



Fraunhofer Institut
Integrierte Schaltungen

***CorePool* FHG_FFT**

Databook

- subject to change without notice -

Version: v1.2

Date: 06.08.02

Document History

Table 1: Document History

| Version | Date | Responsible | Description |
|---------|----------|-------------|--------------------------|
| v1.0 | 01.12.00 | Bes | databook created |
| v1.1 | 10.01.01 | Bes | add overflow control |
| v1.2 | 19.03.01 | Bes | add parameter LOG_FFT_LE |
| | | | |
| | | | |
| | | | |

Contact

CorePool
Fraunhofer Institute Integrated Circuits

Am Wolfsmantel 33
91058 Erlangen
Germany

Phone: +49 (0) 9131 776 777
Fax: +49 (0) 9131 776 499
Email: info@corepool.com
Internet: <http://www.corepool.com>

Table of Contents

| | |
|---|-----------|
| Document History | 2 |
| Contact | 2 |
| Purpose | 7 |
| Features | 7 |
| Design Kit | 7 |
| Requirements | 7 |
| References | 8 |
| Block Diagram | 8 |
| General Information | 8 |
| Signal Description | 9 |
| Parameter Description | 10 |
| Interfaces | 10 |
| VHDL Usage through Component Instantiation | 11 |
| Architecture | 14 |
| Waveforms and timing tables | 15 |
| Application notes | 18 |

List of Tables and Figures

| | |
|---|-----------|
| Document History | 2 |
| Block Diagram | 8 |
| Signal Description | 9 |
| Parameter Description | 10 |
| Fourier Transformation of order N | 15 |
| „butterfly“ representation | 15 |
| Waveform of external data write access | 16 |
| Timing table of external data write access | 16 |
| Waveform of external data read access | 17 |
| Timing table of external data read access | 17 |
| Application Example | 18 |

Purpose

The CorePool component FHG_FFT computes a parameterizable and synthesizable Fast Fourier Transformation radix-2 algorithm.

The tunable parameters are the number of values of the FFT (must be a power of 2), the wordlength of the complex input data, the wordlength of the sine and cosine coefficients.

Features

- Fully Parameterized Fast Fourier Transformation
- Fully synchronous design
- Fully synthesizable
- FFT order parameterized
- Wordlength of complex input data parameterized
- Wordlength of the sine and cosine coefficients parameterized
- Registered Inputs and Outputs
- Inputs and Outputs data in Two's complement Format

Design Kit

- Technology Independent Implementation as Synopsys Design Ware Components
- VHDL Source Code Simulation Models
- VHDL Compliance Test Suite
- Auxiliary Simulation Models for User Testbenches
- Synthesis and Testsynthesis Scripts
- Design Support, Netlist Synthesis Service and Consulting available

Requirements

Simulation

- VHDL IEEE-1076 Simulator

Synthesis

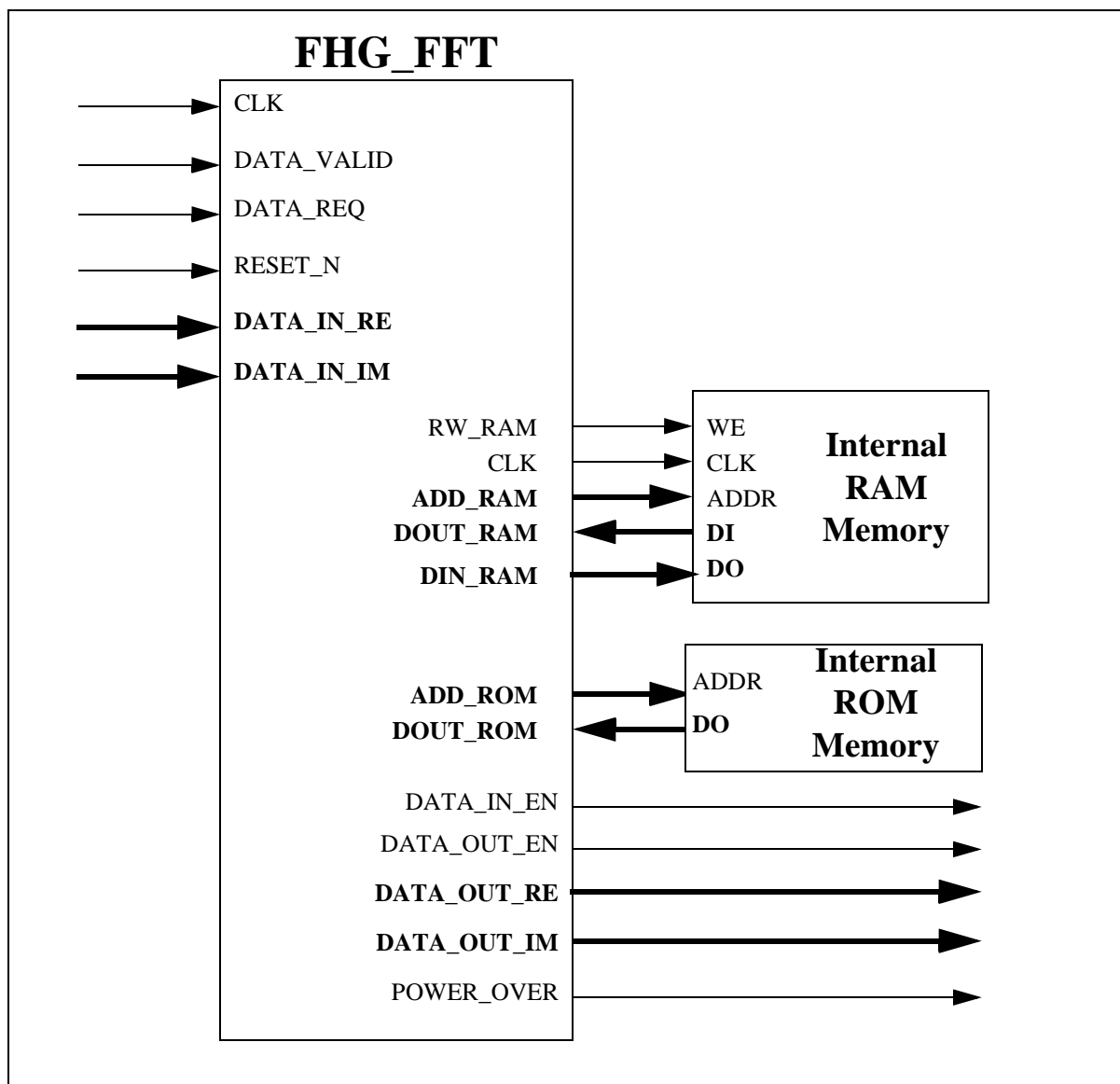
- Synopsys Design Compiler

References

For more details about the FHG_FFT refer to standard literature about the fast Fourier Transformation Algorithm.

Block Diagram

Figure 1: Block Diagram



General Information

It is assumed that the complex input data are applied parallely to the input of the FFT core.

Signal Description

Table 2: Signal Description

| Pin | Direction | Bitwidth | Description |
|-------------|-----------|--------------------------|--------------------------------------|
| CLK | IN | 1 | System clock. |
| RESET_N | IN | 1 | Low active asynchronous reset. |
| DATA_VALID | IN | 1 | High active |
| DATA_REQ | IN | 1 | High active |
| DATA_IN_RE | IN | bw_data | Real data input bus |
| DATA_IN_IM | IN | bw_data | Imaginary data input bus |
| DATA_IN_EN | OUT | 1 | High active |
| DATA_OUT_EN | OUT | 1 | High active |
| DATA_OUT_RE | OUT | bw_data +2 | real data output bus. |
| DATA_OUT_IM | OUT | bw_data +2 | Imaginary data output bus. |
| POWER_OVER | OUT | ceil(log ₂ L) | 2 power, multiplication of data_out. |

The input port CLK is the system clock. The data are applied parallelly to the input of the FFT core with the corresponding clock rate.

The input port RESET_N is a low active asynchronous reset for the FHG_FFT. An asynchronous reset should be performed at the beginning of the Fourier transformation process.

The input port DATA_IN consists of the complex input data bus.

The input data format is :

- imaginary part = DATA_IN_IM[bw_data -1 : 0]
- real part = DATA_IN_RE[bw_data -1 : 0]

The input port DATA_VALID is a high active signal. When valid, the FFT core knows that the incoming data are valid. At first the data are loaded in RAM, if DATA_VALID is high and the output port DATA_IN_EN low. In fact DATA_IN_EN when active (high) prevent the input data to be loaded and means that a process is already active.

DATA_IN_EN returns to „low“ state only when the output data have been read by the external device.

The output port DATA_OUT_EN signals to the external device , when active (high), that the data have been processed and that they are available for reading. The reading process starts when data are requested by the external device (input port DATA_REQ active (high)).

DATA_IN_EN and DATA_OUT_EN return „low“ when all the data have been read.

The output port DATA_OUT consists of the complex output data bus.

The output data format is :

- imaginary part = DATA_OUT_IM[bw_data +1 : 0]
- real part = DATA_OUT_RE[bw_data +1 : 0]

The output port POWER_OVER consists of the power of 2, with which the DATA_OUT can be multiplied in order to recover the original value.

(DATA_OUT <= DATA_OUT * 2^{POWER_OVER})

Parameter Description

Tabelle 3: Parameter Description

| Name | Type | Default Value | Value Range | Description |
|------------|---------|---------------|-------------|---|
| FFT_LENGTH | Integer | none | | FFT_LENGTH = $\log_2(N)$ N = FFT value number (power of 2) ex : N = 1024, FFT_LENGTH = 10 |
| BW_DATA | Integer | none | | Data Wordlength |
| BW_COEF | Integer | none | | sine and cosine coefficients Wordlength |
| ROM_EXP | Integer | none | | ROM_EXP = $\log_2(N/8) + 1$ with N/8 = ROM Memory depth |
| LOG_FFT_LE | Integer | none | | $\text{ceil}(\log_2(\text{FFT_LENGTH}))$ |

The parameter FFT_LENGTH is the number of steps to process the whole FFT. It is calculated with $\text{FFT_LENGTH} = \log_2(N)$. The parameter N is the size of the Fast Fourier Transformation. It must be a power of 2 value, because the design uses the radix-2 algorithm.

The parameter BW_DATA is the wordlength of the real and imaginary input data. The total wordlength is $2 * \text{BW_DATA}$.

The parameter BW_COEF is the wordlength of the sine and cosine coefficients. The total wordlength is $2 * \text{BW_COEF}$. The BW_COEF LSB represent the cosine value, while the BW_COEF MSB represent the sine value.

The parameter ROM_EXP is calculated with $\text{ROM_EXP} = \log_2(N/8) + 1$. The parameter „N/8“ is the ROM size. The ROM contains cosine and sine values from 0 to $\pi/4$ at $\pi/(N/8)$ intervals, multiplied by $2^{\text{BW_COEF} - 1}$ to obtain 2's complement integer of BW_COEF bits.

The parameter LOF_FFT_LE is the result of $\text{ceil}(\log_2(\text{FFT_LENGTH}))$. It is used to calculate the number of bits of the output POWER_OVER.

Interfaces

The control interface of the FHG_FFT consists of the system clock CLK, the low active asynchronous reset input RESET_N, the high active DATA_VALID, DATA_REQ, DATA_IN_EN and DATA_OUT_EN signals.

The data interface consists of the data input bus DATA_IN_RE (real data) and DATA_IN_IM (imaginary part) and the Fast Fourier Transformed data output bus DATA_OUT_RE (real) and DATA_OUT_IM (imaginary).

VHDL Usage through Component Instantiation

```
library IEEE, WORK;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use WORK.FFT_PKG.all;

entity MY_FFT is
  generic( BW_DATA : integer := 16;
           LOG_FFT_LE: integer := 4;
           FFT_LENGTH: integer := 10);
  port(CLK      : in  std_logic;
       RESET_N  : in  std_logic;
       DATA_VALID : in  std_logic;
       DATA_REQ  : in  std_logic;
       DATA_IN_RE : in  std_logic_vector(BW_DATA-1 downto 0);
       DATA_IN_IM : in  std_logic_vector(BW_DATA-1 downto 0);
       DATA_IN_EN : out std_logic;
       DATA_OUT_EN: out std_logic;
       DATA_OUT_RE: out std_logic_vector(BW_DATA+1 downto 0);
       DATA_OUT_IM: out std_logic_vector(BW_DATA+1 downto 0);
       POWER_OVER  : out std_logic_vector(FFT_LOG_LE-1 downto 0));
end MY_FFT;

architecture MY_FFT_RTL of MY_FFT is
  constant BW_COEF : integer := 16;
  constant ROM_EXP : integer := 8;
  -----
  component FHG_FFT
    generic(FFT_LENGTH: integer;
           LOG_FFT_LE  : integer;
           BW_DATA     : integer;
           BW_COEF     : integer;
           ROM_EXP     : integer);
    port(CLK      : in  std_logic;
         RESET_N  : in  std_logic;
         DATA_VALID : in  std_logic;
         DATA_REQ  : in  std_logic;
         DATA_IN_RE : in  std_logic_vector(BW_DATA-1 downto 0);
         DATA_IN_IM : in  std_logic_vector(BW_DATA-1 downto 0);
         DOUT_RAM    : in  std_logic_vector(2*BW_DATA+3 downto 0);
         DOUT_ROM    : in  std_logic_vector(2*BW_COEF-1 downto 0);
         DATA_IN_EN : out std_logic;
         DATA_OUT_EN: out std_logic;
         DATA_OUT_RE: out std_logic_vector(BW_DATA+1 downto 0);
         DATA_OUT_IM: out std_logic_vector(BW_DATA+1 downto 0);
         RD_WR_RAM  : out std_logic;
         ADD_RAM    : out std_logic_vector(L-1 downto 0);
         DIN_RAM    : out std_logic_vector(2*BW_DATA+3 downto 0);
```

```
        ADD_ROM      : out std_logic_vector(ROM_EXP-1 downto 0);
        POWER_OVER   : out std_logic_vector(FFT_LOG_LE-1 downto 0));
end component;
```

```
-----
component RAM
  generic(L      : integer;
         BW_DATA: integer);
  port(WE : in  std_logic;
       ADDR : in  std_logic_vector(L-1 downto 0);
       CLK : in  std_logic;
       DI  : in  std_logic_vector(2*BW_DATA+3 downto 0);
       DO  : out std_logic_vector(2*BW_DATA+3 downto 0));
end component;
```

```
-----
component ROM
  generic(BW_COEF: integer;
         ROM_EXP : integer);
  port(ADDR : in  std_logic_vector(ROM_EXP-1 downto 0);
       DO   : out std_logic_vector(2*BW_COEF-1 downto 0));
end component;
```

```
-----
signal RW_RAM      : std_logic;
signal ADD_RAM     : std_logic_vector(L-1 downto 0);
signal DIN_RAM     : std_logic_vector(2*BW_DATA+3 downto 0);
signal DOUT_RAM    : std_logic_vector(2*BW_DATA+3 downto 0);
signal ADD_ROM     : std_logic_vector(ROM_EXP-1 downto 0);
signal DOUT_ROM    : std_logic_vector(2*BW_COEF-1 downto 0));
```

```
begin
MY_FFT_CORE : FHG_FFT
  generic map(L      => L,
             BW_DATA=> BW_DATA,
             BW_COEF=> BW_COEF,
             ROM_EXP=> ROM_EXP)
  port map(CLK      => CLK,
          RESET_N   => RESET_N,
          DATA_VALID => DATA_VALID,
          DATA_REQ  => DATA_REQ,
          DATA_IN_EN => DATA_IN_EN,
          DATA_OUT_EN=> DATA_OUT_EN,
          DATA_IN_RE => DATA_IN_RE,
          DATA_IN_IM => DATA_IN_IM,
          DATA_OUT_RE=> DATA_OUT_RE,
          DATA_OUT_IM=> DATA_OUT_IM,
          RW_RAM     => RW_RAM,
          ADD_RAM    => ADD_RAM,
          DIN_RAM    => DIN_RAM,
          DOUT_RAM   => DOUT_RAM,
          DOUT_ROM   => DOUT_ROM,
```

```
        ADD_ROM    => ADD_ROM,  
        POWER_OVER => POWER_OVER);  
-----
```

```
MY_RAM_CORE : RAM  
  generic map(L    => L,  
             BW_DATA=> BW_DATA)  
  port map(CLK=> CLK,  
          WE   => RW_RAM,  
          ADDR => ADD_RAM,  
          DI   => DIN_RAM,  
          DI   => DOUT_RAM);  
-----
```

```
MY_ROM_CORE : ROM  
  generic map(BW_COEF=> BW_COEF,  
             ROM_EXP  => ROM_EXP)  
  port map(DO=> DOUT_ROM,  
          ADDR => ADD_ROM);
```

```
end MY_FFT_RTL;
```

Architecture

This part of the document shows a general overview of the Fast Fourier Transformation algorithm.

Given a discrete time series $x[0], x[1], \dots, x[N-1]$, its Fourier Transformation is defined as :

$$X[n]_{n=0, N-1} = \sum_{k=0}^{N-1} x[k] e^{-j2\pi kn/N} \quad (1)$$

Where :

Where

- N is the number of values of the DFT (size of the frame)
- $x[n]_{n=0 \text{ to } N-1}$ is the input complex values table
- $X[n]_{n=0 \text{ to } N-1}$ is the result complex values table
- $e^{-jx} = \cos(x) - j\sin(x)$

The points are complex integers.
 N must be a power of 2 value.

The following figure illustrates the radix-2 FFT algorithm that has been implemented. The $N=2^L$ inputs points $x(n)$ are stored in the work table $u(k)$ (RAM memory), following the reversed bit order. Then the processing runs in L steps.

In each step, $N/2$ complex multiplications are required to combined the results of the previous stage. This correspond to $N/2$ „butterfly“ calculations. A „butterfly“ is defined by the operations described in the figure 3.

The total number of clock cycles needed for the whole process is : $2*(L+1)*N$ clock cycles, as explained in the following lines.

- N clock cycles for the loading phase (one clock cycle per data)
- $2*N*L$ for the computing process (4 clock cycles per butterfly, $N/2$ butterflies per step, L step)
- N clock cycles for the reading phase (one clock cycle per data)

Therefore the minimal clock frequency for the full FFT processing is : $2*(L+1)*N*T_{symbol}$ with $T_{symbol} = \text{input clock frequency} / \text{number of samples}$.

Figure 2: Fourier transformation process of order N

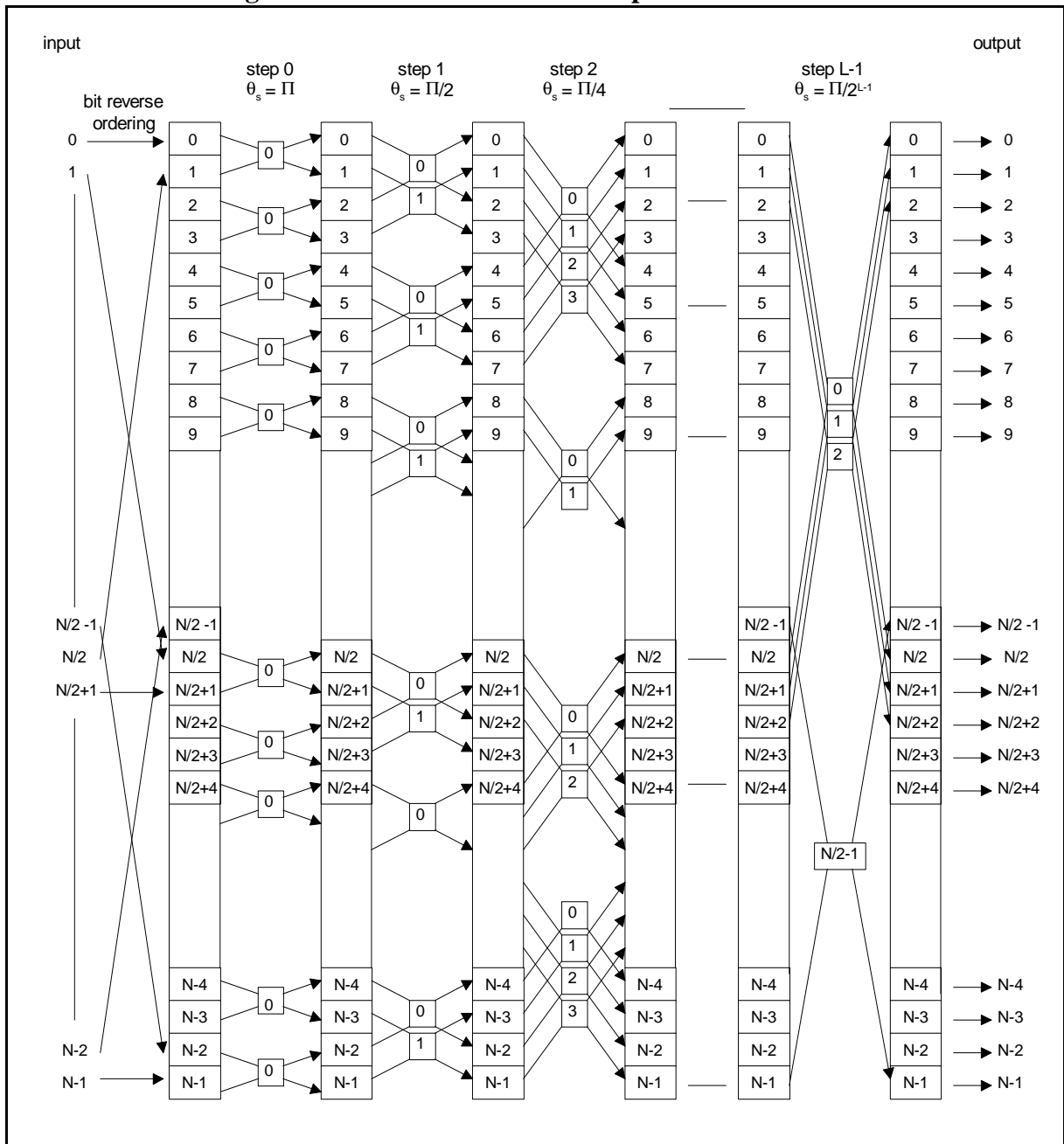
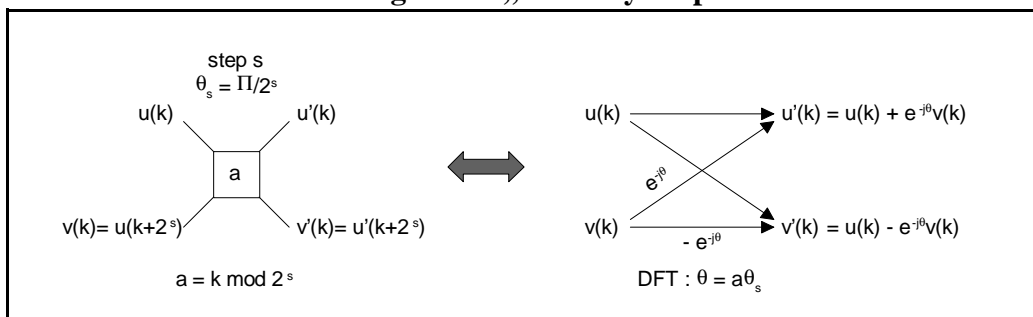
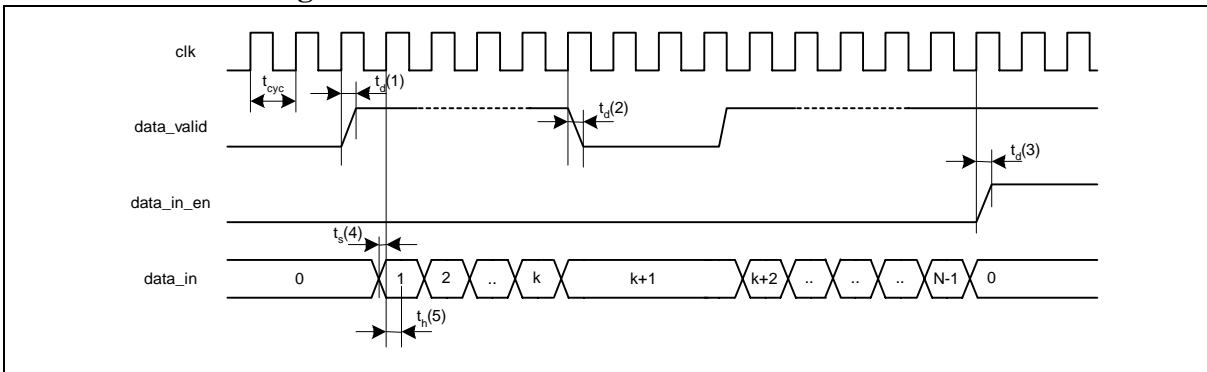


Figure 3: „butterfly“ representation



Waveforms and timing tables

Figure 4: Waveform of external data write access



As long as the RAM memory is not full, the process does not start. When the RAM memory is full, then any further writing is not allowed, until, DATA_EN is de-validated.

Table 4: Timing table of external data write access

| Symbol | Name | Item | Notes | Min | Max | Unit |
|------------|---|---|-------|-----|-----|------|
| t_{cyc} | $t(\text{CLK}_{\text{rise}}-\text{CLK}_{\text{rise}})$ | CLK cycle | | | | ns |
| $t_{d(1)}$ | $t_d(\text{CLK}_{\text{rise}}-\text{DATA_VALID}_{\text{rise}})$ | Delay from rising CLK to rising DATA_VALID | | | | ns |
| $t_{d(2)}$ | $t_d(\text{CLK}_{\text{rise}}-\text{DATA_VALID}_{\text{fall}})$ | Delay from rising CLK to falling DATA_VALID | | | | ns |
| $t_{d(3)}$ | $t_d(\text{CLK}_{\text{rise}}-\text{DATA_IN_EN}_{\text{rise}})$ | Delay from rising CLK to rising DATA_IN_EN | | | | ns |
| $t_{s(4)}$ | $t_s(\text{DATA_IN}-\text{CLK}_{\text{rise}})$ | Setup time for DATA_IN to rising clock edge | | | | ns |
| $t_{h(5)}$ | $t_h(\text{CLK}_{\text{rise}}-\text{DATA_IN})$ | Hold time for DATA_IN to rising clock edge | | | | ns |

Figure 5: Waveform of external data read access

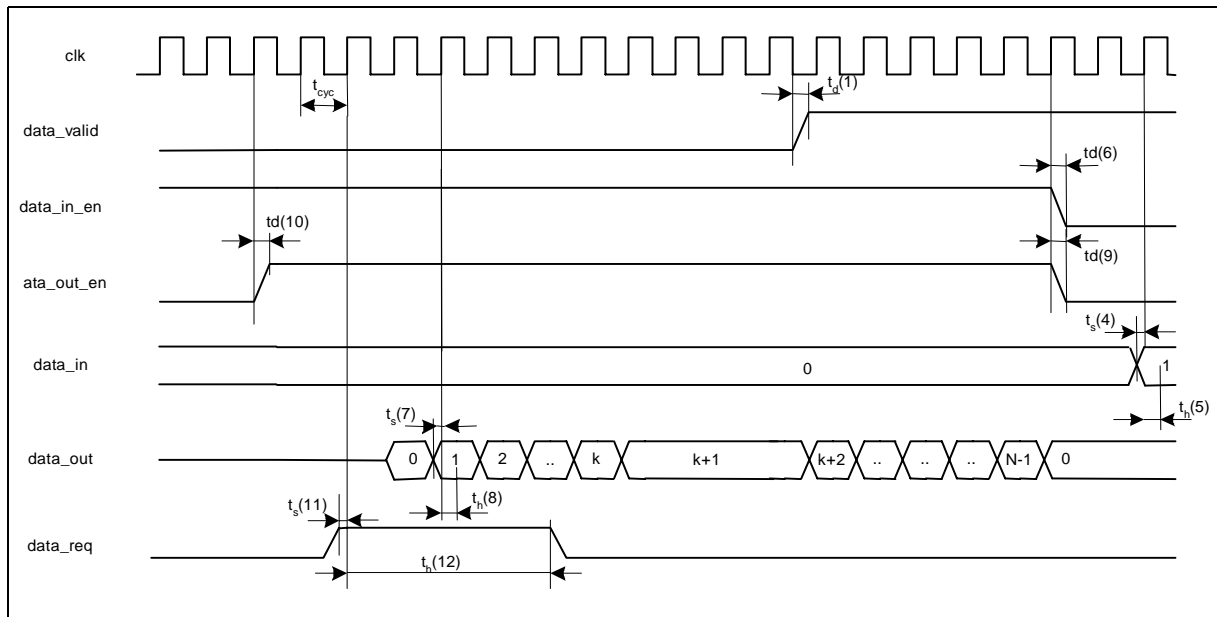


Table 5: Timing table of external data read access

| Symbol | Name | Item | Notes | Min | Max | Unit |
|-----------|--|--|-------|-----|-----|------|
| t_{cyc} | $t(\text{CLK}_{\text{rise}}-\text{CLK}_{\text{rise}})$ | CLK cycle | | | | ns |
| $t_d(6)$ | $t_d(\text{CLK}_{\text{rise}}-\text{DATA_IN_EN}_{\text{fall}})$ | Delay from rising CLK to falling DATA_IN_EN | | | | ns |
| $t_s(7)$ | $t_s(\text{DATA_OUT}-\text{CLK}_{\text{rise}})$ | Setup time for DATA_OUT to rising clock edge | | | | ns |
| $t_h(8)$ | $t_h(\text{CLK}_{\text{rise}}-\text{DATA_OUT})$ | Hold time for DATA_OUT to rising clock edge | | | | ns |
| $t_d(9)$ | $t_d(\text{CLK}_{\text{rise}}-\text{DATA_OUT_EN}_{\text{fall}})$ | Delay from rising CLK to falling DATA_OUT_EN | | | | ns |
| $t_d(10)$ | $td\text{CLK}_{\text{rise}}-\text{DATA_OUT_EN}_{\text{rise}}$ | Delay from rising CLK to rising DATA_OUT_EN | | | | ns |
| $t_s(11)$ | $t_s(\text{DATA_REQ}-\text{CLK}_{\text{rise}})$ | Setup time for DATA_REQ to rising clock edge | | | | ns |
| $t_h(12)$ | $t_h(\text{CLK}_{\text{rise}}-\text{DATA_REQ})$ | Hold time for DATA_REQ to rising clock edge | | | | ns |

Application notes

The following example shows the parameters for the FHG_FFT for a special application. This parameter set is also used in section „VHDL Usage through Component Instantiation“.

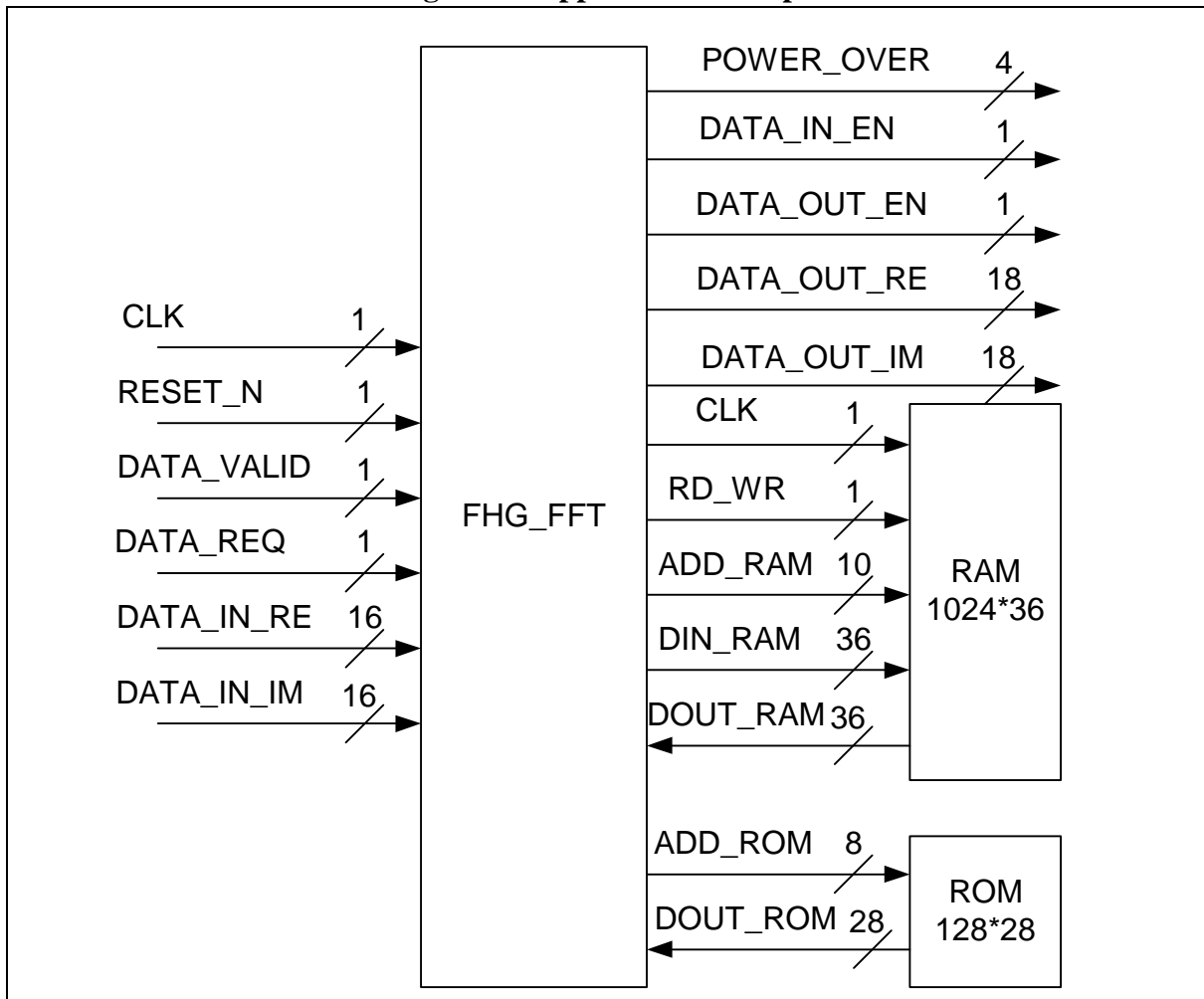
EX :

- 1024 value FFT
- 16 bits for real and imaginary signed input data
- 16 bits for coefficients bit width (the two MSB must not be inserted in the parameter)

PARAMETERS :

- L = 10 (=log₂(1024))
- BW_DATA = 16
- BW_COEF = 14 (= 16-2)
- ROM_EXP = 8 (=log₂(1024/8) +1)

Figure 6: Application Example



ex :

input clock = 300 000 samples per second,

one symbol = 1024 samples

Tsymbol = 300000/1024 = 292,97

$$\text{Minimal clock frequency} = (2 * (10+1) * 1024) * 292,97 = 6,6 \text{ MHz}$$