE RLW String	IN1	STD_LOGIC_ VECTOR	Input	Source value
ShiftRotate_W_U_e  IN1[150] OUT1[150]	N	STD_LOGIC_ VECTOR	Input	Number of bit positions to be rotated.
N[150] inst	OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.
	tring := "F FD_LOGIC_VE FD_LOGIC_VE	ECTOR ( 15 de	ownto 0);	

#### 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)



#### 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)

Quart	Quartus Schematic Block Symbol			Parameter	Data type	Direction	Description		
	Parameter	Value	Туре		MODE	STRING	Generic	Defines Mode: "RLW": rotate left	
	MODE	RLW	String		IN1	STD_LOGIC_ VECTOR	Input	Source value	
Shir	ftRotate_DW_L		124 01		N	STD_LOGIC_ VECTOR	Input	Number of bit positions to be rotated.	
ins	IN1[310] N[150] t	0011	[310]		OUT1	STD_LOGIC_ VECTOR	Output	Destination address of the value specified at the IN1 input.	
	<pre>entity ShiftRotate_DW_U_e is     generic (</pre>								

#### 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:
Parameter Value Type				"RRW": rotate
MODE RRW String				right
MODE KKW String	EN	STD_LOGIC	Input	Enable input
ShiftRotate_W_e	ENO	STD_LOGIC	Output	Enable output
Cim (Notato_VV_0	IN1	DWORD	Input	Source value
RST ENO	N	WORD	Input	Number of bit
CLK OUT1[150]				positions to be
CLKEN	OUT1	DWORD	Output	rotated.  Destination
STOP	0011	DWORD	Output	address of the
EN EN				value specified at
IN1[150]				the IN1 input.
N[150]				
inst				
IIISC				
			l	
<pre>entity ShiftRotate W</pre>	e <b>is</b>			
generic (	_			
MODE : s	tring := "	RRW"		
);	_			
Port (				
RST : in S	TD LOGIC:			
	TD LOGIC;			
CLKEN: in S				
STOP: in S				
	TD LOGIC;			
		ECTOR ( 15	downto 0	١.
		ECTOR ( 15		
	TD_LOGIC_V TD LOGIC;	ECTOR ( 13	down to	, ,
	_	ECHOD / 15	<b>4</b>	
OUT1 : out S	TD_TOGIC_A	ECTOR ( 15	downto 0	,
);				
<pre>end ShiftRotate_W_e;</pre>				

#### 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 S	ymbol		Parameter	Data type	Direction	Description
				MODE	STRING	Generic	Defines Mode: "RRW": rotate right
Parameter	Value	Туре	<b>!</b>	IN1	DWORD	Input	Source value
MODE	RRW	String					
ShiftRotate_W_U	_e			N	WORD	Input	Number of bit positions to be rotated.
IN1[150] N[150]	OUT1	[150]		OUT1	DWORD	Output	Destination address of the value specified at the IN1 input.
inst							
е	_		otate_	W_U_e is			
	gene	eric (					
	•	MODE	:	string := "	'RRW"		
	);						
	Port	•		amp			
				STD_LOGIC_V			
				STD_LOGIC_V			
		OUTI	: out	STD_LOGIC_V	ECTOR ( 15	downto 0	)
_	);						
e	nd Shif	tkota	te_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 Bloc	Parameter	Data type	Direction	Description						
		MODE	STRING	Generic	Defines Mode: "RRW": rotate right					
Parameter Value		EN	STD_LOGIC	Input	Enable input					
MODE RRW	String	ENO	STD_LOGIC	Output	Enable output					
ChiffDotate DW e	IN1	DWORD	Input	Source value						
ShiftRotate_DW_e  RST	ENO	N	WORD	Input	Number of bit positions to be rotated.					
CLKEN STOP EN	Г1[310]	OUT1	DWORD	Output	Destination address of the value specified at the IN1 input.					
IN1[310] N[150] inst										
<pre>entity ShiftRotate_DW_e is     generic (</pre>										
	CLK : i CLKEN: i STOP : i EN : i IN1 : i N : i ENO : o OUT1 : o	n STD_LOG n STD_LOG n STD_LOG n STD_LOG ut STD_LOG ut STD_LOG	IC; IC; IC; IC; IC, IC_VECTOR ( IC_VECTOR (	15 downto	0);					
<pre>end ShiftRotate_DW_e;</pre>										

# 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)

Quar	Quartus Schematic Block Symbol			Parameter	Data type	Direction	Description				
	<b>Parameter</b> MODE	<b>Value</b> RRW	<b>Type</b> String		MODE	STRING	Generic	Defines Mode: "RRW": rotate right			
/	<u>I</u>			1	IN1	DWORD	Input	Source value			
Sh	ShiftRotate_DW_U_e				N	WORD	Input	Number of bit positions to be rotated.			
in	N1[310] N[150]	OUT1	[310]		OUT1	DWORD	Output	Destination address of the value specified at the IN1 input.			
	<pre>entity ShiftRotate_DW_U_e is     generic (</pre>										
	<pre>end ShiftRotate_DW_U_e;</pre>										

### 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)

Parameter Value Type  MODE SLW String  EN STD_LOGIC Input Enable input  ENO STD_LOGIC Output Enable output  IN1 WORD Input Source value  N WORD Input Number of bir positions to b shifted.  OUT1 WORD Output Destination address of the string	Quartus S	chematic	<b>Block</b>	Symbol		Parameter	Data type	Direction	Description
Parameter Value Type MODE SLW String  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 b shifted.  OUT1 WORD Output Destination address of th value specific the IN1 input.  entity ShiftRotate W e is generic ( MODE: string:= "SLW" ); Port ( RST: in STD_LOGIC; CLK: in STD_LOGIC;						MODE	STRING	Generic	Defines Mode:
MODE SLW String    ENO   STD_LOGIC   Output   Enable output   IN1   WORD   Input   Number of bit positions to be shifted.	D.	- v- vt- v	Value	Tuna					"SLW": shift left
IN1 WORD Input Source value  N WORD Input Number of bit positions to b shifted.  OUT1 WORD Output Destination address of th value specific the IN1 input.  EN IN1[15.0]  N[15.0]  entity ShiftRotate_W_e is generic (     MODE : string := "SLW" );  Port (     RST : in STD_LOGIC;     CLK : in STD_LOGIC;									-
RST ENO OUT1[15.0]  CLK OUT1[15.0]  EN IN1[15.0]  N WORD Input Number of bit positions to be shifted.  OUT1 WORD Output Destination address of the iN1 input.  EN IN1[15.0]  N[15.0]  entity ShiftRotate W e is generic (     MODE : string := "SLW" );  Port (     RST : in STD_LOGIC;     CLK : in STD_LOGIC;	M	ODE	SLW	String					
ShiftRotate_W_e  RST ENO CLK OUT1[150] CLKEN STOP EN IN1[150] N[150]  entity ShiftRotate_W_e is generic (     MODE : string := "SLW" ); Port (     RST : in STD_LOGIC;     CLK : in STD_LOGIC;									
RST ENO CLK OUT1[15.0] CLKEN STOP EN IN1[15.0] N[15.0]  entity ShiftRotate_W_e is generic (     MODE : string := "SLW" ); Port (     RST : in STD_LOGIC; CLK : in STD_LOGIC;	ShiftRo	otate_W_e				N	WORD	Input	
CLK OUT1[15.0] WORD Output Destination address of th value specific the IN1 input.  entity ShiftRotate W e is generic (     MODE : string := "SLW" ); Port (     RST : in STD_LOGIC;     CLK : in STD_LOGIC;					1				
clken stop EN IN1[15.0] N[15.0]  entity ShiftRotate_We is generic (     MODE : string := "SLW" ); Port (     RST : in STD_LOGIC;     CLK : in STD_LOGIC;	RS	Т				OUT1	WORD	Output	Destination
entity ShiftRotate_W_e is  generic (  MODE : string := "SLW"  );  Port (  RST : in STD_LOGIC;  CLK : in STD_LOGIC;	CLF	K		OUT1[150]					address of the
<pre>entity ShiftRotate_W_e is     generic (          MODE : string := "SLW"     );     Port (          RST : in STD_LOGIC;          CLK : in STD_LOGIC;</pre>	CLF	KEN							value specified at
<pre>entity ShiftRotate_W_e is     generic (          MODE : string := "SLW" );     Port (          RST : in STD_LOGIC;          CLK : in STD_LOGIC;</pre>	STC	OP							the IN1 Input.
<pre>entity ShiftRotate_W_e is     generic (</pre>	EN								
<pre>entity ShiftRotate_W_e is     generic (</pre>	IN1	[150]							
<pre>entity ShiftRotate_W_e is     generic (          MODE : string := "SLW" ); Port (          RST : in STD_LOGIC;          CLK : in STD_LOGIC;</pre>									
<pre>entity ShiftRotate_W_e is     generic (          MODE : string := "SLW" ); Port (          RST : in STD_LOGIC;          CLK : in STD_LOGIC;</pre>	1,11	00]							
<pre>entity ShiftRotate_W_e is     generic (          MODE : string := "SLW" ); Port (          RST : in STD_LOGIC;          CLK : in STD_LOGIC;</pre>									
<pre>generic (</pre>	inst				_				
<pre>generic (           MODE : string := "SLW" ); Port (         RST : in STD_LOGIC;         CLK : in STD_LOGIC;</pre>	<u> </u>								
<pre>generic (           MODE : string := "SLW" ); Port (         RST : in STD_LOGIC;         CLK : in STD_LOGIC;</pre>								<u> </u>	
<pre>MODE : string := "SLW" ); Port (     RST : in STD_LOGIC;     CLK : in STD_LOGIC;</pre>		en	tity	ShiftRo	tate V	√e <b>is</b>			
); Port (  RST : in STD_LOGIC; CLK : in STD_LOGIC;			ger	neric (	_	_			
<pre>Port (     RST : in STD_LOGIC;     CLK : in STD_LOGIC;</pre>				MODE :	strin	ng := "SLW'	п		
RST : in STD_LOGIC; CLK : in STD_LOGIC;			);						
CLK : in STD_LOGIC;			Por	rt (					
<del>-</del>				RST :	in S	STD_LOGIC;			
CLKEN: in STD_LOGIC;				CLK :	in S	STD_LOGIC;			
				CLKEN:	in S	STD_LOGIC;			
STOP : in STD_LOGIC;									
EN : in STD_LOGIC;									
<pre>IN1 : in STD_LOGIC_VECTOR ( 15 downto 0);</pre>									
<pre>N : in STD_LOGIC_VECTOR ( 15 downto 0);</pre>							VECTOR ( 15	downto	));
ENO : out STD_LOGIC;						_			
OUT1 : out STD_LOGIC_VECTOR ( 15 downto 0)				OUT1 :	out S	STD_LOGIC_V	VECTOR ( 15	downto	))
);									
<pre>end ShiftRotate_W_e;</pre>		en	<b>d</b> Shi	iftRotat	e_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
ParameterValueTypeMODESLWString	MODE	STRING	Generic	Defines Mode: "SLW": shift left
ShiftRotate_W_U_e	IN1	WORD	Input	Source value
IN1[150] OUT1[150]	N	WORD	Input	Number of bit positions to be shifted.
N[150] inst	OUT1	WORD	Output	Destination address of the value specified at the IN1 input.
entity ShiftRotate generic (				
MODE :	string := "SLW"			
Port (				
N : in	STD_LOGIC_VECTOR STD_LOGIC_VECTOR STD_LOGIC_VECTOR	( 15 down	to 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 S	Symbol		Parameter	Data type	Direction	Description
-	l	_	1	MODE	STRING	Generic	Defines Mode: "SLW": shift left
Parameter	Value	Type	1	EN	STD_LOGIC	Input	Enable input
MODE	SLW	String		ENO	STD_LOGIC	Output	Enable output
			1	IN1	DWORD	Input	Source value
ShiftRotate_DW_e	9		]	N	WORD	Input	Number of bit positions to be shifted.
RST CLK CLKEN STOP EN IN1[310] N[150]	OU T	ENO 1[310]		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)

uartus Schematic	Block S	ymbol		Parameter	Data type	Direction	Description
				MODE	STRING	Generic	Defines Mode: "SLW": shift left
Parameter	Value	Туре		IN1	DWORD	Input	Source value
MODE	SLW	String					
ShiftRotate_DW_I		124 01	7	N	WORD	Input	Number of bit positions to be shifted.
IN1[310] N[150] inst	OUT	[[310]		OUT1	DWORD	Output	Destination address of the value specified a the IN1 input.
e	_	ShiftRo	tate_D	W_U_e <b>is</b>			
	);	MODE :	strin	g := "SLW"			
	Port	IN1 :		TD_LOGIC_VE			
		OUTT1 ·	out S	rd_Logic_vi	ECTOR ( 31	downto 0)	

### 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)

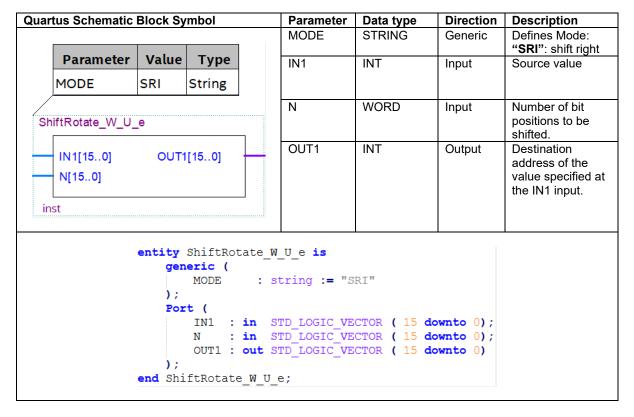
MODE STRING Generic Define	ription
	es Mode:
"SRI"	: shift right
	e input
	e output
	e value
	er of bit
position	ns to be
RST ENO Shifted	
Out   INT   Output   Destin	
	ss of the specified at
	1 input.
STOP	i iliput.
EN EN	
IN1[150]	
N[150]	
N[130]	
inst	
entity ShiftRotate W e is	
generic (	
MODE : string := "SRI"	
);	
Port (	
RST : in STD LOGIC;	
CLK : in STD LOGIC;	
<u> </u>	
<u> </u>	
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)	
);	
<pre>end ShiftRotate_W_e;</pre>	

#### 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)



### 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:
Parameter Value Type		070 10010		"SRI": shift right
	EN	STD_LOGIC	Input	Enable input
MODE SRI String	ENO	STD_LOGIC	Output	Enable output
ShiftRotate_DW_e	IN1 N	DINT	Input	Source value
	N	WORD	Input	Number of bit positions to be
RST ENO				shifted.
CLK OUT1[310]	OUT1	DINT	Output	Destination address of the
CLKEN				value specified at
STOP				the IN1 input.
EN				
IN 1[310]				
N[150]				
inst				
<u>\</u>				
	•		•	
<pre>entity ShiftRotate_D</pre>	W_e <b>is</b>			
generic (				
MODE : string	g := "SRI"			
);				
Port (				
RST : in S'	<pre>FD_LOGIC;</pre>			
CLK : in S	TD_LOGIC;			
CLKEN: in S	TD_LOGIC;			
STOP : in S'	TD_LOGIC;			
EN : in S'	TD LOGIC;			
IN1 : in S	rd Logic V	ECTOR ( 31	downto 0)	;
N : in S'	rd_logic_v	ECTOR ( 15	downto 0)	;
ENO : out S	TD_LOGIC;			
OUT1 : out S	rd Logic V	ECTOR ( 31	downto 0)	
);				
<b>end</b> 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)

Qua	rtus Schematic	Block Sy	mbol		Parameter	Data type	Direction	Description
	_				MODE	STRING	Generic	Defines Mode: "SRI": shift right
	Parameter	Value	Тур	2	IN1	DINT	Input	Source value
	MODE	SRI	String					
Sh	iftRotate_DW_L	J_e			N	WORD	Input	Number of bit positions to be shifted.
	N[150]	OUT1	[310]		OUT1	DINT	Output	Destination address of the value specified at the IN1 input.
in	st							
	ent	tity Sh gener		tate_DW	<b>N_</b> U_e <b>is</b>			
		_	_	string	g := "SRI"			
		Port	(					
		I	N1 :	in S	rD_LOGIC_V	ECTOR ( 3	1 downto 0	));
		N	: :	in S	rd_logic_v	ECTOR ( 1	5 downto 0	);
		0	UT1 :	out S	rD_LOGIC_V	ECTOR ( 3	1 downto 0	))
	end	); i Shift	Rotat	בו ושת ב	٠.			
	CII	DIIIIC	nocac		_~,			

# 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)

tus schematic	Block Sy	mbol		Parameter	Data type	Direction	Description
				MODE	STRING	Generic	Defines Mode:
D	Malus.	<b>T</b>					"SRW": shift right
Parameter	Value	Туре		EN	STD_LOGIC	Input	Enable input
MODE	SRW	String		ENO	STD_LOGIC	Output	Enable output
				IN1	WORD	Input	Source value
hiftRotate_W_e				N	WORD	Input	Number of bit
			7				positions to be
RST		ENO		OUT1	WORD	Output	shifted.
CLK	OUT1	[150]		0011	WORD	Output	Destination address of the
	0011	[100]					value specified at
CLKEN							the IN1 input.
STOP							are ner input.
EN							
IN1[150]							
N[150]							
nst			_				
				l .	I.		
	entity	ShiftRo	tate_W	_e <b>is</b>			
	gen	eric (					
		MODE	: 51	tring := "S	TATA II		
		ПООП		cring D	KW"		
	);	НОВЫ		oring b	KW"		
	); Por	t (			KW"		
		t ( RST :	in S	<pre>FD_LOGIC;</pre>	KW.		
		t ( RST : CLK :	in St	TD_LOGIC;	KW.		
		t ( RST : CLK : CLKEN:	in Stin Stin Stin	TD_LOGIC; TD_LOGIC; TD_LOGIC;	.FKM		
		t (  RST :  CLK :  CLKEN:  STOP :	in Sin Sin Sin Sin	<pre>FD_LOGIC; FD_LOGIC; FD_LOGIC; FD_LOGIC;</pre>	ĸw.		
		t (  RST :  CLK :  CLKEN:  STOP :  EN :	in Si in Si in Si in Si	<pre>FD_LOGIC; FD_LOGIC; FD_LOGIC; FD_LOGIC; FD_LOGIC;</pre>			
		t ( RST : CLK : CLKEN: STOP : EN : IN1 :	in Sin Sin Sin Sin Sin Sin Sin Sin Sin S	<pre>FD_LOGIC; FD_LOGIC; FD_LOGIC; FD_LOGIC; FD_LOGIC; FD_LOGIC;</pre>	CTOR ( 15 do		
		t ( RST : CLK : CLKEN: STOP : EN : IN1 : N :	in Sin Sin Sin Sin Sin Sin Sin Sin Sin S	PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC_VE PD_LOGIC_VE			
		RST : CLK : CLKEN: STOP : EN : IN1 : N :	in Sin Sin Sin Sin Sin Sin Sin Sin Sin S	PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC_VE PD_LOGIC_VE PD_LOGIC;	CTOR ( 15 do	ownto 0);	
	Por	RST : CLK : CLKEN: STOP : EN : IN1 : N :	in Sin Sin Sin Sin Sin Sin Sin Sin Sin S	PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC_VE PD_LOGIC_VE PD_LOGIC;	CTOR ( 15 do	ownto 0);	
	Por );	RST : CLK : CLKEN: STOP : EN : IN1 : N : ENO : OUT1 :	in Sin Sin Sin Sin Sin Sin Sin out Sin out Sin out Sin	PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC_VE PD_LOGIC_VE PD_LOGIC;	CTOR ( 15 do	ownto 0);	
	Por );	RST : CLK : CLKEN: STOP : EN : IN1 : N :	in Sin Sin Sin Sin Sin Sin Sin out Sin out Sin out Sin	PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC; PD_LOGIC_VE PD_LOGIC_VE PD_LOGIC;	CTOR ( 15 do	ownto 0);	

# 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 Sy	mbol		Parameter	Data type	Direction	Description
		-	]	MODE	STRING	Generic	Defines Mode: "SRW": shift right
Parameter	Value	Type		IN1	WORD	Input	Source value
MODE	SRW	String					
ShiftRotate_W_U_	<u>e</u>			N	WORD	Input	Number of bit positions to be shifted.
IN1[150] N[150] inst	OUT1	[150]		OUT1	WORD	Output	Destination address of the value specified at the IN1 input.
en	gene:	ric (		U_e <b>is</b>	SRW"		
	1	IN1 :	in ST	D_LOGIC_V	ECTOR ( 15 ECTOR ( 15 ECTOR ( 15	downto 0)	;
end	);	:Rotate			-		

### 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)

rtus Schematic	Block Sy	/mbol		Parameter	Data type	Direction	Description
				MODE	STRING	Generic	Defines Mode:
Parameter	Value	Tuno					"SRW": shift righ
Parameter	value	Туре		EN	STD_LOGIC	Input	Enable input
MODE	SRW	String		ENO	STD_LOGIC	Output	Enable output
/				IN1	WORD	Input	Source value
ChiftPotato DW				N	WORD	Input	Number of bit
ShiftRotate_DW_e	=		1				positions to be shifted.
RST		ENO		OUT1	WORD	Output	Destination
				0011	WORD	Output	address of the
- CLK	OUT1	[310]					value specified a
CLKEN							the IN1 input.
STOP							'
EN EN							
IN1[310]							
N[150]							
inst			_				
IIIot							
	ge		_	W_e is			
	);						
	Do	rt (					
	FU						
		RST :		TD_LOGIC;			
		RST :	in S	TD_LOGIC;			
	FO	RST : CLK : CLKEN:	in S	TD_LOGIC;			
		RST : CLK : CLKEN: STOP :	in S in S in S	TD_LOGIC; TD_LOGIC; TD_LOGIC;			
		RST : CLK : CLKEN: STOP : EN :	in S in S in S in S	TTD_LOGIC; TTD_LOGIC; TTD_LOGIC; TTD_LOGIC;	CTOP ( 31 dos	wnto O):	
		RST : CLK : CLKEN: STOP : EN : IN1 :	in S in S in S in S	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC_VE	CTOR ( 31 do		
		RST : CLK : CLKEN: STOP : EN : IN1 :	in S in S in S in S in S in S	TTD_LOGIC;  TTD_LOGIC;  TTD_LOGIC;  TTD_LOGIC;  TTD_LOGIC_VE  TTD_LOGIC_VE	CTOR ( 31 dow	_	
		RST : CLK : CLKEN: STOP : EN : IN1 : N :	in S in S in S in S in S out S	TTD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC_VE STD_LOGIC_VE STD_LOGIC_VE	CTOR ( 15 down	wnto 0);	
		RST : CLK : CLKEN: STOP : EN : IN1 : N : ENO : OUT1 :	in S in S in S in S in S out S	TTD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC_VE STD_LOGIC_VE STD_LOGIC_VE	•	wnto 0);	
	);	RST : CLK : CLKEN: STOP : EN : IN1 : N : ENO : OUT1 :	in S in S in S in S out S out S	TD_LOGIC; TD_LOGIC; TD_LOGIC; TD_LOGIC; TD_LOGIC_VE TD_LOGIC_VE TD_LOGIC_VE	CTOR ( 15 down	wnto 0);	

# 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)

Quar	tus Schematic	Block Sy	mbol		Parameter	Data type	Direction	Description
					MODE	STRING	Generic	Defines Mode: "SRW": shift right
	Parameter	Value	Туре		IN1	WORD	Input	Source value
	MODE	SRW	String		N	WORD	Input	Number of bit
Sh	iftRotate_DW_l	J_e				WORD	Input	positions to be shifted.
	IN1[310] N[150]	OUT1	[310]		OUT1	WORD	Output	Destination address of the value specified at the IN1 input.
in	st							
	е	ntity S	hiftRo	tate DV	√U e is			
		gene	ric (	_				
			MODE :	string	g := "SRW"			
		);						
		Port	(					
			IN1 :	in ST	TD LOGIC VE	CTOR ( 31	downto 0);	
			N :	in Si	rd Logic VE	CTOR ( 15	downto 0);	
					_	CTOR ( 31	_	
		);				•	•	
	е	nd Shif	tRotate	e 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)

EN ENO IN1	STRING  STD_LOGIC  STD_LOGIC  WORD	Input Output Input	Defines Mode: "WAND": AND operation Enable input Enable output
ENO IN1 IN2	STD_LOGIC WORD	Output	operation Enable input Enable output
ENO IN1 IN2	STD_LOGIC WORD	Output	Enable input Enable output
ENO IN1 IN2	STD_LOGIC WORD	Output	Enable output
IN1 IN2	WORD		
IN2		Input	First value of logic
	WORD		operation
0.1.=		Input	Second value of logic
Q : :== /		•	operation
OUT1	WORD	Output	Result word of the
			logic operation
	WAND"		
_			
_			
_			
_			
_			
_	_		
n STD_LOG	IC_VECTOR (	15 downto	0);
ut STD_LOG	IC;		
ut STD LOG	IC VECTOR (	15 downto	0)
_	_		
e:			
	n STD_LOG: ut STD_LOG:	tring := "WAND"  n STD_LOGIC; n STD_LOGIC; n STD_LOGIC := '1'; n STD_LOGIC := '0'; n STD_LOGIC := '1'; n STD_LOGIC_VECTOR ( n STD_LOGIC_VECTOR ( ut STD_LOGIC; ut STD_LOGIC_VECTOR (	tring := "WAND"  n STD_LOGIC; n STD_LOGIC; n STD_LOGIC := '1'; n STD_LOGIC := '0'; n STD_LOGIC := '1'; n STD_LOGIC_VECTOR ( 15 downton STD_LOGIC_VECTOR ( 15 downtout STD_LOGIC; ut STD_LOGIC_VECTOR ( 15 downtout STD_LOGIC;

# 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)

Qua	artus Schema	tic Block	Symbol		Parameter	Data type	Direction	Description
	Parameter	Value	Туре		MODE	STRING	Generic	Defines Mode: "WAND": AND operation
10	MODE /ordLogic_W_U_0	WAND	String		IN1	WORD	Input	First value of logic operation
VV	IN 1[150]	OUT1[	15 01		IN2	WORD	Input	Second value of logic operation
	- IN2[150]	0011	.130]		OUT1	WORD	Output	Result word of the logic operation
ir	nst							
		_			_W_U_e <b>is</b>			
		ge	neric (					
			MODE		: string	= "WAND"		
		);						
		Po	rt (					
			IN1		_		( 15 downto	
			IN2				( 15 downto	
		);		: 0	ut STD_LOG	IC_VECTOR	( 15 downto	0)
		end Wo	rdLogic	_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:
Parameter Value Type				"WAND": AND
	EN	STD LOCIC	Innut	operation
MODE WAND String	ENO	STD_LOGIC STD_LOGIC	Input Output	Enable input Enable output
	IN1	DWORD	Input	First value of logic
WordLogic_DW_e	1111	DWOND	iliput	operation
	IN2	DWORD	Input	Second value of
RST ENO			•	logic operation
— CLK OUT1[310] <del>- </del>	OUT1	DWORD	Output	Result double word
CLKEN				of the logic
STOP				operation
EN				
IN1[310]				
IN2[310]				
inst				
<pre>entity WordLogic_DW_</pre>	_e <b>is</b>			
generic (				
MODE : strin	ng := "WAN	D <b>"</b>		
);				
Port (				
	STD_LOGIC;			
	STD_LOGIC;			
CLKEN: in	STD_LOGIC	:= '1';		
STOP : in	STD_LOGIC	:= '0';		
EN : in S	STD LOGIC	:= '1';		
IN1 : in 8	STD LOGIC	VECTOR ( 31	downto 0	));
		VECTOR ( 31		
	STD LOGIC;			
OUT1 : out		VECTOR ( 31	downto 0	))
);				
<pre>end WordLogic DW e;</pre>				

# 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
	Parameter	Value	Туре		MODE	STRING	Generic	Defines Mode: "WAND": AND operation
,	MODE	WAND	String		IN1	DWORD	Input	First value of logic operation
W	ordLogic_DW_U	_e			IN2	DWORD	Input	Second value of logic operation
in	IN1[310] IN2[310]	OUT1[	310]		OUT1	DWORD	Output	Result double word of the logic operation
	e	gene	ric (	_	<pre>J_U_e is string :=</pre>	"WAND"		
		);						
	ei		IN1 : IN2 : OUT1 :	in out	STD_LOGIC_ STD_LOGIC_	VECTOR ( 31 VECTOR ( 31 VECTOR ( 31	downto	);

# 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
Parameter Value Type	EN	STD LOGIC	Input	Enable input
MODE WOR String	ENO	STD LOGIC	Output	Enable output
	IN1	WORD	Input	First value of logic
WordLogic_W_e			-	operation
RST ENO	IN2	WORD	Input	Second value of logic operation
CLK OUT1[150]	OUT1	WORD	Output	Result word of the
			- '	logic operation
CLKEN				
STOP				
EN EN				
IN1[150]				
IN2[150]				
inst				
entity WordLogic	_W_e <b>is</b>			
generic (				
MODE : st	tring := "\	WAND"		
);				
Port (				
RST : ii	n STD LOG	IC;		
CLK : i	n STD LOG	IC;		
CLKEN: i	_	IC := '1';		
STOP : in	_	IC := '0';		
	n STD LOG			
IN1 : in	_	IC VECTOR (	15 downto	0).
IN1 : I		IC_VECTOR (		
			15 downto	J 0],
	ut STD_LOG		45 4	
	ut STD_LOG	IC_VECTOR (	15 downto	) U)
);				
<pre>end WordLogic_W_e</pre>	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)

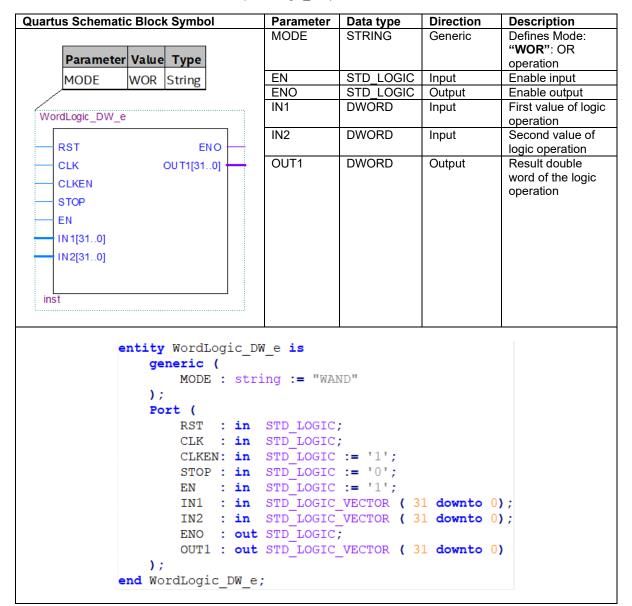
Qua	rtus Schemat	tic Block	c Symbol		Parameter	Data type	Direction	Description
					MODE	STRING	Generic	Defines Mode: "WOR": OR operation
	Parameter	Value	Туре		IN1	WORD	Input	First value of logic
	MODE	WOR	String					operation
10/	ord ogic W II d				IN2	WORD	Input	Second value of logic operation
in	ordLogic_W_U_e IN1[150] IN2[150]		1[150]		OUT1	WORD	Output	Result word of the logic operation
	•	entity	WordLo	gic	W_U_e is			
		ge	neric (					
			MODE		: string :	= "WOR"		
		);						
		Po	rt (					
						_	( 15 downto	
					_	_	( 15 downto	
			OUT1	: 01	it STD_LOG	C_VECTOR	( 15 downto	0)
	•	); end Wo	rdLogic	<b>W</b> _t	J_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)



# 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)

<b>Quartus S</b>	chematic	Block S	Symbol		Parameter	Data type	Direction	Description
Pa	rameter	Value	Type		MODE	STRING	Generic	Defines Mode: "WOR": OR operation
MOI	DE	WOR	String		IN1	DWORD	Input	First value of logic operation
WordLo	gic_DW_U_	_e		$\neg$	IN2	DWORD	Input	Second value of logic operation
	[310] [310]	OUT1	[310]		OUT1	DWORD	Output	Result double word of the logic operation
	en	gene	JordLo eric ( MODE	_	W_U_e is string :=	"WOR"		
		Port	(					
			IN1	: in	STD_LOGIC_	VECTOR (	31 downto	0);
				: in		_		
			OUT1	: out	STD_LOGIC_	VECTOR (	31 downto	0)
	en	); d Word	lLogic	_D <b>W</b> _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:				
Parameter Value Type				"WXOR": XOR				
		070 10010		operation				
MODE WXOR String	EN	STD_LOGIC	Input	Enable input				
	ENO	STD_LOGIC	Output	Enable output				
WordLogic_W_e	IN1	WORD	Input	First value of logic operation				
RST ENO	IN2	WORD	Input	Second value of logic operation				
CLK OUT1[150]	OUT1	WORD	Output	Result word of the				
CLKEN				logic operation				
STOP								
EN								
IN1[150]								
IN2[150]								
1112[10::0]								
inst								
\								
generic (  MODE : Si ); Port (  RST : ir CLK : ir CLKEN: ir	<pre>MODE : string := "WAND" ); Port (</pre>							
	n STD_LOG							
	n STD_LOG							
IN1 : i:	n STD_LOG	IC_VECTOR (	15 downto	0);				
IN2 : <b>i</b> :	n STD_LOG	IC_VECTOR (	15 downto	0);				
ENO : or	ut STD LOG	IC;						
OUT1 : or	ut STD LOG	IC VECTOR (	15 downto	0)				
);	_							
end WordLogic W e	e:							
	- r							

# 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)

Qua	Quartus Schematic Block Symbol				Parameter	Data type	Direction	Description
	Parameter	Value	Туре		MODE	STRING	Generic	Defines Mode: "WXOR": XOR operation
	MODE	WXOR	String		IN1	WORD	Input	First value of logic operation
W	ordLogic_W_U_	<b>e</b>			IN2	WORD	Input	Second value of logic operation
ir	IN1[150] IN2[150] st	OUT1[	150]		OUT1	WORD	Output	Result word of the logic operation
		entity	WordLo	ogic_	_W_U_e <b>is</b>			
		ge:	neric	(				
			MODE		: string	= "WXOR"		
		);						
		Po	rt (					
			IN1	: i:	n STD_LOG	C_VECTOR	( 15 downto	0);
			IN2	: i:	n STD_LOG	C_VECTOR	( 15 downto	0);
			OUT1	: 01	ut STD_LOG	C_VECTOR	( 15 downto	0)
		);			_	_		
		end Wo	rdLogio	w t	J е;			
			_		_			

# 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:
Parameter Value Type				"WXOR": XOR
	ENI	CTD I OCIC	lana est	operation
MODE WXOR String	ENO ENO	STD_LOGIC STD_LOGIC	Input Output	Enable input Enable output
	IN1	DWORD	Input	First value of logic
WordLogic_DW_e	IINI	DWORD	iliput	operation
	IN2	DWORD	Input	Second value of
RST ENO			·	logic operation
CLK OUT1[310]	OUT1	DWORD	Output	Result double word
CLKEN				of the logic
				operation
STOP				
EN				
IN1[310]				
IN2[310]				
inst				
· · · · · · · · · · · · · · · · · · ·				
<pre>entity WordLogic_DW</pre>	_e <b>is</b>			
generic (				
MODE : stri	ng := "WAN	D"		
);				
Port (				
	STD_LOGIC;			
	STD_LOGIC;			
	STD_LOGIC			
	STD_LOGIC	•		
	STD_LOGIC			
		VECTOR ( 31		
		VECTOR ( 31	downto	0);
	STD_LOGIC;			
OUT1 : out	STD_LOGIC_	VECTOR ( 31	downto	0)
);				
<pre>end WordLogic_DW_e;</pre>				

# 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)

<b>Quartus Schematic</b>	Block Sy	/mbol		Parameter	Data type	Direction	Description
Parameter	Value	Туре	•	MODE	STRING	Generic	Defines Mode: "WXOR": XOR operation
MODE	WXOR	String		IN1	DWORD	Input	First value of logic operation
WordLogic_DW_U_	e			IN2	DWORD	Input	Second value of logic operation
IN1[310] IN2[310] inst	OUT1[	310]		OUT1	DWORD	Output	Result double word of the logic operation
en	tity Wo	ordLog	gic DV	V U e <b>is</b>			
	gene	ric (	_				
	1	MODE	:	string :=	"WXOR"		
	);						
	Port	(					
		IN1	in	STD LOGIC	VECTOR ( 3	1 downto	0);
			in		VECTOR ( 3		
	(	OUT1 :	out	STD LOGIC			
	);						
en	d Word	Logic	D <b>W</b> 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	FM 352-5 Symbolic	Description
	Trains	Name	
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
. 2 00	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
1 5 00	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,
FB 94	EROA Rithport22	BITINSERT32	unlatched logic Inserts a bit into a DINT, 32-Bits
1 D 34	FB94_BitInsert32 FB94_BitInsert32_U	- DITINGER 132	Inserts a bit into a DINT, 32-bits Inserts a bit into a DINT, 32-bits, unlatched
	า มอ <del>า</del> _มแแจ <del>ะ</del> แจะ_บ		logic
FB 95	FB95_BitInsert16	BITINSERT16	Inserts a bit into an INT, 16-Bits
FB 95	FB95_BitInsert16 FB95_BitInsert16_U	BITINSERT16	

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.

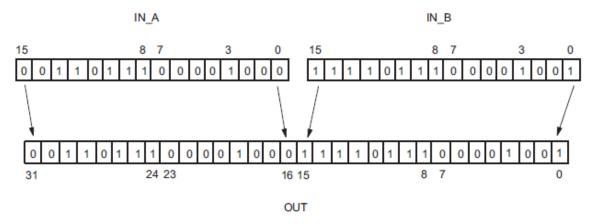
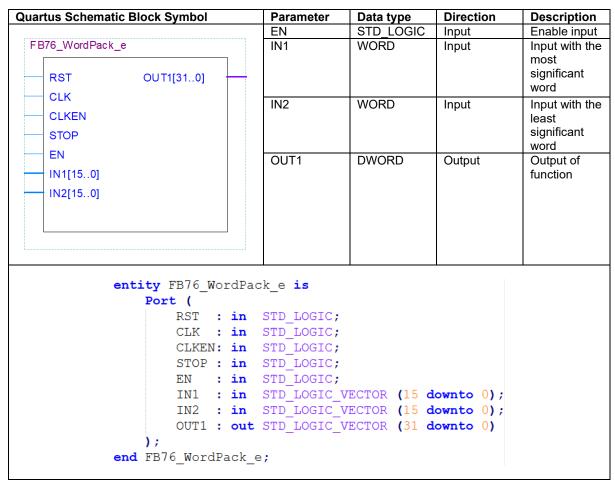


Figure 1: Example of WordPack

Table 55: TM FAST function WordPack (FB76\_WordPack)



# 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.

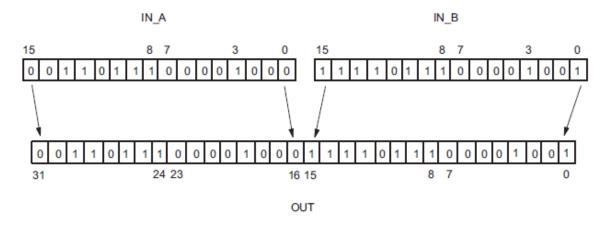
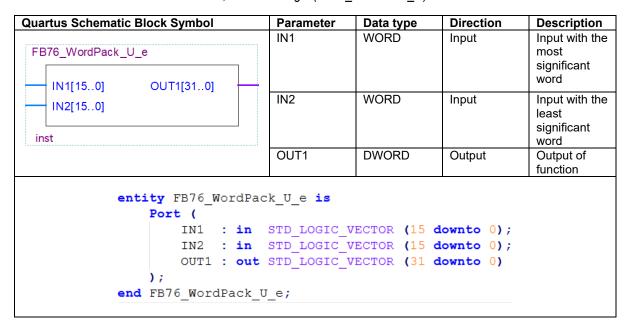


Figure 2: Example of WordPack

Table 56: TM FAST function WordPack, unlatched logic (FB76\_WordPack\_U)



# 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.

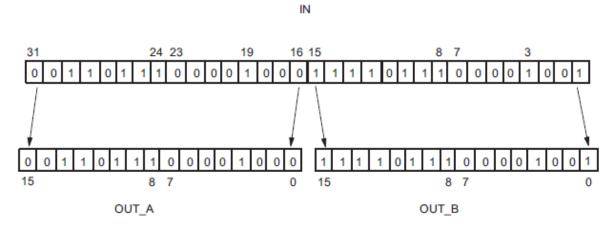


Figure 3: Example of WordCast

Table 57: TM FAST function WordCast (FB77\_WordCast)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description	
	EN	STD_LOGIC	Input	Enable input	
FB77_WordCast_e	IN1	STD_LOGIC	Input	Input of	
		_VECTOR		function	
RST OUT1[150]	OUT1	STD_LOGIC	Output	Output with	
CLK OUT2[150]		_VECTOR		the most	
				significant word	
CLKEN	OUT2	STD LOGIC	Output	Output with	
STOP	00.2	VECTOR	Catpat	the least	
— EN		_		significant	
IN1[310]				word	
IN [S10]					
entity FB77_WordCas	t_e <b>is</b>				
:	STD LOGIC:				
RST : in STD_LOGIC; CLK : in STD LOGIC;					
CLKEN: in STD LOGIC;					
STOP : in	<del></del>				
	STD LOGIC;				
IN1 : in	STD LOGIC VE	CTOR (31 dow	mto 0);		
OUT1 : out	STD_LOGIC_VE	CTOR (15 dow	mto 0);		
OUT2 : out	STD_LOGIC_VE	CTOR (15 dow	mto 0) );		
<pre>end FB77_WordCast_e;</pre>					

# 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.

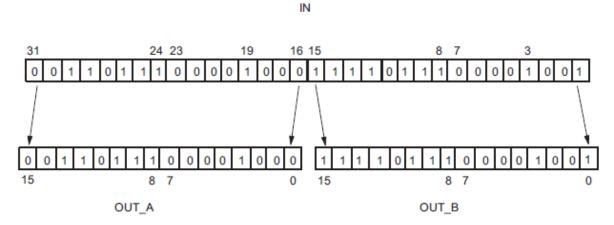


Figure 4: Example of WordCast\_U

Table 58: TM FAST function WordCast, unlatched logic (FB77\_WordCast\_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
FB77_WordCast_U_e	IN1	STD_LOGIC _VECTOR	Input	Input of function
OUT1[150] OUT2[150]	OUT1	STD_LOGIC _VECTOR	Output	Output with the most significant word
inst	OUT2	STD_LOGIC _VECTOR	Output	Output with the least significant word
OUT1 : out	STD_LOGIC_VE STD_LOGIC_VE STD_LOGIC_VE	CTOR (31 down) CTOR (15 down) CTOR (15 down)	nto 0);	

#### 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
                                               STD_LOGIC
                                  ΕN
                                                            Input
                                                                         Enable input
 FB78_BitSum_e
                                  IN1
                                               STD_LOGIC_
                                                                         Variable in
                                                            Input
                                               VECTOR
                                                                         which bits are
    RST
                OUT1[15..0]
                                                                         counted
                                  OUT1
                                               STD_LOGIC_
                                                            Output
                                                                         Value output
    CLK
                                               VECTOR
    CLKEN
    STOP
    ΕN
    IN1[31..0]
  inst
            entity FB78 BitSum e is
                 Port (
                      RST
                           : in STD LOGIC;
                      CLK : in STD LOGIC;
                      CLKEN: in STD LOGIC;
                      STOP : in STD_LOGIC;
                            : 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
FB78_BitSum_U_e	IN1	STD_LOGIC_ VECTOR	Input	Variable in which bits are counted
IN1[310] OUT1[150]	OUT1	STD_LOGIC_ VECTOR	Output	Value output
<pre>entity FB78_BitSu</pre>	m_U_e <b>is</b>			
Port (				
IN1 : in	STD_LOGIC	_VECTOR (31	downto 0);	
OUT1 : ou	t 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.

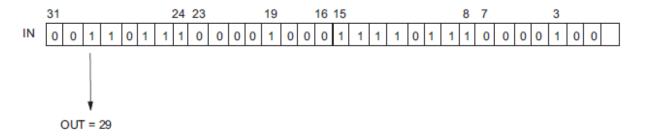


Figure 5: Example of Encode

Table 61: TM FAST function Encode (FB79\_Encode)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
FB79_Encode_e	IN1	STD_LOGIC_ VECTOR	Input	Variable in which bits are counted
CLK CLKEN STOP EN IN1[310]	OUT1	STD_LOGIC_ VECTOR	Output	Value output
entity FB79_Encode	e_e <b>is</b>			
Port (				
RST : in	STD_LOGIC;			
CLK : in	STD LOGIC;			
CLKEN: in	STD LOGIC;			
	STD LOGIC;			
	STD LOGIC;			
IN1 : in		VECTOR (31	downto 0):	
		VECTOR (15		
);	<u>-</u> <u>-</u> -			
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.

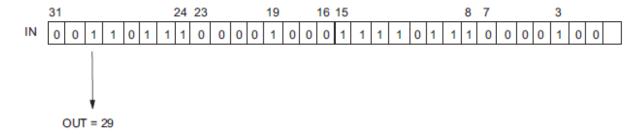


Figure 6: Example of Encode

Table 62: TM FAST function Encode, unlatched logic (FB79\_Encode\_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description			
FB79_Encode_U_e	IN1	STD_LOGIC_ VECTOR	Input	Variable in which bits are counted			
IN1[310] OUT1[150]	OUT1	STD_LOGIC_ VECTOR	Output	Value output			
	entity FB79_Encode_U_e is						
Port (							
<pre>IN1 : in STD_LOGIC_VECTOR (31 downto 0); OUT1 : out STD_LOGIC_VECTOR (15 downto 0) ); end FB79 Encode U e;</pre>							
1273_2110040_0	,						

#### 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.

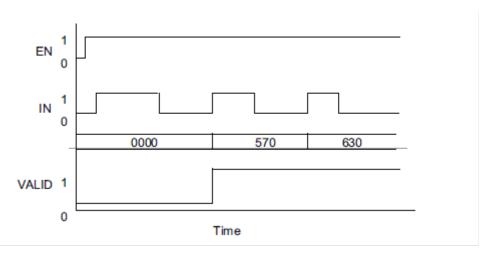


Figure 7: Example of FB80\_Period32, FB81\_Period16

#### If Phases were used:

(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<sup>32</sup>-1) microseconds. Periods greater than 2,147,483,647 (2<sup>31</sup>-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<sup>16</sup>-1) microseconds. Periods greater than 32767 (2<sup>15</sup>-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:

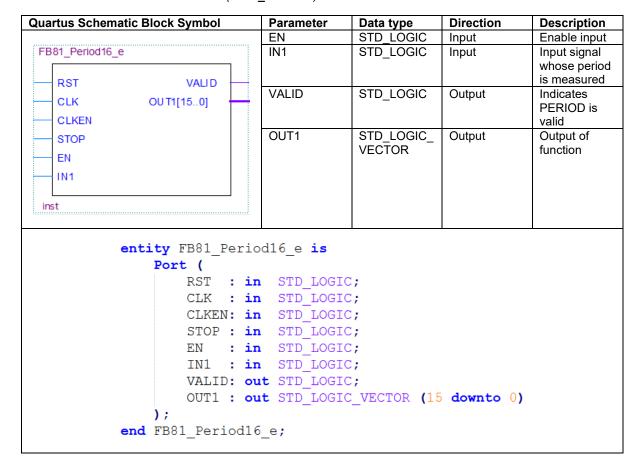
F = F\_CLK\_USER \* [ (OUT1) / (periode \* PHASES) ]

- 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)

EN			Description
EIN	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
d32_e <b>is</b>			
STD_LOGIC	;		
STD_LOGIC	;		
STD LOGIC	;		
STD_LOGIC	;		
STD_LOGIC	;		
STD_LOGIC	;		
		downto 0)	
_	_		
_e;			
	VALID  VALID  OUT1  OUT1	VALID STD_LOGIC  VALID STD_LOGIC  OUT1 STD_LOGIC_ VECTOR  d32_e is  STD_LOGIC; STD_LOGIC_VECTOR (31)	VALID STD_LOGIC Output  OUT1 STD_LOGIC_ Output  VECTOR Output  OUT1 STD_LOGIC_ Output  VECTOR  OUT1 STD_LOGIC_ Output  VECTOR  OUT1 STD_LOGIC_ Output  VECTOR Output  VECTO

Table 64: TM FAST function Period16 (FB81\_Period16)



### 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.

#### If Phases were used:

```
(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<sup>-4</sup> Hz to 500 kHz (periode durations of 2 to 4,294,967,295 (2<sup>32</sup>-1) microseconds). Frequencies lower than 4,66\*10<sup>-4</sup> Hz will appear negative. VALID will be 0 if the periode duration exceeds 4,294,967,295 (2<sup>31</sup>-1) microseconds.

FB83\_Freq16 is used to measure frequencies of 15,26 Hz to 500 kHz (periode durations of 2 to 65535 (2<sup>16</sup>-1) microseconds). Frequencies greater than 32767 (2<sup>15</sup>-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.

#### 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 * 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_QU ANTITY =14)	Phases not used PHASES =1 (PHASE_QUANTITY = 0)					
	F_CLK_USE R =15MHz	F_CLK_USE R = 5MHz	F_CLK_USE R= 15MHz	F_CLK_USE R= 25MHz	F_CLK_USE R= 50MHz	F_CLK_USE R= 75MHz	
Min. Frequency (FB82_Freq 32)	4,66*10 <sup>-4</sup> Hz [1000000/ (2 <sup>31</sup> -1) Hz]	2,33*10 <sup>-3</sup> Hz [5000000/ (2 <sup>31</sup> -1) Hz]	6,98 *10 <sup>-3</sup> Hz [15000000/ (2 <sup>31</sup> -1) Hz]	1,16 *10 <sup>-2</sup> Hz [25000000/ (2 <sup>31</sup> -1) Hz]	2,33 *10 <sup>-2</sup> Hz [50000000/ (2 <sup>31</sup> -1) Hz]	3,49 *10 <sup>-2</sup> Hz [75000000/ (2 <sup>31</sup> -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 <sup>15</sup> -1) Hz]	76,3 Hz [5000000/ (2 <sup>15</sup> -1) Hz]	228,9 Hz [15000000/ (2 <sup>15</sup> -1) Hz]	381,5 Hz [25000000/ (2 <sup>15</sup> -1) Hz]	763 Hz [50000000/ (2 <sup>15</sup> -1) Hz]	1144,4 Hz [75000000/ (2 <sup>15</sup> -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)

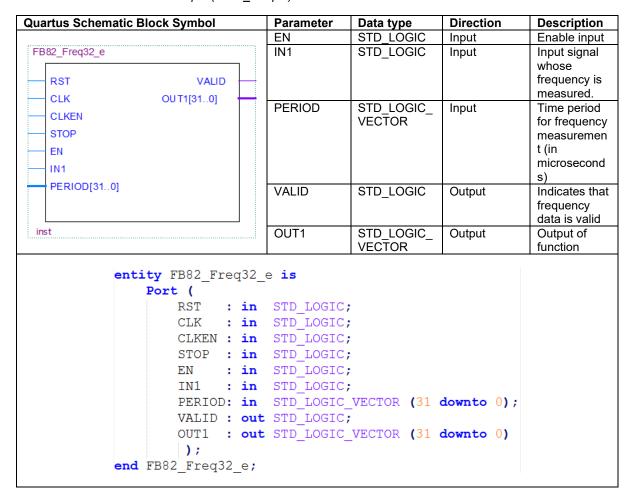
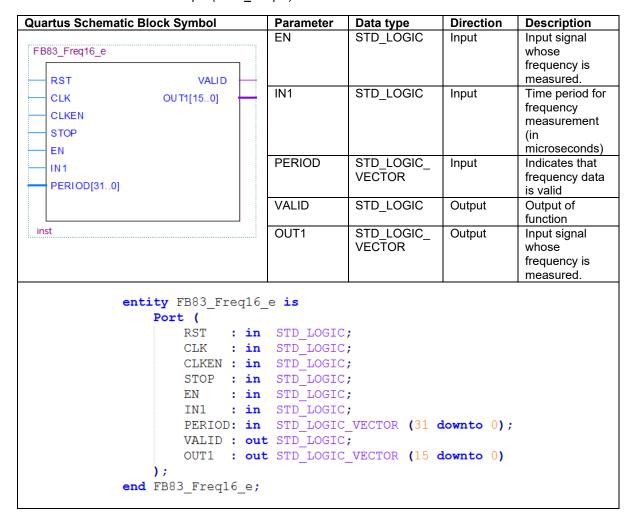


Table 66: TM FAST function Freq16 (FB83\_Freq16)



## 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 S	Symbol	Parameter	Data type	Direction	Description
		EN	STD_LOGIC	Input	Enable input
FB124_Shift_e		SH_CLK	STD_LOGIC	Input	Rising edge
Det DAT	A OUTTO O				that shifts the
	A_OUT[00]				data through
CLK					the shift register
CLKEN		SH_RESET	STD_LOGIC	Input	Resets all
EN					stages of the
SH_CLK					shift register.
SH_RESET		DATA_IN	STD_LOGIC	Input	Data input for
D ATA_IN[00]		- · - · · - · ·	_VECTOR		the shift register
D_LENGTH [150]		D_LENGTH	STD_LOGIC	Input	Length of the
inst		DATA OUT	_VECTOR	0 1 1	shift register
	<i>i</i>	DATA_OUT	STD_LOGIC	Output	Output of the
			_VECTOR		shift register
entity I	B124_Shift_e	is			
Port		15			
	RST : in	STD LOGIC;			
	CLK : in	STD LOGIC;			
	CLKEN : in	STD LOGIC;			
	EN : in	STD LOGIC;			
	SH CLK : in	STD LOGIC;			
	SH RESET: in	<del>-</del>			
	DATA IN : in			ownto 0):	
	_		-		
):					
end FB12	24 Shift e:				
); end FB12	D_LENGTH: in DATA_OUT: ou 24_Shift_e;				

Table 68: TM FAST function Shift2 (FB125\_Shift2)

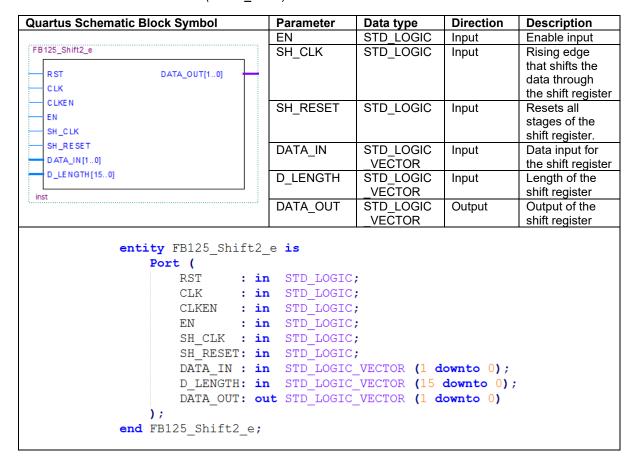


Table 69: TM FAST function Shift4 (FB126\_Shift4):

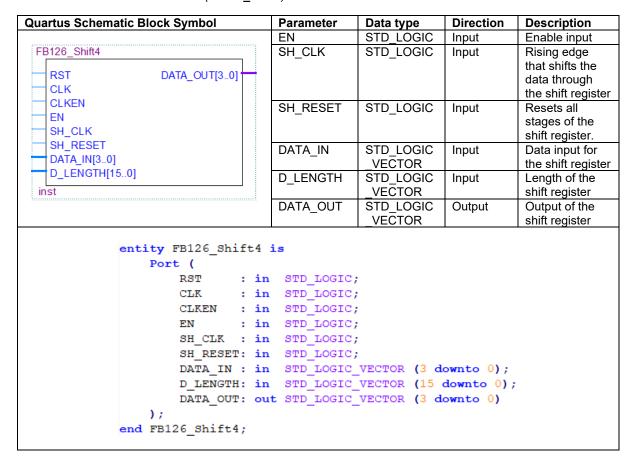


Table 70: TM FAST function Shift8 (FB127\_Shift8):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
FB127_Shift8_e	SH_CLK	STD_LOGIC	Input	Rising edge
RST DATA_OUT[70]				that shifts the
CLK				data through
				the shift register
CLKEN	SH_RESET	STD_LOGIC	Input	Resets all
EN				stages of the
SH_CLK				shift register.
SH_RESET	DATA_IN	STD_LOGIC	Input	Data input for
DATA_IN[70]		_VECTOR		the shift register
D_LENGTH[150]	D_LENGTH	STD_LOGIC	Input	Length of the
		_VECTOR		shift register
inst	DATA_OUT	STD_LOGIC	Output	Output of the
		_VECTOR		shift register
77107 01:510				
entity FB127_Shift8	_e <b>1s</b>			
Port (	amp			
	sTD_LOGIC;			
CLK : ii				
CLKEN : ii	<del>-</del>			
EN : iı				
SH_CLK : ii				
	n STD_LOGIC;			
	sTD_LOGIC_			
	stD_LOGIC_			
DATA_OUT: OI	it STD_LOGIC_	VECTOR (7 do	wnto 0)	
);				
<pre>end FB127_Shift8_e;</pre>				

Table 71: TM FAST function Shift16 (FB85\_Shift16):

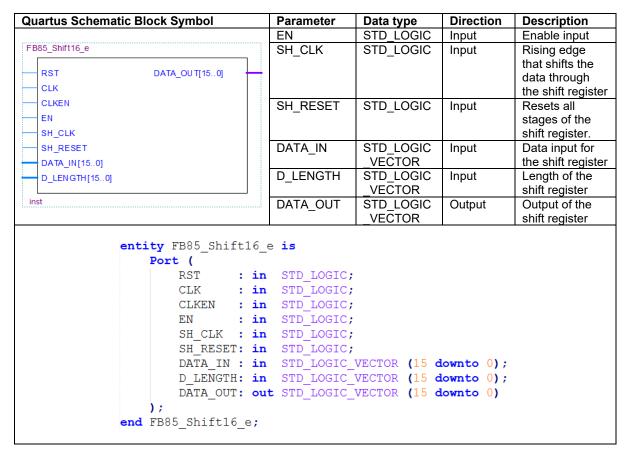
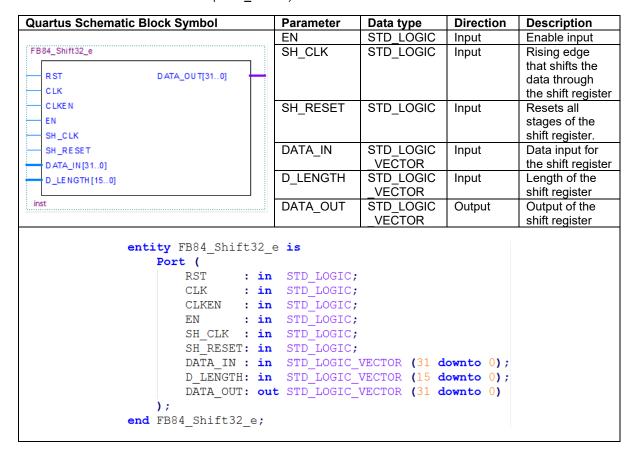


Table 72: TM FAST function Shif32t (FB84\_Shift32):



## 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.

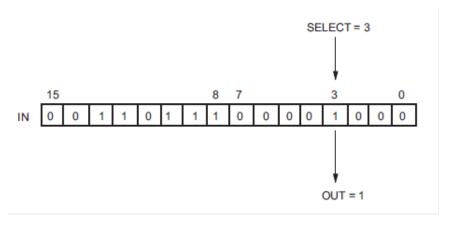
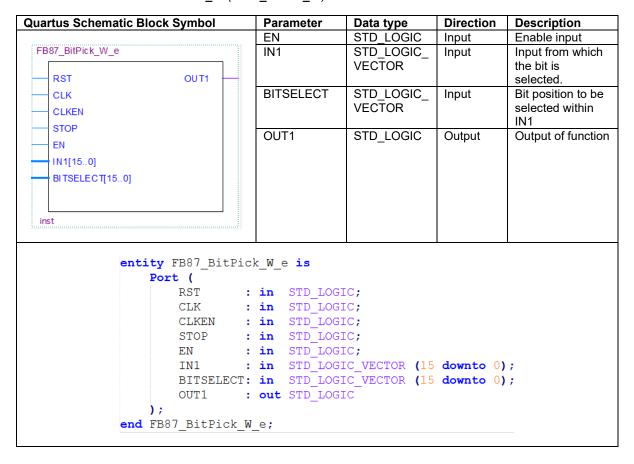


Figure 8: Example of FB86\_BitPick\_DW, FB87\_BitPick\_W

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
FB86_BitPick_DW_e	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
CLK CLKEN	BITSELECT	STD_LOGIC_ VECTOR	Input	Bit position to be selected within IN1
STOP EN	OUT1	STD_LOGIC	Output	Output of function
IN1[310]				
BITSELECT[150]				
inst				
entity FB86_BitPi	ck_DW_e <b>is</b>			
Port (	: in STD LO	CTC.		
	: in STD_LO			
	_			
	: in STD_LO			
	: in STD_LO	· · · · · ·		
	: in STD_LO			
		GIC_VECTOR (3)		
		GIC_VECTOR (1	downto 0	) <i>i</i>
	: out STD_LO	GIC		
); end FB86_BitPick_	DW_e;			

Table 74: TM FAST function BitPick\_W (FB87\_BitPick\_W)



## 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.

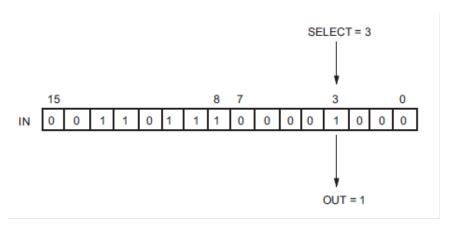


Figure 9: Example of FB86\_BitPick\_DW\_U, FB87\_BitPick\_W\_U

Table 75: TM FAST function BitPick\_DW\_U (FB86\_BitPick\_DW\_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
FB86_BitPick_DW_U_e	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
IN1[310] OUT1	BITSELECT	STD_LOGIC_ VECTOR	Input	Bit position to be selected within IN1
inst	OUT1	STD_LOGIC	Output	Output of function
entity FB86_BitPic	ck_DW_U_e is			
IN1	: in STD LOG	GIC VECTOR (31	downto 0)	;
	: in STD_LOG : out STD_LOG	GIC_VECTOR (15	downto 0)	;
);				
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
FB87_BitPick_W_U_e	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
IN1[150] OUT1	BITSELECT	STD_LOGIC_ VECTOR	Input	Bit position to be selected within IN1
inst	OUT1	STD_LOGIC	Output	Output of function
entity FB87 BitPic	k W U e <b>is</b>			
Port (				
IN1 :	in STD_LOGI	C_VECTOR (15	downto 0)	;
BITSELECT:	in STD LOGI	C VECTOR (15	downto 0)	;
OUT1 :	out STD LOGI	rc		
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	_			
);				

## 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.

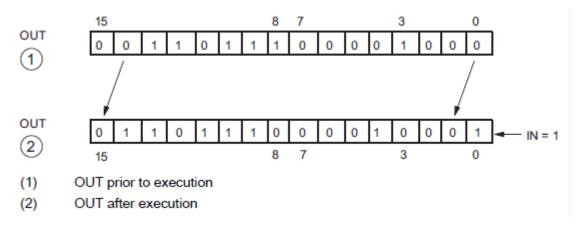
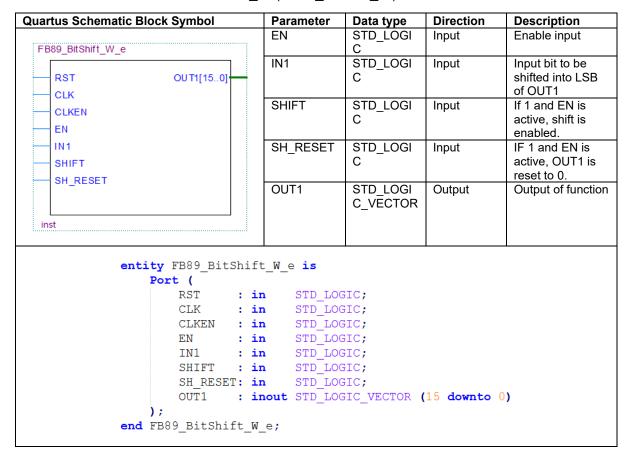


Figure 10: Example of FB88\_BitShift\_DW, FB89\_BitShift\_W

Table 77: TM FAST function BitShift\_DW (FB88\_BitShift\_DW)

Quartus Schematic Bl	ock Symbol	Parameter		Direction	Description
		EN	STD_LOGIC	Input	Enable input
FB88_BitShift_DW_e		IN1	STD_LOGIC	Input	Input bit to be
RST	OUT1[310]				shifted into LSB of OUT1
CLK		SHIFT	STD_LOGIC	Input	If 1 and EN is
CLKEN					active, shift is
EN					enabled.
IN1		SH_RESE	T STD_LOGIC	Input	IF 1 and EN is
SHIFT					active, OUT1 is
SH_RESET		OUT4	CTD LOCIC	O street	reset to 0.
		OUT1	STD_LOGIC _VECTOR	Output	Output of function
			_VECTOR		
inst					
	. =====================================				
	ty FB88_BitSh	iit_Dw_e 1	5		
	Port (		T.00T.0		
		in STD_	<del>-</del>		
		in STD_	-		
	CLKEN :	_	LOGIC;		
		in STD_	<del>-</del>		
		in STD_	-		
		_	LOGIC;		
	SH_RESET:	_	LOGIC;		
	OUT1 :	inout STD_	LOGIC_VECTOR	(31 downto	0)
	);				
end	FB88_BitShift	_DW_e;			

Table 78: TM FAST function BitShift\_W (FB89\_BitShift\_W):



## 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.

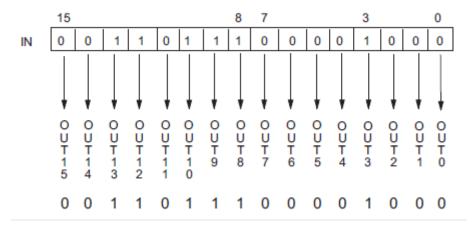


Figure 11: Example of FB90\_BitCast\_DW, FB91\_BitCast\_W

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
FB90_BitCast_DW_e	IN1	STD_LOGIC_ VECTOR	Input	Input to be converted
RST OUT0	OUTn	STD_LOGIC	Output	Output of function
CLK OUT1				
CLKEN OUT2				
STOP OUT3				
EN OUT4				
IN1[310] OUT5				
00 Т6				
OU T7				
оитв —				
00 Т9				
OUT10				
OUT11				
OUT12				
OUT13 -				
OUT14				
OUT15				
OUT16				
OUT17				
OUT18 -				
OUT19				
OUT20				
OUT21				
OUT22				
OUT23				
OUT24				
OUT25				
OUT26				
OUT27				
OUT28				
OUT29				
ОПТ30				
ОПТ31				

```
entity FB90 BitCast DW e is
    Port (
             : in STD LOGIC;
       RST
       CLK : in STD LOGIC;
        CLKEN : in STD LOGIC;
        STOP : in STD LOGIC;
       EN : in STD_LOGIC;
            : in STD_LOGIC_VECTOR (31 downto 0);
       IN1
       OUTO : 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):

uartus Schematic	Block Symbol	Parameter	Data type	Direction	Description
FB91_BitCast_W_		EN IN1	STD_LOGIC STD_LOGIC	Input	Enable input Input to be
rbsi_bitcast_w_t		IIN I	VECTOR	Input	converted
RST	оито	OUTn	STD_LOGIC	Output	Output of
			_	'	function
CLK	OUT1				
C LKE N	OUT2				
STOP	OUT3				
— EN	OUT4				
IN1[150]	OUT5				
' '	оит6				
	OU T7				
	OUT8				
	OUT9				
	OUT10				
	OUT11				
	OUT12				
	OUT13				
	OUT14				
	OUT15				
	Port ( RST : in	n STD LOGIC	;		
	CLK : in				
	CLKEN: in	sTD_LOGIC	;		
	STOP : in	_			
	EN : ir	_	•		
	IN1 : in		_VECTOR (15	downto 0);	
		it STD_LOGIC			
		it SID_LOGIC			
		at STD_LOGIC			
		at STD LOGIC			
		it STD_LOGIC			
	OUT6 : 01	ut STD_LOGIC	;		
		it STD_LOGIC			
		at STD_LOGIC			
		at STD_LOGIC			
		at STD_LOGIC			
		it STD_LOGIC			
		it STD_LOGIC it STD LOGIC			
		it SID_LOGIC			
		it STD_LOGIC			
	);				
en	d FB91_BitCast_	We;			

## 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.

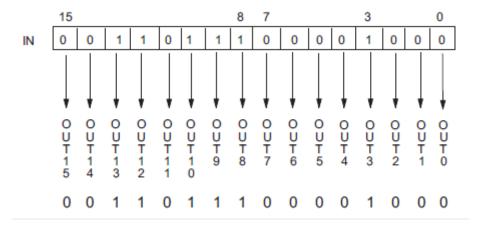


Figure 12: Example of FB90\_BitCast\_DW, FB91\_BitCast\_W

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_	Input	Input to be converted
FB90_BitCast_DW_U_e	OUTn	VECTOR STD_LOGIC	Output	Output of function
IN1[310] OUT0	00111	STD_LOGIC	Output	Output of function
OUT1 -				
0U T2				
00 Т3				
OUT4				
OU T5				
ол 16				
OU T7				
оитв —				
00 Т9				
OUT10				
OUT11				
OUT12				
OUT13				
OUT14				
OUT15				
OUT16				
OUT17 -				
OUT18				
OUT19				
OUT20				
OUT21				
OUT22 -				
OUT23 -				
OUT24				
OUT25				
OUT26				
OUT27				
OUT28				
OUT29				
OUT30				
OUT31				
inst				
ii				

```
entity FB90 BitCast DW U e is
    Port (
              : in STD_LOGIC_VECTOR (31 downto 0);
        IN1
        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_	Input	Input to be
FB91_BitCast_W_U_e		VECTOR	1	converted
IN1[150] OUT0	OUTn	STD_LOGIC	Output	Output of function
OUT1				lanction
OUT2				
OUT3				
OUT4				
OUT5				
OUT6				
0017				
OUT8				
OUT9				
OUT10				
OUT11				
OUT12				
OUT13				
OUT14				
OUT15				
inst				
<pre>entity FB91_BitCa Port (</pre>	st_W_e <b>is</b>			
RST : in				
CLK : in	_			
CLKEN: in	_			
EN : in				
IN1 : in			downto 0);	
	t STD_LOGIC			
OUT1 : ou	t STD_LOGIC	;		
	t STD_LOGIC;			
	t STD_LOGIC;			
	it STD_LOGIC; it STD LOGIC;			
	it STD_LOGIC;			
	t STD LOGIC			
	t STD LOGIC			
OUT9 : ou	it STD_LOGIC;	;		
	t STD_LOGIC;			
	t STD_LOGIC;			
	t STD_LOGIC;			
	it STD_LOGIC; it STD LOGIC;			
	it STD_LOGIC			
);				
<pre>end FB91_BitCast_</pre>	₩_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.

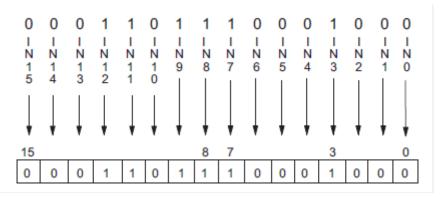


Figure 13: Example of FB92\_BitPack\_DW, FB93\_BitPack\_W

Table 83: TM FAST function BitPack\_DW (FB92\_BitPack\_DW)

Quartus Scher Symbol	matic Block	Parameter	Data type	Direction	Description
		EN	STD_LOGIC	Input	Enable input
FB92_BitPack_D\		INn	STD_LOGIC	Input	Inputs to be packed
RST CLK	OUT1[310]	OUT1	STD_LOGIC_VECTOR	Output	Output of function
CLKEN					
STOP					
EN EN					
IN0					
IN1					
IN2					
IN3					
IN4					
IN5					
IN6					
IN7					
IN8					
IN9					
IN10					
IN11					
IN12					
IN13					
IN14					
IN15					
IN16					
IN17					
IN18					
IN19					
IN20					
IN21					
IN22					
IN23					
IN24					
IN25					
IN26					
IN27					
IN28					
IN29					
IN30					
IN31					
inst		)			
				ĺ	

```
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)

uartus Schematic Block Symbol	Paramet		Direction	Description
FB93_BitPack_W_e	EN	STD_LOGIC	Input	Enable input
DET CULTULE OF	INn	STD_LOGIC	Input	Inputs to be
RST 0UT1[150]	OUT4	OTD 1 0010 1/50705	0.144	packed
CLK	OUT1	STD_LOGIC_VECTOR	Output	Output of function
CLKEN				
STOP				
EN				
INO				
IN1				
IN2				
IN3				
IN4				
IN5				
IN6				
IN7				
INS				
IN9				
- IN10				
IN11				
IN12				
IN13				
IN14				
- IN15				
inst				
Port (		STD_LOGIC;		
CLI		STD_LOGIC;		
		STD_LOGIC;		
STO EN		STD_LOGIC; STD_LOGIC;		
IN		STD_LOGIC;		
IN		STD_LOGIC;		
IN		STD LOGIC;		
IN		STD_LOGIC;		
IN'		STD_LOGIC;		
IN		STD_LOGIC; STD_LOGIC;		
TN				
IN:	LO : in			
IN	l0 : <b>in</b> l1 : <b>in</b>	STD LOGIC;		
IN: IN:	l1 : <b>in</b>			
IN: IN: IN:	11 : in 12 : in 13 : in	STD_LOGIC; STD_LOGIC; STD_LOGIC;		
IN: IN: IN: IN:	l1 : in l2 : in l3 : in l4 : in	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC;		
IN: IN: IN: IN: IN:	11 : in 12 : in 13 : in 14 : in 15 : in	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC;		
IN: IN: IN: IN: OU:	11 : in 12 : in 13 : in 14 : in 15 : in	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC;	lownto 0)	
IN: IN: IN: IN: IN: IN: OU:	11 : in 12 : in 13 : in 14 : in 15 : in 71 : out	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC_VECTOR (15 c	lownto 0)	
IN: IN: IN: IN: OU:	11 : in 12 : in 13 : in 14 : in 15 : in 71 : out	STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC; STD_LOGIC_VECTOR (15 c	downto 0)	

# 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.

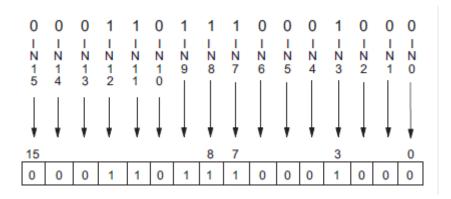


Figure 14: Example of FB92\_BitPack\_DW\_U, FB93\_BitPack\_W\_U

Table 85: TM FAST function BitPack\_DW\_U, unlatched logic (FB92\_BitPack\_DW\_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
FB92_BitPack_DW_U_e	INn	Data type STD_LOGIC	Input	Inputs to be packed
	OUT1	STD_LOGIC_VECTOR	Output	Output of function
IN0 OUT1[310]				
IN1				
IN2				
IN3				
IN4				
IN5				
IN6				
IN7				
IN8				
IN9				
IN10				
IN11				
IN12				
IN13				
IN 14				
IN15				
IN 16				
IN 17				
IN 18				
IN 19				
IN20				
IN21				
IN22				
IN23				
IN24				
IN25				
IN26				
IN27				
IN28				
IN29				
IN30				
IN31				
inst				

```
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 Blo Symbol	ck	Parame	eter Data type		Direction	Description
FB93_BitPack_W_U_e		INn	STD_LOGIC		Input	Inputs to be packed
IN0 OUT1[18 IN1 IN2 IN3 IN4 IN5 IN6 IN7 IN8 IN9 IN10 IN11 IN12 IN13 IN14 IN15	50]	OUT1	STD_LOGIC	C_VECTOR	Output	Output of function
	Port (  IN0 IN1 IN2 IN3 IN4 IN5 IN6 IN7 IN8 IN9 IN10 IN11 IN12 IN13 IN14 IN15	: in : in : in : in : in : in : in : in	STD_LOGIC;	SCTOR (15	downto	0)

## 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.

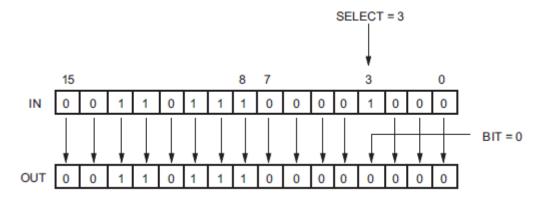


Figure 15: Example of FB94\_BitInsert32, FB95\_BitInsert16

Table 87: TM FAST function BitInsert32 (FB94\_BitInsert32)

Quartus Schematic	Block Symbol	Parameter	Data type	Direction	Description
		EN	STD_LOGIC	Input	Enable input
FB94_BitInsert32_e		IN1	STD_LOGIC_	Input	Input from which the bit
Бот	01174704 01		VECTOR		is selected.
RST	OUT1[310]	SEL_IN	STD_LOGIC_	Input	Bit position to be
CLK			VECTOR		replaced
CLKEN		BIT_IN	STD_LOGIC	Input	Bit to be inserted
STOP		OUT1	STD_LOGIC_	Output	Output of function
EN			VECTOR		
IN1[310]					
SEL_IN[150]					
BIT_IN					
inst					
	entity FB94_B: Port (	itInsert32_	e <b>is</b>		
	RST	: in STD	LOGIC;		
	CLK	: in STD	LOGIC;		
	CLKEN	: in STD_	LOGIC;		
	STOP	: in STD_	LOGIC;		
	EN	: in STD_	LOGIC;		
	IN1	_	LOGIC_VECTOR	•	
		_	LOGIC_VECTOR	(15 downto	0);
	<u> </u>	N: in STD_	_		
		: out STD_	LOGIC_VECTOR	(31 downto	0)
	);				
	end FB94_BitI	nsert32_e;			

Table 88: TM FAST function BitInsert16 (FB95\_BitInsert16):

FB95_BitInsert16_e					
FB95_BitInsert16_e		EN	STD_LOGIC	Input	Enable input
		IN1	STD_LOGIC VECTOR	Input	Input from which the bit is selected.
RST CLK	OUT1[150]	SEL_IN	STD_LOGIC VECTOR	Input	Bit position to be replaced
CLKEN		BIT IN	STD LOGIC	Input	Bit to be inserted
STOP EN		OUT1	STD_LOGIC VECTOR	Output	Output of function
IN1[150]			_vLoTott		
SEL_IN[150]					
BIT_IN					
inst					
	STOP EN IN1	: in STD	LOGIC; LOGIC; LOGIC; LOGIC;		
	_		LOGIC;	•	
	OUT1	: out STD	LOGIC_VECTOR	(15 downto	0)
	); end FB95_BitI	nsert16_e;	_ <del>_</del>		

## 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.

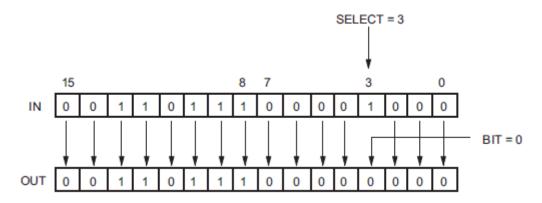


Figure 16: Example of FB94\_BitInsert32\_U, FB95\_BitInsert16\_U

Table 89: TM FAST function BitInsert32\_U, unlatched logic (FB94\_BitInsert32\_U)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
FB94_BitInsert32_U_e	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
IN1[310] OUT1[310]	SEL_IN	STD_LOGIC_ VECTOR	Input	Bit position to be replaced
SEL_IN[150]	BIT_IN	STD_LOGIC	Input	Bit to be inserted
BIT_IN	OUT1	STD_LOGIC_ VECTOR	Output	Output of function
inst				
entity FB94_Bit	:Insert32_U	e <b>is</b>		
Port (				
IN1 :	in STD L	OGIC VECTOR	(31 downto	0);
SEL IN:	in STD L	OGIC VECTOR	(15 downto	0);
BIT IN:	in STD L	OGIC;		
OUT1 :	out STD L	OGIC VECTOR	(31 downto	0)
);	_	_	-	-
end FB94_BitIns	ert32_U_e;			

Table 90: TM FAST function BitInsert16, unlatched logic (FB95\_BitInsert16\_U):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
FB95_BitInsert16_U_e  IN1[150] OUT1[150]	IN1	STD_LOGIC_ VECTOR	Input	Input from which the bit is selected.
SEL_IN[150] BIT IN	SEL_IN	STD_LOGIC_ VECTOR	Input	Bit position to be replaced
	BIT_IN	STD_LOGIC	Input	Bit to be inserted
inst	OUT1	STD_LOGIC_ VECTOR	Output	Output of function
entity FB95_Bi	tInsert16_U	_e		
IN1	: in STD_L	OGIC_VECTOR	(15 downto	0);
_	: in STD_L : in STD_L	OGIC_VECTOR	(15 downto	0);
OUT1	: out STD L	OGIC VECTOR	(15 downto	0)
); end FB95_BitIn	 sert16_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.

	Scan cycle n	Scan cycle n+1	Scan cycle n+2
(256) (255) (254) (254)	Output conditions Entry 1 = 5 Entry 2 = 100 Entry 3 = 125 Entry 4 = -1	Entry 1 = 1 Entry 2 = 100 Entry 3= 125 Entry 4 = -1 Entry 5 = 654	Entry 1 = 100 Entry 2 = 125 Entry 3= -1 Entry 4 = 654
(3) (2) (1) = EMPTY	ENTRIES = 4 FULL = 0 EMPTY = 0 OUT = 5 IN = 654 WRITE = 1 READ_NEXT = 0	ENTRIES = 5 FULL = 0 EMPTY = 0 OUT = 5 IN = 0 WRITE = 0 READ_NEXT = 1	ENTRIES = 4 FULL = 0 EMPTY = 0 OUT = 100 IN = 0 WRITE = 0 READ_NEXT = 0
1) Entry 2) No entry			

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
FB96_FIFO32_e		WR	STD_LOGIC	Input	If active and FULL =
RST — CLK	FULL EMPTY				0, DATA_IN is written into the FIFO
CLKEN EN WR RD_NEXT	ENTRIES[150]  DATA_OUT[310]	RD_NEXT	STD_LOGIC	Input	If active and EMPTY = 0, next entry is placed in DATA_OUT
FIFORESET  DATA_IN[310]		FIFORESET	STD_LOGIC	Input	FIFO entries are reset to 0
inst		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_FIFO	032_e <b>IS</b>			
RST	: IN STD	TOCTC:		
CLK	_			
	_	LOGIC;		
	_	LOGIC;		
EN	_	LOGIC;		
WR	_	LOGIC;		
RD_NEXT	: IN STD_	LOGIC;		
FIFORESE	ET: 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 <b>DOWNTO</b>	0);
		LOGIC VECTOR		
);	_	_		-
END FB96_FIF032_	_e;			

Table 92: TM FAST function FIFO16 (FB97\_FIFO16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
FB97_FIFO16_e  RST FULL CLK EMPTY CLKEN ENTRIES(150)	WR	STD_LOGIC	Input	If active and FULL = 0, DATA_IN is written into the FIFO
CLKEN ENTRIES[150]  EN DATA_OUT[150]  WR  RD_NEXT  FIFORESET	RD_NEXT	STD_LOGIC	Input	If active and EMPTY = 0, next entry is placed in DATA_OUT
DATA_IN[150]	FIFORESET	STD_LOGIC	Input	FIFO entries are reset to 0
IIBL	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_FIF01 Port (	.6_e <b>is</b>			
RST	: in STD L	OGIC;		
CLK	: in STD L	•		
CLKEN	: in STD L	OGIC;		
EN	: in STD_L	OGIC;		
WR	: in STD_L			
<del>-</del>	: in STD_L			
FIFORESET	_	•		
DATA_IN		OGIC_VECTOR	(15 downto	0);
FULL EMPTY	: out STD_LG : out STD LG	•		
ENTRIES		OGIC, OGIC VECTOR	(15 downto	0):
		OGIC_VECTOR		
);			,	-,
end FB97_FIF016_e	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.

	Scan cycle n	Scan cycle n+1	Scan cycle n+2
IN OUT 1 = FULL	Output conditions Entry 1 = 5 Entry 2 = 100 Entry 3 = 125 Entry 4 = -1	Entry 1 = 5 Entry 2 = 100 Entry 3 = 125 Entry 4 = -1 Entry 5 = 654	Entry 1 = 5 Entry 2 = 100 Entry 3 = 125 Entry 4 = -1
(254) ENTRIES  (3) (2) (1) = EMPTY  1) Entry	ENTRIES = 4 FULL = 0 EMPTY = 0 OUT = -1 IN = 654 WRITE = 1 READ_NEXT = 0	ENTRIES = 5 FULL = 0 EMPTY = 0 OUT = 654 IN = 0 WRITE = 0 READ_NEXT = 1	ENTRIES = 4 FULL = 0 EMPTY = 0 OUT = -1 IN = 654 WRITE = 0 READ_NEXT = 0
2) No entry			

Figure 18: Example of FB98\_LIFO32, FB99\_LIFO16

Table 93: TM FAST function LIFO32 (FB98\_LIFO32)

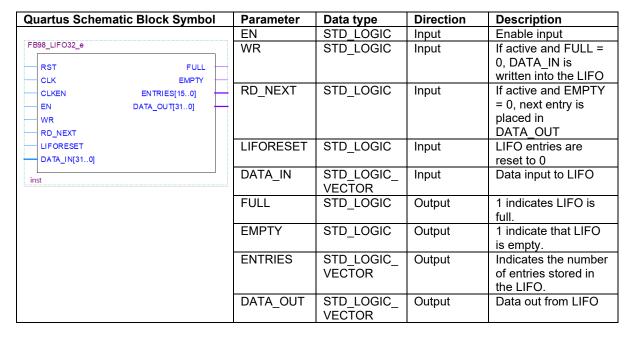


Table 94: TM FAST function LIFO16 (FB99\_LIFO16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable
FB99_LIFO16_e				input
RST FULL	WR	STD_LOGIC	Input	If active and
CLK EMPTY				FULL = 0,
CLKEN ENTRIES[150]				DATA_IN is
EN DATA_OUT[150]				written into
WR .				the LIFO
RD_NEXT	RD_NEXT	STD_LOGIC	Input	If active and
LIFORESET				EMPTY = 0,
DATA_IN[150]				next entry is
				placed in
inst				DATA_OUT
	LIFORESET	STD_LOGIC	Input	LIFO entries
		_	·	are reset to
				0
	DATA IN	STD LOGIC	Input	Data input
	_	VECTOR _	·	to LIFO
	FULL	STD_LOGIC	Output	1 indicates
		_	•	LIFO is full.
	EMPTY	STD LOGIC	Output	1 indicate
		_	·	that LIFO is
				empty.
	ENTRIES	STD LOGIC	Output	Indicates
		VECTOR	-	the number
				of entries
				stored in the
				LIFO.
	DATA_OUT	STD_LOGIC_	Output	Data out
		VECTOR		from LIFO
<pre>entity FB99_LIF016_</pre>	e <b>is</b>			
	in STD LOGI	rc.		
	in STD_LOGI			
	_			
	in STD_LOGI			
	in STD_LOGI	•		
	in STD_LOGI	•		
	in STD_LOGI			
LIFORESET:				
_		C_VECTOR (15	downto 0);	
	out STD_LOGI			
	out STD_LOGI	•		
		C_VECTOR (15		
DATA_OUT :	out STD_LOGI	C_VECTOR (15	downto 0)	
);				
<pre>end FB99_LIF016_e;</pre>				
<del>_</del>				

#### 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 Schemat	ic Block Sym	bol	Parameter	Data type	Direction	Description
,			REQ	STD_LOGIC	Input	Enable the multiply
FB100_FMMul32_e						operation on a 0 to 1
						change. It must
RST	DONE					remain 1 until DONE =
CLK	OVF					1; otherwise, the
CLKEN	OUT1[310]	<b>—</b>	IN1	STD_LOGIC_	Input	multiply terminates.  Input multiplicand
STOP			IINI	VECTOR	_   Iliput	input multiplicand
REQ			IN2	STD LOGIC	Input	Input multiplier
IN1[310]				VECTOR	_  p.s	
IN2[310]			DONE	STD_LOGIC	Output	1 = result is available
			OVF	STD_LOGIC	Output	1, if multiplication
						results in overflow.
inst			OUT1	STD_LOGIC_	Output	Output value = IN1 x
				VECTOR		IN2
		100 =	MM-122 a 4a			
	Port	_	MMul32_e is			
		•	in STD LO	octc.		
	_		in STD LO	•		
			in STD LO			
			in STD LO	-		
			in STD LO			
		~	_	GIC VECTOR	(31 downto	0);
	I		_	GIC VECTOR	•	• •
	Г	ONE :	out STD LO	GIC;		
	C	VF :	out STD LO	GIC;		
	C	UT1:	out STD LO	OGIC VECTOR	(31 downto	0));
	end FB100			_		
		_	_			

Table 96: TM FAST function FMMul16 (FB101\_FMMul16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	REQ	STD_LOGIC	Input	Enable the multiply
FB101_FMMul16_e				operation on a 0 to 1
				change. It must
RST DONE				remain 1 until DONE
CLK OUT1[310]	_			= 1; otherwise, the
CLKEN				multiply terminates.
	IN1	STD_LOGIC_	Input	Input multiplicand
STOP		VECTOR		
REQ INMITE OF	IN2	STD_LOGIC_ VECTOR	Input	Input multiplier
IN1[150]	DONE	STD LOGIC	Output	1 = result is available
IN2[150]	OUT1	STD_LOGIC_	Output	Output value = IN1 x
		VECTOR		IN2
inst				
entity FB101	_FMMull6_e is	3		
Port (				
RST	_	•		
	: in STD_LC	•		
CLKE	J: in STD_LC	GIC;		
STOP	: in STD_LC	GIC;		
REQ	: in STD_LC	GIC;		
IN1	: in STD LC	GIC_VECTOR (	15 downto	0);
IN2	: in STD LO	GIC VECTOR (	15 downto	0);
DONE	: out STD LO	GIC;		
OUT1	: out STD LO	GIC VECTOR (	31 downto	0));
end FB101 FMM				
	,			I
<u>i</u>				

#### 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)

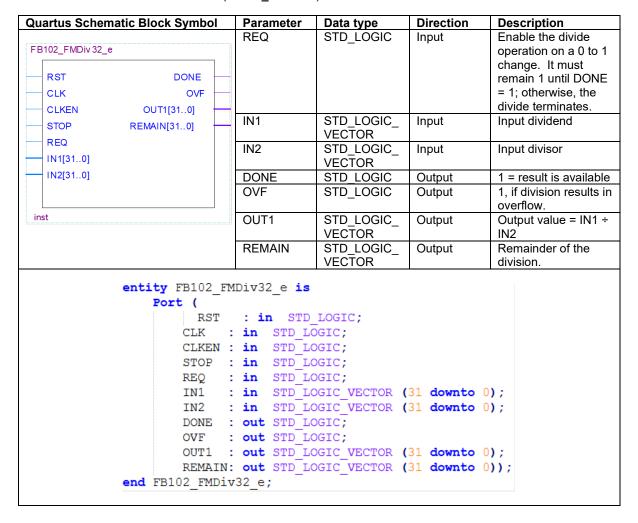


Table 98: TM FAST function FMDiv16 (FB103\_FMDiv16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
REQ		STD_LOGIC	Input	Enable the divide
FB103_FMDiv16_e				operation on a 0 to 1
				change. It must
RST DONE				remain 1 until DONE
CLK OVF				= 1; otherwise, the
CLKEN OUT1[150]				divide terminates.
STOP REMAIN[150]	IN1	STD_LOGIC_	Input	Input dividend
REQ	11.10	VECTOR		
IN1[310]	IN2	STD_LOGIC_	Input	Input divisor
IN2[150]	DONE	VECTOR	0	4
TN2[150]	DONE	STD_LOGIC	Output	1 = result is available
	OVF	STD_LOGIC	Output	1, if division results in overflow.
inst	OUT1	STD_LOGIC_	Output	Output value = IN1 ÷
		VECTOR		IN2
	REMAIN	STD_LOGIC_	Output	Remainder of the
		VECTOR		division.
entity FB103 F	MDiv16 e <b>is</b>			
Port (				
	: in STD L	OGIC:		
	in STD L	•		
CLKEN	_	•		
STOP				
	in STD L	•		
	_	OGIC VECTOR	(31 downto	0):
	_	OGIC VECTOR	-	
	out STD L		(15 downto	017
	: out SID_L : out STD L			
		OGIC VECTOR	/15 downto	0) .
	_	_		
REMAIN	. out STD_L	OGIC_VECTOR	(13 downto	<b>U</b> )
);	-1.0			
end FB103_FMDi	лто_е;			

#### 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
                                                                  Direction
                                                                             Description
                                Parameter
                                             Data type
FB104_FMAbs32_e
                                             STD_LOGIC_VECTOR
                                IN1
                                                                             Input value
                                                                  Input
                                OUT1
                                             STD_LOGIC_VECTOR
                                                                  Output
                                                                             Output
   RST
                OUT1[31..0]
                                                                             value:
                                                                             Absolute
   CLK
                                                                             value of the
   CLKEN
                                                                             input value.
   IN1[31..0]
 inst
                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
FB105_FMAbs16_e	IN1	STD_LOGIC _VECTOR	Input	Input value
RST OUT1[150]  CLK  CLKEN IN1[150]	OUT1	STD_LOGIC _VECTOR	Output	Output value: Absolute value of the input value.
entity FB105 FMAbs16	i e <b>is</b>			
Port (	_			
CLK : in S CLKEN: in S IN1 : in S	TTD_LOGIC; TTD_LOGIC_VEC TTD_LOGIC_VEC			

# 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	Directio n	Description
FB106_FMAdd32_e	IN1	STD_LOGIC_ VECTOR	Input	Input value 1
RST OVF	IN2	STD_LOGIC_ VECTOR	Input	Input value 2
CLKEN	OVF	STD_LOGIC	Output	1, if addition results in overflow.
IN1[310] IN2[310]	OUT1	STD_LOGIC_ VECTOR	Output	Output value = IN1 + IN2
entity FB106_F	MAdd32_e <b>i</b>	s		
RST : CLK : CLKEN: IN1 : IN2 : OVF : OUT1 :	in STD_L out STD_L out STD_L	OGIC; OGIC; OGIC_VECTOR OGIC_VECTOR	(31 downt	0);
end FB106_FMAd	ld32_e;			

Table 102: TM FAST function FMAdd16 (FB107\_FMAdd16)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
FB107_FMAdd16_e	IN1	STD_LOGIC_ VECTOR	Input	Input value 1
RST OVF OUT1[150]	IN2	STD_LOGIC_ VECTOR	Input	Input value 2
CLKEN CONTROL OF	OVF	STD_LOGIC	Output	1, if addition results in overflow.
IN1[150] IN2[150]	OUT1	STD_LOGIC_ VECTOR	Output	Output value = IN1 + IN2
inst				
entity FB107_F	MAdd16_e <b>i</b>	s		
CLK :	_	OGIC;		
IN1 :		OGIC_VECTOR OGIC_VECTOR		
OVF :	out STD_L	_		
end FB107_FMAd	_		(10 10 10 10 10 10 10 10 10 10 10 10 10 1	

#### 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
                                                        Directio
                                                                  Description
                                           Data type
FB108_FMSub32_e
                               IN1
                                           STD LOGIC
                                                                  Input value 1
                                                        Input
                                            VECTOR
                               IN2
                                           STD_LOGIC
                                                        Input
                                                                  Input value 2
   RST
                      OVF
                                            VECTOR
   CLK
               OUT1[31..0]
                               OVF
                                           STD_LOGIC
                                                        Output
                                                                  1, if addition results in
   CLKEN
                                                                  overflow.
   IN1[31..0]
                                           STD_LOGIC
                               OUT1
                                                        Output
                                                                  Output value = IN1 - IN2
                                           _VECTOR
   IN2[31..0]
 inst
              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
FB109_FMSub16_e	IN1	STD_LOGIC_	Input	Input value 1
		VECTOR		
RST OVF	IN2	STD_LOGIC_	Input	Input value 2
CLK OUT1[150]	0) /=	VECTOR		
CLKEN	OVF	STD_LOGIC	Output	1, if addition results in
IN1[150]	OUT1	STD LOGIC	Output	overflow. Output value = IN1 -
	0011	VECTOR	Output	IN2
IN2[150]		VEOTOR		1112
inst				
INSL				
entity FB109 F	MSub16 e i	2		
Port (				
	in STD Lo	OGTC.		
CLK	_	•		
CLKEN:	_			
	_	OGIC VECTOR	(15 downto	0).
IN2	_	OGIC VECTOR		
	out STD Lo	_	(15 downed	, ,,
	_	OGIC VECTOR	(15 downto	. 0)
);	Out SID_IN	OGIC_VECTOR	(15 downed	, ,,
• •	.b16 o.			
end FB109_FMSt	mro_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
ED440 DetCel22 e	IN1	STD_LOGIC_	Input	Input value 1
FB110_DatSel32_e  RST OUT1[310]	IN2	VECTOR STD_LOGIC_ VECTOR	Input	Input value 2
CLK CLKEN SEL IN1[310]	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.
IN2[310]	OUT1	STD_LOGIC_ VECTOR	Output	Output value: IN1 if SEL = 0 IN2 if SEL = 1
entity FB110_D Port (	atSel32_e	is		
RST :	in STD_L	OGIC;		
CLK :	in STD_L	OGIC;		
CLKEN:	in STD_L	OGIC;		
SEL :	in STD_L	OGIC;		
IN1 :		OGIC_VECTOR		
		OGIC_VECTOR		
OUT1 :	out STD_L	OGIC_VECTOR	(31 downto	0)
); end FB110 DatS	e132 e:			
12110_2465				

Table 106: TM FAST function DatSel16 (FB111\_DatSel16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
FB111_DatSel16_e	IN1	STD_LOGIC_ VECTOR	Input	Input value 1
RST OUT1[150]	IN2	STD_LOGIC_ VECTOR	Input	Input value 2
CLK CLKEN SEL IN1[150]	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.
IN2[150] inst	OUT1	STD_LOGIC_ VECTOR	Output	Output value: IN1 if SEL = 0 IN2 if SEL = 1
CLK CLKEN SEL IN1 IN2	LOGIC; LOGIC; LOGIC; LOGIC; LOGIC_VECTOR LOGIC_VECTOR LOGIC_VECTOR	(15 downto	0);	

# 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.

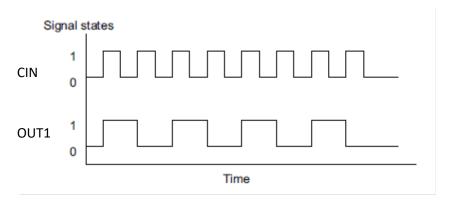


Figure 19: Timing Diagram for Binary Scaler (FB112\_BiScale)

Table 107: TM FAST function Period32 (FB80\_Period32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	EN	STD_LOGIC	Input	Enable input
FB112_BiScale_e	CIN	STD_LOGIC	Input	Input to be
	OUT1	STD_LOGIC	Output	converted. Output of the
RST OUT1	0011	015_20010	Cutput	function.
CLK				
CLKEN				
EN EN				
CIN				
inst				
entit	y FB112_Bis	Scale_e <b>is</b>		
1	Port (	_		
	RST : i	.n STD_LOGIC	;	
	CLK : i	.n STD_LOGIC	;	
	CLKEN: i	n STD LOGIC	;	
	EN : i	n STD LOGIC	;	
	CIN : i	.n STD_LOGIC	;	
	OUT1 : c	out STD_LOGIC	:	
	;			
end H	FB112_BiScal	.e_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.

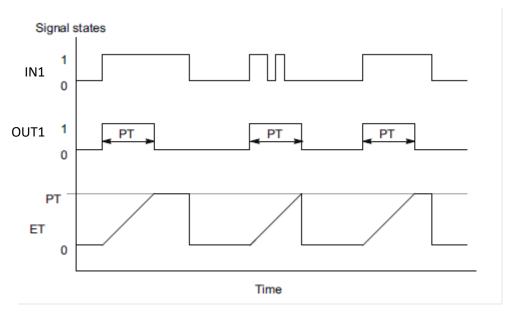


Figure 20: Timing Diagram for Pulse Timer (FB113\_TP32, FB116\_TP16)

Table 108: TM FAST function TP32 FB113\_TP32)

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
-	IN1	STD_LOGIC	Input	Start input
FB113_TP32_e  RST OUT1  CLK ΕΤ[310]  CLKEN	PT	STD_LOGIC_VECTOR	Input	Duration of the pulse in 10usec units. PT must be a positive constant.
CLK100K STOP IN1	CLK100k	STD_LOGIC	Input	Connect to the ClkTic100k logic
PT[310]	OUT1	STD_LOGIC	Output	Status of the time.
inst	ET	STD_LOGIC_VECTOR	Output	Elapsed time.

Table 109: TM FAST function TP16 (FB116\_TP16):

Quartus Schematic Block Symbol	Parameter	Data type	Direction	Description
	IN1	STD_LOGIC	Input	Start input
FB116_TP16_e	PT	STD_LOGIC _VECTOR	Input	Duration of the pulse in 10usec units. PT must be a positive constant.
CLK E∏150]	CLK100k	STD_LOGIC	Input	Connect to the ClkTic100k logic
	OUT1	STD_LOGIC	Output	Status of the time.
CLK100K STOP IN1	ET	STD_LOGIC _VECTOR	Output	Elapsed time.
Ρπ[150]				
inst	216 0 1-			
entity FB116_TF Port (	10_e 18			
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.

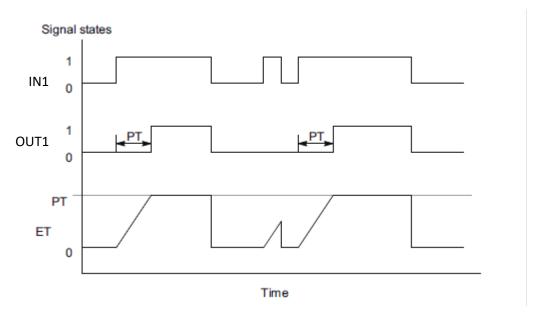


Figure 21: Timing Diagram for On-Delay Timer (FB114\_TOn32, FB117\_TOn16)

Table 110: TM FAST function TOn32 FB114\_TOn32)

Quartus Schematic Block Symb	ol Parameter	Data type	Direction	Description
	IN1	STD_LOGIC	Input	Start input
FB114_TOn32_e	PT	STD_LOGIC	Input	Duration of the pulse in
		_VECTOR		10usec units. PT must
RST OUT1				be a positive constant.
CLK ET[310]	CLK100k	STD_LOGIC	Input	Connect to the
CLKEN				ClkTic100k logic
	OUT1	STD_LOGIC	Output	Status of the time.
CLK100K	ET	STD_LOGIC	Output	Elapsed time.
STOP		_VECTOR		
IN1				
PT[310]				
inst	·			
	L		1	<u> </u>

Table 111: TM FAST function Ton16 (FB117\_TOn16):

Quartus Schematic	Block Symbol	Parame	eter Data type	Direction	Description
		IN1	STD_LOGIC	Input	Start input
FB117_TOn16_e		PT	STD_LOGIC_	Input	Duration of the pulse
			VECTOR		in 10usec units. PT
RST	OUT1				must be a positive
CLK	ET[150]	01.144.04	OTD 1 0010		constant.
CLKEN		CLK100	Ok STD_LOGIC	Input	Connect to the
CLK100K		OUT1	STD LOGIC	Output	ClkTic100k logic Status of the time.
STOP		FT	STD_LOGIC	Output	Elapsed time.
		'	VECTOR	Output	парэси шпс.
IN1					
PT[150]					
inst					
en	tity FB117_TO Port (	_			
	RST		STD_LOGIC;		
	CLK		STD_LOGIC;		
	CLKEN		STD_LOGIC;		
			STD_LOGIC;		
	STOP		STD_LOGIC;		
	IN1		STD_LOGIC;		
	PT		STD_LOGIC_VECTOR	(15 downto	0);
	OUT1		STD_LOGIC;		
	ET	: out	STD_LOGIC_VECTOR	(15 downto	0)
	);				
en	<b>d</b> FB117_TOn1	o_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.

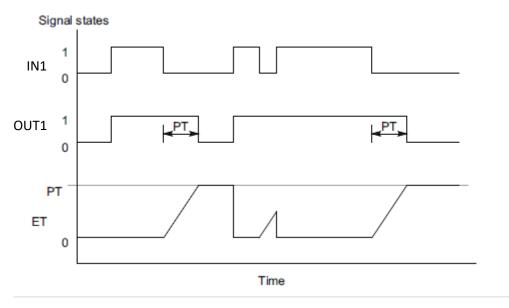
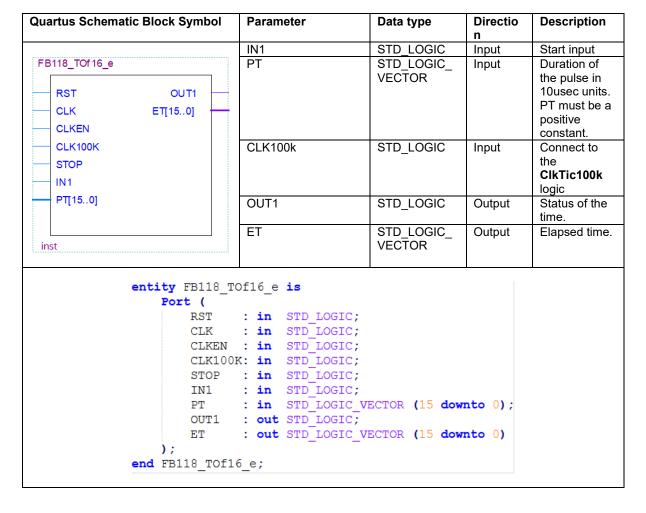


Figure 22: Timing Diagram for Off-Delay Timer (FB115\_TOf32, FB118\_TOf16)

Table 112: TM FAST function TOf32 FB115\_TOf32)

Quartus Schematic Block Symbol	Parameter	Data type	Directio	Description
	IN1	STD LOGIC	n Input	Start input
FB115_TOf32_e  RST OUT1 CLK EΠ[310]	PT	STD_LOGIC_ VECTOR	Input	Duration of the pulse in 10usec units. PT must be a positive constant.
CLK100K STOP IN1	CLK100k	STD_LOGIC	Input	Connect to the ClkTic100k logic
Ρ∏310]	OUT1	STD_LOGIC	Output	Status of the time.
inst	ET	STD_LOGIC_ VECTOR	Output	Elapsed time.

Table 113: TM FAST function TOf16 (FB118\_TOf16):



#### 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.

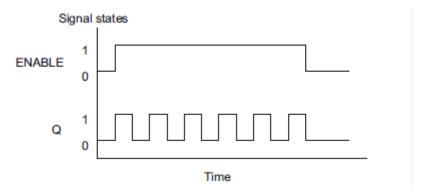


Figure 23: Timing Diagram for Clock Pulse Generator (FB119\_CP\_Gen)

Table 114: TM FAST function CP\_Gen (FB119\_CP\_Gen)

Quartus Schematic Block Symbol	Parameter	Data type	Directio	Description
			n	
	ENABLE	STD_LOGIC	Input	Enable input
FB119_CP_Gen_e	PERIOD	STD_LOGIC _VECTOR	Input	The number of 20usec steps in the period.
CLK	CLK100k	STD_LOGIC	Input	Connect to the ClkTic100k logic
CLK100k ENABLE	Q	STD_LOGIC	Output	Status of the time
PERIOD[150]				
inst				

#### 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
FB120_CTUD32_e	CU	STD_LOGIC	Input	Count up input.
	CD	STD_LOGIC	Input	Count down input
RST QU CLK QD	R	STD_LOGIC	Input	Reset input. R is dominant over CU.
CLK QD CLKEN CV[310]	LOAD	STD_LOGIC	Input	Load input. LOAD input is dominate over CD.
CU CD R LOAD	PV	STD_LOGIC_ VECTOR	Input	Preset value. The counter is preset to PV when the signal level at the LOAD input is 1.
PV[310] inst	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;
           : in STD LOGIC;
       R
       LOAD : in STD LOGIC;
       PV : in STD LOGIC VECTOR (31 downto 0);
          : out STD_LOGIC;
       QU
       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
FB123_CTUD16_e	CU	STD_LOGIC	Input	Count up input.
	CD	STD_LOGIC	Input	Count down input
RST QU	R	STD_LOGIC	Input	Reset input. R is dominant over CU.
CLK QD CLKEN CV[150]	LOAD	STD_LOGIC	Input	Load input. LOAD input is dominate over CD.
CU CD R LOAD	PV	STD_LOGIC_V ECTOR	Input	Preset value. The counter is preset to PV when the signal level at the LOAD input is 1.
PV[150]	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_V ECTOR	Output	Counter value

```
entity FB123 CTUD16 e is
   port (
        RST : in STD LOGIC;
       CLK : in STD_LOGIC;
CLKEN: in STD_LOGIC;
           : in STD_LOGIC;
        EN
        CU : in STD_LOGIC;
        CD : in STD LOGIC;
            : in STD LOGIC;
        LOAD : in STD LOGIC;
        ΡV
           : in STD LOGIC VECTOR (15 downto 0);
           : out STD LOGIC;
        QU
            : out STD LOGIC;
        QD
           : 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
                                                               Direction
                                                                          Description
                                                 Data type
                                 ΕN
                                                 STD LOGIC
                                                               Input
                                                                          Enable input
FB121_CTU16_e
                                 CU
                                                 STD LOGIC
                                                               Input
                                                                          Count up input.
                                 R
                                                 STD LOGIC
                                                                          Reset input. R is
                                                               Input
                                                                          dominant over CU.
    RST
                                 PV
                                                 STD LOGIC
                                                               Input
                                                                          Preset value. The
    CLK
                   CV[15..0]
                                                 VECTOR
                                                                          counter is preset to
    CLKEN
                                                                          PV when the signal
                                                                          level at the LOAD
   ΕN
                                                                          input is 1.
   CU
                                                 STD LOGIC
                                                               Output
                                 S
                                                                          Status of the
   R
                                                                          counter: QU has
                                                                          the following value:
   PV[15..0]
                                                                          1 if CV > PV
                                                                          0 in all other
                                                                          situations
 inst
                                 CV
                                                 STD LOGIC
                                                               Output
                                                                          Counter value
                                                 VECTOR
               entity FB121 CTU16 e is
                   port (
                              : in STD LOGIC;
                        RST
                        CLK : in STD LOGIC;
                        CLKEN: in STD LOGIC;
                              : in STD LOGIC;
                        EN
                              : in STD LOGIC;
                        CU
                              : in STD_LOGIC;
                        R
                              : in STD_LOGIC VECTOR (15 downto 0);
                        PV
                              : out STD LOGIC;
                        Q
                              : out STD LOGIC VECTOR (15 downto 0)
                        CV
               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	Directio n	Description
	EN	STD_LOGIC	Input	Enable input
FB122 CTD16	CD	STD_LOGIC	Input	Count down input
RST Q CLK CV[150]	LOAD	STD_LOGIC	Input	Load input. The LOAD input overrides the CD input.
CLKEN EN CD LOAD	PV	STD_LOGIC_ VECTOR	Input	Preset value. The counter is preset to PV when the signal level at the LOAD input is 1.
PV[150] inst	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 CTD	16 is			
Port (	10 15			
	n STD LOGIC			
CLK : i	_			
CLKEN: i	_			
EN : i	_	•		
	n STD_LOGIC			
LOAD : i	_	•		
	_	•	dormto 01	
PV : i	_	VECTOR (15	downto U)	i
	ut STD_LOGIC		J	
);	_	_VECTOR (15	aownto ∪)	
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
CIKTICK100K  RST CLKEN100KHZ  CLK  CLKEN  inst	CLKEN100KHZ	STD_LOGIC	Output	Clock-tic signal (100kHz) needed by other logic blocks.
entity ClkTick100k is				
port (				
	RST :	in STD_LOG	IC;	
	CLK :	in STD_LOG	ic;	
	CLKEN :	in STD_LOG	IC;	
	CLKEN100KHZ:	OUT STD_LOG	ic	
);				
end ClkT	ick100k;			

# 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, 1064-Bit – 24V HTL, Pulse/Dir or RS422 ABN
SSI	SSI Encoder interface, 1064-Bit
SSI_ListenIn	SSI ListenIn Encoder interface, 1064-Bit

# 4.1 Incremental Encoder (Inc\_Enc)

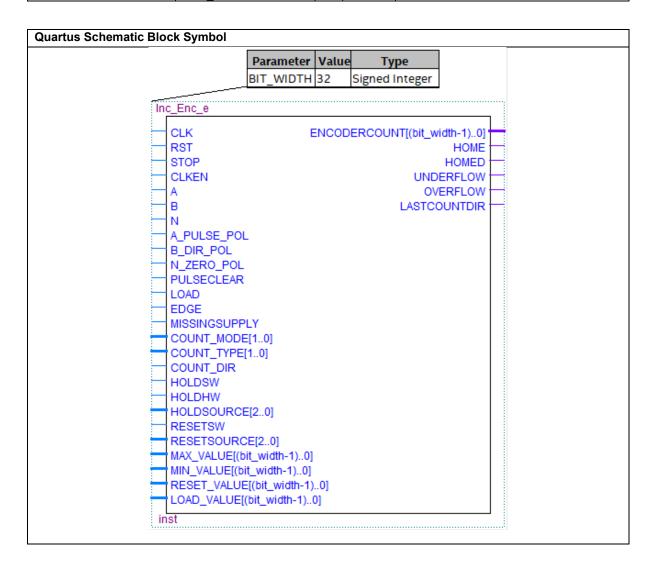
# Description

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
Α	STD LOGIC	Input	Encoder A / Pulse Input
В	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
COUNT_TYPE	STD_LOGIC_VECTOR	Input	<ul> <li>0 -&gt; single count</li> <li>1 -&gt; double count</li> <li>2 -&gt; quad count</li> <li>3 -&gt; pulse/direction</li> </ul> Counting Type
			<ul> <li>0 -&gt; continuous count</li> <li>1 -&gt; periodic count</li> <li>2 -&gt; single count</li> </ul>
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
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.



```
entity Inc Enc e is
    Generic (
        -- bit width of the counter value
        BIT WIDTH : natural := 32
    );
    Port (
         -- Logic-Clock and -Reset
                     : in std logic;
                      : in std logic;
         RST
                      : in std_logic;
         STOP
         CLKEN : in std logic;
         -- Counter Signals A/B/N
                    : in std_logic;
                      : in std_logic;
                      : 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;
                       : in std logic;
         EDGE
         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.

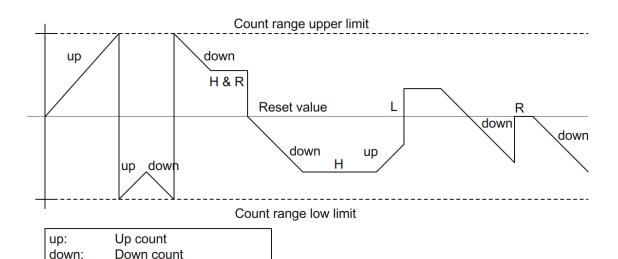
When counting up, the module increments to the maximum value, then rolls over to the minimum value and continues counting. (This rollover is reported in the overflow status bit.) When counting down, the module decrements to the minimum value, then rolls over to the maximum value and continues counting. (This rollover is reported in the underflow status bit.)

The figure below illustrates how continuous counting functions.

Hold is active

Reset is active Loading is active

H: R:



#### **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.

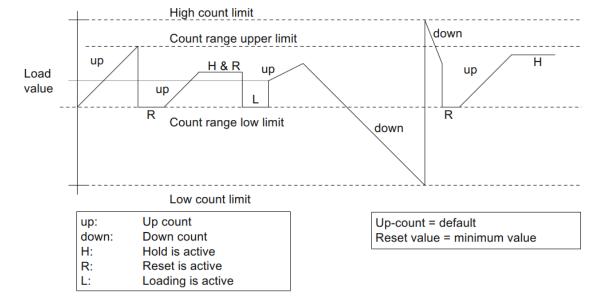
When the "main count direction" is set to Count Up, the counter behaves in the following ways:

- 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.

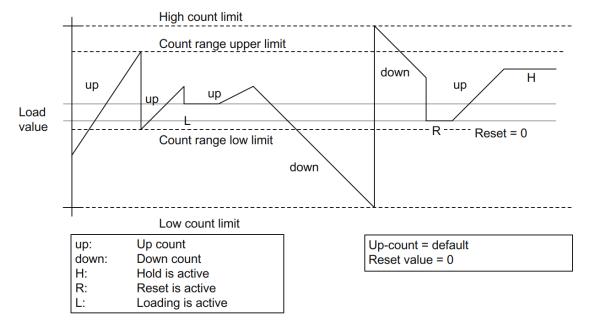
When the "main count direction" is set to Count Up, the counter behaves in the following ways:

- 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**

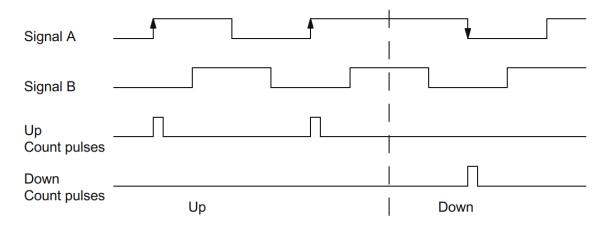
- Count\_MODE = 0: single count
- Count MODE = 1: double count
- Count\_MODE = 2: quad count
- Count MODE = 3: pulse/direction

### 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.

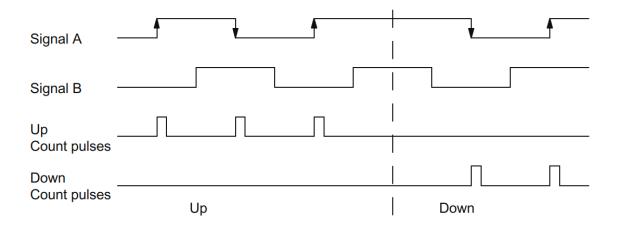


#### **Double Evaluation (double count)**

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.

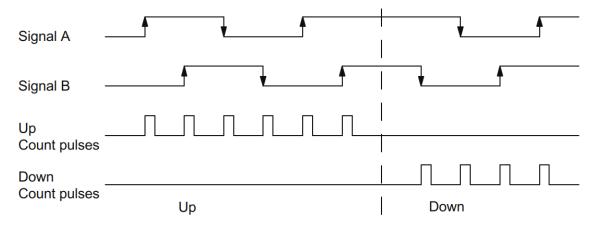


## **Quadruple Evaluation (quad count)**

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.

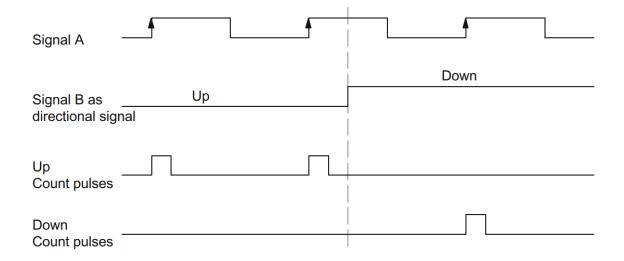
The figure below illustrates quadruple evaluation of the signals.



#### 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.



## 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		
			<ul> <li>0 -&gt; Absolute binary</li> </ul>		
			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
                                   0
                                         Signed Integer
                      CLOCKSEL
                      SHIFT DIR
                                   right
                                         String
                      SHIFT_COUNT
                                   0
                                         Signed Integer
                      FRAME WIDTH
                                   13
                                         Signed Integer
                SSI e
                  RST
                            ENCODERCOUNT[(frame width-1)..0]
                  CLK
                                               SSI CLOCK
                  GRAY_BIN_N
                                            DATAAVAILABLE
                  SSI DATA IN
                                          FRAME OVERRUN
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;
                                             := "right";
        SHIFT_DIR : string
        SHIFT COUNT : integer range 0 to 64 := 0;
        FRAME WIDTH: integer range 10 to 64 := 13
    );
    Port (
        RST
                     : in STD LOGIC;
                     : in STD LOGIC;
        CLK
        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		Encoder counter value			
FRAME_OVERRUN	STD_LOGIC	Output	Frame overrun error			
DATAAVAILABLE	STD_LOGIC	Output	Valid data is available			
<b>Quartus Schematic Blo</b>	ck Symbol					
	Parameter	Value Typ	e			
	BIT WIDTH	16 Signed In				
	SHIFT DIR	right String				
	SHIFT COUNT	0 Signed In	teger			
	CLOCKSEL	0 Signed In				
	FRAME WIDTH		0			
	<u> </u>		tege.			
	SSIListenIn_e					
	RST	RST ENCODERCOUNT[(bit_width-1)0]				
	CLK					
	GRAY_BIN_N	<del>-</del> -				
	SSICLOCK					
	SSI_DATA_IN					
	inst1					

```
entity SSIListenIn e is
       Generic (
       BIT WIDTH
                   : integer :=
       SHIFT DIR : string := "right";
       SHIFT COUNT : integer :=
                : integer :=
       CLOCKSEL
                                     0;
       FRAME WIDTH : integer :=
                                    13
       );
   Port (
                    : in STD LOGIC;
       RST
                    : in STD LOGIC;
       CLK
       GRAY BIN N : in STD LOGIC;
                    : in STD LOGIC;
       SSICLOCK
       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.