
Part I

DIGITAL AUTOCORRELATION SPECTROMETERS FOR SPACE BORNE APPLICATIONS

Part II

DESIGN OF A PARALLEL FFT PROCESSOR USING FIXED POINT ARITHMETIC AND CSD-MULTIPLICATION

Roland Weigand

XRM Section

ESA/ESTEC

31/8/1995

ACKNOWLEDGEMENTS

The author would like to thank the following persons, who supported this work:

Mr. R. COIRAULT
Head of XR division

Mr. G. GATTI,
Head of XRM section

Mr. M. HOLLREISER, XRM section,
who supervised this work, giving useful impulses and assisting in case of problems.

Mr. L. FANUCCI, XRM section,
for his support with the Synopsys software tools.

Mr. J. SONNEVELD, BCS section,
for his support with the XR Sun Cluster.

This report is the result of the work performed by Roland Weigand as a Young Graduate Trainee within the XRM section from 1/9/94 to 31/8/95. The work was supervised by Mr. Martin Hollreiser.

It has been a very fruitful experience for an electronic engineer, to work at ESA. Besides the basic techniques of ASIC design, special knowledge, related to the space environment, such as low power design and power consumption estimation was acquired during this period.

Since two subjects have been treated independently, this report is divided into two parts. The first part concerns Digital Autocorrelation Spectrometers and the second part is about a Parallel FFT Processor.

NOTE

Some of the figures of this report are not available in electronic format, leaving blank spaces in the document.

Part I

DIGITAL AUTOCORRELATION SPECTROMETERS FOR SPACE BORNE APPLICATIONS

CONTENTS

1 INTRODUCTION 3

1.1 BACKGROUND AND MOTIVATION FOR DIGITAL AUTOCORRELATION SPECTROMETERS 3
1.2 CURRENT STATE-OF-THE-ART 3
1.3 DESCRIPTION OF THE WORK 3

2 ARCHITECTURES OF ACS 4

2.1 THE HYBRID AND MULTIPLEXED APPROACH 4
2.2 THE CORRELATOR 4
2.3 THE MULTIPLICATION METHODS 5

3 PERFORMANCE OF ACS 9

3.1 DEFINITIONS OF 'SENSITIVITY', 'DEGRADATION FACTOR' AND 'INTEGRATION TIME' 9
3.2 DEGRADATION AS A FUNCTION OF THE QUANTISER THRESHOLDS (VTH) 9

4 THE SPREADSHEET FOR SYSTEM TRADE-OFF 11

4.1 THE INPUT SECTION 11
4.2 MODELLING OF THE CORRELATOR CHIP 12
4.3 THE ANALOG CIRCUITS 13
4.4 THE OUTPUT SECTION 13
4.5 MACRO FUNCTIONS FOR OPTIMISING THE DATA 14

5 SOPRANO AND MASTER SPECTROMETER TRADE-OFF 16

5.1 SPECIFICATIONS: 16
5.2 TRADE-OFF GUIDELINES 16
5.3 THE SOPRANO SPECTROMETER 17
5.4 THE MASTER SPECTROMETER 21

6 THE VHDL MODEL 24

6.1 OUTLINE OF THE MODEL 24
6.2 SIMULATIONS IN VHDL AND MATLAB 26
6.3 SYNTHESIS: SCHEMATICS, SPEED AND COMPLEXITY 28

7 CONCLUSION 29

7.1 ACHIEVEMENTS **29**
7.2 RESULTS **29**
7.3 PROSPECTS AND SUGGESTIONS **30**

8 REFERENCES **31**

9 APPENDIX A : SCHEMATICS OF THE SYNTHESISED AUTOCORRELATOR **32**

10 APPENDIX B - LISTINGS OF THE VHDL MODEL **38**

1 INTRODUCTION

1.1 Background and motivation for Digital Autocorrelation Spectrometers

Microwave Spectrometers have been used in radio astronomy for many years. In the beginning on earth-based telescopes, they are now implemented on board of spacecraft.

During the last years, aeronomy has become an important discipline within the space-based earth observation activities. We encounter therefore an increasing need of space-borne microwave spectrometers, such as they are planned in different projects of the European Space Agency (ESA).

Compared to other technologies (acousto-optical spectrometers), digital autocorrelation spectrometers (ACS) promise a higher mechanical and thermal stability, and a lower mass. Their reconfigurable design (bandwidth and resolution) allows the use of one instrument for different applications, which would be a considerable advantage on a satellite. Due to their high power consumption, they have been rejected for the use in space until today.

With the recent advances in digital technology, the power consumption of ACS can be considerably reduced. This increased performance incites to investigate the feasibility of space borne digital ACS, which was the subject of this work.

1.2 Current state-of-the-art

The use of coarse quantisation in digital correlators has been studied theoretically by [1, 2, 3]. One- and two-bit representations of the signal provide simple signal processing and therefore high operation speeds. Autocorrelation spectrometers using coarse quantisation have been designed and built for earth-based radio observatories [4, 5, 6]. Coarse quantisation is also the key technique to reduce the power consumption of instruments in space.

Studies about space borne spectrometers have been executed between 1992 and 1994 by different European companies (Matra Marconi Space, GB; Edge/Omnisys, Sweden; DASA, Germany) under ESA contract. Estimates for performance, complexity and power consumption disagree by orders of magnitude and the analog part of hybrid spectrometers is not included.

1.3 Description of the work

This project aims to work out detailed information about the power consumption, complexity and performance of ACS and to show their feasibility, particularly of the digital ASIC. The results could be an input to further contracts about studies and industrial realisations of earth observation payloads.

The following steps have been carried out:

1. Literature browse and selection of the design guidelines (architecture, coding schemes).
2. Development of a spread-sheet to evaluate power consumption and complexity of ACS. A trade-off has been performed between different design variants, based on SOPRANO and MASTER specifications.
3. Evaluation of the performance/sensitivity of different variants by matlab simulations.
4. VHDL-Design of an ASIC, performing the autocorrelation function. This model can be set to different design variants and specifications by the means of parameters. Simulation and synthesis based on a 0.6 µm CMOS technology (Matra MHS) have been carried out. The results of the functional simulations have been reported to the spread-sheet.

2 ARCHITECTURES OF ACS

2.1 The hybrid and multiplexed approach

In many cases, it is impossible to sample and calculate the autocorrelation function over the total bandwidth, because the involved digital circuits can not acheive the required speed. Even when this is theoretically possible, the power consumption would be, in many cases, very high. Hybrid and multiplexed architectures are used to fix this problem.

A hybrid spectrometer consists of a filterbank, dividing the total frequency band into J subbands, each of which is then separately sampled and auto-correlated. The subbands can be, in post-processing, recomposed to form the total band. Each of the autocorrelators has 1/J of the total lags, running at 1/J times the original sampling frequency (without subbanding). The power consumption for the digital part, being (in CMOS) proportional to the lag number times their clock frequency is reduced by a factor J. On the other hand, additional power is required for the analog circuitry. For the whole system (digital and analog part), we can determine a minimum as afunction of J. A hybrid architecture is described in [4].

In some cases, the number of subbands J may be limited, due to mass and stability constraints. The sampling frequency can then be too high for the digital technology. A multiplexing correlator can be used in this case. This means, the sampled data stream of Fs-frequency can be time-multiplexed to M streams of Fs/M frequency. Using a matrix of correlators, as described in [6], requires a M times more complex digital circuit. Thus, the power balance is not reduced by this technique and its use will be restricted to the cases mentioned here.

2.2 The correlator

The autocorrelator is a DSP - Chip, designed to compute the discrete autocorrelation function:

$$\mathbf{C(n)} = \sum_{\mathbf{i}} \mathbf{x(i)x(i-nT)}, \qquad \text{where T is the clock period.}$$

This means, we need a delay function, a multiplier, and an adder/integrator.

One lag calculates one coefficient c(n). The chip consists of a large number of identical lags, shown in Figure 2.1. Each lag implements a delay of one clock period. When all lags are cascaded, the n-th lag has nT - delayed data at its input (numbering starting at 0). The integrator is divided into 2 parts: 1 accumulator (adder + registers) running at high speed, 1 asynchronous counter running at the (lower) carry-out frequency of the accumulator. The bits of the accumulator are not skewed, because we are interested only in its highest carry bit, integrated over many samples. For speed reasons, the chip is pipelined after the multiplier and after the accumulator.

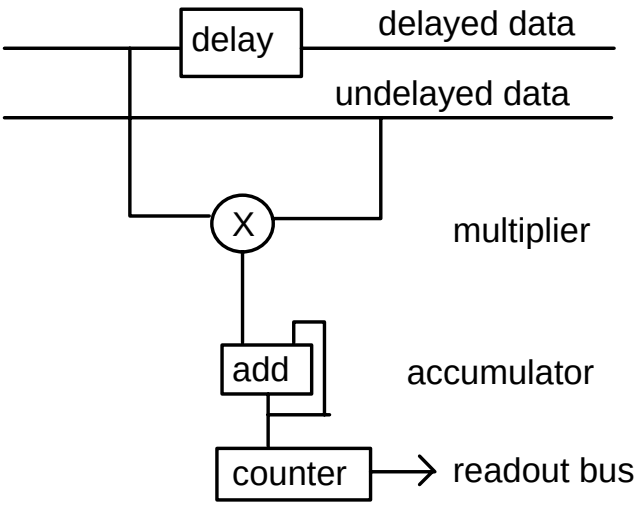

Figure 2.1 Block diagram of one autocorrelator lag

2.3 The multiplication methods

The autocorrelator uses coarse quantisation (1 or 2 bit). Before any estimation or implementation, the coding schemes of multiplicands and product must be fixed in order to obtain simple logic equations. Coarse quantisation and coding tricks require in many cases the use of correcting transforms to calculate the true data from the spectrometer readout. Corrections for the influence of coarse quantisations are given in [1] and [7]. In our application (earth observation), we measure a noise, whose autocorrelation function is some orders of magnitude below 1, except for the zero lag. The correction can therefore be reduced to a linear function. Corrections for the coding effects are given in this text. The following multiplication methods have been considered in this project (a short name is given for each of them in parentheses, this name appears throughout this document and the data files):

2.3.1 The 1x1-bit multiplication (bit_1)

Data is 1-bit quantized, this means that only the sign information of the signal is considered. A negative value is quantized to -1 (logic level 0), a positive value to +1 (logic level 1). The two different product values (-1 and +1) are also coded into 0 resp. 1. Two multiplication tables can be set up, where P is the true product and P' is the coded product:

P	-1	+1
-1	+1	-1
+1	-1	+1

Table 2.1

P'	0	1
0	1	0
1	0	1

Table 2.2

The transform P→P' requires a correction of the readout. If the product P' is accumulated over N samples, the true readout is: $\sum \mathbf{P} = 2\sum \mathbf{P'} - \mathbf{N}$

P' is realised by a XNOR function: $\mathbf{P' = XNOR(X, Y) = \overline{X \oplus Y}}$

2.3.2 The 3-level multiplication (level_3)

Data is 3-level quantized and coded into 2-complement numbers. The product P (-1, 0, +1) is offset to positive numbers P' (0, 1, 2). The two multiplication tables are Table for the true product and Table for the coded product:

P	-1	0	+1
-1	+1	0	-1
0	0	0	0
+1	-1	0	+1

Table 2.3

P'	11	00	01
11	10	01	00
00	01	01	01
01	00	01	10

Table 2.4

The transform P→P' requires a correction of the readout. If the product P' is accumulated over N samples, the true readout is: $\sum \mathbf{P} = \sum \mathbf{P'} - \mathbf{N}$

The multiplication is performed by the following logic equations:

$$\mathbf{P_0' = \overline{X_0 Y_0}} \quad \textbf{and} \quad \mathbf{P_1' = \overline{X_1 \oplus Y_1 + P_0'}}$$

2.3.3 The reduced two-bit four-level multiplication (level_4_rn and level_4_r0)

Multiplication tables

A common way to simplify a 4 x 4 - level (= 16 values) multiplication is to assign 0 to the low-level products, as described in [1]. The efficiency of this correlator is 0.87 instead of 0.88 for a full 2-bit multiplication. The multiplication outputs a 5 state system which requires 3 bits (see Table).

Another method consists in assigning n to the low-level products, as shown in Table . This multiplication scheme has at its output only 4 different values. In some cases, these 4 states may be coded by two bit numbers. The multiplier and the accumulator are considerably simplified, the efficiency is 0.84, between the reduced (to 0) method and the three level correlation.

Full 2 bit multiplication
level_4

x	-n	-1	1	n
-n	n^2	n	-n	$-n^2$
-1	n	**1**	**-1**	-n
1	-n	**-1**	**1**	n
n	$-n^2$	-n	n	n^2

Table 2.5

Low-level set to 0
level_4_r0

x	-n	-1	1	n
-n	n^2	n	-n	$-n^2$
-1	n	**0**	**0**	-n
1	-n	**0**	**0**	n
n	$-n^2$	-n	n	n^2

Table 2.6

Low-level set to n
level_4_rn

x	-n	-1	1	n
-n	n^2	n	-n	$-n^2$
-1	n	**n**	**-n**	-n
1	-n	**-n**	**n**	n
n	$-n^2$	-n	n	n^2

Table 2.7

Coding and Implementation of level_4_rn (see Table)

The quantized values are coded in a sign-magnitude scheme. The MSB represents the sign (0 if ≥ 0, 1 else), the LSB the absolute value (1 if $|x| \geq v_0$, 0 else).

q(x)	-3	-1	1	3
X1;X0	11	10	00	01

Table 2.8

If we assign n = 3, the possible products are P = -9, -3, 3, 9. These values are equidistant. The simple linear transform P' = (P+9) / 6 leads to P' = 0, 1, 2, 3 and the multiplication Table .

$P'_1P'_0$	11	10	00	01
11	11	10	01	00
10	10	10	01	01
00	01	01	10	10
01	00	01	10	11

Table 2.9

The logic equations for P' are $P'_1 = \overline{X_1 \oplus Y_1}$ and $P'_0 = P'_1 \oplus \overline{X_0 Y_0}$.

The multiplier contains one XOR, one XNOR and one NAND gate, the accumulator must add 2 bits.

The case n = 5 leads to a more complicated multiplier and needs a 3 bit accumulator. Since efficiency and complexity are very similar to level_4_r0, this case is not treated here.

Coding and Implementation of level_4_r0

This correlator has been implemented by Van Herzen. Considering that the different bits of the result must not be obtained at the same clock cycle (since the result is integrated anyway), the following multiplier has been implemented here.

The quantized samples are coded in the sign-magnitude scheme and weighted with n = 3. The product (Table) is transformed to positive values as P' = (P+9) / 3. The resulting values (0, 2, 3, 4, 6) are represented (in binary) in the multiplication Table :

$P'_1P'_0$	11	10	00	01
11	110	100	010	000
10	100	011	011	010
00	010	011	011	100
01	000	010	100	110

Table 2.10

The logic equations to satisfy this table are:

$$\mathbf{P'_0 = \overline{X_0 + Y_0}} \quad ; \quad \mathbf{P'_1 = \overline{[\overline{X_0 Y_0} \oplus (X_1 \oplus Y_1)]\overline{P'_0}}} \quad \textbf{and} \quad \mathbf{P'_2 = \overline{(X_1 \oplus Y_1) + P'_0}}$$

2.3.4 The full two-bit multiplication (level_4)

The coding of the samples is again the sign-magnitude system, the values are weighted by n = 3. The transform P' = (P+9) / 2 is applied to the product, which gives the output values (0, 3, 4, 5, 6, 9). The multiplication table translates therefore into its binary form Table :

$P'_3P'_2P'_1P'_0$	11	10	00	01
11	1001	0110	0011	0000
10	0110	0101	0100	0011
00	0011	0100	0101	0110
01	0000	0011	0110	1001

Table 2.11

The logic equations are:

$$\mathbf{P'_0 = \overline{(X_1 \oplus Y_1) \oplus P'_1}}$$
$$\mathbf{P'_1 = X_0 \oplus Y_0}$$
$$\mathbf{P'_2 = \overline{(X_0 + Y_0)[X_0 Y_0 + (X_1 \oplus Y_1)]}}$$
$$\mathbf{P'_3 = \overline{\overline{X_0 Y_0} + (X_1 \oplus Y_1)}}$$

2.3.5 Summary of multiplication methods

Table compares the different multiplication methods. The efficiency (= 1/sqrt(rel. integration time) is taken or derived from [1] and [7]. The number is valid only for an optimal adjustment of the quantiser thresholds.

Method	Efficiency	Threshold	Complexity
bit_1	0.64	--	1 XNOR
level_3	0.81	0.6	1 XOR, 2 NAND/NOR, 2 bit accu
level_4_rn, n = 3	0.838	0.782	2 X(N)OR, 1 NAND, 2 bit accu
level_4_rn, n = 5	0.859		(1 XOR, 2 NAND/NOR, 1 Inverter, 3 bit accu)
level_4_r0	0.872	0.906	2 XOR, 3 NOR/NAND, 1 Inverter, 3 bit accu
level_4	0.88	0.95	3 X(N)OR, 1NOR, 2 AND/OR, 1 OR-NAND, 4 bit accu

Table 2.12 Summary of multiplication methods

3 PERFORMANCE OF ACS

3.1 Definitions of 'sensitivity', 'degradation factor' and 'integration time'

Different measures are used in literature to specify the performance of autocorrelators. The aim of this section is to clear the confusions created by these definitions.

The spectrometer measures spectral lines hidden in noise, thus, we have a very low signal to noise ratio (S/N). The ACS allows to concentrate the (supposedly uncorrelated) noise power in the zero-lag of the autocorrelation function. After elimination of this 'DC' component, the S/N of the Fourier-transformed spectrum will be the higher, the more samples are integrated.

The coarse quantisation introduces an additional noise to the signal, it therefore degrades the signal to noise ratio of the spectrum. This fact can be compensated by integrating more samples.

Since the output S/N always depends on the properties of the input signal, the performance of coarsely quantised ACS must be specified normalised to an ideal (analog, multibit) ACS. We use here the definition of the degradation factor D, given in [2]:

$$\mathbf{D} = \frac{\text{output S/N of analog ACS}}{\text{output S/N of digital ACS}} = \frac{\text{output } \boldsymbol{\sigma} \text{ of digital ACS}}{\text{output } \boldsymbol{\sigma} \text{ of analog ACS}}$$

Another measure is the efficiency or sensitivity S, as defined in [1]: **S = 1 / D.**

To compensate the degradation D, the integration time must be increased by a factor I, which is given in [7] by the ratio of the variances (σ^2), therefore: **$I = D^2$.**

3.2 Degradation as a function of the quantiser thresholds (Vth)

The degradation D (or sensitivity) of the 2 (and more) bit correlators depends on the right adjustment of the decision levels V_{th} of the quantisers. For 2 bit, one $|V_{th}|$ is relevant. This voltage is adjusted relative to the RMS value of the signal (σ). The parameter is V_{th}/σ.

From statistical considerations [1, 2, 3, 5] the function D vs. V_{th} can be derived. The result is a curve presenting a minimum for a certain optimal value of V_{th}. Often, only this minimum value is given, as for example in Table , but one must always know that this is only achieved with a suitable adjustment of V_{th}.

Optimum efficiencies	bit_1	level_3	level_4_rn	level_4_r0	level_4
Efficiency (theory)	0.64	0.81	0.838	0.872	0.88
at Vth / σ (theory)	--	0.6	0.782	0.906	0.95
Efficiency (simulated)	0.62	0.81	0.83	0.86	0.87
at Vth / σ (simulated)	--	0.6	0.65	1	1.025

Table 3.1 Optimal thresholds and efficiencies, theory and matlab simulations

Figure 3.1 shows these curves for correlators, simulated in matlab. The simulations show similar minima (see Table), or a constant efficiency for the 1 bit method, which is independent of V_{th}. Nevertheless, the simulations are subject to some unexpected distortions, which can not be eliminated by increasing the integration time. After a certain number of samples, the simulation begins even to diverge. This effect can be explained by the combination of two facts:

- The theoretical approach, as it is shown for example in [2] (see Figure 3.2), assumes, that the signal level is very low compared to the noise level (S/N --> 0 at the input of the spectrometer). Only in this case, signal and noise can be treated indpendently, as assumed in the literature.
- To confirm this theory, simulations should be carried out with a low input S/N, which implies that the process becomes numerically unstable, is subject to distortions and, finally, may diverge towards a totally different output.

This type of simulation requires high CPU times and is numerically not stable. Nevertheless (see Table) we can approximately reproduce the theoretical minima.

Figure 3.1 Degradation D vs. V_{th}. Matlab simulation + polynomial cuve-fit 4th order

Figure 3.2 Degradation D vs. V_{th}. Theoretical curve, level_4_r0 and level_4. Source: [2], p. 380

4 THE SPREADSHEET FOR SYSTEM TRADE-OFF

To evaluate and trade-off the system design, a spread-sheet in MS-Excel was developed, taking into account the global power consumption of digital **and** analog circuits. Different designs are considered: Realisation in GaAs or CMOS, hybrid and multiplexed architectures. Starting from a specification, a trade-off can be worked out. Two macros help to optimise the partitioning parameters to minimise the power consumption. The spread-sheet is divided into input- model- and output-sections, each of which can be separately displayed or hidden by the means of the outlining features (+ and -).

4.1 The Input Section

The INPUT section contains fields for the specifications of the ACS, the choice of architecture (partitioning) and the properties of the technologies.

4.1.1 ACS specifications

The main specifications are the total bandwidth B_T and the number of spectrometer channels N, which specifies the frequency resolution $\Delta f = B_T/N$. The architecture is determined by the number of hybrid channels J and the multiplex factor M, according to the explanations in chapter 2. The fully digital and not multiplexed spectrometer is then represented by J=M=1. A filter shape S and an oversampling factor O are introduced to take into account the eventual use of band overlapping or oversampling techniques. The latter method, Fs > Nyquist Rate, can improve the sensitivity of the instrument [7]. Nyquist sampling means O=1. The filter shape factor is > 1 for non ideal filters and = 1 for ideal filters.

4.1.2 Architecture partitioning

In this second table, some important values for the partitioning of the architecture are computed from the specifications. Note: no values are to be inserted by the user into this table.

Frequency resolution $\Delta f = B_T/N$
Bandwidth per correlator (1 hybrid channel) BpCorr = $S*(B_T/J)$
Sampling frequency $f_S = 2*O*$ BpCorr
Number of lags per channel LpCh = $0.5\ f_S/\Delta f$
Number of correlators NbCorr = $J*M^2$
Number of lags per correlator LpCorr = LpCh/M
Clock frequency of the correlators $f_C = f_S/M$
Multiplexing indicator MuxUsed = True, if M>1, false if M=1
Total Lags = NbCorr*LpCorr
Lags*Clock = LagsTimesClock = TotalLags * f_C (in MHz/Million)

4.1.3 Technology parameters

All power parameters are given in µW per Gate per MHz in CMOS and in mW per Gate in GaAs. An accurate power modelling requires distinction between gate intrinsic power (A), fan out load (B) and wiring load (C), these parameters are multiplied with the activity for each node. Actually, only A is used in the spreadsheet, since the parameters B and C of most technologies are not known.

The gate count of different functions (xor, adders etc.) can be defined in the technology part, since these values may differ depending on the technology.

The Clock Tree Size is a factor which defines the number of clock-buffers to be used in function of the number of gates contained in the clocked Flip Flops.

The use of these parameters is explained later in this chapter.

4.2 Modelling of the Correlator Chip

The chip consists in a large number of identical lags, shown in Figure 2.1 on page 4. Within one correlator, all lags operate at the same clock frequency. The model is evaluating the power of one lag, and the total power dissipation will be given by the value for one lag times the number of lags. The node activities of one lag were determined by VHDL simulations. We use these node activities for all the lags of the spectrometer. This simplification is accurate for all lags, except the zero lag.

4.2.1 Power dissipation of one autocorrelator lag

In first approximation, the power dissipation of CMOS circuits is proportional to a sum over all gates times their respective operating frequency. A precise power model is given by:

$$\mathbf{P} = \mathbf{f_s} \sum_{\mathbf{i,j}} \mathbf{activity_i} * (\mathbf{A} + \mathbf{B} * \mathbf{FanOut} + \mathbf{C} * \mathbf{W_i}) \qquad (1)$$

where f_S = sampling/global clock frequency
activity = duty factor = (operating frequency in block i) / f_S
A = load independent part of the gate losses
B = fan out load factor
C = wiring capacitance load factor
FanOut = fan out of the node
W_i = length of the interconnection wires to the following inputs

A, B, C, are the technology parameters which must be provided in the input table.

The activities (duty factors) are the data bit commutation rates, i. e., the toggling frequency of a node vs. the clock frequency of the whole chip. The activities could be determined from probabilistic considerations of the design and the input signal, but the values used here are extracted from simulations of the VHDL model.

The integration counter, implemented as an asynchronous ripple counter, requires a special consideration: the activity of the LSB (i. e., the counter input) is divided by two at each bit. The series 1 + 1/2 + 1/4 + 1/8 + ... converges to 2. Thus, a n-bit counter can be considered as two flip-flops running with the duty factor of the LSB. Therefore, the counter has a special formula to compute its power consumption : P(per lag and MHz) = A * NbGates * Activity * 2/NbEl. NbGates is the gate count of the whole counter (which is a multiple of NbEl, the number of counter flip-flops). Thus, the factor NbEl, contained in NbGates, is eliminated and replaced by a 2.

Tristate buffers (for readout bus): 3 Gates per counter bit, from which only one counts for the power consumption. For the same reasons as for the counter FF's, the power is independent of the number of counter bits and equivalent to two buffers. Therefore, two buffers, one gate active => P = 2 * Activity.

In the power model presented in [8], the estimation is based on average values of activity, fan out and interconnection length, assumed for the whole chip. The average consumption of one gate is multiplied with the total gate count. Since one autocorrelator lag is a particularly simple circuit, we can consider here a single autocorrelator lag in detail. Thus, the activity of each gate can be determined by simulations. The interconnection wiring W still must be estimated globally, because it is impossible to know the wire length without having a physical layout. Anyway, at the moment, wiring- and fan out loads are included in the main parameter A and are not considered separately. The activities, multiplied by the gate counts, of each node, are summed over one lag.

The global clock driving power must be considered in the case of an autocorrelator chip using a high clock frequency and several clock driven cells per lag, especially, when pipelined logic is used. We can derive the clock power from the number of clock driven flip-flops (or their gate count) by giving the number of gates necessary to perform the clock distribution in a tree (ClockTreeSize = Nb. of clock tree gates / Nb. of flip flop gates). The number of gates in the clock tree is added to the power consumption (the clock tree has always activity = 1). The total number, multiplied by the technology parameters, gives the lag power dissipation in μW per MHz, which appears in the last line of each lag model.

Since the correlator result is integrated in an accumulator and counter system, the readout frequency is several orders of magnitude less than the clock frequency ($2^{[\text{nb. of counter bits}]}$). The power dissipation of the readout bus and the off chip driving is therefore neglected in first approximation.

The lag models also contain a column for GaAs implementation. Each gate type has a specific power consumption assigned to it, which is independent of the operating frequency and appears in the technological parameters. The power model sums this number over all the gates/blocks of one lag, delivering a number in mW / MHz. In the GaAs implementation, the integration counter is divided into a fast GaAs part whose length is specified by the parameter GaAsCounter-Bits and a CMOS ripple counter (length = NbEl), whose operating frequency is considerably lower than of the rest of the circuit.

4.2.2 Correlator summary table

A dedicated table summarises all relevant values per lag for the different quantization methods:

GateNb	= Number of gates
LagPower	= power in μW per lag per MHz clock frequency (for CMOS)
	= power in mW per lag (for GaAs).
CompNb	= number of quantizer comparators

The power per lag is the quantity LagPower = A * [(NbGates * Activity), summed over the rows of the lag model, + the above mentioned exception for the asynchronous counter + the clock tree gates]. The gate number GateNb is the sum over the 'NbGate' column. CompNb is a constant. The AD-convertor will always contain one less comparators than there are quantization levels.

4.3 The Analog Circuits

A table is dedicated to the architecture of the front end, down conversion and digitizer circuit. The power dissipation is split into a fixed value Pf, independent from the partitioning of the spectrometer, a part depending on the number of hybrid channels (J) Pj, the power dissipation of one comparator Pc and the consumption of the multiplexer Pm per multiplexing path[1].

4.3.1 Front end architecture

The model for the front end is based on the structure proposed in [9]. The block diagram is represented in Figure 4.1 on page 15. A first local oscillator (LO) of around 10 GHz converts the frequency in a SSB mixer to two bands between 0.5 and 1 GHz. This second IF is down converted by a set of 2 SSB modules into four subbands which are digitized and sampled.

The structure can be duplicated and allows an entire multiple of 4 subbands to be implemented according to different frequencies of the first LO. One set of two 'second' LO's can be used for all subbands, this saves power and complexity. The second LO power is therefore split into a fixed part and a part depending on J, since at higher J, on LO must drive several mixers.

4.4 The Output Section

The output tables combine the information of the different sets of input specifications and parameters with the models of correlators and analog circuits.

4.4.1 Power Consumption

The formula for the power consumption is ('MUX used' is the Boolean indicator computed in the partitioning table):

$$\mathbf{P = LagPower * f_C + Pf + (Pj + Pc * CompNb + [MuxUsed] * Pm * M) * J}$$

4.4.2 Efficiency and Performance

A table summarises the performance of the quantization method. The relative sensitivity σ and the corresponding integration time T ($T = 1/\sigma^2$) is given for coarse quantisation and Nyquist sampling compared to a multibit quantisation. For the higher quantizations, the third column contains the required values of comparator thresholds (V_0/σ) to obtain these sensitivities.

[1]The power consumption of the A/D-convertor and the multiplexer depends partly on the sampling frequency. This fact is not yet implemented in the spreadsheet, but could be introduced easily.

4.4.3 Energy Merit

The quantity **(Integration time) * (Power consumption)** is a measure for the energy required to compute the power spectrum of the specified frequency band within a specified signal to noise ratio. This can be a convenient number to perform the trade-off between different architectures, if there is no strict requirement for the efficiency, which imposes a certain quantisation method.

4.5 Macro functions for optimising the data

The file ACSMACRO.XLM contains three macros designed to work with the data in the spreadsheet. Each of them is called from the active ACS.XLS with a CTRL-hotkey. The data to treat is determined by the selections in the worksheet.

4.5.1 Mux_Min: hotkey CTRL - m

Prompts for a maximum clock frequency for the correlators (default 300 MHz) and computes the multiplexing factor M to achieve this constraint in all the selected specs. Note that only one cell per spec (i. e. per column) should be selected. The selection can be in multiple, distinct areas.

4.5.2 J_Min: hotkey CTRL - j

Optimises the selected values in the power consumption table by changing the number of hybrid channels J. Only one value per column may be optimised (and selected) at once. Multiple selections (in different columns) are allowed.

The built-in solver of excel does not converge in this case. Even if it did so, it would be much to slow, since it is searching for a solution in the real numbers, whereas this problem requires an integer solution. This algorithm is of the bisectional type: the step width, starting from a power of 2 value, is successively reduced by a factor 2 to approach the solution. The following parameters can be changed in the macro sheet (the letters in (..) appear in the listing):

(a) Initial value for J ("JA"), entire value.

(b) Initial step width ("Delta"), must be a power of 2.

(c) Minimum step width where the iteration stops ("Precision"), >=1, power of 2.

4.5.3 Display_chart: hotkey CTRL - d

Displays the selected data in a power consumption chart (3D bar diagram). The selection must always have 5 rows. Multiple areas are accepted, but only the first selection will be displayed. Only the data area should be selected, without any labels, which are added by the macro automatically.

This macro can be used for CMOS or GaAs data, partial (front end, digital part) or total power consumption. The title of the chart must then be adapted by hand. The spec no. in the chart always starts counting by 1, whatever columns have been selected.

The program is convenient to present the results obtained with the worksheet. Designed for power consumption of 5 different data series (e. g., the 5 multiplication methods considered in this work), it can also be adapted to other data by some minor modifications in the macro sheet.

Figure 4.1 Outline of the front end architecture [9]

5 SOPRANO AND MASTER SPECTROMETER TRADE-OFF

The trade-off has been performed with the spread-sheet presented in the previous chapter. According to different studies conducted by ESA, the specifications for Soprano and Master - Spectrometers, given in Table , have been analysed.

5.1 Specifications:

Spectrometer type	Total bandwidth Bt (MHz)	Frequency resolution (MHz)
Soprano high bandwidth/low resolution	5300	3
Soprano low bandwidth/high resolution	20	0.3
Master high bandwidth / low resolution	10000	50
Master low bandwidth / high resolution	600	3

Table 5.1 Specifications

5.2 Trade-Off guidelines

5.2.1 Objectives

The main objective is the minimisation of the power consumption.

Other objectives concern the complexity and mass of the system. In order to avoid very high gate counts for the digital circuits, we must exclude highly multiplexed systems, since the complexity is increasing with M.

For reasons of high mass and to ensure a satisfying stability of the analog part, a too high number of subbands (J) must be avoided as well.

5.2.2 Constraints

1. Sampling frequency. Fs depends on the AD convertor chosen. We assume a two bit convertor, working at max. 2.5 GHz or alternatively a 1 GHz convertor.
2. Clock frequency. Fc is limited by the path delays of the digital circuits. For CMOS correlators, 150 MHz is the upper limit.
3. The number of hybrid channels. If the filter bank is implemented as 2 stage SSB converters (see previous chapter), the number of channels J is most efficiently a multiple of 4.

5.2.3 Procedure

The main variables of the trade-off are the number of subbands/hybrid channels (J) and the multiplexing factor (M).

Constraint (1) imposes a minimum value on J: Jmin = 2 Bt / Fsmax.

Constraint (2) imposes a minimum value on the product J*M: (JM)min = 2 Bt / Fcmax.

Practically, we try different values of M (1, 2, 3, ...) and find a value for J which minimises the power consumption while respecting the constraints. J is then rounded to the next multiple of 4 still respecting the constraints.

5.3 The Soprano Spectrometer

INPUT

Spectrometer Specifications

Specification	Soprano high bandwidth				low bandwidth	
Bandwidth Bt (MHz)	5300	5300	5300	5300	20	20
Nb. of channels N	1767	1767	1767	1767	67	67
Hybrid channels J	20	24	28	36	2	1
Multiplex factor M	4	3	3	2	1	1

Partitioning Parameters

Parameter	Spec1	Spec2	Spec3	Spec4	Spec5	Spec6
Freq. resolution (MHz)	2,99943	2,99943	2,99943	2,99943	0,29851	0,29851
Bandw./correlator (MHz)	265	220,833	189,286	147,222	10	20
Sampling freq. (MHz)	530	441,667	378,571	294,444	20	40
Lags/Hybrid-channel	89	74	64	50	34	67
Nb. of corr.	320	216	252	144	2	1
Lags/corr.	23	25	22	25	34	67
Clock freq. (MHz)	132,5	147,222	126,19	147,222	20	40
Multiplexing used	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
Total Lag Number	7360	5400	5544	3600	68	67

OUTPUT

Chip Complexity (total gate count)

Specification	Soprano low resolution				low bandwidth	
1x1 bit	1911540	1402488	1439888	934992	17661	17402
3x3 level	2170317	1592352	1634815	1061568	20052	19757
2x2 bit (Level 4 R0)	2354906	1727784	1773859	1151856	21758	21438
2x2 bit (Level 4 RN)	2185037	1603152	1645903	1068768	20188	19891
2x2 bit (Level 4 full)	2517415	1847016	1896270	1231344	23259	22917

Power consumption (Watt), Front End

1x1 bit	23,575	25,08	29,085	32,595	2,3025	1,67625
3x3 level	26,075	28,08	32,585	37,095	2,5525	1,80125
Level 4	28,575	31,08	36,085	41,595	2,8025	1,92625

Power consumption (Watt), Total CMOS

1x1 bit	31,5092	31,5481	34,7769	36,9071	2,31356	1,69805

Table 5.2 The Soprano Spectrometer Trade-Off

3x3 level	43,2073	42,0466	44,8756	46,406	2,57639	1,84833
2x2 bit (Level 4 R0)	53,8506	51,6851	54,2175	55,3318	2,83775	1,99571
2x2 bit (Level 4 RN)	48,3536	47,2039	50,274	52,3442	2,83008	1,9806
2x2 bit (Level 4 full)	60,3041	56,9461	58,8472	58,8391	2,84675	2,01345

Wide band / low resolution specification

This spectrometer is determined by its high number of channels (1767). It tends therefore to have a high complexity and power consumption. The minimum values for J are 8 (for the 2.5 GHz convertor) resp. 12 (using a 1 GHz convertor). Both lead to high gate counts and power (see Table and Figure).

Specs 3 and 4 are based on low (2) or no multiplexing, the power consumption can then be optimised with J = 28 resp. 36. Spec. 3 leads to very similar results with J = 24.

Thus, the power consumption achieved for the two bit correlation is ~ 50 - 70 W.

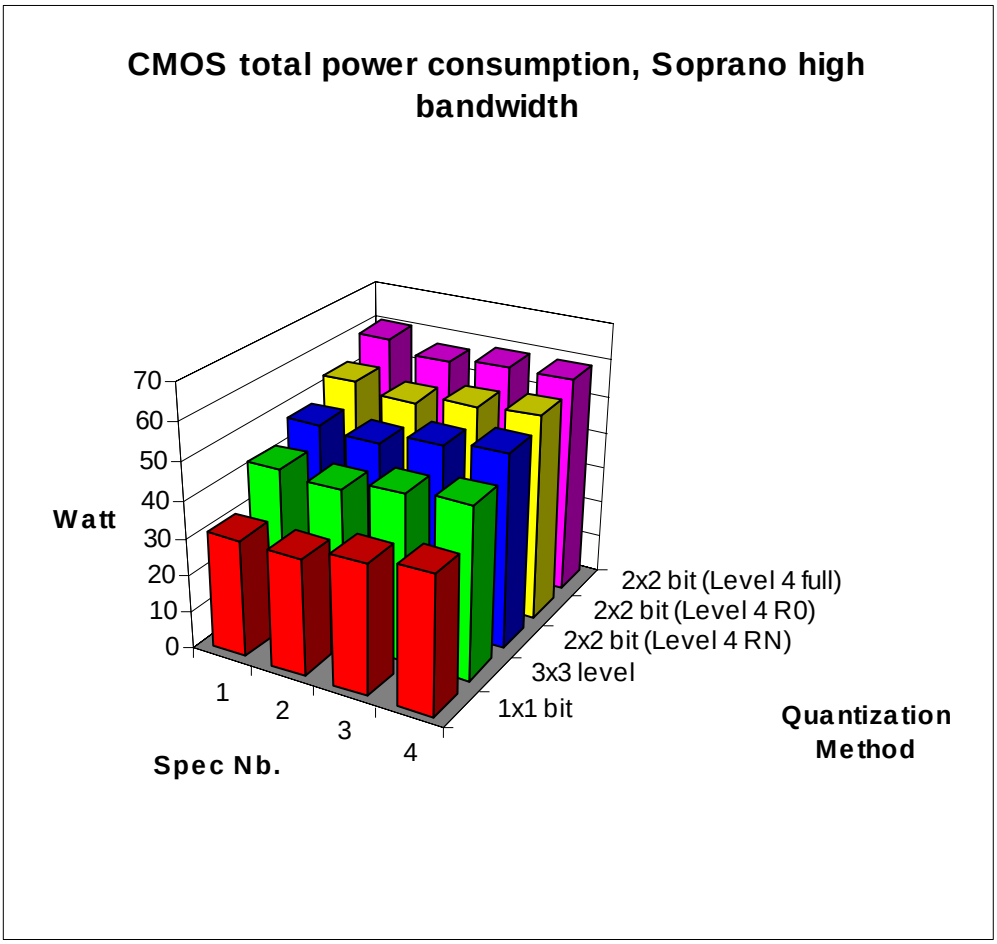

Figure 5.1 Soprano wide band trade-off

5.3.1 Narrow band / high resolution specification

This spectrometer does not present a major difficulty. It can be implemented as a single correlator, without subbanding, and leads to small ASICs and a power consumption of ~ 2 W. See Table and Figure .

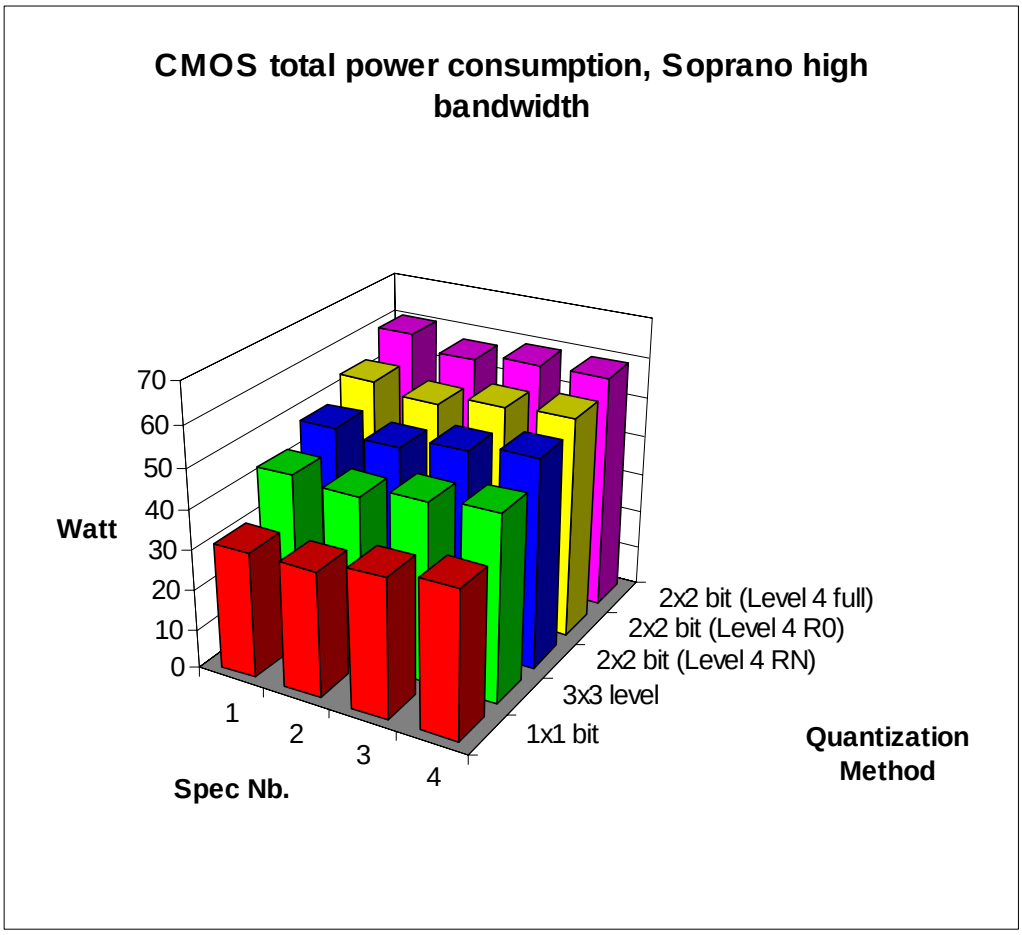

Figure 5.2 Soprano narrow band trade-off

5.4 The Master Spectrometer

Table 5.3 The Master Spectrometer Trade-Off

Wide band / low resolution specification

The difficulty here is the high bandwidth, which requires a high product J*M. A solution without multiplexing seems to be impossible, it would in fact require 68 subbands. Specs 1 and 2 (see Table and Figure) lead to a slightly higher, but not excessive power consumption. Specs 3, 4 and 5 optimise the power to values ~ 30 - 40 W. Spec. 5 requires an AD convertor faster than 1 GHz, but ensure the lowest number of subbands.

Figure 5.3 Master wide band trade-off

5.4.1 Narrow band / high resolution specification

This specification is optimised by a hybrid spectrometer with 4 subbands, which satisfies the constraint Fc < 300 MHz (see Table and Figure). The power consumption is ~ 5 - 8 W.

Figure 5.4 Master narrow band trade-off

5.4.2 Summary for Level_4_R0

Spectrometer type	J	M	P	gates/subband
Soprano High BW / Low Res.	24	2	68.74	94708
Soprano Low BW / High Res.	1	1	2.07	42875
Master High BW / Low Res.	20	4	38.94	25597
Master Low BW / High Res.	4	1	7.67	31996

Table 5.4 Summary of trade-offs

Table lists a convenient trade-off for each type of spectrometer with the corresponding number of subbands J, multiplexing factor M, power consumption for the Level_4_R0 correlator and the number of gates per subband. This means, the complexity of the ASICs will be, in any case, max. 100000 gates. Each subband being equipped with its own correlator, the system will need J chips of the above mentioned complexity.

6 THE VHDL MODEL

6.1 Outline of the model

This section gives a short outline of the model and describes briefly the files, located in the '/home/rweigand/acs' directory at the XR - Sun cluster. More detailed comments can be found in the source files themselves.

The whole model is parametrisable (multiplication method, word lengths, number of lags) by changing parameters in the package 'constants'.

According to Figure 2.1 on page 4, the model is divided in lags, and each lag is subdivided in a delay, multiplier, accu and counter. These components contain their sequential part in a process with edge construct. The design uses leading edge triggered flip-flops without or with asynchronous reset. Clock and Clear_N are distributed to each component.

All lags are placed in a top level entity 'acs'. The lag and top-level entities do not implement registers. Two test benches are provided, 'tb_lag' to evaluate the functionality of a single lag (test the multipliers), and 'tb_acs', to simulate multiple lags, starting from previously matlab-created data samples and writing output (autocorrelation function) into a text file. Several auxiliary files (scripts) are provided as described in the following lists.

6.1.1 VHDL files

- constants: package of global parameters for the configuration of the system and data types.
- tb_acs: instantiates L lags, applies data samples to the ACS (entity getsignal) and writes the result to file (entity result).
- getsignal: read (quantised) samples from 'noise.txt', used in tb_acs
- result: writes autocorrelation function to file 'correlation.txt', usable in matlab.
- tb_lag: instantiates 1 lag, applies all combinations of input samples to this lag.
- acs: top level entity, instantiates L lags, implements address demultiplexer for the readout bus.
- lag: one lag, instantiates delay, multiplier, accu and C times counter1 .
- delay: implements one bit of a shift register for N bit data words.
- multiplier: implements different multipliers, according to the equations given in chapter 2, selected by the METHOD parameter in the constants package.
- accu: implements the function SUM := SUM + PRODUCT with adders and registers, word length of SUM is parametrisable in constants. The carry of the MSB is fed as counter pulse to the output.
- counter1: implements one flip-flop, used for the asynchronous integration counter.
- noisegen: model for a generator of aleatoric numbers, using a shift register and XOR-function in the feedback. NOT a part of ACS.

6.1.2 Auxiliary files

- make_acs: makefile to analyse the hierarchy with make until the acs entity.
- make_tb_acs: makefile to analyse the hierarchy with make until tb_acs.
- open: unix command procedure to open the VHDL files in textedit.
- noise.m: matlab program, creates Gaussian noise samples, superposes a sine wave, digitises them and saves them in 'noise.txt'; executes autocorrelation in matlab (for comparison with VHDL-results).
- setup_sim: vhdlsim script to initialise the simulation.

6.1.3 Files in sub directory 'synthesis'

- *.db: examples of synthesis results, can be loaded into design analyzer.
- *.script script files for synthesis at the different levels of hierarchy.

Results of simulations and some matlab programs to process those data are located in other subdirectories of acs. Simulations and the synthesised circuit will be presented in the following sections of this chapter.

a) after 4096 samples b) after 262100 samples

c) S/N ratio vs. sample number

Figure 6.1 Autocorrelation function and power spectrum after different integration times.

6.2 Simulations in VHDL and Matlab

Simulations have been performed for two reasons:

- to confirm the correct functioning of the VHDL-Design in two steps:
  1. VHDL - simulation at single lag level to test that the multipliers satisfy the equations given in chapter 2. This is not reported here.
  2. VHDL - simulation of an autocorrelator with several lags, stimulated with a typical signal (weak sine wave hidden in a strong Gaussian noise). The same configuration was simulated by matlab and compared to the VHDL results.
- to evaluate the performance (sensitivity, efficiency) of different multiplication methods.

All matlab files are situated in the directory '/home/rweigand/matlab'.

6.2.1 VHDL simulations

Figure 6.1 shows the (full 2 bit) correlator output to an input signal with a S/N of -18 dB. The signal peak is hidden in noise after a few samples (a), but comes out clearly after a high sample number (b). The general trend of S/N vs. sample number can be seen in (c). This confirms the statement made above, that the signal degradation can be compensated by an increase of the integration time.

6.2.2 Cross-check with matlab simulations

The curve shown above (Figure 6.2-c) has been simulated in VHDL and in matlab for all multipliers and is represented in Figure 6.2-a and -b. Figure 6.2-c shows the difference between the two simulations. The difference is not zero, as one could expect. In fact, the accumulator implemented in VHDL is not skewed at its input. This means, that different bits of the product are not added at the same time, the integration counter can run a few pulses behind the true result.

This limited offset has an impact only at low sample numbers, but can be negliged after ~ 10^4 samples. In a practical use, the sample number will be several orders of magnitude higher than 10^4. Therefore, the model can be considered as working correctly.

a) VHDL model simulations, S/N in dB

b) Matlab simulations, S/N in dB

c) Difference S/N(Matlab) - S/N(VHDL) in dB

Figure 6.2 S/N ratio vs. number of samples

6.3 Synthesis: schematics, speed and complexity

All synthesis files are located in the directory 'home/rweigand/acs/synthesis'.

The VHDL model of a single lag was completed by the top design acs.vhd. This file describes an address decoder for the readout logic and instantiates L lags in the spectrometer. Synthesis has been verified for a low lag number (4, 16). For a high lag number, the address decoder is to be validated and the distribution of global signals such as CLK, CLK_N, CLEAR_N, UNDELAYED must be buffered.This may add some complexity to the circuit.

Synthesis of the model has been done in MG_MHSLIB. The synthesis procedure of this repetitive design is straightforward. Three different script files (mult.script, lag.script and acs.script) achieve a hierarchical synthesis step by step. The file lag.script contains mult.script and acs.script contains lag.script. The created schematics are shown in appendix A. Table 6.1 summarises complexity and speed of the different correlators (for 1 lag), obtained under ‘worst case commercial’ conditions

:

Method	max. delay	counter bits	comb. gates	seq. gates	total gates
bit_1	1.85 ns	24	31	228	259
level_3	2.73 ns (A)	22	46	246	292
level_4_rn	2.91 ns	22	48	246	294
level_4_r0	3.11 ns	21	61	255	316
level_4	3.00 ns	20	73	264	337

(A) max. delay path is situated in the accumulator (2 x XNOR), for all other methods, the slowest path is in the multiplier.

Table 6.1Summary of the synthesis results, speed and complexity

The Tri State Buffers for the readout bus are considered by Synopsys as a sequential element and therefore listed with the flip-flops.

The above total gate number does not include the clock tree and matches the gate number which is given in the last row of each lag model in the spreadsheet.

The delays (worst case commercial) allow a clock speed of >= 300 MHz for two bit and >= 500 MHz for one bit quantization.

7 CONCLUSION

7.1 Achievements

In the present work, the concept of digital Autocorrelation Spectrometers (ACS) has been investigated for the use in spacecraft, such as Earth Observation payloads. Different architectures have been studied, such as hybrid spectrometers (using analog subbanding), multiplexing and pure digital spectrometers; the digital autocorrelator has been studied with different coarse multipliers (1 and 2 bit), which allows to vary the sensitivity and the cost of the instrument according to the specifications.

The power consumption and the complexity of all combinations of architectures and multipliers can be estimated in a spread-sheet which was developed in MS-Excel. Trade-offs have been elaborated with this spread-sheet for several specifications of spectrometers, taken out of ESA projects (SOPRANO and MASTER).

The estimations are based on a design of the digital autocorrelator, which was elaborated in VHDL and synthesised in a cell based ASIC library (Matra MHS). This design was validated by different series of simulations, on an elementary level checking the outputs of the multipliers, as well as on a higher level, using real-like input signals and measuring the output spectra. The latter type of simulations was cross-checked by simulations in matlab, which were in a good correlation to the VHDL simulations.

7.2 Results

It has been shown in this work, that ACS are expected to fulfil the requirements for power consumption and complexity, using the following concepts:

- Coarse quantisation (1 or 2 bit)
- Hybrid architecture (analog filter bank + digital ACS)
- Cell based ASIC design in space qualified/qualifiable technologies

A new multiplication scheme was developed, where the low level products are not cancelled (set to 0), but set to the next non zero value (+/- n). The complexity, power consumption and the performance (sensitivity) of this method were shown to be situated in between level_4_r0 (low level cancelled) and the full level_4 method.

Simulations about the dependence of the efficiency on the quantiser decision thresholds have been performed. The results show similar minima, but they are affected by numerical noise.

7.3 Prospects and suggestions

With increasing sampling frequencies, errors in the analog-digital converters become more important. The investigation of their impact on the sensitivity of the spectrometer is therefore of a big interest for wide band spectrometers.

Some theoretical studies on quantisation errors are given in [3, 5]. Simulations have been proven to be insufficient to improve our knowledge about these phenomena. Therefore, an ACS should be executed as a hardware prototype:

1. Implementation of the digital autocorrelator in a FPGA.
2. Sampling and quantisation circuit with a multibit (e. g. 8 bit) converter.

The FPGA will contain correlators using different coarse quantisation concepts. A multibit converter allows then to simulate converter errors (variations of the comparator thresholds, offset and gain) in the test runs.

Detailed knowledge about the impact of quantisation errors is a condition to evaluate the total sensitivity of ACS. It is then possible to select a trade-off between the complexity (and cost) of the system and the minimal sensitivity required in the specifications for earth observation payloads.

On the other hand, concepts to compensate the mentioned errors are expected from a hardware study. This compensation should be obtained by a suitable design of the spectrometer (particularly the sampling and quantising circuit), and by special algorithms in post processing.

8 REFERENCES

[1] B. F. C. Cooper, *Correlators with two-bit quantization*
1970, Aust. J. Phys., 23, p. 521 - 527

[2] F.K. Bowers, R.J. Klingler, *Quantization Noise Of Correlation Spectrometers*
1974, Astron. Astrophys. Suppl. 15, p. 373 - 380

[3] L. R. D'Addario, A. R. Thompson, F. R. Schwab, J. Granlund
Complex cross correlators with three-level quantization: Design tolerances
1984, Radio Science, 19, No. 3, p. 931-945

[4] Sander Weinreb, *Analog-Filter Digital-Correlator Hybrid Spectrometer*
1985, IEEE Trans. on Instrumentation and Measurement, IM-34, No. 4, p. 670 - 675.

[5] Stephen Padin, Martin S. Ewing, *A Fast 2-Bit Digitizer for Radio Astronomy*
IEEE Transactions on Instrumentation and Measurement, vol. 38, no. 6, December 1989

[6] Brian Von Herzen, *VLSI Partitioning of a 2-Gs/s Digital Spectrometer*
IEEE J. of Solid-State Circuits, Vol. 26, No.5, p. 768-772 (May 1991)

[7] J.B. Hagen, D.T. Farley, *Digital-correlation techniques in radio science,*
1973, Radio Science Vol. 8 no. 8/9, p. 775 - 784

[8] Dake Liu, Christer Svensson, *Power Consumption Estimation in CMOS VLSI Chips,*
IEEE Journal of Solid-State Circuits, vol. 29, no. 6, p. 663 - 670, June 1994

[9] N.D. Whyborn, *Investigation of Digital Autocorrelation Spectrometers,* Final Report, Chalmers Technical University / Deutsche Aerospace, March 1993

9 APPENDIX: SCHEMATICS OF THE SYNTHESISED AUTOCORRELATOR

Figure 9.1 ACS Top level (4 lags)

Figure 9.2 One lag (4 bit accumulator, 8 bit counter). Flattened except the multiplier.

Figure 9.3 The level_4 multiplier.

Figure 9.4 The level_4_r0 multiplier.

Figure 9.5 The level_4_rn multiplier.

Figure 9.6 The level_3 multiplier.

10 APPENDIX: LISTINGS OF THE VHDL MODEL

Part II

DESIGN OF A PARALLEL FFT PROCESSOR USING FIXED POINT ARITHMETIC AND CSD-MULTIPLICATION

CONTENTS

1 INTRODUCTION 2

2 ALGORITHM 3

2.1 FFT RADIX 4 3
2.2 COMPLEXITY OF THE ALGORITHM 4
2.3 CANONIC SIGNED DIGIT MULTIPLICATION 5

3 ESTIMATION OF THE COMPLEXITY 6

3.1 ADDERS 6
3.2 FLIP FLOPS AND TOTAL AREA 7

4 PERFORMANCE AS A FUNCTION OF THE COMPUTING PRECISION 8

4.1 SIMULATION PROCEDURE AND DEFINITION OF THE PERFORMANCE 8
4.2 RESULTS AND EXPLANATIONS 8

5 VHDL DESIGN 13

5.1 OUTLINE 13
5.2 5.2. IMPLEMENTATION OF THE CSD MULTIPLIERS 15
5.3 SYNTHESIS OF THE DESIGN 15

6 CONCLUSION 16

7 REFERENCES 16

1 INTRODUCTION

The Fourier transform, particularly the FFT algorithm, is used frequently in many applications of digital signal processing (DSP), in communications as well as the Synthetic Aperture Radar (SAR).

The market provides us with dedicated FFT chips, computing usually in floating point arithmetic [2], but using serial processing, which leads to low computing speeds (spectra/second). Besides these performance considerations, off-the-shelf components always impose certain specifications, such as number of points, precision etc.

Two reasons lead to this work:

1. Increasing the speed of the FFT by using parallel processing.
2. Having a parametrisable design (number. of points, precision) for custom IC - development.

Therefore, the following approach was followed for the design:

1. Designing a parallel radix 4 FFT algorithm for 16, 64, 256 and 1024 points.
2. Using a Canonic Signed Digit (CSD) representation for the multiplication coefficients.
3. Pipelining after each butterfly of the FFT algorithm.
4. Fixed point arithmetic with a variable precision, set by design parameters. A floating point arithmetic was sacrificed for reasons of speed and complexity.

The main questions were (for different configurations):

1. Complexity of the chip
2. Speed of the chip.
3. Noise introduced by truncation errors in the fixed-point arithmetic.

2 ALGORITHM

2.1 FFT Radix 4

A detailed description of algorithms can be found in [1]. On page 715 and further, the radix 4 algorithm is explained, only the main ideas are given here.

Radix 4 can be applied, if the number of data points N is a power of 4. This can always be obtained by filling the data series with zeros, to complete $N = 4^S$, where S is the number of stages. Using the symmetry of complex numbers, the number of multiplications is significantly reduced. One stage in radix 4 applies the following algorithm:

$$\begin{bmatrix} X(0,q) \\ X(1,q) \\ X(2,q) \\ X(3,q) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} W_N^0 F(0,q) \\ W_N^q F(1,q) \\ W_N^{2q} F(2,q) \\ W_N^{3q} F(3,q) \end{bmatrix} \qquad (1)$$

X(p,q) is an element of the discrete Fourier transform of x(n):

X(p,q) = X(Np/4 + q)

F(p,q) is the discrete Fourier transform of the of the data points x(p,m). Here, p is a parameter and m is the summation variable. The data points x(p,m) are mapped to the FFT input as follows:

x(p,m) = x(4m + p)

The variables have the following intervals:

p = (0, 1, 2, 3) ; m, q = (0, 1, 2, ... , N/4 - 1)

The phase coefficient W_N^k is defined as:

$$W_N^k = e^{-j2\pi k/N}$$

Thus, through equation (1), the N point transform is split into 4 (p = 0, 1, 2, 3) transforms of N/4 points. This operation can be applied recursively, as long as the length of the partial transforms is again a multiple of 4. Thus, in the next stage, each of the 4 transforms of N/4 points can be split, by applying equation (1), into 4 transforms of N/16 points. In this way, with 4 series of N/16 points, we must, at the end, compute the matrix (1) 4*N/16 = N/4 times. This can be shown the same for all stages. The recursion stops, if a data 'series' F(p,q) is just a single point. Thus, we need S stages, if $N = 4^S$.

Equation (1) involves three multiplications with the complex W-coefficients, called phase factors (W_N^0 is always 1). The matrix multiplies with unitary elements 1, -1, j, -j, which translate into simple additions of complex numbers in rectangular representation. Each line of the matrix requires 3 complex additions, 12 in total. But the matrix can be split into 2 matrices, as it is shown in equation (2). This technique requires only 8 complex additions (one per each line of each matrix):

$$\begin{bmatrix} X(0,q) \\ X(1,q) \\ X(2,q) \\ X(3,q) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & j \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} W_N^0 F(0,q) \\ W_N^q F(1,q) \\ W_N^{2q} F(2,q) \\ W_N^{3q} F(3,q) \end{bmatrix} \qquad (2)$$

In a similar way to the radix 2 algorithm, the radix 4 can be drawn in the typical 'butterfly' structure shown in Figure 2.1. One of the big points represents the matrix multiplication of equation (1), involving 4 data points. This is called one butterfly. The numbers indicate the complex multiplications executed in the last vector of equation (1). They correspond to the index k in the coefficient W_N^k.

Figure 2.1 A 16 point radix 4 algorithm. Source: [1] p. 717

2.2 Complexity of the algorithm

The computation needs 8 complex additions and 3 complex multiplications per butterfly. One stage needs N/4 butterflies and the complete FFT needs S = ld(N)/2 stages. The complexity of the FFT radix 4 is therefore:

(3 / 8) N ld(N) **complex multiplications**

N ld(N) **complex additions**

Some simplifications apply to these numbers:

- The first stage after the data input has no multiplications (q = 0).
- If the FFT of pure real data is calculated, half of the operations of the first stage is eliminated. Since, in this case, also the output spectrum is symmetrical, the last stage is also reduced to half complexity.

2.3 Canonic Signed Digit Multiplication

The Canonic Signed Digit (CSD) representation uses a ternary, or signed-digits code. The possible digits are -1, 0, 1. This concept has been presented in [3].

A binary number, having a high number of consecutive non zero digits, can be represented with a lower number of non zero digits in CSD code. This shall be illustrated by an example:

$$31_d = (\,11111\,)_b = 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 2^5 - 2^0 = (\,1\;0\;0\;0\;0\;-1\,)_{CSD}.$$

The multiplication $X * 31_d$ can be executed in binary or CSD :

$$X * (\,11111\,)_b = (2^4 * X) + (2^3 * X) + (2^2 * X) + (2^1 * X) + (2^0 * X) \quad \text{==> 4 additions.}$$

$$X * (\,1\;0\;0\;0\;0\;-1\,)_{CSD} = (\,2^5 * X\,) - (\,2^0 * X\,) \quad \text{==> 1 subtraction.}$$

The multiplication with a power of 2 is a straightforward bit-shifting operation. The binary number has 5 non zero bits, the CSD number only 2. A multiplication with this number implies therefore 4 additions in binary and only one subtraction (or one change-sign operation and one addition) in CSD.

In our case, a parallel implementation of an FFT, numerous multiplications must be implemented, where the multiplier is a constant, the phase coefficient W_N^k. The complexity can be considerably reduced, when CSD code is used for the phase factors. The first step of this project was therefore the development of a tool to compute the optimised CSD code.

This task was accomplished in form of a matlab routine. The program ter.m in the directory /home/rweigand/fft/matlab, and its subprogram ter_sub.m compute the code in a recursive way.

3 ESTIMATION OF THE COMPLEXITY

The complexity depends on the number of single bit adders and the number of flip flops in the chip. Therefore, we must know, how many complex additions and multiplications are performed, the ratio between complex operation and one real addition and how many non zero bits are contained in CSD code.

3.1 Adders

As it has been shown in the last chapter, the FFT algorithm needs (3/8) N ld(N) complex multiplications and N ld (N) complex additions. In rectangular representation, one complex multiplication requires 4 real multiplications and 2 real additions, one complex addition requires 2 real additions.

It has been shown empirically, that in CSD code, the average number of non zero bits in numbers from -2^C to $+2^C$ (thus being max. C+1 bits long) is:

$$\mathbf{N_C = 1.75 + (C - 4) / 3} \quad \text{(valid for C>= 4).}$$

Compare to binary, two-complement code, where $N_B = (C + 1) / 2$.

A real CSD multiplication with a C+1 bit coefficient requires therefore $N_C - 1$ real additions.

Finally, the first stage does not have multipliers. Therefore, the number of complex multipliers must be multiplied by (1 - 1/S). If we use real data, the complexity of the butterflies in the first and the last stage is reduced by 50 %. This is equivalent to saying, that one stage has no butterflies and the number of complex adders is reduced by the same factor (1 - 1/S). Thus, the total complexity will be multiplied by (1 - 1/S).

With this information, we can determine a first estimation (N ld(N) is common factor):

Nb. / N ld (N)	Nb. of cpx op's	real / cpx op's	Nb. of real op's	Nb. of real add's
Mult	3/8	4 Mult + 2 Add	3/2	3/2 [3/4 + (C-4)/3]
Add	1	2 Add	2 + 3/4	11/4

Table 3.1

Adding up the last column of Table gives the number of additions = N ld(N) (1.875 + C/2). We round up (1.875 to 2), correct by the factor (1 - 1/S) mentioned above and substitute S = log4(N) = ld(N) / 2:

$$\mathbf{N_{add} = N (S-1) (C + 4)}$$

Most operations carry over the internal word length (I + 1 bits). If a one bit adder needs 10 gates, then we can estimate the number of gates for the adders (combinational area):

$$\mathbf{N_{gates\_comb.} = 10 N (S-1) (C + 4) (I + 1)}$$

The size of the adders might be a bit smaller in the first stage, where only D + 1 bit words are added, but there is also some additional complexity for inverters (sign change inverters) and clock tree. Therefore, we keep this equation, independent of D. This was justified empirically by several trials of synthesis.

3.2 Flip Flops and total area

The number of flip flops can be determined easily for the three levels of pipelining, input registers, intermediate (between stages) registers and output registers. Table 3.2 is valid for a FFT with real-data input:

	Input Registers	**Inter-stages Reg.**	**Output Registers**
Number of FF's	(D + 1) N	2 (I+1) N (S - 1)	(I + 1) N

Table 3.2

The number of gates in sequential area is therefore (1 register = 6 gates):

$$N_{gates\_seq.} = 6\,N\,[\,(D+1)+(I+1)(2S-1)\,]$$

The total complexity is given by:

$$N_{gates} = N\,[\,6(D+1)+(I+1)\{\,6(2S-1)+10(C+4)(S-1)\,\}\,]$$

The formula will be used in the tables of the next chapter.

4 PERFORMANCE AS A FUNCTION OF THE COMPUTING PRECISION

4.1 Simulation procedure and definition of the performance

The FFT-Model has been simulated in different configurations, in order to evaluate the degradation of the spectrum due to the truncation errors during calculation.

All simulations are based on a N=1024 points FFT. The simulation procedure (unix-batch files, matlab- and simulator command scripts) are stored in directory 'home/rweigand/fft/sim_results'.

The design is fed with a sine wave of frequency 5 (5 periods in 1024 points). The computed complex spectrum is converted in matlab into a power spectrum, from which the Signal to Noise Ratio (SNR) and the Spurious Free Dynamic Range (SFDR) are computed. If X_6 is the value of the 6. bin (for frequency 5), these values are defined as follows:

$$\mathbf{SNR} = 10 * \log_{10}\left(\frac{\mathbf{X}_6}{\sum_{\mathbf{i}\neq 6}\mathbf{X}_\mathbf{i}}\right); \qquad \mathbf{SFDR} = 10 * \log_{10}\left(\frac{\mathbf{X}_6}{\max_{\mathbf{i}\neq 6}(\mathbf{X}_\mathbf{i})}\right)$$

Series of simulations have been carried out with different numbers of internal bits I = 7, 9, 11, 13, 15 (all bit numbers are given without sign bit, as in the constants-file of the model, thus, I = 7 means, the output is an 8 bit two's complement number, -127 to 127).

These series carry over different values of D (input data bits) and C (bits of W-coefficients).

The results are summarised in the tables and plots below.

4.2 Results and explanations

When Inf (infinite) appears in the tables, all other bins, except X_6 are 0. We have an ideal FFT engine and we achieve maximum performance. This max. performance, of course, is not infinite, but limited by the quantisation noise and is < 20 $\log_{10}$ (2^I). In the case of a sine wave, we obtain 6 dB (factor 2 in absolute value) below this value, since the Fourier-transformed sine function never uses the output dynamic range at its full extent.

The infinite values should therefore be replaced by the max. not inf. value in each curve. Thus, the curve finishes into a horizontal pane (dashed line in the plots).

SFDR and SNR show slightly different values (SFDR > SNR), but their basic behaviour is the same. It is therefore sufficient to know one of both values, to trade off a configuration.

All curves for D >= I - S are identical. We obtain no gain of precision for a higher input word length than I - S, in some cases, there is even a degradation (I = 7).

Depending on D, the increase of C over a certain value does not improve the performance either.

The tables also show the complexity, obtained with the formula from the last chapter.

C =	2	4	6	8	10	12	14
	Results in dB for I=15						
SNR	(Signal to Noise Ratio)						
D = 5	13.38	28.76	35.48	36.26	36.39	36.42	36.45
D = 7	13.36	29.34	40.69	48.55	50.06	50.33	50.49
D = 9	13.35	29.36	41.13	53.19	63.63	71.97	70.66
D = 11	13.35	29.37	41.19	53.68	64.28	84.28	Inf
D = 13	13.35	29.37	41.11	53.70	64.03	84.29	Inf
SFDR	(Spurious Free Dynamic Range)						
D = 5	16.91	36.33	45.37	44.35	44.45	44.45	44.45
D = 7	16.87	36.30	47.09	57.79	57.80	57.80	58.22
D = 9	16.87	36.33	47.24	59.89	67.37	74.73	72.23
D = 11	16.87	36.31	47.25	59.90	67.38	84.28	Inf
D = 13	16.87	36.31	47.09	60.39	67.38	84.29	Inf
D	Complexity						
11	4.89E+06	6.20E+06	7.51E+06	8.82E+06	1.01E+07	1.14E+07	1.28E+07

Table 4.1 Performance and complexity of the 1024 point FFT with I = 15

Figure 4.1 Performance of the 1024 point FFT with I = 15

C =	2	4	6	8	10	12	14
SNR	**Results in dB for I=13**						
D = 5	13.45	29.30	37.06	37.99	38.05	38.06	38.05
D = 7	13.43	29.77	42.29	53.68	57.01	57.57	58.97
D = 9	13.43	29.78	42.69	57.45	72.23	Inf	Inf
D = 11	13.42	29.80	42.78	57.76	69.23	Inf	Inf
SFDR							
D = 5	16.93	36.43	46.46	45.19	45.20	45.20	45.19
D = 7	16.89	36.43	47.59	62.19	62.65	62.65	62.65
D = 9	16.91	36.47	48.34	62.22	72.23	Inf	Inf
D = 11	16.89	36.48	48.35	62.23	72.24	Inf	Inf
D	Complexity						
9	4.28E+06	5.42E+06	6.57E+06	7.72E+06	8.86E+06	1.00E+07	1.12E+07

Table 4.2 Performance and complexity of the 1024 point FFT with I = 13

Figure 4.2 Performance of the 1024 point FFT with I = 13

C =	2	4	6	8	10	12	14
SNR	**Results in dB for I=11**						
D = 5	13.75	31.27	44.30	46.95	46.75	47.42	47.42
D = 7	13.74	31.37	48.37	Inf	Inf	Inf	Inf
D = 9	13.71	31.20	48.40	Inf	Inf	Inf	Inf
D = 11	13.74	31.19	48.41	Inf	Inf	Inf	Inf
SFDR							
D = 5	17.05	37.15	50.44	50.42	50.43	50.43	50.43
D = 7	17.01	36.69	53.14	Inf	Inf	Inf	Inf
D = 9	17.04	36.73	53.17	Inf	Inf	Inf	Inf
D = 11	17.05	37.33	53.18	Inf	Inf	Inf	Inf
D	Complexity						
7	3.66E+06	4.64E+06	5.63E+06	6.61E+06	7.59E+06	8.58E+06	9.56E+06

Table 4.3 Performance and complexity of the 1024 point FFT with I = 11

Figure 4.3 Performance of the 1024 point FFT with I = 11

C =	2	4	6	8	10	12	14
SNR	**Results in dB for I=9**						
D = 5	14.94	36.50	Inf	Inf	Inf	Inf	Inf
D = 7	14.73	35.79	Inf	Inf	Inf	Inf	Inf
SFDR							
D = 5	17.74	38.42	Inf	Inf	Inf	Inf	Inf
D = 7	17.49	38.55	Inf	Inf	Inf	Inf	Inf
D	Complexity						
5	3.05E+06	3.87E+06	4.69E+06	5.51E+06	6.32E+06	7.14E+06	7.96E+06
7	3.06E+06	3.88E+06	4.70E+06	5.52E+06	6.34E+06	7.16E+06	7.97E+06

Table 4.4 Performance and complexity of the 1024 point FFT with I = 9

Results in dB for I=7

D	SNR	SFDR	N_gates
3	19.42	20.36	2.43E+06
5	17.98	19.82	2.45E+06
7	17.30	19.16	2.46E+06

Table 4.5 Performance and complexity of the 1024 point FFT with I = 7

Figure 4.4 Performance of the 1024 point FFT with I = 9

5 VHDL DESIGN

A parallel hardware design can not use the recursive algorithm, shown in chapter 2. The butterfly structure must be generated in loops. As one can see in Figure 2.1 on page 4, the design is partitioned in stages. Each stage contains N/4 butterflies and (except for the first stage) three multiplications per butterfly. But all the stages are wired in a different way, they are not identical.

5.1 Outline

Figure 5.1 shows the outline of the FFT processor with its signal and component names.

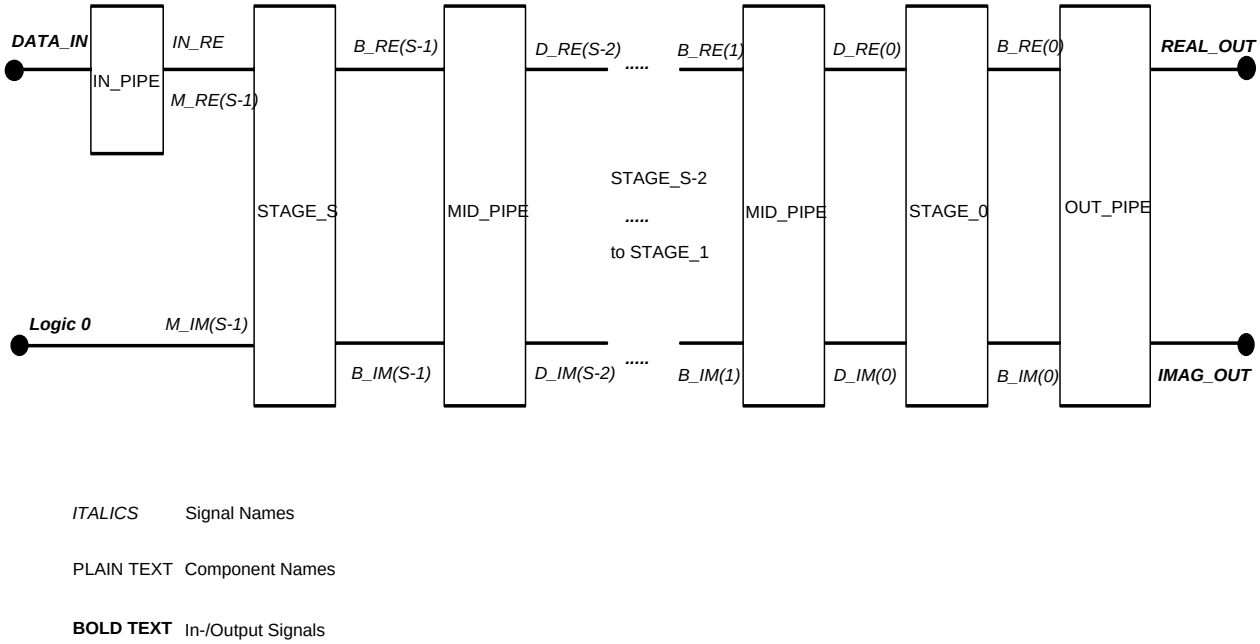

Figure 5.1 Block schematic of the FFT module

The first stage (just after the input) is the entity STAGE_S. This stage just infers butterflies, since no multipliers are needed. The other stages (STAGE_N for N = 0 to S-2) are instances of the entity STAGE, with a generic parameter N. In STAGE, multipliers and butterflies are inferred. The parameter N determines the wiring of the butterflies, as well as the W(N,K) coefficients for the multipliers. After each stage, MID_PIPE or OUT_PIPE creates the pipelining flip flops.

The design can process fully complex data. But the imaginary input is actually fed with zeros, as well as OUT_PIPE reads only half of the output signals (from 0 to N/2-1). This means, that during synthesis/optimisation, the operations, corresponding to the not used signals, are eliminated. The change to a fully complex FFT is a relatively straightforward procedure. Definitions of DATA_IN, REAL_OUT and IMAG_OUT must be changed to the full set of data (N real and N imag. data points), IN_PIPE and OUT_PIPE can be replaced by MID_PIPE.

The directory /home/rweigand/fft contains the following VHDL files:

constants: configuration, global constants and data types, W(N,K) coefficients

fft: top level entity

in_pipe, mid_pipe, out_pipe: pipelining flip flops

stage_s: first stage (after the input)

stage: lower stages (S-2 to 0), generic N: nb. of stage

mult: multiplier with CSD coefficient, generic K = nb. of coefficient

mult_csd: same as mult

mult_bin: multiplier with integer coefficient, generic K = nb. of coefficient

butterfly: performs the butterfly matrix multiplication

tb_fft: stimulates fft with input data, writes output data to a text file

cfg: configuration of the model, for simulation use

The directory /home/rweigand/fft/matlab contains the following files:

w_sin.m: computes the W(1024,K) in integer representation and stores them in a text file 'sin_data.txt'.

w_csd.m: computes the W(1024,K) in CSD representation, resolution C=16 bit and stores them in the text file csd_coeff.txt'.

ter.m: main program to convert integer to CSD (ternary code), se chapter 2.

ter_sub.m: recurrent subprogram to compute the optimal CSD code.

terbin.m: compares CSD to its binary equivalent, to check ter_sub.

itobin.m: converts integer to binary.

The directory /home/rweigand/fft/sim_results contains diverse matlab and text files for simulation of the performance:

result.m: computes SNR and SFDR (see chapter 4) of the spectrum obtained with VHDL simulations.

post.m: plots SNR and SFDR graphs.

result_*: results of simulation (SNR and SFDR values)

sim.script: VHDL simulation command script.

simu: unix command file to execute simulations

simu.prn: print/documentation file for the simulation procedure.

The directory /home/rweigand/fft/synthesis contains synthesis scripts and diverse matlab programs to create the synthesis scripts.

The directories /home/rweigand/fft/fft16, -fft64, -fft256 contain configurations (constants.vhd) and results (.db files) of different synthesis runs.

5.2 Implementation of the CSD multipliers

The design MULT (or MULT_CSD) performs the multiplication of a number of I+1 bits by a CSD coefficient, which is determined by the generic parameter K. The coefficient is read from the global constant CSD_CO, defined in the CONSTANTS package. CI and CR are the real and imaginary part of W(N,K), truncated to C+1 bits.

According to the rules of a complex multiplication (in rectangular representation), 4 products (PR1, PR2, PI1, PI2) and two additions must be performed. The products are arrays of integer ranges corresponding to the product's word length (I+C+1). Each element of the array corresponds to one bit of the coefficient. In the loop PART_PROD, the bit shifted value of one element is assigned to the following, adding or subtracting eventually (depending on the value of the coefficient bit -1, 0, or 1) the input multiplicand. In this way, an addition or subtraction is created only, when there is a non zero bit in the CSD multiplicator.

The partly products are added in the last two statements. Here, a type conversion applies to avoid constraint errors during simulation. In fact an integer addition of two numbers of N bits creates a sum of N+1 bits (the carry bit). But in our case, since the complex coefficients are pure phase factors, |W(N,K) | = 1. This means, that our sum never uses the carry bit. Nevertheless, the simulator creates it during initialisation and requires a corresponding target data type. Therefore, the addition is performed between 'signed' data types, where the carry bit is ignored.

The sum is also divided by 2**C. This truncation applies to come back to the internal word length (I+1) in the fixed point arithmetic.

5.3 Synthesis of the design

Several runs of synthesis have been performed using the tools (script files) in the directory '/home/rweigand/fft/synthesis'. The best way to run the synthesis is compiling all the designs one by one, saving and deleting them before the next one is read. Designs with generic parameter must be read with the 'elaborate' command.

Table 5.1 compares the synthesised area of two examples (16 and 64 points) with the complexity estimated by the formula of chapter 3. The synthesised and optimised designs are stored in 'home/rweigand/fft/fft16/fft_16_csd_3.db' and 'home/rweigand/fft/fft64/fft64_opt.db'.

N	S	C	D	I	Synth. Area	Est. Area	Max. delay
16	2	6	7	9	19456	19648	43 ns
64	3	5	4	7	105116	109440	32.3 ns

Table 5.1

The maximum delay is between 32 and 43 ns. This means, that the speed is between 20 and 30 MHz (or 20 - 30 M spectra per second). Note that the speed does not depend on the number of points, but on the precision of the computations (wordlength, C and I).

The synthesis of a 256 and 1024 points design did not succeed. The reason is a bug in Synopsys 3.2, which is documented in the 'Synthesis Release Note 3.2b', with the ID-Nr. STAR 2291. This defect causes the process to run out of memory when arrays of arrays are used in the design, which is the case in our FFT model. The problem should be fixed in version 3.3 (Release Note 3.3a). The release 3.3a was available, but could not be installed successfully at ESTEC until the end of this project. With the new release, the synthesis for N = 256 and 1024 is expected to be successful.

6 CONCLUSION

A FFT module has been designed in VHDL. The model is parametrisable to different complexities (number of points N = 16, 64, 256, 1024) and a varying precision (input, internal and coefficient word length).

The concept uses a Radix 4 algorithm and a CSD representation for the phase coefficient. The FFT is computed in parallel, this means, that at each clock cycle, one complete FFT is coming out of the module. The computations are executed in fixed point arithmetic. The latency (the number of clock cycles between the input and its corresponding transformed output) depends on the number of stages (S = ld(N) / 2).

The complexity of different configurations has been estimated. As a result, the FFT for N=1024 points requires around 10^7 gates. This is not feasible in current ASIC technology. For N=256 and less, we can achieve a complexity <= 10^6 gates.

The influence of more or less coarse fixed point arithmetic was studied by VHDL simulations. The results show that the most important parameter which decides about the performance is the word length of the internal computations. Input- and coefficient word length have optimal values depending on the internal word length, but their impact on the overall result (signal to Noise ratio) is small.

Logic synthesis has been executed in several trials. The model passes for N = 16 and 64 points. The synthesised area matches the estimated area with less than 5% difference. The speed of the module depends on the precision (computation word length), but not on the number of data points N. In most cases, the design is expected to run at a clock frequency around 20 - 30 MHz.

The synthesis for N = 256 and 1024 did not work due to a bug in the Synopsys software which was actually used. This defect has been fixed in the latest release (3.3a) of the software. The model is expected to work successfully when release 3.3a. will be installed correctly on the XR Sun cluster.

7 REFERENCES

[1] Preaches, John G., *Introduction to digital signal processing,* New York (Macmillan Publishing Company) 1988, page 698 - 720.

[2] Report Alcatel Espace, *A 16 bit real FFT ASIC,* When and Where ?

[3] something about CSD .