An Application-Specific Processor Hard Macro for Real-time Control

Xiaofeng Wu, Vassilios Chouliaras and Roger Goodall
Department of Electronic and Electrical Engineering, Loughborough University
Loughborough, Leicestershire, UK LE11 3TU

Abstract—This paper presents an application-specific processor architecture for real-time control applications. The processor is specially designed for 1-bit processing, which is a new technique in real-time control. The targeted processor is low-power, small and fast, and has been implemented as a hard macro in a high-performance silicon process.

I. INTRODUCTION

In the area of real-time control, there is always a particular requirement of small and fast microprocessor for implementing control applications. The main reasons are that, on the one hand, many physical systems are becoming more and more compact; on the other hand, control laws are becoming more and more complex. These trends mean that a processor has to be small and at the same time, clock at very high speed to be able to calculate possibly hundreds of equations in one sampling cycle (less than 1ms in many cases). Much of the previous work takes standard processor devices to accommodate control laws by control engineers. Much of the hardware developed to date, however, has tended to be a general-purpose microprocessor or digital signal processor (DSP) with little analysis of the underlying requirements of control systems. From the prospect of control, the solution is to adopt a system approach and bring together the control and electronic design requirements. This is related to hardware and software co-design, a major subject which has seen substantial research progress over the last few years [2]. This paper, however, proposes an application-specific processor architecture to satisfy the challenge of space and speed for embedded real-time control, and will give a brief description of a new control concept at the beginning.

The remainder is organised as follows. Section II describes the concept of 1-bit processing, which shows great advantages over conventional designs. Section III details the application-specific processor architecture and benchmarks with other standard processors given a 4th order controller. Section IV gives a 4th order low pass filter and its simulation results. Section V concludes.

II. CONTROL ALGORITHM

Many analogue-to-digital and digital-to-analogue converters use ∆Σ modulation to convert signals into a simple digital form for high quality conversion. This simple digital signal can be stored in a 1-bit register. A multibit representation of the signal is achieved with a decimating filter for analogue-to-digital converters. This multibit signal is studied by control engineers and controlled by many control strategies. Differently, 1-bit processing manipulates that simple digital signal directly with the purpose of reducing the silicon area and achieving very high sampling rate.

A. ∆Σ modulation

∆Σ modulation relies on a fast sampling frequency to obtain a high precision after decoding. Fig. 1 shows a 2nd order ∆Σ modulator developed by Ritchie [8], in which several integrators are cascaded in the forward loop to create a higher order filter, with each integrator receiving an additional input from the quantiser. The output of the quantiser is a binary value, ±1, hence it can be stored in a 1-bit register. For decoding, decimating is required. The multi-bit format of the input, \( U \), is calculated as:

\[
\hat{U} = \frac{1}{N} \sum_{i=1}^{N} q(x_n),
\]

where \( N \) is the number of samples, \( x_n \) is the last integrator’s output and \( q(x_n) \) is the output of the 1-bit quantiser. If \( x_n \) is over 0, \( q(x_n) \) is +1. If \( x_n \) is below 0, then \( q(x_n) \) is −1.

However, the decimator is removed in 1-bit processing. \( \hat{U} \) is an average value of \( q(x_n) \) over \( N \) samples. So, in fact, each sample has its corresponding part on the original input but with an error. The truth is that 1-bit signals contain all the useful information of the input [1], although this information is buried within the quantization noise. However, when the sampling frequency is high enough, this quantization noise can be ignored as the noise spectra within the signal bandwidth is much smaller than the input signal spectra.

B. 1-bit processing

1-bit processing for control is shown in Fig. 2. The analogue signal is encoded into its corresponding 1-bit format by passing a simple analogue ∆Σ modulator. The output of the controller is in multi-bit format, hence we place a digital
The principles of sampling are more detailed in [9], and are rates much higher considering the signal-to-noise issues. With 1-bit processing, however, sample delays [5]. With 1-bit processing, however, sample delays are much higher considering the signal-to-noise issues. The principles of sampling are more detailed in [9], and are not presented in detail in this paper.

III. SYSTEM-ON-CHIP DESIGN

A. Numerical representation

The signal range requirements are usually modest in well-designed digital control algorithms and floating-point arithmetic is usually expensive in terms of power consumption, execution time and complexity. Thus the $\Delta \Sigma$-CSP, $\Delta \Sigma$-based

Control System Processor, adopts fixed-point word format. For 1-bit processing, as the sampling frequency is very high, it results in long word length for both coefficients and state variables. So the word length needs to be carefully chosen to ensure that the full value and dynamic range of the variables involved in the calculation can be accommodated.

A simple criterion used to determine the number of fraction bits is described in [3]. A reasonable number of fraction bits would be in the range of 8-16 bits, which will support a wide range of controllers. Fig. 4 shows a general format for the coefficients and state variables. This format accommodates a signal with an amplitude between -128 to 128, which is sufficient for most control applications considering the input is only -1 or 1. There are no overflow or underflow bits specified as we believe they are unnecessary in 1-bit processing.

B. Processor architecture

The control design is reasonably simple. Fig. 5 shows a block diagram of the $\Delta \Sigma$-CSP. The ALU takes three inputs from I/O, accumulator and data RAM respectively. A special conditional negator and accumulator (CNA) is designed. This CNA calculates $D = A \times B + C$ in one clock cycle and writes the result back to the accumulator. Here the symbol ‘*$’ means a conditional negation, not a multiplication. $A$ is the I/O input in 1-bit format, $B$ is the coefficient and $C, D$ are the state variables, all in 24-bit fixed-point format. $D$ is stored in the accumulator in every cycle, ready for the next calculation. The system is fully pipelined such that there is a latency between instruction issues, with the result being written back to the accumulator.

The processor has only four instructions (see Table I). Each opcode is specified in 16-bit format as shown in Fig. 6. Except $C$, both $A$ and $B$ are indicated by the source address. There
TABLE I

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLT</td>
<td>0 → D</td>
<td>Wait and clear accumulator</td>
</tr>
<tr>
<td>CAS</td>
<td>A * B + C → D</td>
<td>Conditional negator and accumulator or Signed right shift</td>
</tr>
<tr>
<td>READ</td>
<td>B → D</td>
<td>Read the data from data ROM</td>
</tr>
<tr>
<td>WRT</td>
<td>C → D</td>
<td>Write the data</td>
</tr>
</tbody>
</table>

Fig. 6. Operation code format.

are four digital inputs and four digital outputs in the I/O block. The data address uses 10 bits of the operation code, which allows access to a $1k \times 24b$ RAM. The CAS instruction comprises the CNA and shift together. The 14th bit of the operation code decides which function is used. In this design, 1 is shift operation, and 0 is CNA. CNA can also execute an add operation by taking $A$ as 1. The constant 1 is stored in the I/O register. There is no left shift operation in the processor because in practice the scaling coefficient $k$ in Fig. 3 is always fractional. The WRT operation has four functions depending on the ‘write mode’ (see Table II). When programming, these functions use keywords ‘RAM’, ‘IO’, ‘PC’ and ‘TIMER’ to distinguish them. For example we use ‘WRT TIMER’ to set the timer.

The program counter contains an initial address, which indicates the control loop begins. It starts from 0 and keeps increasing until the instruction ‘HLT’ is read. At this point, the program counter jumps to the initial address. The program counter waits here for the sample clock to trigger. Then the control loop begins, and the program counter keeps increasing till the instruction ‘HLT’ is read again.

The timer can be configured to generate the sampling frequency. It reads an initial value from the RAM. The input clock cycles are counted from 0 using a counter. The sample clock keeps low till the counter’s value reaches to the timer initial value. Then the sample clock goes to high, and the counter is reset to 0. The sample clock stays at high for 1 clock cycle, and it returns to low when a new loop begins.

The program, coefficients and initial data are stored in Flash ROMs allowing reprogramming of the controller. Two ROM blocks are used in the ΔΣ-CSP. One stores the program, and is called program ROM. The other stores the coefficients and initial data, and is called data ROM. The program ROM is $1k \times 16b$, and the data ROM is $64 \times 24b$. A USB1.1 serial communication port is provided, which allows the ΔΣ-CSP to exchange data with the host computer. The communication scheme between the ΔΣ-CSP and the computer is shown in Fig. 7.

C. Complexity and speed

The processor is synthesized with Synplify ASIC 3.0.2 targeting the umc 0.13μm 8-metal process. It results in an overall cell count of fewer than 2,200 cells and a delay of less than 2ns. The size of the ΔΣ-CSP is $400 \times 220(\mu m^2)$. Fig. 8 shows the processor layout after Place & Route. It is likely to consume only a few milliwatts of power. As such the ΔΣ-CSP is a small and low-power core capable of implementing most real-time control laws.

The ΔΣ-CSP runs at over 500MHz. To benchmark this processor, we compared it with other processors, and the results are summarised in Table III. The benchmark is carried out by comparing the computation delays of a $4^{th}$ order Butterworth filter. It includes results for a 50MHz CSP, a 60MHz TMS320C31 DSP, a 160MHz TMS320C54 DSP, a 25MHz C167 microcontroller, a 233MHz Strong-ARM processor and a 500MHz Pentium III. The CSP [7] is also an application specific processor for real-time control, the difference being that it uses conventional bit-parallel arithmetic, hence multipliers are used. From Table III, ΔΣ-CSP is the fastest processor for real-time control.
Table 3  
**BENCHMARK.**

<table>
<thead>
<tr>
<th>Processor</th>
<th>1-BitP</th>
<th>CSP</th>
<th>TMS320C31</th>
<th>TMS320C54</th>
<th>C167</th>
<th>Strong-ARM</th>
<th>PentiumIII</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency(MHz)</td>
<td>300</td>
<td>50</td>
<td>60</td>
<td>160</td>
<td>25</td>
<td>233</td>
<td>300</td>
</tr>
<tr>
<td>Clock cycles/instruction</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1.49</td>
<td>3.34</td>
<td>1.79</td>
<td>1.13</td>
</tr>
<tr>
<td>Number of instructions</td>
<td>24</td>
<td>23</td>
<td>48</td>
<td>450</td>
<td>194</td>
<td>43</td>
<td>49</td>
</tr>
<tr>
<td>Computation time(µs)</td>
<td>0.048</td>
<td>0.46</td>
<td>1.603</td>
<td>4.190</td>
<td>25.92</td>
<td>0.331</td>
<td>0.113</td>
</tr>
<tr>
<td>Max. sampling(kHz)</td>
<td>20833</td>
<td>2175</td>
<td>623</td>
<td>238</td>
<td>38.5</td>
<td>3021</td>
<td>8823</td>
</tr>
<tr>
<td>Normalised time</td>
<td>1</td>
<td>0.98</td>
<td>33.40</td>
<td>87.29</td>
<td>540</td>
<td>6.90</td>
<td>2.35</td>
</tr>
</tbody>
</table>

![TABLE III](image1.png)

![Fig. 9](image2.png)

**IV. SIMULATION RESULTS**

The RTL simulation was done on Modelsim. We implemented a 4th order 1Hz Butterworth low pass filter with the processor. The transfer function of the filter is:

\[ H(s) = \frac{1}{(1 + 1.4\frac{s}{w} + \frac{s^2}{w^2})^2}, \]

where \( w = 2\pi \). The transfer function is converted into the modified canonic \( \delta \) form as shown in Fig. 3. With a sampling frequency of 1kHz, the coefficients are:

\[ p_0 = 0.0418, p_1 = p_2 = p_3 = p_4 = 0, \]
\[ q_0 = 0.0418, q_1 = 1.4565, q_2 = 2.5614, q_3 = 2.2519, \]
\[ k = 2^{-7}. \]

The result is obtained by applying a step input with an amplitude of 0.5. It is cross-referenced against the continuous output obtained by a Matlab program. Because the \( \Delta\Sigma \)-CSP only works on 1-bit signals, the step input is preprocessed by a 2nd order \( \Delta\Sigma \) modulator to create the input stream for the processor. The \( \Delta\Sigma \) modulator is modelled with a VHDL entity. The filter is programmed and compiled. Fig. 9 shows the program and the 16-bit operation codes. The operation codes are downloaded to the program ROM to execute the program.

As mentioned in Section II, 1-bit signals contain all the useful information covered by noises. Wavelets analysis [9] has been used to de-noise the 1-bit output using Matlab. Fig. 10 is the simulation results. Graph (a) is the step input after passing \( \Delta\Sigma \) modulator; Graph (b) is the processor output after executing the program; Graph (c) is the results of continuous output and de-noised processor output; Graph (d) shows the error between the continuous output and the processor output. The errors are less than 3%. They are mainly produced by the wavelet filter. This demonstrates that the \( \Delta\Sigma \)-CSP specification is correct and can be implemented in practical applications.

**V. CONCLUSION**

The \( \Delta\Sigma \)-CSP is an extremely small and fast application-specific processor. It is likely to cost less than $0.50 for manufacturing and consume only a few milliwatts of power. It is important to appreciate that, although the \( \Delta\Sigma \)-CSP outperforms even the fastest of the other processors by a significant margin, it is much more simple by not utilizing multipliers.

In the next stage, a Magnetic Levitation (MAGLEV) project will be undertaken, in which a magnetically-suspended vehicle will be controlled by the \( \Delta\Sigma \)-CSP. The MAGLEV controller is 46th order. This will verify the processor’s capability of handling very complex control systems further. At the same time, we are investigating the possibility of very high parallel processing through integrating hundreds of \( \Delta\Sigma \)-CSPs.
REFERENCES