AdcFrame

Marc Defossez
Sr. Staff Applications Engineer

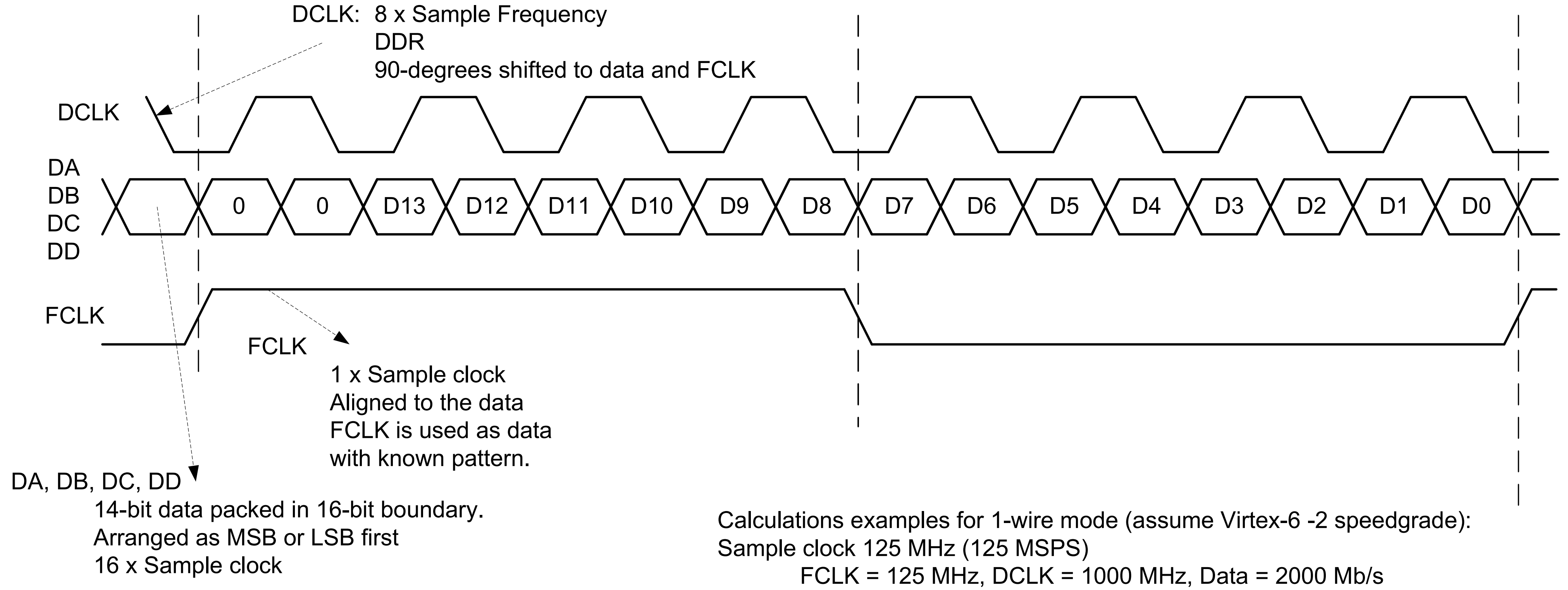
Created: February 6, 2008
Modified: April 6, 2011

DISCLAIMER:

Contact: e-mail hotline@xilinx.com phone + 1 800 255 7778

This page is intentionally left blank.

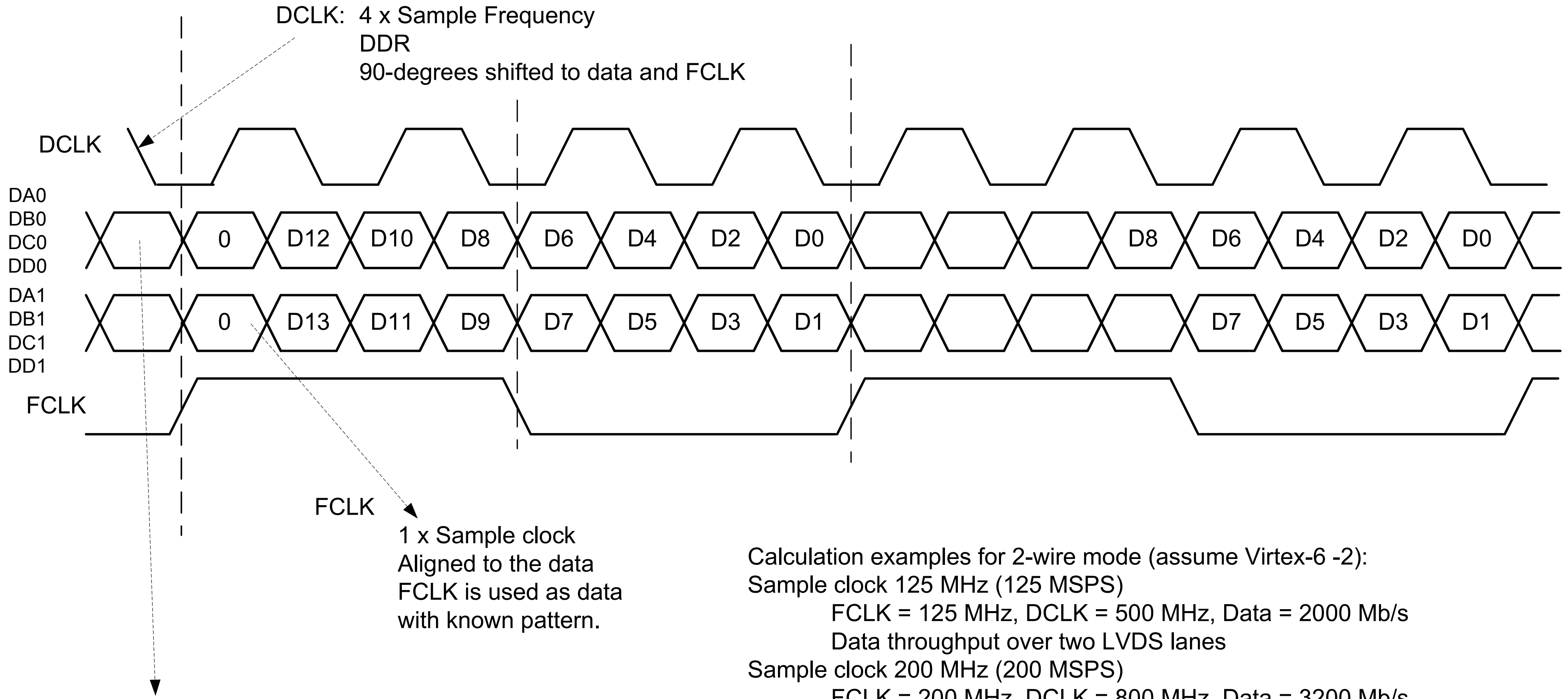
1-WIRE



Assume FCLK = 80 MHz or 12.5 ns
Then the DCLK is 640 MHz or 1.5625 ns
The DCLK is equal to the CLK internally used.
From that CLK a CLKDIV is generated. The CLKDIV is the clock that latches the serial shifted data into a parallel register for treatment.
The CLKDIV is 4 times CLK because 8-bits are used at a time.
CLKDIV is 160 MHz or 6.25 ns.

Calculations examples for 1-wire mode (assume Virtex-6 -2 speedgrade):
Sample clock 125 MHz (125 MSPS)
    FCLK = 125 MHz, DCLK = 1000 MHz, Data = 2000 Mb/s
    This is to fast for the regional clock inputs
Sample clock 105 MHz
    FCLK = 105 MHz, DCLK = 840 MHz, Data = 1680 Mb/s
    This is to fast for the regional clock inputs
Sample clock 100 MHz
    FCLK = 100 MHz, DCLK = 800 MHz,Data = 1600 Mb/s
    This is OK for the regional clock inputs
For a resolution of 16-bits (or 14-bits packed in 16-bit boundaries) and in 1-wire mode the sample speed of the ADC is limited to 100 MHz.

2-WIRE

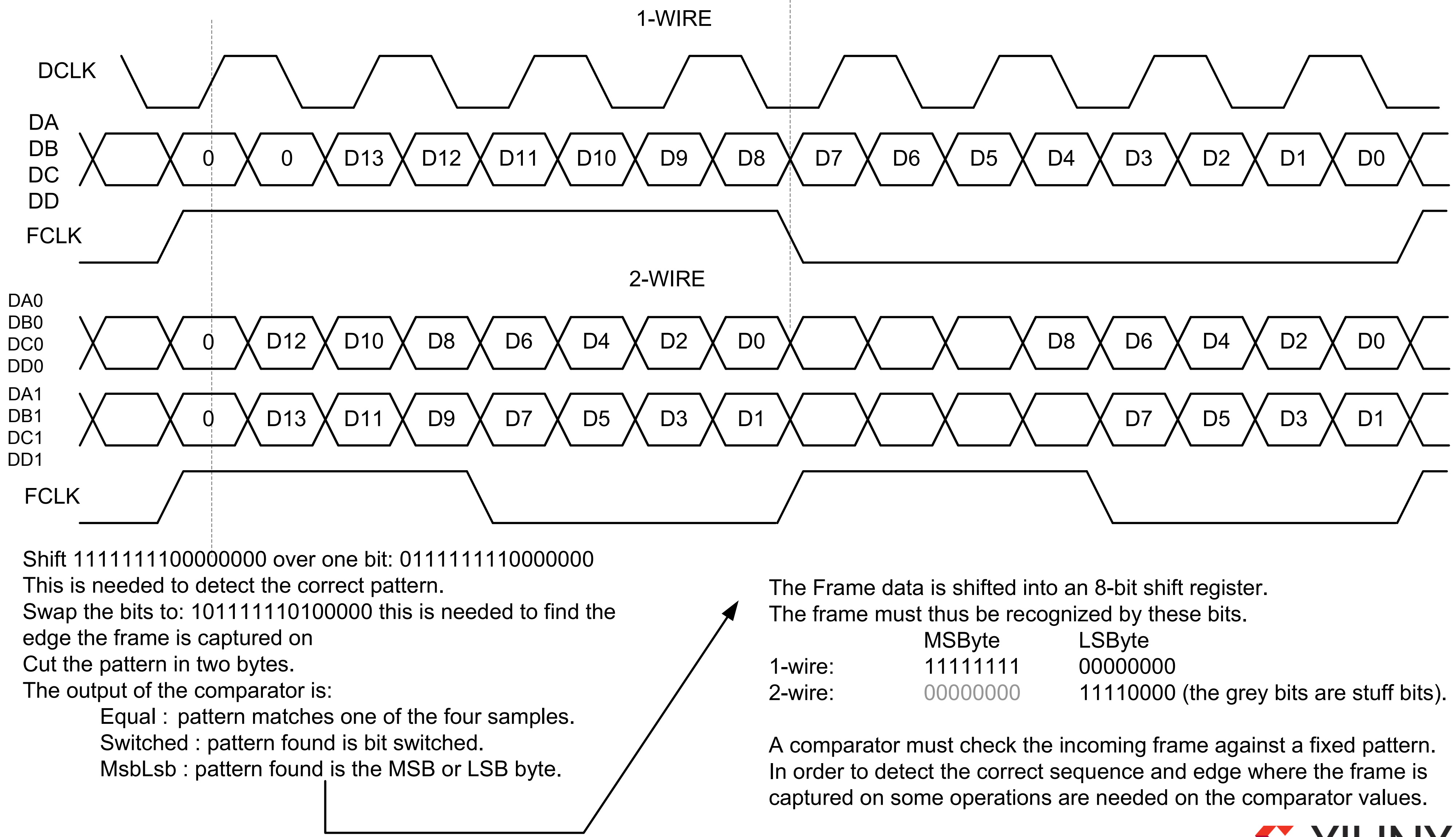


DA0, DA1, DB0, DB1, DC0, DC1, DD0, DD1
14-bit data packed in 16-bit boundary and divided over two LVDS lanes. Arranged as MSB or LSB first. The data rate is 8 x sample frequency per channel.

Calculation examples for 2-wire mode (assume Virtex-6 -2):
Sample clock 125 MHz (125 MSPS)
    FCLK = 125 MHz, DCLK = 500 MHz, Data = 2000 Mb/s
    Data throughput over two LVDS lanes
Sample clock 200 MHz (200 MSPS)
    FCLK = 200 MHz, DCLK = 800 MHz, Data = 3200 Mb/s
    Data throughput over two LVDS lanes

For a resolution of 16-bits (or 14-bits packed in 16-bit boundaries) and 2-wire mode the sample speed of the ADC is maximal to 200 MHz

Frame Recognition



Shift 1111111100000000 over one bit: 0111111110000000
This is needed to detect the correct pattern.
Swap the bits to: 1011111101000000 this is needed to find the edge the frame is captured on
Cut the pattern in two bytes.
The output of the comparator is:

Equal : pattern matches one of the four samples.
Switched : pattern found is bit switched.
MsbLsb : pattern found is the MSB or LSB byte.

The Frame data is shifted into an 8-bit shift register.
The frame must thus be recognized by these bits.

	MSByte	LSByte
1-wire:	11111111	00000000
2-wire:	00000000	11110000 (the grey bits are stuff bits).

A comparator must check the incoming frame against a fixed pattern. In order to detect the correct sequence and edge where the frame is captured on some operations are needed on the comparator values.

Recognition example of a byte wide frame

(2-wire mode operation of the ADC)

Capturing the Frame signal from a ADC is capturing over and over the same data, because a frame is a slow clock signal that is treaded as normal data. This is done to easy align the real data to correct boundaries
It can be done because the Frame and Data are phase aligned and 90-degrees phase shifted to the clock.
As shown in previous sheets, it is possible that data gets captured starting at a rising or falling edge of the bit clock (CLK).
When one or the other happens it is possible that the captured bits are presented in a bit-swapped state at the output of the ISERDES.
When this happens a multiplexer must swap the bits back in correct position.

How can a bit swapped captured word be recognized?
The only way to do this is again via the FRAME signal because a regular pattern can be easily examined.
Example:
Assume that the FPGA interface captures data from a (14)16-bit ADC, the ADC transmits the bits to the FPGA in 2-wire mode. The frame, data and clock will look as figure A.

The FRAME seen as data result then in a data bit pattern as:
Frame ……...111100001111…………..
The frame capturing ISERDESses will thus capture a pattern as:

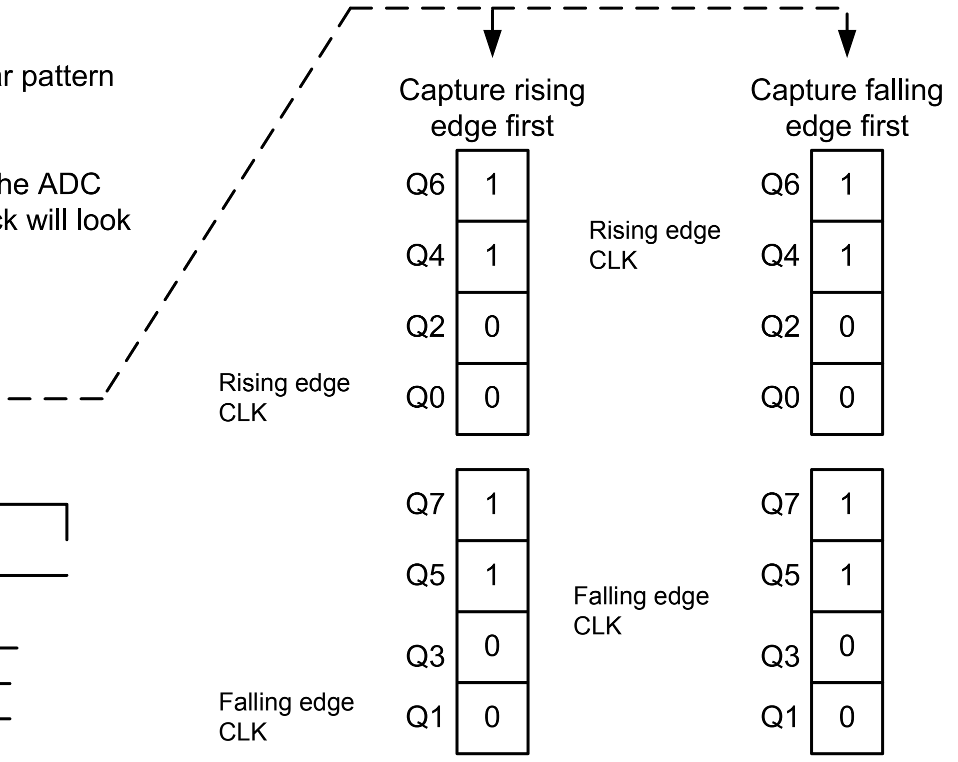
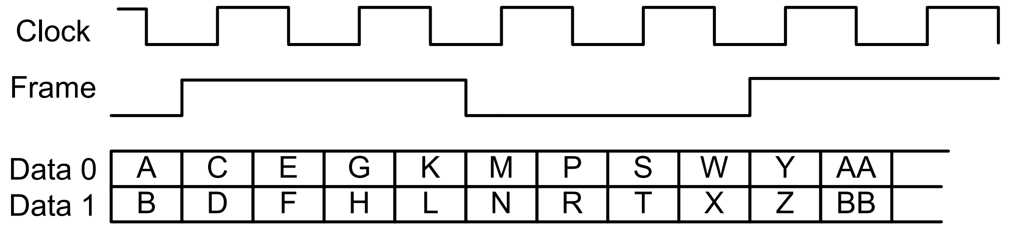


Figure A



When the straight pattern is captured it can thus not been told if the bits are swapped or not.
When capturing **11110000** and the captured bit pattern is swapped, the result is still: **11110000**
But the frame signal is the only signal we can permanently use to find if bits are swapped or not.
The captured pattern must thus be adapted.
The ADC can not change the pattern of the Frame signal, thus the FPGA needs to do this.

How can we do this?
Detect the pattern somewhere in-between the 1 and 0 stream, a place where there are 1's and 0's to align for.

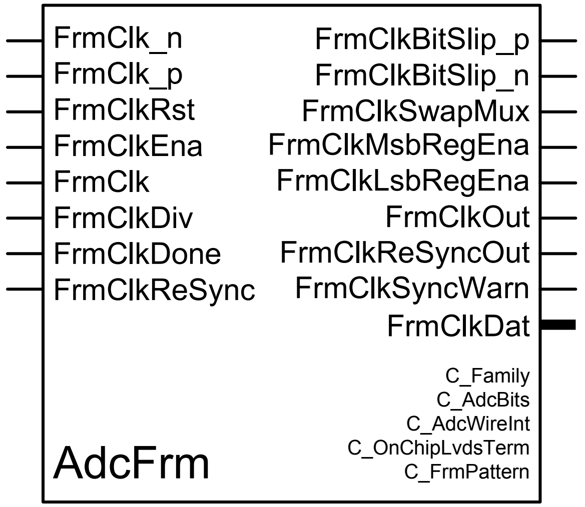
To not have to build to much logic let us capture the frame pattern one bit shifted.
When that pattern is detected an extra shift in the correct direction will align data and frame to the correct boundaries.
Thus to be able to detect a one bit shifted pattern this pattern must be constructed in the HDL code.

ThIs is the effective Frame pattern, in case of our example: **11110000**
From that pattern, by means of VHDL functions the shifted pattern is constructed.

The HDL transforms the **11110000** pattern into a bit shifted pattern, being: **01111000**
Knowing that the pattern can be captured rising or falling edge first resulting in a bit swapped output, a bit swapped check pattern must be constructed. This is also done with a VHDL function and the result is then: **10110100**
This bit shifting and swapping is only needed when the frame pattern is a symmetric clock signal.
Meaning that the number of 1 and/or 0 are equal and equally spread in the word, so that it can not be told if bit swapping occurred or not.
There is thus a check for the symmetry of the pattern build in the HDL.
The sheet "Pattern Repetition" of the XLS spreadsheet file shows that an 8-bit pattern only has 16 repetitive patterns.
Doing this allows that any pattern can be used a frame pattern.

REMARK:
A function in VHDL doesn't generate any logic.
By performing the shift and swap operations no logic is thus generated.

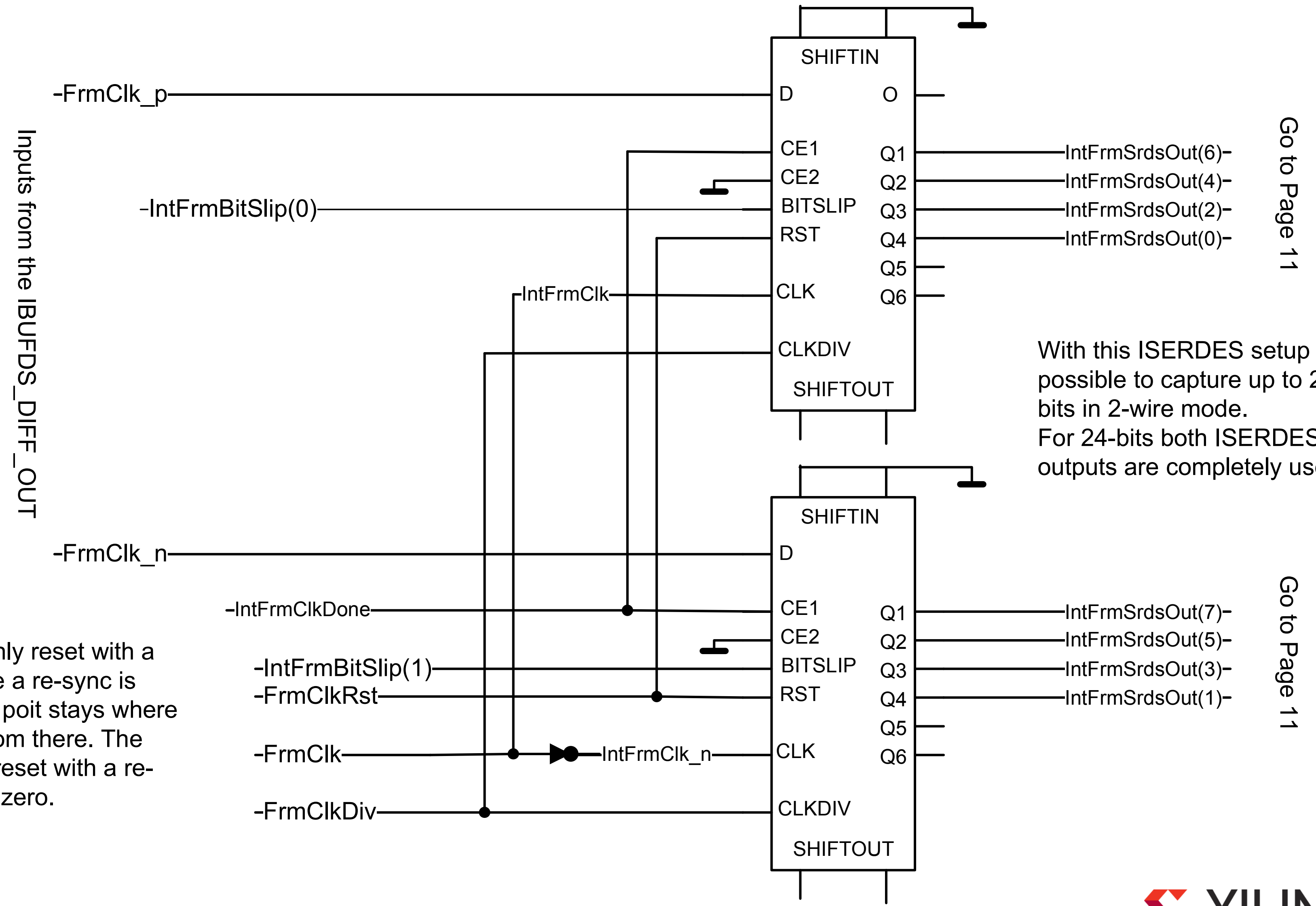
AdcFrm



The goes to the data capturing ISERDES part of the design.

These are outputs to the application behind the interface

ISERDES

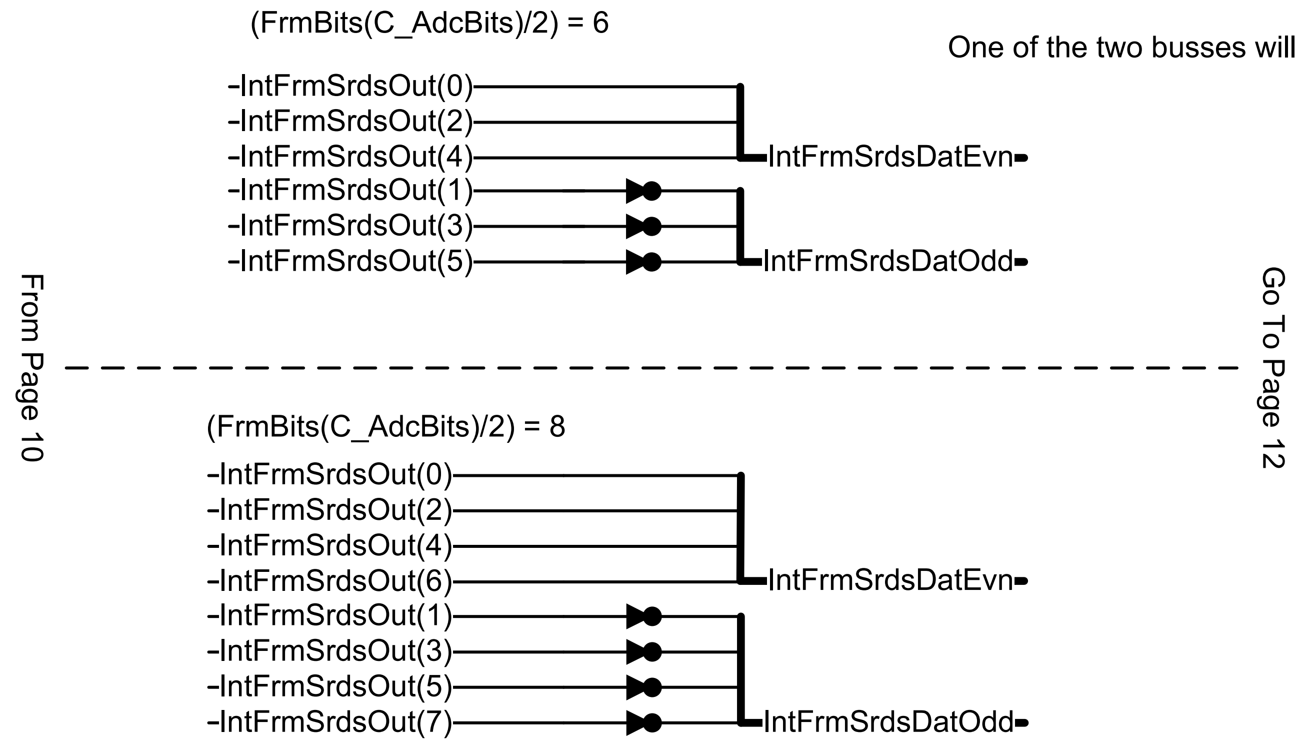


With this ISERDES setup it is possible to capture up to 24-bits in 2-wire mode.
For 24-bits both ISERDES outputs are completely used.

The ISERDES are only reset with a system reset. In case a re-sync is necessary the bitslip poit stays where it is and continues from there. The logic in the circuit is reset with a re-sync and starts from zero.

Bus Selection

One of the two busses will be created.



Double Nibble Detection

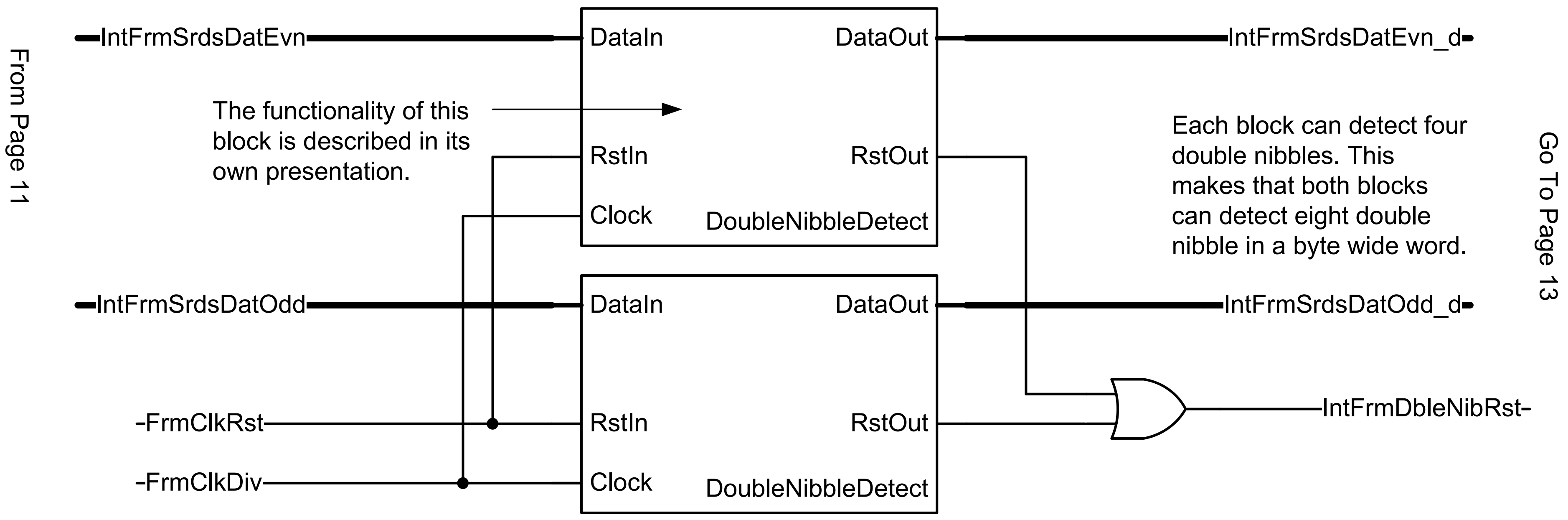
! Only needed when the ADC is used in 1-wire mode. !

The “wire” modes are explained from page 4 to page 8.
Page 21 and 22 highlight the “Double Nibble” phenomena.

This block (hierarchical level) is added to the design when 1-wire mode is selected.

In 2-wire mode this design is not present.

```
IntFrmSrdsDatEvn_d <= IntFrmSrdsDatEvn;
IntFrmSrdsDatOdd_d <= IntFrmSrdsDatOdd;
IntFrmDbleNibRst <= Low;
```



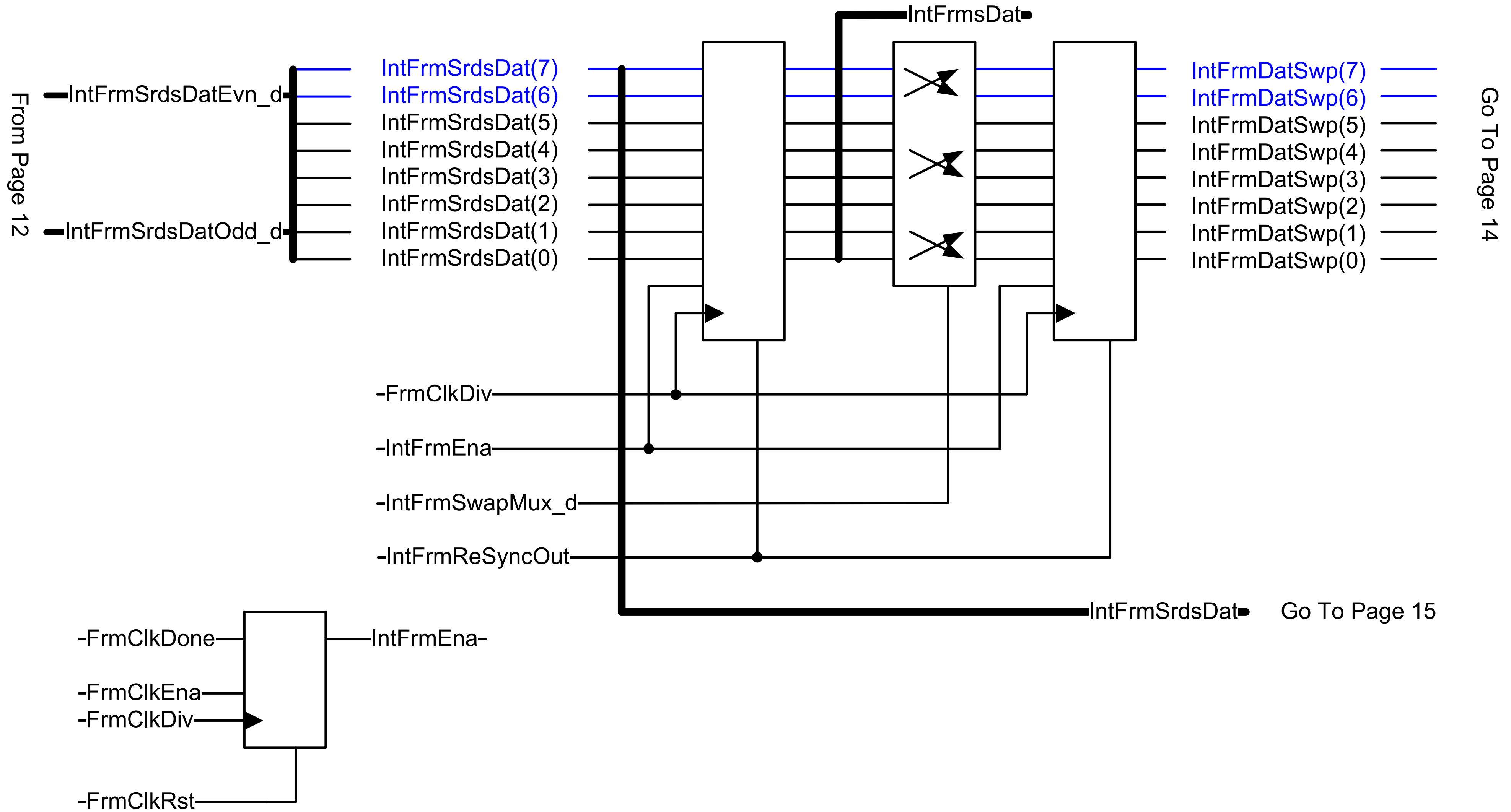
The functionality of this block is described in its own presentation.

Each block can detect four double nibbles. This makes that both blocks can detect eight double nibble in a byte wide word.

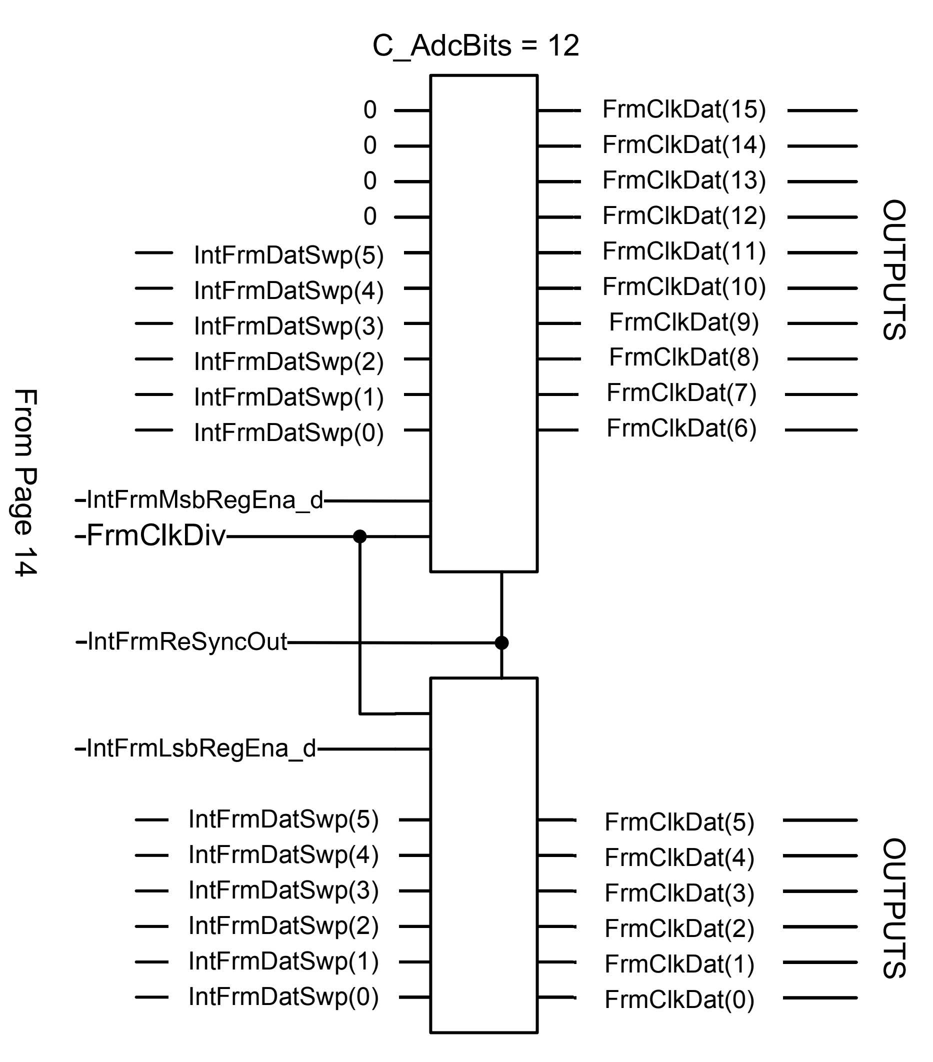
This block has a build in reset delay. When the reset is released the block starts functioning, while the block is still in reset the output is zero.
The “RstOut” signal is generated when a fifth DoubleNibble is detected.
Then the internal logic is reset and also external the block measures can be taken.

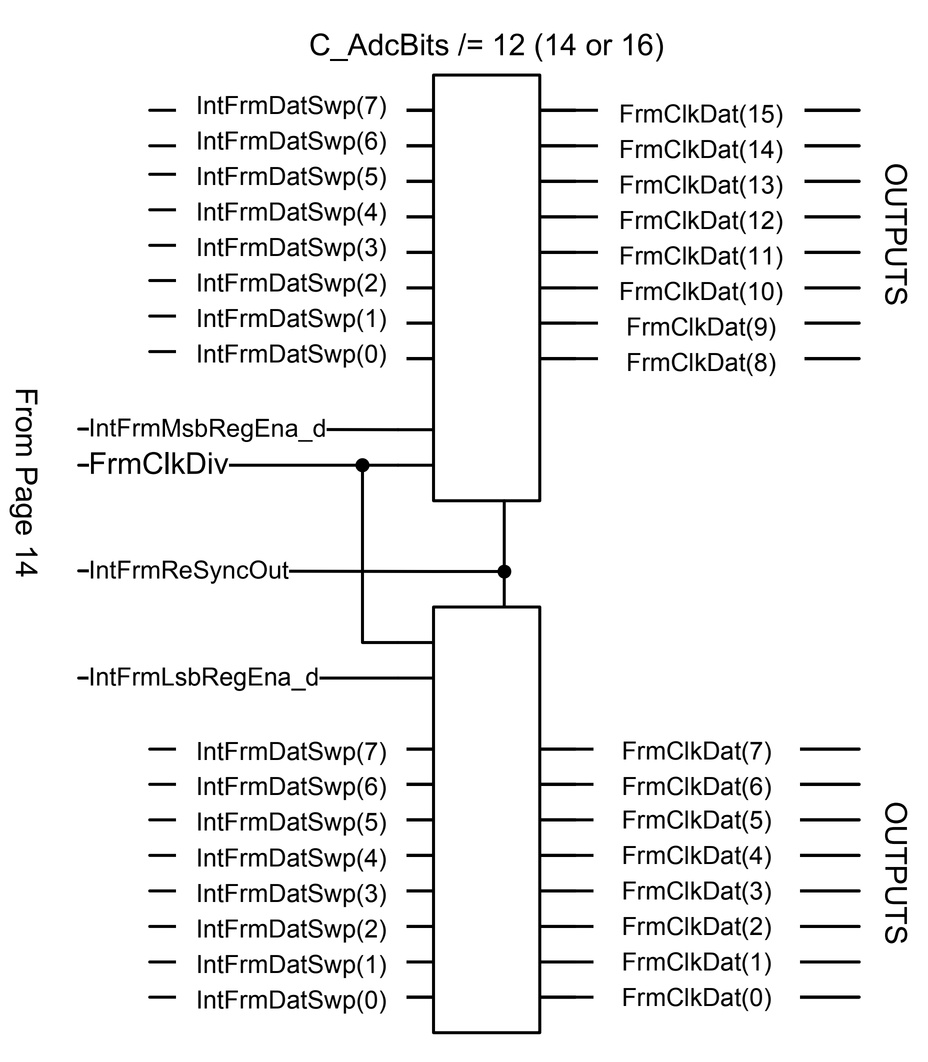
Bit Swap Multiplexer

For a 12-bit Frame detection circuit the **blue** signals are not available.

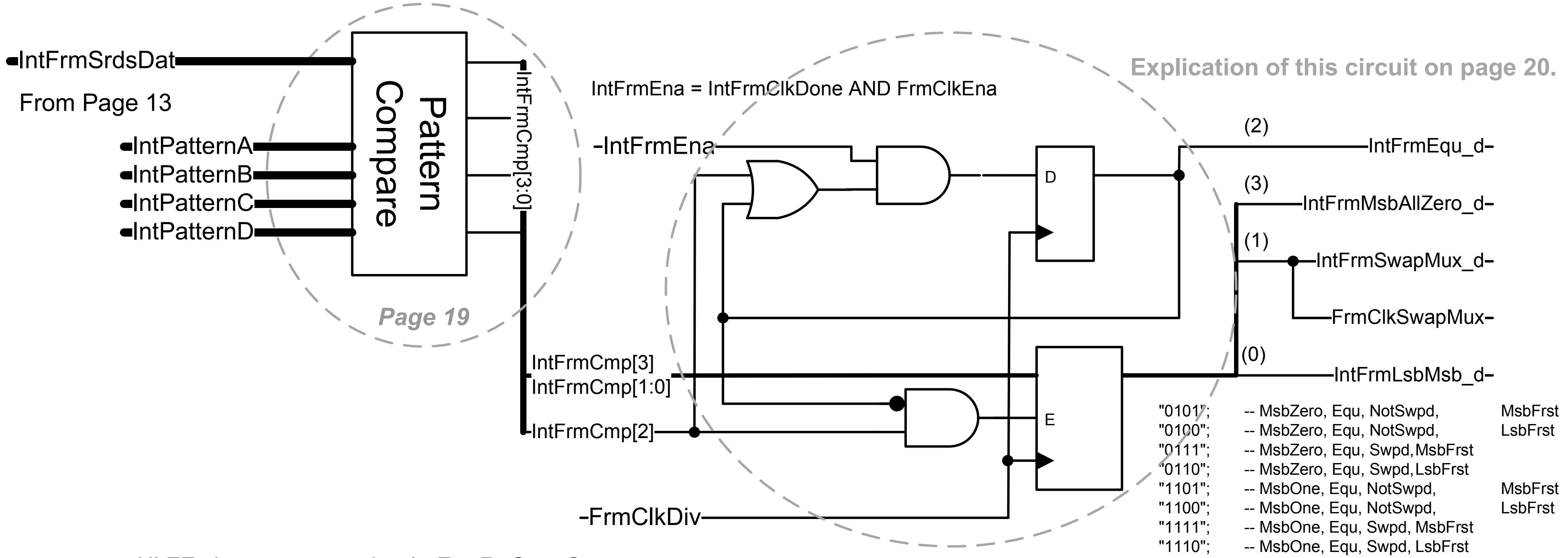


Output Registers





Pattern Check

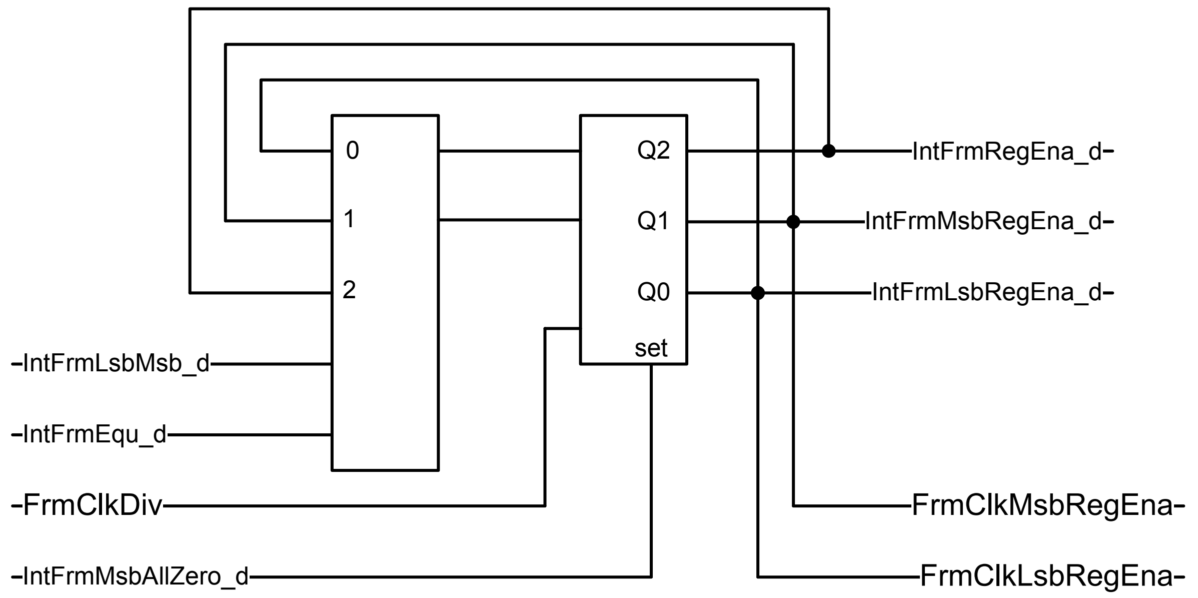


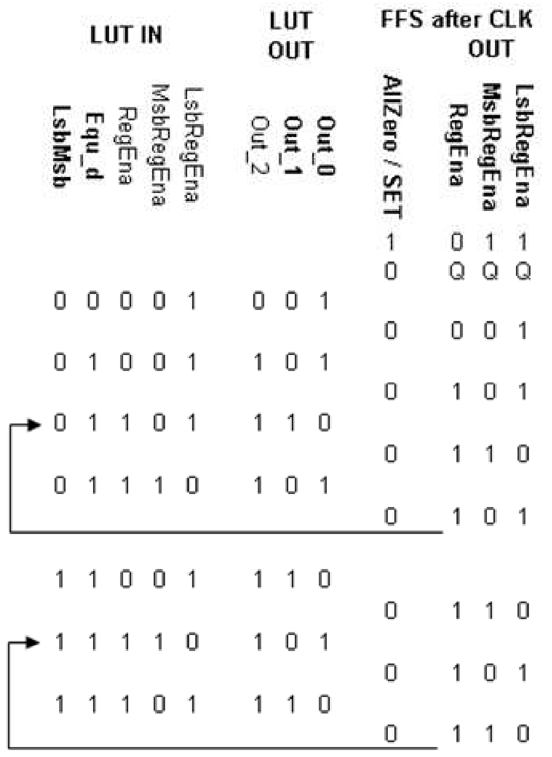
"0101";	-- MsbZero, Equ, NotSwpd,	MsbFrst
"0100";	-- MsbZero, Equ, NotSwpd,	LsbFrst
"0111";	-- MsbZero, Equ, Swpd,MsbFrst	
"0110";	-- MsbZero, Equ, Swpd,LsbFrst	
"1101";	-- MsbOne, Equ, NotSwpd,	MsbFrst
"1100";	-- MsbOne, Equ, NotSwpd,	LsbFrst
"1111";	-- MsbOne, Equ, Swpd, MsbFrst	
"1110";	-- MsbOne, Equ, Swpd, LsbFrst	

All FFs here are reset by: IntFrmReSyncOut

Page 21 +

Select Output Register



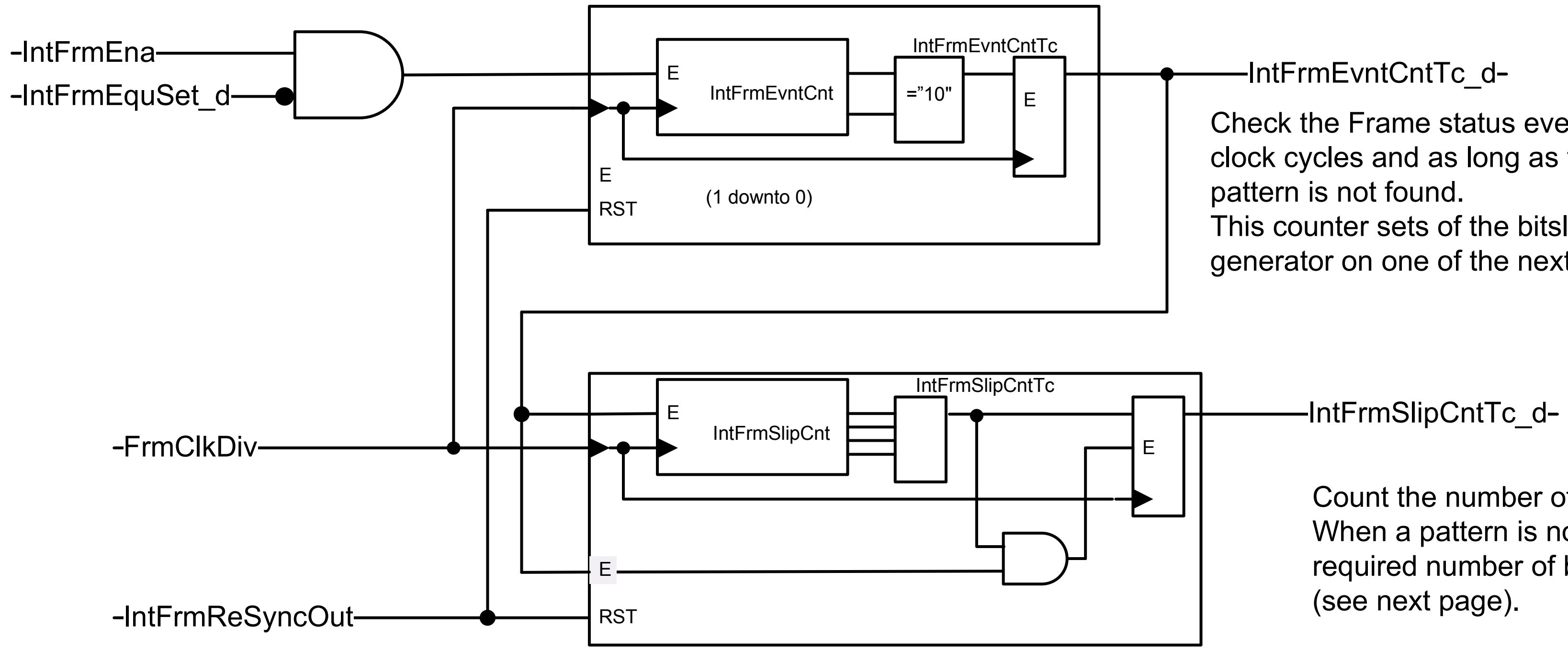


LUT IN					LUT OUT			FFS after CLK OUT			
LsbMsb	Equ_d	RegEna	MsbRegEna	LsbRegEna	Out_2	Out_1	Out_0	AllZero / SET	RegEna	MsbRegEna	LsbRegEna
								1	0	1	1
								0	Q	Q	Q
0	0	0	0	1	0	0	1	0	0	0	1
0	1	0	0	1	1	0	1	0	1	0	1
0	1	1	0	1	1	1	0	0	1	1	0
0	1	1	1	0	1	0	1	0	1	0	1
1	1	0	0	1	1	1	0	0	1	1	0
1	1	1	1	0	1	0	1	0	1	0	1
1	1	1	0	1	1	1	0	0	1	1	0

Microsoft Excel Worksheet

Sample Counters

Count 0, 1, 2, 3 ... 15
Terminal count at 15



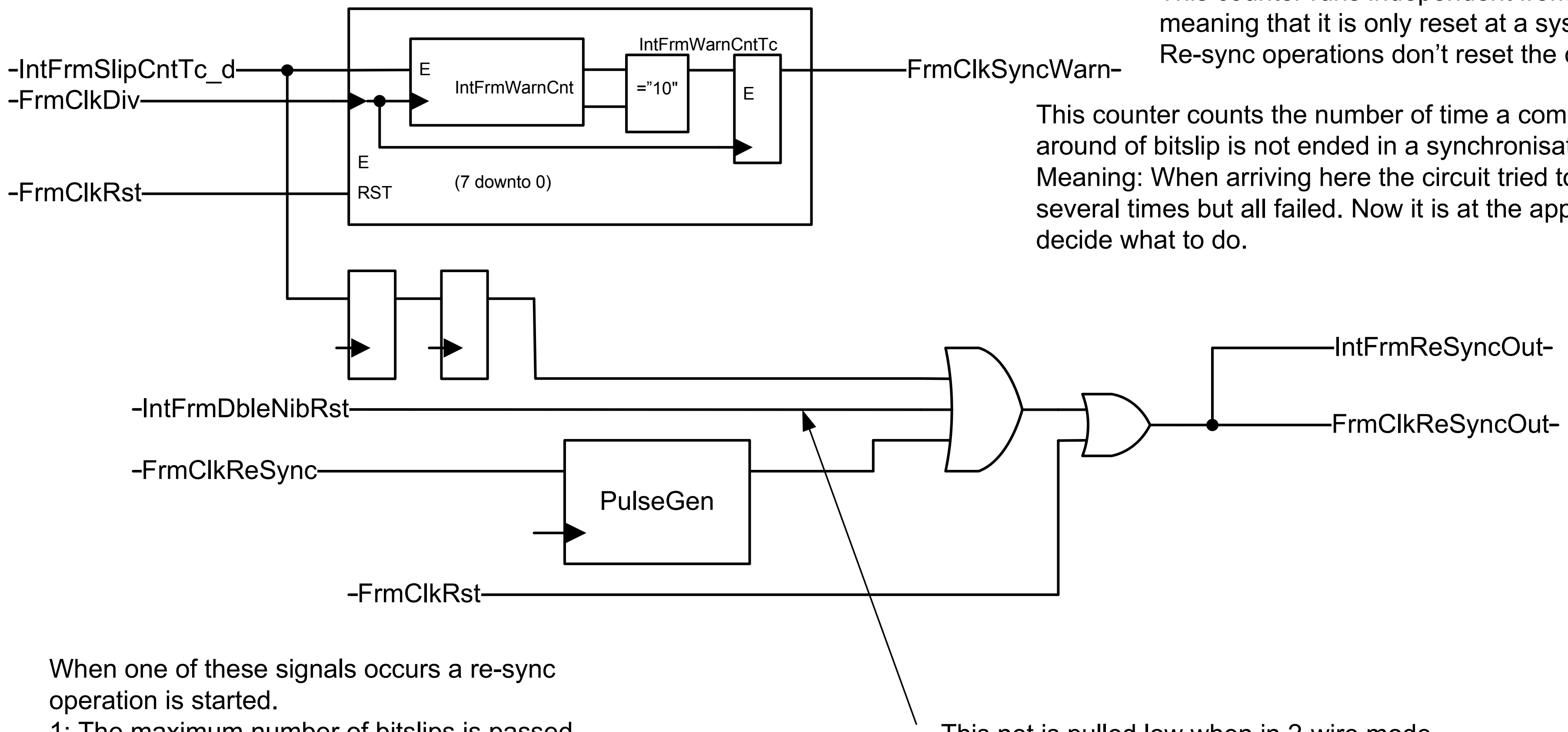
Check the Frame status every 16 clock cycles and as long as the pattern is not found.
This counter sets of the bitslip pulse generator on one of the next pages.

Count the number of bitslips occurred.
When a pattern is not recognized in the required number of bitslips do something (see next page).

Sizable counter in the source code for the Frame signal.

Counter programmable for 1-wire or 2-wire mode and 12-bit or 14/16-bit.
The counter counts the number of bitslips occurred.
For 2-wire mode it signals one over-run for 1-wire mode it signals two over-runs.

Reset and Re-Sync



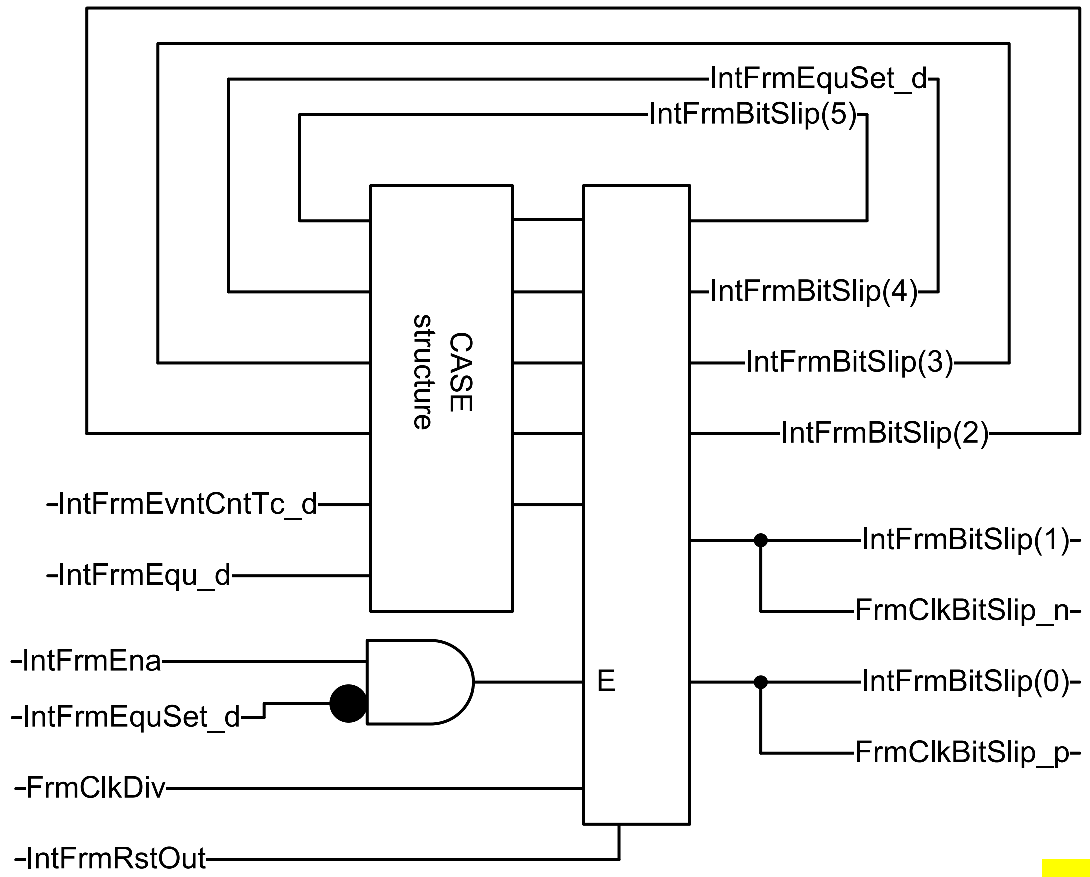
This counter runs independent from all logic, meaning that it is only reset at a system reset. Re-sync operations don't reset the counter.

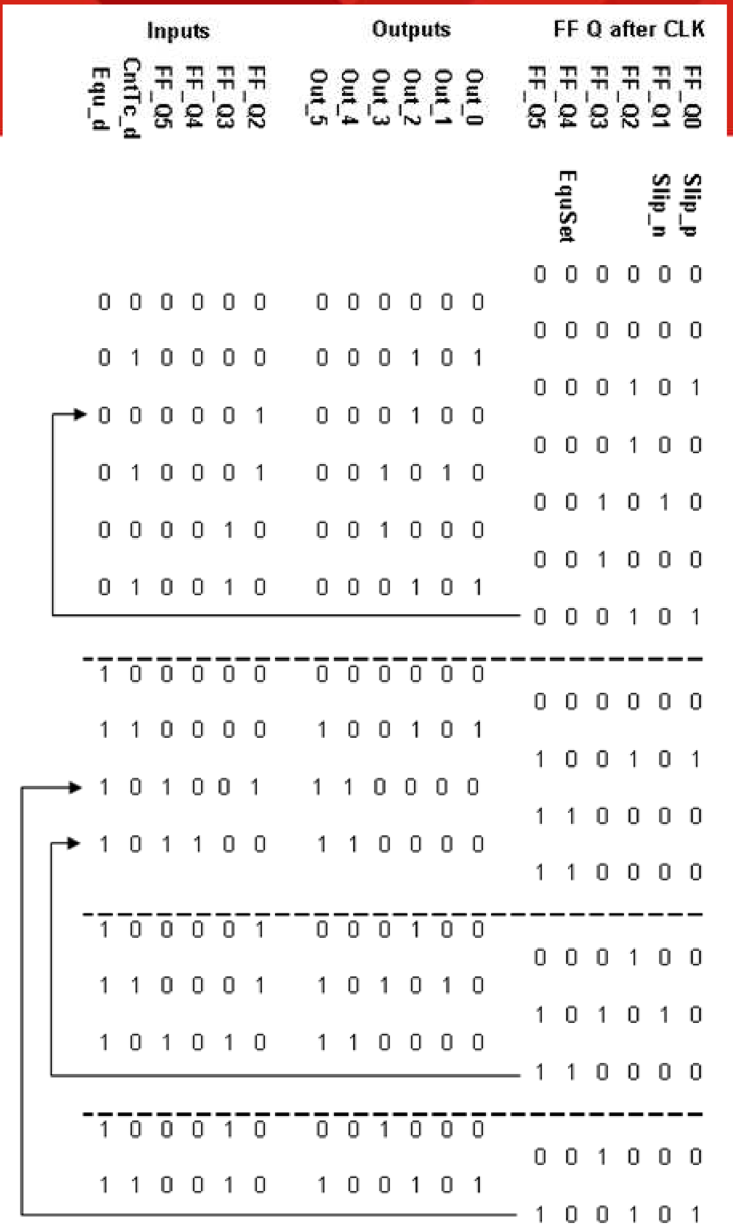
This counter counts the number of time a complete turn around of bitslip is not ended in a synchronisation. Meaning: When arriving here the circuit tried to lock for several times but all failed. Now it is at the application to decide what to do.

When one of these signals occurs a re-sync operation is started.
1: The maximum number of bitslips is passed without detection of a matching pattern.
2: One of the two ISERDES gave at its output twice the same value, more about this on page 21.
3: the application asked for a re-sync.

This net is pulled low when in 2-wire mode.

Bitslip State Machine



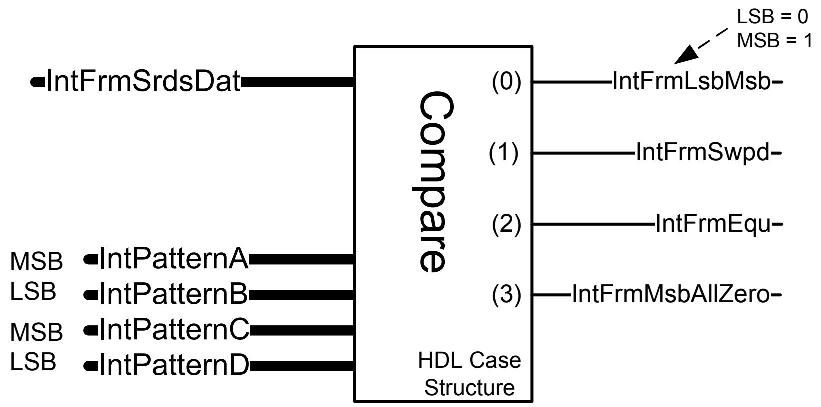


Inputs						Outputs						FF Q after CLK					
Equ_d	CntTc_d	FF_Q5	FF_Q4	FF_Q3	FF_Q2	Out_5	Out_4	Out_3	Out_2	Out_1	Out_0	FF_Q5	FF_Q4 EquSet	FF_Q3	FF_Q2	FF_Q1 Slip_n	FF_Q0 Slip_p
												0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0						
												0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0	1						
												0	0	0	1	0	1
0	0	0	0	0	1	0	0	0	1	0	0						
												0	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	1	0						
												0	0	1	0	1	0
0	0	0	0	1	0	0	0	1	0	0	0						
												0	0	1	0	0	0
0	1	0	0	1	0	0	0	0	1	0	1						
												0	0	0	1	0	1
1	0	0	0	0	0	0	0	0	0	0	0						
												0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	1	0	1						
												1	0	0	1	0	1
1	0	1	0	0	1	1	1	0	0	0	0						
												1	1	0	0	0	0
1	0	1	1	0	0	1	1	0	0	0	0						
												1	1	0	0	0	0
1	0	0	0	0	1	0	0	0	1	0	0						
												0	0	0	1	0	0
1	1	0	0	0	1	1	0	1	0	1	0						
												1	0	1	0	1	0
1	0	1	0	1	0	1	1	0	0	0	0						
												1	1	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0	0						
												0	0	1	0	0	0
1	1	0	0	1	0	1	0	0	1	0	1						
												1	0	0	1	0	1

Microsoft Excel Worksheet

Compare Frame Patterns (1)

The given example is for 8-bit and 16-bit patterns but the same is done for 6-bit and 12-bit patterns!



Patterns are entered in the HDL code as 16-bit values.
In case of a 8-bit frame pattern must have the MSB byte of the given pattern set to all zero.

Entered Frame patterns in HDL:

	MSB	LSB
8-bit :	00000000	11110000
or		
16-bit :	11111111	00000000

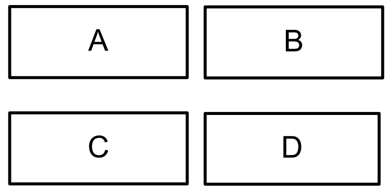
When synthesizing the design the synthesis report shows the settings of the pattern to search for. This is also shown in the on screen report under the AdcFrm synthesis section.

This results in the table below.
The table is constructed with HDL functions and requires no logic to be made.

A and B is the given pattern shifted by one. Read the start of this document for more on this C and D is the shifted pattern with swapped bit positions. Read the start of this document for more about this.

	8-bit	16-bit
IntPatternA	00000000	11111110
IntPatternB	00011110	00000001
IntPatternC	00000000	11111101
IntPatternD	00101101	00000010

Pattern shifted for one bit.
Shifted pattern bitswapped

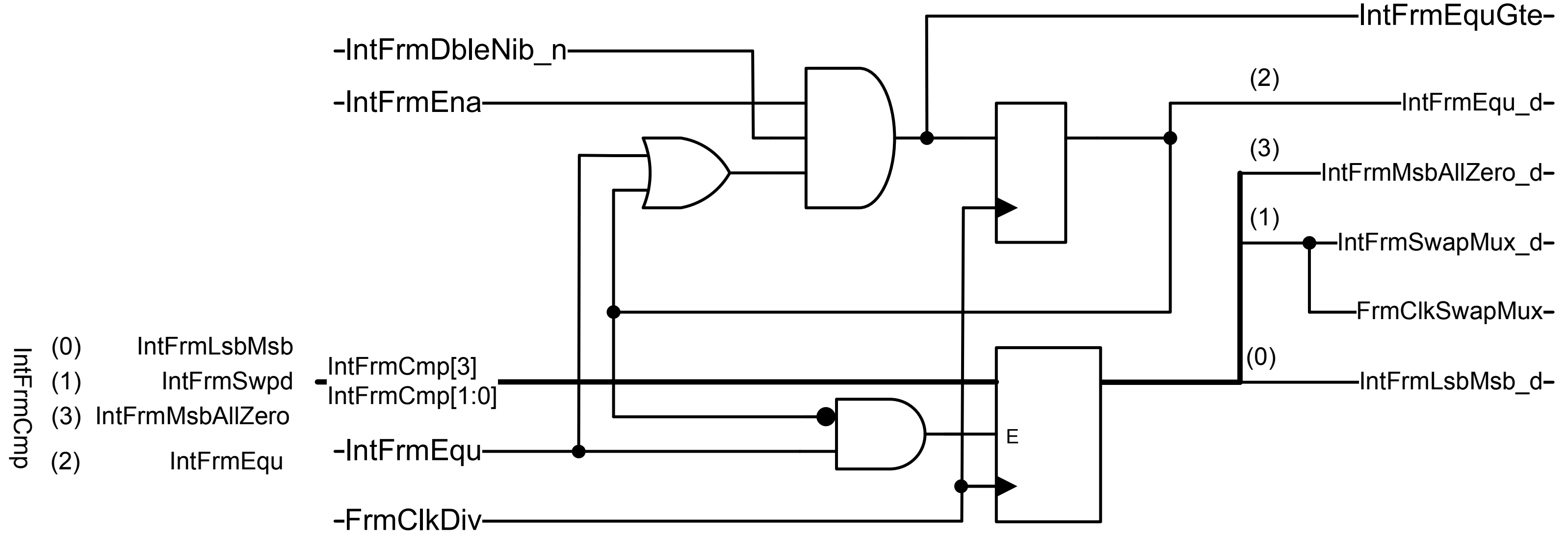


When pattern "A" is found it means that the MSByte is found.
Register the received data in MSB register and the next received byte in the LSB register.
When pattern "B" is found it means LSByte is found.
Register the next received data in the MSB regsiter and the next data in the LSB register.
Do the same for "C" and "D" but then also invoke the bit swap multiplexer.

Compare Frame Patterns (2)

- "IntFrmEna" needs to high before the circuit can start working.
- Pattern detection:
  - As long as "intFrmDbleNibble" is not active, low in this case, the pattern comparator can pass data to the "IntFrmEqu_d" FF.
  - The "IntFrmEqu" status and three other signals are registered whenever a matching frame pattern is found.
  - The combinatorial "IntFrmEqu" signal immediately disables (blocks) the "Double Nibble" circuit.
  - That disable signal is then overtaken by the registered version of the "IntFrmEqu" signal (IntFrmEqu_d"
  - When the "IntFrmEqu_d" signal goes high, indicating that a pattern has been found, it disables the registers of the other three other status bits. This action prevents that these signals change status after detection of the required pattern.

- Double Nibble detection
  - This circuit is described later in this document.
  - Only one thing here:
    - Whenever a "frame pattern" is found this circiuit is disabled, blocked, immediately.



XILINX®

Compare Equal Nibbles (1)

! Only needed when the ADC is used in 1-wire mode. !

When an ADC is used in 1-wire mode the frame pattern must be detected aver a range of 16-bits.

An ISERDES outputS then sometimes twice the same data. This is shown in figure 1.
This has no effect at all on regular source synchronous interface that synchronize through data training pattern.
It only delays the time the synchronisation occurs.

In case of the ADC interface this behaviour has a big impact!
The interface is synchronised on one signal, the frame clock and all data inputs are shifted (bitslipped) along with the frame pattern while it is synchronizing. Thus when something happens with the frame signal it has impact on the data capturing.

This is what happens:
- Due to the fact that the ISERDES bitslip operation doesn't occur for both ISERDES at the same time, one ISERDES will output twice the same data before the other.
- This effect will complete destroy any possibility for synchronisation.
- The results from figure 1 taken apart:

- After a couple of bitslip operations the output of the ISERDES is "3" and "E" this results in the byte "AD".
- The next output of the ISERDES is "C" and "1" resulting in a byte "52".
- The "…_p" ISERDES gets a bitslip request and executes it. The result should be "1" and "E" resulting in "A9".
- Instead the ISERDES outputs a second time "C" and the resulting byte is now "F8" ("C" and "E").
- The next CLKDIV clock edge the earlier operated bitslip occurs at the output of the ISERDES, resulting in: "1" and "1" or "03".
- And so on.

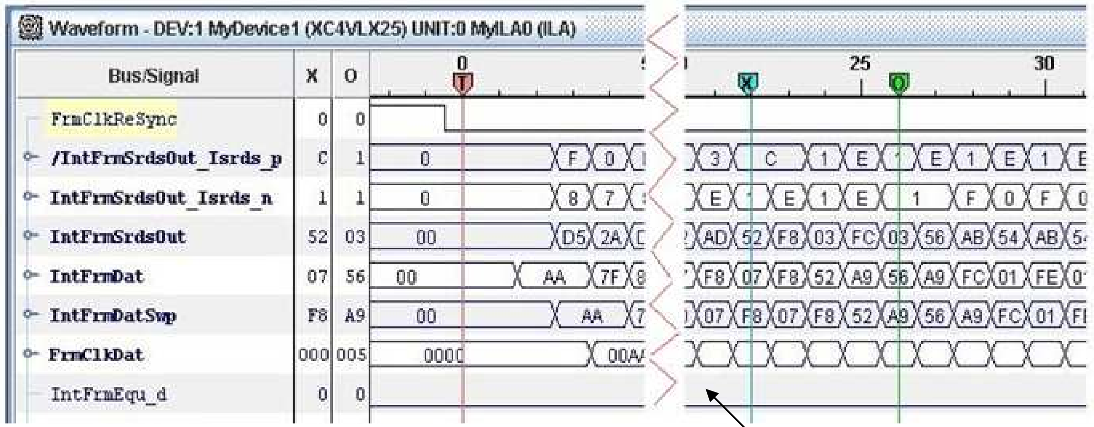


Figure 1

A nible got a bitslip and is suddenly delayed with one CLKDIV period.

Next page show this graphically.

Start of explication

Compare Equal Nibbles (2)

! Only needed when the ADC is used in 1-wire mode. !

This is the pattern seen in the Figure 1 on previous page.

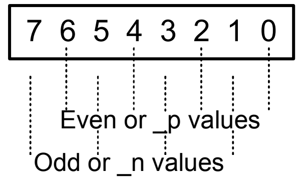
...SrdsOut_Isrds_p	...SrdsOut_Isrds_n
C	1
3	E
C	1
C	E
1	1
E	E
1	1
E	1
1	F
E	0

The "…_p" ISERDES gets a bitslip request here. This should result in a bitslipped patten here:

0011 after bitslip = 0001

The bitslip doesn't occur, in stead the ISERDES outputs again the previous value "C".

It is only at the next CLKDIV clock edge that the output of the ISERDES shows the bitslipped value "1".



"3" → 0011
"E" → 1110
Results in:
1 0 1 0 1 1 0 1 = "AD"

The same happen at this side. In stead of outputting immedately the bitslipped value "F", the ISERDES outputs the previous value "1" one more time before it shows the bitslipped value.

If everything would workout well the ISERDES pattern output should look as:

"C"	"1"	→ "52"
"3"	"E"	→ "AD"
"C"	"1"	→ "52"
"1"	"E"	→ "A9"
"E"	"1"	→ "56"
"1"	"E"	→ "A9"
"E"	"1"	→ "56"
"1"	"F"	→ "5B"
"E"	"0"	→ "54"

When everything went OK, one or two extra bitslips would have produced the searched pattern "BD6E".

To prevent this from happening, at the same time the frame pattern is searched a circuit should check the output of each ISERDES for a repeating pattern and then take action. This is described on next page.

XILINX