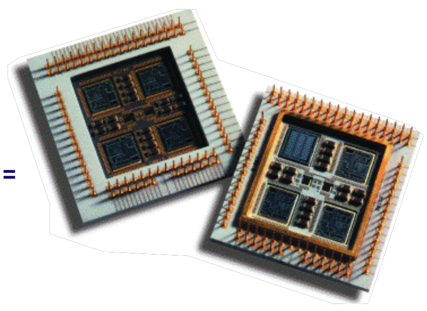
ASICentrum® A COMPANY OF THE SWATCH GROUP
Novodvorska 994, 142 21 Praha 4, Czech Republic
Tel. (+420 2) 4404 3478, Fax: (+420 2) 4149 2691, E-mail: info@asicentrum.cz



ACFFT IP CORE
FFT PROCESSOR

BASIC PARAMETERS

- configurable fixed point FFT processor
- DSP processor architecture
- 2's complement arithmetic
- multiplication by window-weighting function available (rectangle, Gaussian α, Von Hann, Kaiser α and others)
- configurable data and coefficient widths
- configurable precision and output scale ability
- FFT algorithm specified by program in internal ROM
- coefficients stored in internal ROM

Optional enhancement

- CORDIC algorithm to compute magnitude spectrum

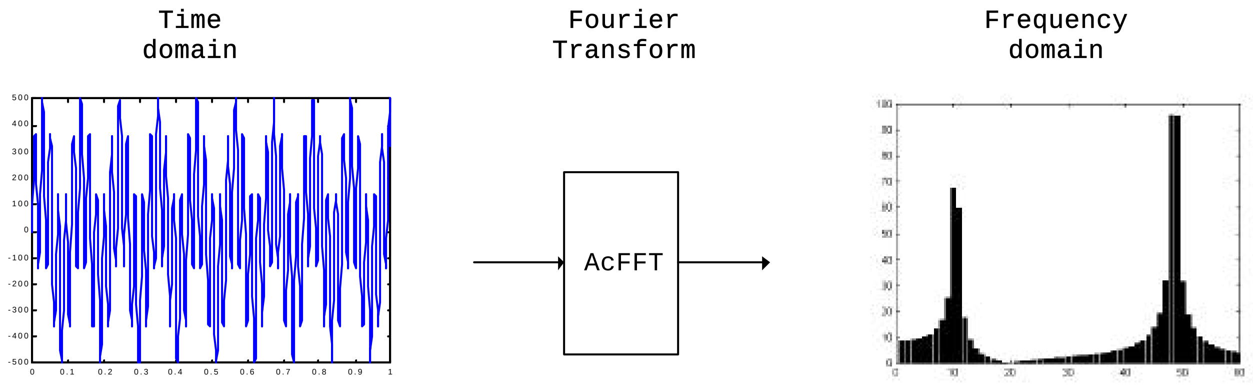
FEATURES

Completely synchronous design
Technological independence
Fully verified according compliance checklist

DELIVERABLES

- Design file formats
  - RTL VHDL synthesisable code
  - netlist
- Technical documentation
  - Data sheet
- Verification tools
  - Testbench
  - Application FPGA board
- Constraints files
- Maintenance

DESCRIPTION

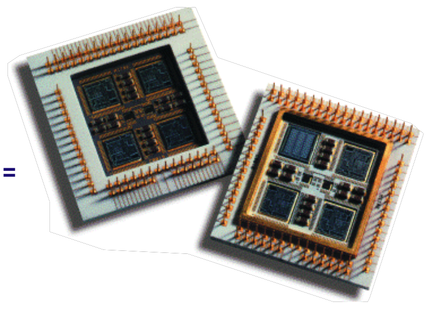


Fast Fourier Transform (FFT) is an algorithm to compute Discrete Fourier Transform (DFT) consuming O (N*log$_2$N) runtime (N - FFT size). DFT transforms time domain to frequency domain using equation :

$$X[k] = \sum_{n=0}^{N-1} x(n) W_N^{kn}$$

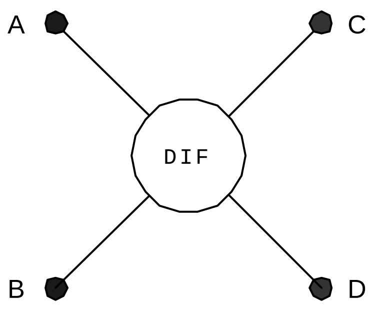
where W$_N$ represents "twiddle factors" :

$$W_N = e^{-j\frac{2p}{N}}$$



PROCESSOR REALIZATION

Processor uses DIF - RADIX2 algorithm which consists of basic computational elements - the butterflies. Each butterfly has two inputs and produce two outputs by the following equations :

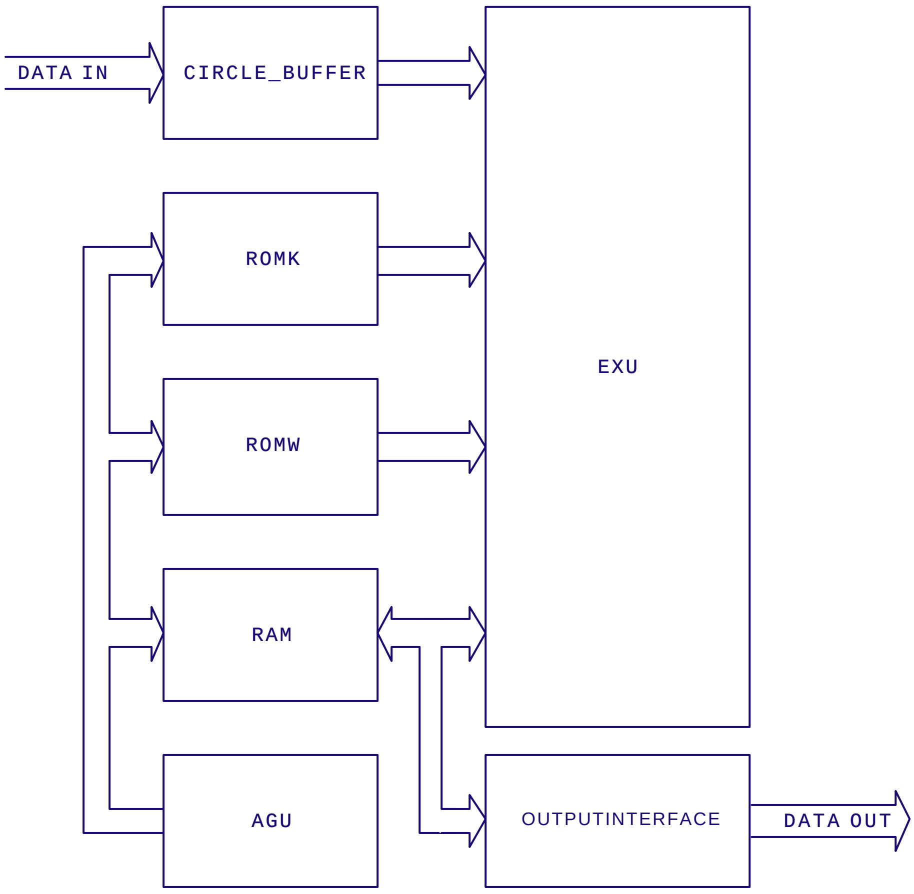


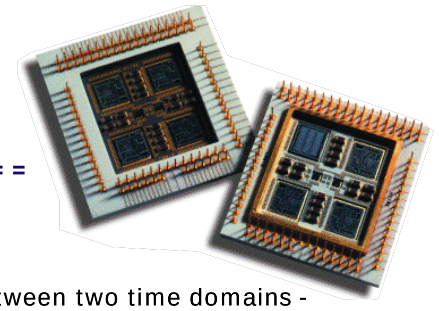
```
Cre = Are + Bre
Cim = Aim + Bim

Dre = (Are - Bre)*Wre - (Aim - Bim)*Wim
Dim = (Are - Bre)*Wim + (Aim - Bim)*Wre
```

Inputs A, B are divided by 2 to avoid overflow.

The processor consists of Circle Buffer Unit, Address Generator Unit (AGU), ROM with "twiddle factors" (ROMW), ROM with window-function coefficients (ROMK), RAM, Executive Unit (EXU) and Output Interface :





FUNCTIONAL DESCRIPTION

Input data are stored in CIRCLE_BUFFER unit, which works as interface between two time domains - write_clk (clocks input data) and clk (system clock). When a sample is read from unit, it is not 'cleared' so that one sample can be read several times. The FIFO (first-in, first-out) rule is observed during reading. If the internal memory is full, the incoming samples overwrite oldest samples. This behaviour provides loading of input data in N-point window.
The macro works in 3 states : LOAD, FFT, LIST. During the LOAD state the data from CIRCLE_BUFFER are loaded and multiplied by window-coefficients. The FFT state computes complex spectra and LIST state send complex spectra out of macro. The LOAD state is started by start_fft signal, FFT state follows immediately, LIST state is started by start_list signal. Complex spectra consists of real and imaginary part.

FFT RUNTIME

The runtime depends on FFT length :

State	Clock cycles
LOAD	$W_L = N + 10$
FFT	$W_F = 2\ N \log_2(N) + 16$
SAVE	$W_S = N + 4$

Save time is proportional shorter when sending less than N samples out of macro (sending one sample takes 1 clock cycle)

SAMPLE RATE

Due to use of CIRCLE_BUFFER the sample rate could be :

$$f_{write\_clk} < \frac{f_{clk}}{2}$$

Let have time of complete cycle FFT :

$W = W_L + W_F + W_S$

$T = W \cdot T_{clk}$

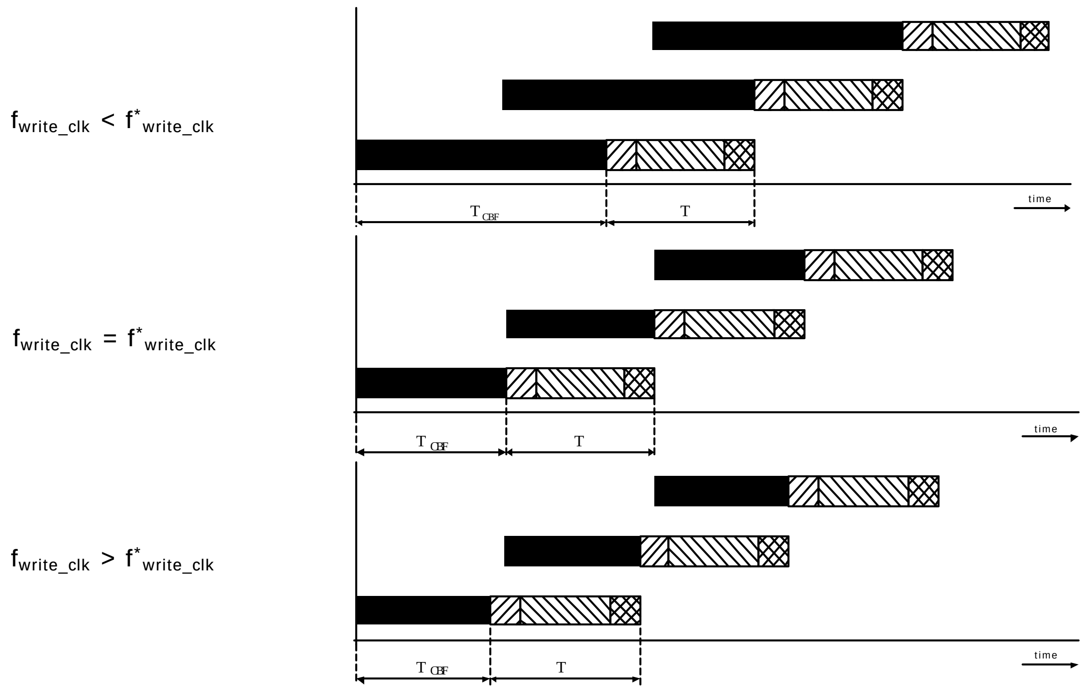
There is N samples stored in CIRCLE_BUFFER. They represent time interval :

$T_{CBF} = N \cdot T_{write\_clk}$

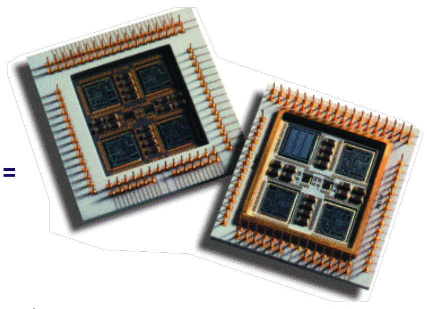
The critical sample rate is :

$$f^*_{write\_clk} = \frac{N}{W} f_{clk}$$

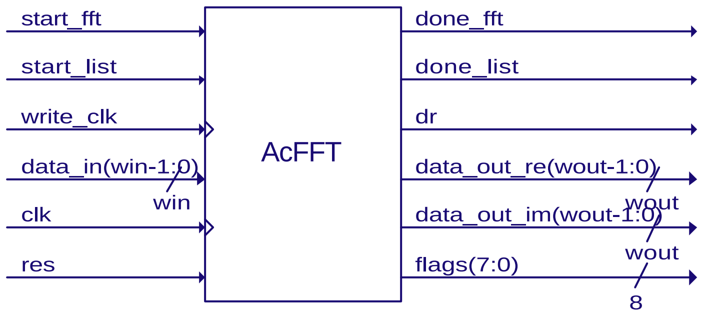
Three cases of relationship between f_{write_clk} and $f^*_{write\_clk}$ can occur :


$f_{write\_clk} < f^*_{write\_clk}$
T_{CBF}
T
time
$f_{write\_clk} = f^*_{write\_clk}$
T_{CBF}
T
time
$f_{write\_clk} > f^*_{write\_clk}$
T_{CBF}
T
time

■ input samples stored in CIRCLE_BUFFER memory and used to compute FFT ▨ LOAD state ▧ FFT state ⊠ SAVE state



ACFFT PINOUT



Port name	Direction	Description
CLK	Input	System clock (~50MHz)
RES	Input	Asynchronous global reset
WRITE_CLK	Input	Clocks input data
START_FFT	Input	Starts FFT state
START_LIST	Input	Starts LIST state
DATA_IN(WIN-1:0)	Input	Input data
DONE_FFT	Output	State FFT is done
DONE_LIST	Output	State LIST is done
DR	Output	Output data valid
DATA_OUT_RE(WOUT-1:0)	Output	Output data - real part of complex spectra
DATA_OUT_IM(WOUT-1:0)	Output	Output data - imaginary part of complex spectra
FLAGS(7:0)	Output	Test bits

CORE PARAMETERISATION

ACFFT allows easy modification according customer's need :

- input and output data width
- selection of window-weighting function
- coefficient precision (both "twiddle factors" and window-weighting function)
- RAM width

This modification is provided by ASICentrum :

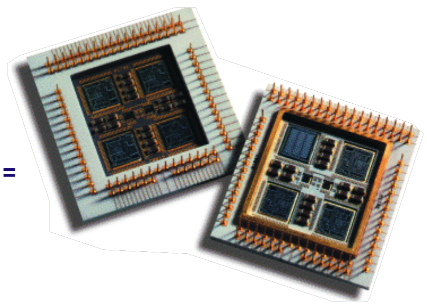
- custom instructions may be included

The list of generic parameters :

Generic	Type	Description
m	Integer	2^m - point FFT
win	Integer	input data width
wout	Integer	output data width
wram	Integer	internal RAM width
wromw	Integer	"twiddle factors" width
wromk	Integer	window-weighting function coefficients width

TECHNOLOGY SPECIFIC ISSUES

Core is implemented as fully synchronous synthesizable RTL VHDL code including RAM.



FPGA IMPLEMENTATION OVERVIEW

FFT size	Generics					Device	System Frequency	Runtime (LOAD+FFT)	Utilization	Block RAM utilization	Equivalent Gates
[points]	win	wout	wram	wromw	wromk		[MHz]	[µs]			
256	12	12	30	10	7	XC2S50-5PQ208	58.9	74.4	82 %	3 of 8	62 992
						XCV50-4CS144	54,5	80.4	88 %	3 of 8	63 031
512	12	12	30	11	7	XC2S50-5PQ208	56.5	172.8	90 %	6 of 8	113 166
						XCV50-4CS144	55.0	177.4	96 %	6 of 8	113 222
1024	12	12	30	12	7	XC2S150-5PQ208	56.1	384.0	46 %	11 of 12	197 194
						XCV150-4PQ240	54.8	392.7	49 %	11 of 12	197 194
1024	12	12	32	14	14	XCV200-4PQ240	52.7	408.3	45 %	11 of 14	201 407
2048	12	12	30	13	7	XCV600-4BQ432	49.4	953.1	14 %	21 of 24	363 715

VERIFICATION METHODS

ACFFT core has been tested using functional and post-layout timing VHDL simulation (after FPGA implementation). Both functional and timing testbenchs are available.

APPLICATION

Spectrum analyzer