An Optimal Architecture for a DDC

Tjerk Bijlsma, Pascal T. Wolkotte, Gerard J.M. Smit
University of Twente, Department of EEMCS
P.O. Box 217, 7500 AE Enschede, The Netherlands
{Bijlsma, Wolkottept, Smit}@cs.utwente.nl

Abstract

Digital Down Conversion (DDC) is an algorithm, used to lower the amount of samples per second by selecting a limited frequency band out of a stream of samples. A possible DDC algorithm consists of two simple Cascading Integrating Comb (CIC) filters and a Finite Input Response (FIR) filter preceded by a modulator that is controlled with a Numeric Controlled Oscillator (NCO). Implementations of the algorithm have been made for five architectures, two Application Specific Integrated Circuits (ASIC), a General Purpose Processor (GPP), a Field Programmable Gate Array (FPGA), and the Montium Tile Processor (TP). All architectures are functionally capable of performing the algorithm. The differences between the architectures are their performance, flexibility and energy consumption. In this paper we compared the energy consumption of the architectures when performing the DDC algorithm. The ASIC is the best solution if digital down conversion is constantly required. When digital down conversion is needed only parts of the time, the Altera Cyclone II is the best solution due to its smaller technology size. In the spare time the reconfigurable architectures can be reconfigured for other tasks of today's multimedia devices.

1 Introduction

Modern mobile multimedia devices need to deliver top performance. The user requires adequate functionality, adaptive behaviour and energy-efficiency. As in all designs a trade off needs to be found for the optimal solution.

With the growing popularity of mobile multimedia devices, the demand for energy-efficient wireless communication increases. A key part of wireless communication is the Digital Down Converter (DDC) [8]. Like most algorithms, it can be performed by dedicated hardware or executed on a kind of processor. Which solution is preferred, depends on the desired adaptive behaviour, processing performance and energy-efficiency.

In a multimedia device like a PDA it is possible that the digital down conversion is only needed occasionally. For example, when the GSM module is activated, to connect to the internet via WLAN, or when the user wants to listen to digital radio via Digital Radio Mondiale (DRM) [10] [11] or Digital Audio Broadcasting (DAB) [9]. It can be useful to reconfigure that part of the chip, which is used for the DDC, to execute other tasks, instead of going to its standby modus. In this case the hardware has a higher utilization factor.

When a DDC is needed in a device like a mobile phone or a single mode digital radio, it has to perform a dedicated task. The algorithm has to be performed continuous and only the parameter settings might be changed. Adaptability and spare performance will not be used in this situation.

In this paper the realization of a DDC on five architectures are compared, two ASICs, an FPGA, a GPP and the Montium Tile Processor. The objective of this paper is to find an energy-efficient solution for a DDC. Therefore the report starts with discussing how the DDC algorithm works in section 2. The sections 3 till 6 discuss the results of mapping the DDC algorithm on the five architectures. In the conclusion of this report the results are summarized and the optimal implementation is suggested.

2 The Digital Down Converter

The Digital Down Converter (DDC) is typically used in mobile communication [13]. The DDC processes the samples from the AD-converter in such a way that it selects a small band of the total frequency range. After selecting the desired frequency band the signal is processed by a concatenation of filters. By attenuating the unwanted frequencies the signal can be resampled at a lower rate. The reduced sample rate relaxes the processing after the DDC. Depending on the selected frequency range, sample rate, and desired quality, different filter combinations can be used to perform the DDC.

In this report several architectures are compared on their performance of the DDC algorithm. To make a fair compar-

Component	Clock/sample rate	Decimation (D)
NCO	64.512 MHz	-
CIC^2	64.512 MHz	16
CIC^5	4.032 MHz	21
125 taps FIR	192 kHz	8
Output	24 kHz	-

Table 1. Clock speed and decimation in a DDC

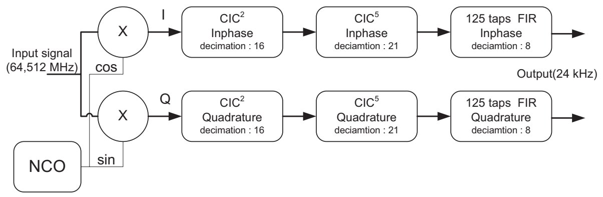
Input signal (64,512 MHz)
X
I
cos
X
Q
sin
NCO
CIC^2 Inphase decimation : 16
CIC^5 Inphase deciamtion : 21
125 taps FIR Inphase deciamtion : 8
CIC^2 Quadrature decimation : 16
CIC^5 Quadrature deciamtion : 21
125 taps FIR Quadrature deciamtion : 8
Output(24 kHz)

Figure 1. DDC algorithm

ison, the architectures should perform similar algorithms. The configurations should also be similar. A configuration for a DDC to select a DRM band has been found and used to compare the architectures. The filter and its configuration are depicted in Figure 1 and Table 1. The desired configuration of the DDC is discussed at the end of this section. First the operations of the algorithm are explained.

2.1 Parts of the DDC algorithm

The first part of the DDC algorithm is the Numerical Controlled Oscillator (NCO) (see Figure 1). This component produces a sine and cosine signal. The NCO calculates these values, e.g. by Taylor series, or reading from a look-up table. The signals from the NCO are used to shift the frequencies. To generate an in-phase (I) signal the input signal is multiplied with the cosine signal. The quadrature part (Q) is derived by multiplying the input signal with the sine signal.

The next step is to filter the shifted signal with a Cascading Integrating Comb (CIC) filter. The CIC filter is used in the parts with the highest sample rates. The high sample rates can be handled by using only additions and no multiplications. The filter consists of a cascaded set of integrating and comb filters [7]. An example of the algorithm is given in Figure 2. This is a CIC^2, which means it has two integrators and two comb filters. The CIC^5 has five integrators and five comb filters.

The integration is done in the first part of the CIC^2. This part adds the previous calculated value $x_{int}[n-1]$ to the input signal $x[n]$ and stores the result $x_{int}[n]$ for the next input sample. The comb filter is a special kind of FIR filter with fixed coefficients. The filter subtracts the previous input value from its current input. The result is given to the next comb filter that delivers the result. The decimation in the CIC^2 is to reduce the sampling rate. According to the decimation factor *D*, it only sends 1 out of *D* samples to the comb part of the filter. Where *D* is the decimation rate.

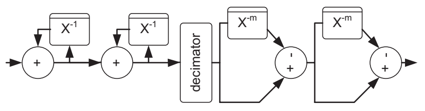

Figure 2. CIC^2

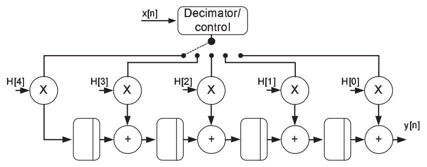

Figure 3. Polyphase FIR filter with 5 taps and a decimation of 5

The drawback of the CIC filters is their sub-optimal frequency attenuation. Therefore an additional Finite Input Response (FIR) filter is added to the filter chain, which has 125 taps. After the FIR filter a final decimation step determines the output sampling rate. Performance can be gained by using a polyphase FIR filter, which combines a FIR filter with the sampling rate reduction. An example of a 5 taps polyphase FIR filter is given in Figure 3. A normal FIR filter stores the input values $x[n] \ldots x[n-M-1]$, where *M* is the number of taps, in a chain of registers. Every clock cycle, the values $x[m]$ from the registers are read and multiplied with a coefficient $h[m]$. The sum of all multiplied values is calculated and delivered as one output sample $y[n]$. After each clock cycle, the oldest value is dropped and all the other values are shifted one position to the right. Due to the decimation 1 out of *D* output samples is actually used. The polyphase FIR filter writes the input values to the correct registers at the input sample rate. But it reads, multiplies and calculates the sum only every *D* cycles for an output sample. In Figure 3 the decimator/control writes the values to the correct registers. In case the decimation of this filter would be five, it would have to calculate the sum once every five clock cycles. The next input value is stored in the right most register. The second value will be stored in the register one from the right and so on. At the fifth clock cycle all registers will be updated, so the summation can be done. To implement a 125-taps polyphase FIR filter, the FIR filter in Figure 3 should be increased with a factor 25.

3 ASIC implementation of a DDC

The implementation of a DDC on an Application Specific Integrated Circuit (ASIC) is the fastest solution, but also the least flexible. In this section, two ASIC implementations are discussed. The implementation of Texas Instruments is commercial available as a single chip solution. This is in contrast to the second DDC implementation, which is not commercial available and is optimised for low-

Parameter	Value
Input speed of filter	Up to 100 MSPS
Input size of filter	14 (4ch.) or 16-bit (3ch.)
Decimation of a channel	32 to 16.384
Output size of filter	12,16,20 or 24-Bit
Energy consumption for a GSM channel	115mW (80 MHz & 2.5 V)

Table 2. Configuration of a TI Quad DDC

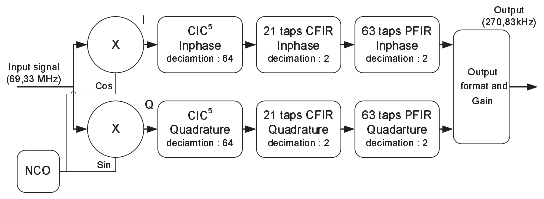
Input signal (69,33 MHz)
X
I
Cos
X
Q
Sin
NCO
CIC5 Inphase deciamtion : 64
21 taps CFIR Inphase decimation : 2
63 taps PFIR Inphase decimation : 2
CIC5 Quadrature deciamtion : 64
21 taps CFIR Quadrature decimation : 2
63 taps PFIR Quadarture decimation : 2
Output format and Gain
Output (270,83kHz)

Figure 4. Channel of the TI GC4016

power consumption.

3.1 Texas Instrument's Quad DDC-chip

Texas Instruments (TI) provides a "GC4016 Multi-standard quad DDC-chip" as ASIC solution for a DDC [17]. First the design of this chip will be discussed. At the end of this section we try to generalize the performance of this chip.

3.1.1 Design and specifications

The schematic drawing for one of the four channels of the Quad receiver chip is given in Figure 4. In Table 2, a part of the specifications can be found. Each channel can perform an independent DDC on a 14-bit input. The chip can be configured to use one or more channels. When the input width of 16 bits is chosen for the DDCs, only three channels are available. The results of the channels can be combined at the output using either a multiplexer or an adder.

The four DDC filters in the chip all have the same design, but have their own parameter settings. The DDCs consist of a 5-stage CIC filter, followed by a 21 taps Complex Finite Input Response (CFIR) filter and a 63 taps Programmable Finite Input Response (PFIR) filter. All the filters perform decimation. The decimation of the CIC^5 can be set to a value ranging from 8 to 4096, the CFIR and the PFIR both decimate by 2 leading to a decimation range from 32 to 16384.

As can be seen in the Table 2, the ASIC is capable of performing digital down conversion on four 14 bits inputs, which are fed with 100 MSPS to the DDCs. The chip is clocked at the speed at which the samples are fed to the chip.

A specification that is missing is the technology size. Since the chip is available from the beginning of 2001 and using 2.5 V internally, it is probably developed with 0.25 μm technology.

3.1.2 Power estimation

The documentation of the GC4016 DDC [17] comes with an example in which digital down conversion is performed for a GSM channel. This example uses a different configuration compared to the reference DDC of section 2.

The example DDC for a GSM channel runs at 80MHz and uses 115mW for a channel. The input rate of the chip is 69.333 MHz and the output rate is 270.833 kHz, which is roughly ten times the required sample rate for a DRM receiver. Because the FIR filters can only decimate with 2, the CIC^5 performs a decimation of 64. The FIR filters are configured to use 68 taps.

When the filter for this example is compared to the layout in section 2, a few things are different. The CIC^2 is not present, the DDC in the example decimates with 256 in stead of 2688, the combination of the two FIR filters provide up to 84 taps instead of 125, the output sample rate is 270.833 kHz instead of 24 kHz. This makes it difficult to compare the reference configuration and the GSM example. Nonetheless, the example is interesting. Reasons are that the DDC is commercially available as a single chip solution and the example has approximately the same amount of input MSPS. The first stages of the DDC consume most of the energy, because this part is working with the highest sample rate.

The main technology size in this paper is 0.13 μm at 1.2 V, while this ASIC is made with 0.25 μm at 2.5 V. The larger transistors and higher voltage can clarify the difference in energy consumption of this chip and the customized low power DDC of section 3.2. According to [14] it is possible to estimate the energy consumption for a smaller technology. The common dependency of the dynamic power consumption is that it is linear related to the total capacitance (C) and frequency and quadratic related to the voltage (V). With reduction from 0.25 μm to 0.13 μm the capacity goes down with a factor 0.25/0.13. The same goes for the voltage that drops with a factor 2.5/1.2. This makes it reasonable that the power consumption decreases with a factor $(\frac{2.5}{1.2})^2 \cdot \frac{0.25}{0.13}$. This would lead to 13.8 mW when the DDC was made in 0.13 μm technology.

3.2 Customised Low Power DDC

The second ASIC implementation is one that can be configured to the chosen filter layout from section 2 [15]. The DDC can perform a maximum decimation of 65536, and a minimum of 2. It has been realized in 0.18 μm technology with a V_{dd} of 1.8 V. The size of the core is 1.7 mm^2.

When performing the digital down conversion at 64.512 MHz, with the configuration of section 2, it consumes 27 mW. The power consumption is based on gate count and activity rate estimation. The power consumption is based on 0.18 μm technology. As described in the

Part of filter	Clock speed	Percentage of clock cycles
NCO	64.512 MHz	50 %
CIC^2-integrating		40 %
CIC^2-cascading	4.032 MHz	3.2 %
CIC^5-integrating		4.4 %
CIC^5-cascading	192 kHz	< 0.5 %
FIR125-poly-phase		< 0.5%
FIR125-summation	24 kHz	1.6 %

Table 3. Division of the DDC code for an ARM

previous section we estimated the energy consumption for a 0.13 μm design at 1.2 V. The energy consumption for the smaller technology size would be $27/((\frac{1.8}{1.2})^2 \cdot \frac{0.18}{0.13}) = 8.7$ mW.

4 Mapping a DDC on a GPP

In this section the program written for a General Purpose Processor (GPP) is discussed. The DDC was written in C-code. This code was compiled for an ARM 9 [5]. This section starts with a short discussion on the ARM 9, followed by a discussion on the implementation. The section finishes with power estimation for the ARM 9.

4.1 The ARM 9

The company ARM is known for the GPP and Digital Signal Processors (DSP) IP-cores that it develops. One of their series is the ARM 9 series, which is quite energy-efficient [5]. These GPPs are developed for the 0.13 μm process and can perform up to 250 MIPS. The design is sold as an IP-core that can be embedded in a SOC. The ARM922T core has two small caches of 8 KB. It can handle 32-bit and 16-bit instructions and 32-bit operands. A multiply-accumulate (MAC) instruction requires several clock cycles. The ARM can fetch and write data from/to the memory in one cycle.

4.2 DDC algorithm on a GPP

4.2.1 The implementation

To see how an ARM GPP would perform a DDC algorithm, the algorithm of section 2 is written in C. For simplicity reasons, the code only performs the in-phase transformation, so the result has to be doubled for the whole DDC. It is assumed that the values for the cosines and the sinus function are fetched from a look-up table. The C-code is compiled to assembler for the ARM processor.

The resulting assembler code is simulated and profiled with the ARM source-level debugger. The profiling results showed that the ARM needs to perform 2865 Mega instructions per second, to perform the inphase transformation. These instructions require $4.870 * 10^9$ clock cycles per second. The I part of the algorithm is equal in size to the Q part, so the amount of instructions and clock cycles per second has to be doubled. To perform the DDC algorithm real-time, the ARM should have a clock frequency of 4870*2=9740 MHz.

4.2.2 Energy consumption

The energy consumption of the ARM can also be calculated. The ARM922T is used with its caches enabled, which reduces the communication with the memory for the DDC algorithm. The lowest power consumption is obtained if the biggest part of the algorithm is loaded in the cache. The documentation states that the core and activated caches consume 0.25 mW/MHz [5](memory access not included). To perform a DDC, the ARM requires 9740 MHz. Multiplying 9740 MHz with 0.25 mW/MHz results in 2.435 Watt to perform the DDC algorithm. This value does not include the memory or the memory access.

Three notes have to be made. First, one ARM is not able to perform the algorithm at the required speed. The calculations are used as an indication of the performance of a general purpose processor. The second note is that the code was not optimized. It should be possible to speed up the algorithm when it is completely optimized for the ARM922T. The last note is that ARM provides an extra DSP instruction set, which is for example available in an ARM946. Using this core did not show a major speed improvement and resulted in an even higher power consumption.

5 Mapping a DDC on an FPGA

The commercial market is offering a wide range of DDC IP-cores that can be used on an FPGA [4], [6], [16]. Since these solutions are not open source and differ from the desired DDC, a custom implementation is made. This section starts with a short discussion about the FPGA used, followed by a discussion about the implementation and the performed power estimation.

5.1 Altera Cyclone FPGA

For the development of the DDC, the Altera Cyclone I [2] and Cyclone II [3] FPGAs are chosen. The Cyclone I is chosen because it is made with 0.13 μm technology, which compares with the technology of the other solutions. The Cyclone II is chosen because it is a state of the art FPGA that is energy efficient realized in 0.09 μm technology. Both FPGAs contain high densities of Logical Elements (LE), ranging from 2,910 to 20,060 LEs for the Cyclone I and from 4,608 to 68,416 LEs for the Cyclone II. The Cyclone I is equipped with 13 to 64 RAM blocks and the Cyclone II with 26 to 250. Each RAM block provides a storage space of 512 bytes. It is also possible to use a Phase Locked Loop

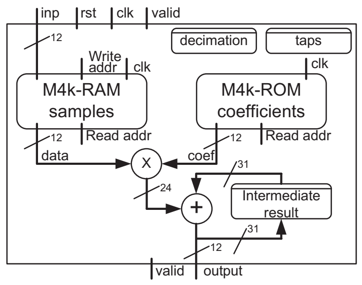

Figure 5. Schema of polyphase FIR

in both FPGAs. Beside this the Cyclone II is equiped with embedded 9-bit multipliers, ranging from 26 to 300.

5.2 DDC algorithm on an FPGA

The implementation corresponds with the DDC discussed in section 2. The DCC was aimed to fit in an Altera Cyclone I EP1C3T100C6 and an Altera Cyclone II EP2C5T144C6, which are both the smallest FPGAs in their series. The implementation is discussed in the following section. This section concludes with a power estimation.

5.2.1 The implementation

For the implementation, the DDC is divided into parts. The DDC consists of a NCO, CIC filters divided in integrating and comb parts and a polyphase FIR filter. The parts are interconnected with a data bus of 12 bits and an output data valid line. For the implementation the decimation values and the amount of taps are read from generics, so they can be altered at design time. The polyphase FIR is implemented with 124 taps, this is done to make the sequential filter run a little more efficiently.

The NCO and the CIC filters have to work at the input sample rate, so they are implemented as explained in section 2. The integrating part of the CIC filter has a counter to register the number of processed inputs. If this part should deliver a value to the comb part, it makes its output valid signal high for one clock cycle. The comb component reads the signal and processes it. This way the comb part of the CIC filters receives decimated information.

The polyphase FIR, however, is receiving information at 192kHz and delivering results at 24kHz. It has been decided to implement the filter as a sequential algorithm. The other option would have been in parallel at a lower clock frequency. This would require a lot of extra hardware that would be idle most of the time. The sequential implementation makes the logic cells run at the full clock speed of 64.512 MHz. The input sample rate of 192ksps and a decimation factor of 8, results in 2688 clock cycles to calculate one single output sample.

	Cyclone I EP1C3T100C6	**Cyclone II EP2C5T144C6**
Total logic elements	1,656 / 2,910 (56 %)	906 / 4,608 (20 %)
Total pins	41 / 65 (63 %)	41 / 89 (46 %)
Total memory bits	6 780 / 59,904 (12 %)	7,686 / 119,808 (6 %)
Embedded 9-bit Multiplier	0 / 0 (0 %)	8 / 26 (30 %)
Total PLLs	0 / 1 (0 %)	0 / 2 (0 %)

Table 4. Synthesis results for Cyclone I and II

Figure 5 depicts the schematic drawing of the polyphase FIR filter. The filter has a RAM block to store the previous inputs and a ROM from which the coefficients can be read. When valid, the new input (inp) is stored at the correct position in the RAM. The filter calculates its result, once it has received D samples from the CIC^5, where D is the decimation rate of the FIR filter. For the 124 taps, this is done in 125 clock cycles. Every cycle a coefficient and the corresponding input are read from the ROM and the RAM. These values are multiplied with each other and the result is added to the intermediate result. When all inputs are processed, the result is delivered on the output and valid becomes active for one clock cycle.

For the internal values in the polyphase FIR filter, the bus size is chosen in such a way that overflow cannot occur. Since the output is 12-bit, the 31-bit intermediate result has to be quantized before the result can be delivered. The result consists of the 11 least significant bits of the intermediate result and a sign bit. In case of saturation, the maximum or the minimum value is returned.

The synthesis result for the DDC algorithm (both I and Q part) in the Altera Cyclone I and Cyclone II are shown in Table 4. The Cyclone I can perform the implementation at a maximum frequency of 66.08MHz, while the Cyclone II can reach 80.87MHz. The final implementation runs at 64.512 MHz, according to the reference design in section 2.

5.2.2 Power estimation

Altera provides the program Quartus II [1] and its "PowerPlay Power Analysis" tool. With these a synthesis of the design and a power estimation of the synthesized design is performed. The amount of bit toggles of the input and inside the FPGA determine the amount of energy used. Because no real input data is available, bit toggling percentages at the input and internal in the chip are used.

The used input bit toggling is 50%, which corresponds to random data. Furthermore, we assumed an internal toggle rate of 10% for both FPGAs. For the Cyclone II, "PowerPlay Power Analysis" estimates the total power consumption is 57.98 mW, which consists of 26.86 mW static and 31.11 mW dynamic power dissipation. The estimations for the Cyclone I are done with several settings for the internal bit toggling, see Table 5. For stimuli with a toggle rate of 50% and an internal toggle rate of 10% the Cyclone I

Internal toggle rate	5%	10%	50%	87.5%
Total Thermal Power Dissipation	120.9 mW	141.4 mW	305.3 mW	458.9 mW
Dynamic Thermal Power Dissipation	72.9 mW	93.4 mW	257.2 mW	410.8 mW
Static Thermal Power Dissipation	48.0 mW	48.0 mW	48.0 mW	48.0 mW

Table 5. Power consumption of Cyclone I (input toggle rate is 50%)

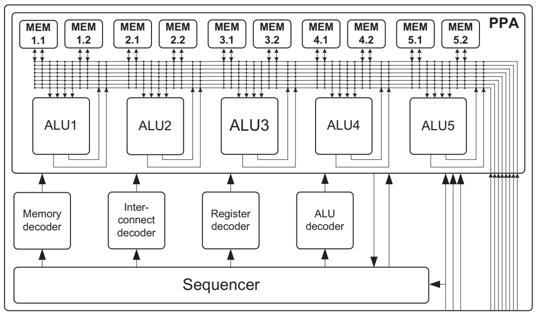

Figure 6. Montium tile processor

consumes around 141.4 mW in regular use.

6 Mapping a DDC on one Montium TP

6.1 The architecture of a Montium TP

The Montium Tile Processor (TP) is a part of a tiled System-On-Chip (SOC). The Montium TP has its roots at the University of Twente and is now commercially available through Recore Systems [2]. A Montium TP is designed to be small, fast and coarse grained reconfigurable [12]. Because a Montium TP can operate independently and communicate with other tiles, additional performance can be gained by adding more Montium tiles to a chip.

Figure 6 depicts the Montium TP with its three major parts: the sequencer, the decoders and the Processing Part Array (PPA). The sequencer implements a state machine. It takes an instruction and sends it to the decoders that give the PPA their configuration parameters. By walking through the instructions as a state machine, the sequencer can perform algorithms. When the decoders receive their parameters they can activate their part of the PPA. The sequencer also controls the PPA, by activating it or by stalling it. The PPA consists of five ALUs, which each has two (small) memories. The memories can be loaded with external data. The connections between the memories and the ALUs can be configured by the interconnect decoder.

The ALU of the Montium is presented in Figure 7. It has four 16-bit inputs, one 17-bit east input, one 17-bit west output and two 16-bit outputs. An ALU can send a result to the ALU west of it via its 17-bit west output. The neighbour ALU receives this value on its 17-bit east input.

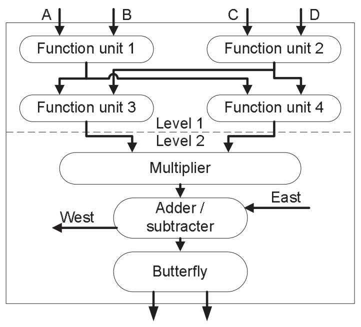

Figure 7. ALU as used in a Montium tile

The ALU is separated in two levels (see Figure 7). Level one has four function units. These can perform logic operations on the incoming data. Examples of these logic operations are ”or”, ”and”, ”not” and ”addition”. At level two, a multiplication can be done with the input values of the ALU or with the output values of level one. The addition/subtraction at level two can choose its input values from the east input, function units three and four, ALU inputs or the multiplier. The last functionality of level two is the butterfly structure. This structure provides the possibility to perform an addition and/or a subtraction of the inputs of the ALU with the result from the adder/substraction.

The Montium TP can perform many combinations of operations in parallel with its five ALUs. Each ALU can perform multiple non-multiply operations and one multiplication in one clock cycle. This makes the Montium an good architecture to for 16-bit digital signal processing algorithms. The possibility to reconfigure the chip makes it flexible, so it can perform a variety of algorithms. The Montium is not as flexible in performing its algorithms as a GPP, but has a higher performance and lower energy consumption in its algorithm domain. When the configuration is not altered every cycle, the architecture has an energy-efficiency close to an ASIC. The possibility to add more Montium tile processors to the chip, to increase the performance, makes it a scalable architecture.

6.2 DDC algorithm on a Montium TP

The Montium TP is programmed using an intermediate level programming language. Implementing the discussed DDC in this intermediate programming language requires knowledge of the Montium TP architecture. The combination of this knowledge and the intermediate level programming language allows an optimal solution to be implemented.

6.2.1 Implementation

The most intensive parts of the DDC implementation are the NCO and the integrating part of the CIC^2. The input signal

Algorithm part	#ALUs	Percentage of time on ALUs
NCO + CIC^2 integrating	3	100%
CIC^2 cascading	2	6.3%
CIC^5 integrating	2	25%
CIC^5 cascading	2	0.9%
FIR^{125}	2	0.5%

Table 6. DDC algorithm on a Montium

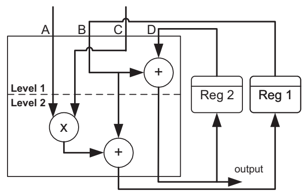

Figure 8. NCO and CIC^2 on a Montium TP ALU

needs to be multiplied with the cosine and sine signal and integrated according the CIC^2 algorithm, for both I and Q part of the algorithm. For this reason two ALUs (one for I and one for Q) in the Montium TP are needed to perform this part of the algorithm at 64.512 MSPS (see Table 6), which leads to a clock frequency of 64.512 MHz. Because the Montium allows only one clock speed for the tile, all ALUs will work at this clock speed.

Figure 8 depicts the configuration of one NCO-CIC ALU. The input signal is placed on input A and the sine or cosine value on input C. The values for the sine and cosine are stored in the local memories, so every clock cycle the values are fetched from a Look-Up Table (LUT). The address for the LUT is generated in an extra ALU, which enables to change the frequency during execution. After the multiplication, the value is integrated twice. The first integration is done with the value from register 1 (Reg 1) via input B. This value is added to the result of the multiplication. The result is written back to the register for the integration in the next clock cycle. The addition takes place in the adder of level two. The second integration of the CIC^2 takes place in level one of the ALU. The result is stored in Reg 2. Both registers (Reg 1 and 2) are mapped on the local register files of the ALU.

The above described part of the DDC requires three ALUs, which leaves two remaining ALUs for the rest of the algorithm. Because that part of the algorithm is not required to run at the input sample rate we are able to time-multiplex the successive filters (CIC^2 (comb-part), CIC^5 and FIR) on two ALUs.

The comb-part of the CIC^2 filter requires two ALUs for one clock cycle per complex sample. The delayed input is stored in the register files and the subtractions are performed in both level 1 and 2 of the ALU. The output of the CIC^2 filter is integrated with the CIC^5 filter that requires two ALUs for four clock cycles. The intermediate results are stored in the register files. Due to the decimation with 16, these five

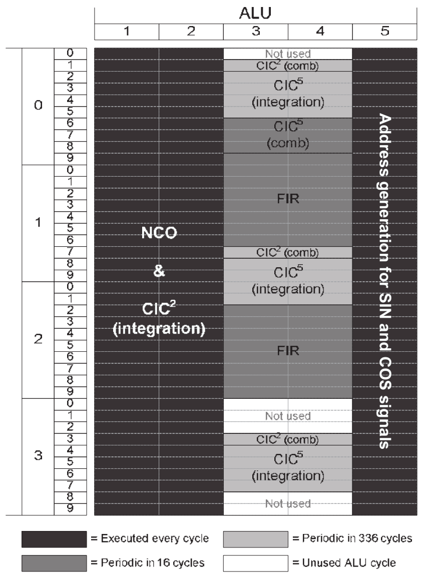

Figure 9. First 40 clock cycles of the DDC

cycles only have to be executed once every 16 clock cycles.

The remaining $\frac{11}{16}$ part of the clock cycles of the two ALUs is used for the comb-part of the CIC^5 filter and the polyphase FIR filter. The CIC^5 part is implemented similar to the CIC^2 filter and requires 3 clock cycles every 16*21 = 336 cycles. The polyphase FIR filter is a specialization of a general FIR filter as discussed in [12]. Due to the last decimation step of 8, the output of the CIC^5 only needs a multiplication with $\lceil \frac{125}{8} \rceil$ out of the 125 filter-coefficients. The intermediate results are stored in the local memories and summed with future intermediate results. Once every 2668 clock cycles a sum of intermediate results is presented at the output of the tile processor.

Figure 9 depicts the first 40 clock cycles of the DDC implementation, operated at 64.512 MHz. In this figure it is visible that the NCO and address generation use three ALUs. The comb part of CIC^2 filter is repeated every 16 cycles. The period of the comb part CIC^5 filter and the polyphase FIR cycles is repeated every 336 cycles, which is not visible in this figure. Table 6 shows the percentages of cycles and ALUs that are used for the current implementation. The implementation compiles to a configuration file of 1110 bytes.

6.2.2 Power estimation

The power consumption of the Montium is measured to be 0.6 mW/MHz [12] in 0.13 μm technology and a V_{dd} of 1.2 V. This implementation of the Montium has a core size of 2.2 mm^2. Using this, we can estimate that a Montium TP needs 38.7 mW to perform the DDC algorithm.

Solution	Size	Freq[MHz]	V_{dd}	Power	Area
TI GC4016	0.25μm	80.0	2.5	115.0 mW	n.a.
	0.13μm (estimated)	80.0	1.2	13.8 mW	n.a.
Customised Low Power DDC	0.18μm	64.512	1.8	27.0 mW	17mm^2
	0.13μm (estimated)	64.512	1.2	8.7 mW	n.a.
ARM922T	0.13μm	6697.0	1.08	2.435 W	3.2mm^2
Altera Cyclone I	0.13μm	64.512	1.5	93.4 mW	n.a.
Altera Cyclone II	0.09μm	64.512	1.2	31.11 mW	n.a.
	0.13μm (estimated)	64.512	1.2	44.94 mW	n.a.
Montium TP	0.13μm	64.512	1.2	38.7 mW	2.2mm^2

Table 7. Summary of results

7 Conclusion

In this report, five architectures and their implementations of the DDC have been highlighted. As noted in the description of the architectures, the flexibility and the energy consumptions of the architectures differ considerably. The implementations are compared on their energy consumption. The results are listed in Table 7. Depending on the scenario we propose to use different architectures, based on their energy consumption.

7.1 Static scenario

Devices that need to perform full-time wireless communication need a small and energy-efficient solution. An ASIC solution seems optimal for this situation. In the discussed examples this would be the customised low power DDC, which uses only 27 mW. When scaling the technology to 0.13 μm this would even drop to 8.7 mW.

When the device needs multiple dedicated DDCs, the GC4016 ASIC of Texas Instruments could also be an option. This ASIC consumes roughly four times more energy compared to the customised low power DDC solution and performs a slightly different algorithm. However, the solution has four DDC channels available on one chip. When this solution is scaled to 0.13 μm, it would only use 13.8 mW.

7.2 Reconfigurable scenario

As expected, an FPGA consumes more energy compared to the ASIC solutions. The Cyclone I and Cyclone II solution are interesting for rapid prototyping, to find design errors. The solution would consume too much energy to be applied as a static scenario in a radio or a mobile phone.

When a device does not need to perform full-time wireless communication, the DDC algorithm would only be active a part of the time. During the in-active periods the architecture could provide other functionality to the user and the device. In this case a dynamic reconfigurable architecture can be considered. The best performing architecture at the reconfigurable area is the Altera Cyclone II due to its smaller technology size. When all architectures are scaled to 0.13 μm the Montium has the lowest power consumption of 38.7 mW.

Acknowledgement

This research is conducted within the Smart Chips for Smart Surroundings project (IST-001908) supported by the Sixth Framework Programme of the European Community.

References

[1] Altera, http://www.altera.com. *Quartus II*. Version : 5.0 Build 148.

[2] Altera Corporation. *Cyclone I Device Handbook, Volume 1*, 2005.

[3] Altera Corporation. *Cyclone II Device Handbook, Volume 1*, 2005.

[4] R. Andraka. High performance digital down-converters for fpgas. *Xilinx Xcell*, 4(38):48–51, 2000.

[5] ARM, Inc., http://www.arm.com. *ARM922T*, 2005.

[6] B. K. C. Court, B. Row. *Digital Down Converter Demo/Framework for HERON modules with FPGA*. Hunt Engineering, 2004. v1.1.

[7] M. Donadio. *CIC Filter Introduction*. www.dspguru.com/info/tutor/cic.htm, 2000.

[8] Elixent. *Digital Down Conversion*, November 2003. AN0012, v1.1.

[9] European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France. *Radio broadcasting systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers*, May 1997. ETSI ETS 300 401.

[10] European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France. *Digital Radio Mondiale (DRM); System Specification*, April 2003. ETSI ES 201 980.

[11] C. H. Frank Hofmann and W. Schafer. Digital radio mondiale (drm) digital sound broadcasting in the am bands. *IEEE Transactions on Breadcasting*, 49(3):319–328, September 2003.

[12] P. M. Heysters. *Coarse-grained reconfigurable processors*. Ctit ph.d. thesis series no. 04-66, University of Twente, 2004. ISBN: 90-365-2076-2.

[13] T. Hollis and R. Weir. The theory of digital down conversion. Technical report, Hunt Engineering, 2003. v1.2.

[14] Committee on Networked Systems of Embedded Computers. *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*. National Academy Press, 2001. ISBN: 0-3090-7568-8.

[15] Personal communication.

[16] A. Rudra. Fpga-based applications for software radio. *RFDESIGN*, pages 24–35, May 2004.

[17] Texas Instruments. *GC4016 multi-standard quad DDC chip, datasheet*, 2001. v1.