HYBRID MULTIPLIER/CORDIC UNIT FOR ONLINE HANDWRITING RECOGNITION

Stephen McInerney, Richard B. Reilly
stephenm@faraday.ucd.ie, richard.reilly@ucd.ie

DSP Group, Dept. of Electronic and Electrical Engineering
University College Dublin
Belfield, Dublin 4, Ireland

ABSTRACT

Traditionally Online Handwriting Recognition (OHR) implementations use general-purpose processor architectures. The pre-processing step of OHR comprises regular array-based tasks such as normalisation, feature extraction and segmentation. Standard processor architectures cannot however efficiently support the varied arithmetic operations required by pre-processing. These tasks would seem ideally suited for custom hardware acceleration. CORDIC offers all the required elementary functions for pre-processing but is inefficient for linear mode operations (multiplication/division) due to its serial nature. A hybrid Multiplier/CORDIC architecture is proposed in which a fast iterative multiplier/MAC shares hardware with a serial CORDIC unit. This multiplier retires 6b/cycle with minor additional hardware requirements. This hybrid offers improved general performance for signal-processing applications and is targeted at the pre-processing task of OHR. Performance results are included.

http://wwwdsp.ucd.ie

1. ONLINE HANDWRITING RECOGNITION PRE-PROCESSING

The growth in use of Personal Digital Assistants generates a need for accurate OHR under significant constraints of processor die area, power and memory.

The OHR task itself involves pre-processing, character recognition and optional postprocessing steps. The preprocessing step involves normalisation, feature extraction and segmentation processes [1]. By implementing the pre-processing in hardware, potential resources on a PDA are released for other applications.

Normalisation aims to correct unwanted variations in the input. Typical processes are rotation, scaling, shear transform, deskewing, extrema location, centre-of-mass location, zone classification, smoothing, threshold-based anomaly exclusion, resampling in the time and/or spatial domain.

Feature extraction gives the recognizer its expected inputs in such formats as Euclidean distance, velocity or acceleration (requiring differentiation and integration), stroke or interstroke direction expressed either as slope, angle, cosine, sine or curvature, and measure of stroke curviness (stick, arc, curve).

Segmentation entails classifying a sequence of sample points into strokes and characters, and relating them by order, connectivity and significance.

Operations typically required by these pre-processing tasks [2]-[5] are listed below.

- MAC-based operations: differentiation/integration, filtering, resampling, scaling, shear transform, centre-of-mass location, dot product
- elementary functions sqrt, sin,cos, tan, sinh,cosh, exp,ln
- elementary transformations: rotation, Euclidean distance
- compound operations e.g. curvature, Hough transform
- comparison-based operations: thresholding, extrema finding, intersection checking

These operations map to general-purpose architectures with varying levels of efficiency.

Sample features were taken from the open-distribution UNIPEN project [6]. The *uptools 3.2* suite was profiled and produced the results in Table 1. Profiling results are expressed in terms of call count to basic functions for each coordinate pair. For illustration the following features were selected: Sc: cumulative displacement, COSF: cosine of running angle and Curv: curvature.

Feature name	*filter / differ*	*mean*	*polar*	*division*	*Euclidean distance*
Sc	**2**	**0**	**0**	**0**	**1**
COSF	**2**	**0**	**0**	**1**	**1**
Curv	**3**	**1**	**1**	**1**	**0**

Table 1 Profiled operation counts for sample features

The results indicate a mixture of multiply-accumulate-intensive tasks with elementary functions. This paper investigates implementations with a broad range of elementary function capabilities targeted at OHR pre-processing.

2. CORDIC APPROACHES

2.1 CORDIC Applicability to OHR

One algorithm which offers an unrivalled range of elementary function capabilities is the CORDIC (COordinate Rotation DIgital Computer) iterative algorithm [7]. This relies on circular and hyperbolic modes [8]. Traditionally CORDIC linear mode for multiplication and division has been neglected as it underperforms standard iterative and array algorithms [9] due to its high latency, low efficiency and limited accuracy.

Implementation parameters are dictated by the application, where the sampling rate of the pen input coordinates is typically 100Hz at resolutions of 200 points/inch [1]. These data values are fully representable by wordlengths of 12-18 bits.

Due to the limited throughput of this application, fixed-point serial approaches with wordlengths of up to 18 bits were selected.

2.2 CORDIC Circular & Hyperbolic Modes

CORDIC implementations exhibit a performance/die area tradeoff curve [10] and in practice one of the extremes is chosen: fully pipelined or serial.

The functionality required for CORDIC includes arithmetic units (X,Y,Z adder/subtractors), shift sequence parameters, Z-coefficient sets, double-iteration and scaling-iteration control flags.

Fully pipelined approaches [11] achieve high speed and area savings due to hardwiring of coefficient values into the arithmetic and shift units of each stage. Furthermore zero rotations may be allowed in later stages due to negligible scaling factor effect [12]; this allows higher-radix multiplicative termination stage(s).

Serial approaches require additional sign-extending barrel-shifters for X,Y values and a ROM and iteration control FSM for handling the coefficients. A serial CORDIC is shown in the diagram below.

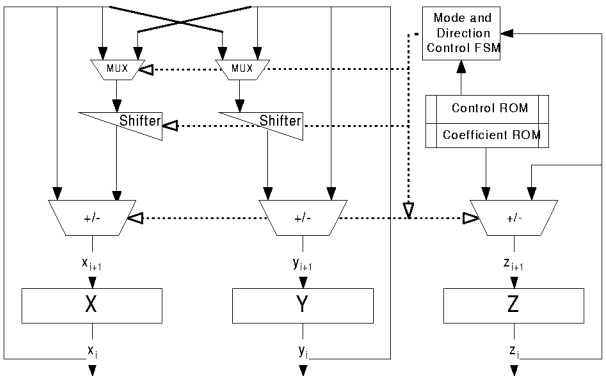

Figure 1 Serial CORDIC Unit

2.3 CORDIC Linear Mode

Linear-mode multiplication (vectoring) and division (rotation) require almost no additional hardware but have not been investigated in the literature due to their poor efficiency.

The CORDIC linear-mode multiplication/MAC algorithm is inefficient in its unmodified form for the following five reasons:

- The result $Y \rightarrow Y + X*Z$ is necessarily truncated to the width of the Y register.
- The algorithm is inexact. It is equivalent to a one-bit-serial signed-digit {1, 1}-recoded MSB-first algorithm. Zero rotations are not allowed, even though scaling factor is not an issue. This absence of zero rotations causes 1 LSB error for even multipliers.
- The inefficiency of the recoding also incurs unnecessary add/sub operations.
- The absence of zero rotations also causes the multiplier to be extended to n bits, requiring leading sign-bit-detection if early termination is desired.
- The Z-unit is inefficiently utilised to perform a simple 1b right-shift operation on the multiplier.

Similarly the unmodified linear-mode division algorithm is inefficient. Both operations are suitable for improvement. This paper focuses on improving multiplication.

3. MULTIPLIER/CORDIC DESIGN

A basic principle is that architectural modifications should justify their additional cost by their performance benefit. Hence due to the dominance of multiply/MAC operations in the OHR feature extraction application, a serial CORDIC unit was chosen for minimal area.

It is possible to reuse the CORDIC logic to implement a fast iterative multiplier structure. This gives a significant improvement on CORDIC performance with limited additional logic requirement. Furthermore, the full precision result can be generated.

The proposed Multiplier/CORDIC unit is shown in Figure 2. It is based on a standard serial fixed-point CORDIC unit using 18b precision (1b overflow, 12b value, 5b guard bits). An iterative triple-radix-4 Booth-recoded multiplier/MAC is formed reusing the X,Y,Z adder/subtractors and shifters to retire 6b/cycle.

The ROM requirement is 28 x 20b by using a heuristic for generating optimal-size coefficient sets [13].

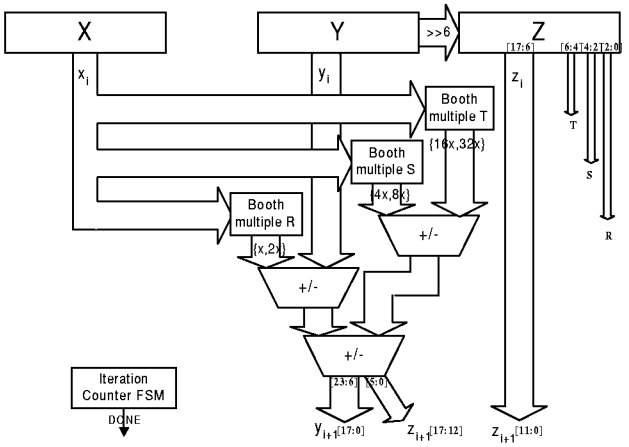

Figure 2 Hybrid Multiplier/CORDIC Unit

The additional hardware requirement of the hybrid design is the three modified radix-4 Booth multiple generators and the datapath multiplexers (not shown) for hardware-sharing. The CORDIC barrel-shifters are not used in multiplier mode.

The register assignments are similar to CORDIC linear mode, except the result is maintained to full precision. The multiplier is stored in Z, the multiplicand is stored in X, and the intermediate result is stored in {Y,Z}. 6b of multiplier are retired per cycle by shifting {Y,Z} right and loading the intermediate result into Z. For MAC operations an 18b accumulate value is loaded into Y; for multiplies Y is reset to zero.

Each of the three Booth stages retires 2b/cycle. Each generator encodes a three-bit slice of Z to produce the required multiple {0,X,2X} with sign bit. Shifting then produces the required multiples up to 32X. These are summed in the add/sub tree.

Precision of multiplicand X is limited to 17b so that the 2X multiple can use the 18b add/sub block. Precision of the final add/sub in the tree must be extended to be 20b.

Precision of multiplier Z is only 17b due to the Booth encoding requirement that the LSB of Z be zero. Additionally in MAC mode the control FSM must be modified to prevent the LSB of the product being scanned as if it was a spurious bit 18 of the multiplier.

4. RESULTS AND EVALUATION

4.1 Multiply/MAC performance

Table 2 below illustrates the sixfold multiply/MAC latency decrease of the proposed design compared to the standard CORDIC architecture. Full precision support is shown.

Architecture	Latency (cycles)	Precision
CORDIC	**18**	**18b x 18b → 17/18b (*)**
M/CORDIC	**3**	**17b x 18b + 18b → 35b**
Array MAC	**1**	**18b x 18b + 18b → 36b**

Table 2 Comparison of algorithm performance
(*) LSB is in error for even multipliers

4.2 Implementation results

The overall design was implemented in VHDL and synthesised with Synopsys 98.02 to a 0.6μm process (Toshiba TC6A library). Representative results derived from prelayout synthesis values are presented in Table 3 for comparison.

Architecture	Area (μm^2)	Cycle time (ns)
CORDIC (serial)	**7,150 (*)**	**19.47**
M/CORDIC	**10,300 (*)**	**22.89**
Array MAC	**16,570**	**17.36**

Table 3 Implementation results for algorithms
(*) Area of 28 x 20b ROM not included.

The speed and area penalty of the M/CORDIC design are 18% and 44% compared to a standard serial CORDIC. If the ROM was included the true area penalty would be significantly lower.

Compared to a full-precision array Multiply-Accumulate design, the Multiplier/CORDIC is 32% slower but 38% smaller (excluding ROM).

5. DISCUSSION

The results of the proposed hybrid represent a significant improvement on standard architectures while yielding compatible implementation parameters and acceptable support for the target application. The implementation penalty for this multiplexed functionality is shown to be acceptable.

Of the three designs compared, the Multiplier/CORDIC hybrid is a significant improvement on the linear-mode performance of a standard CORDIC. Although an array MAC has the lowest multiply latency (one cycle), it offers no capability for other signal-processing functionality such as described in Section 1 above. Hence the Multiplier/CORDIC presented offers a superior architecture for OHR pre-processing.

The optimal implementation was achieved by addressing the tradeoff of speed vs. area. The cycle time of a serial CORDIC is determined by the slower of two path groups: the Z-unit path through the ROM and add/sub unit, and the X/Y-unit arithmetic paths. The Multiplier/CORDIC has an additional timing path in the multiplier passing through a Booth encoder and the tree of carry-propagating add/sub blocks.

These three path groups can be balanced for optimal Multiplier/CORDIC implementation. For the range of implementations obtained for this design, the CORDIC Z-path was critical and hence the speed overhead of the M/CORDIC approach reduced to the delays in the hardware-sharing multiplexers. This limits the attractiveness of applying carry-save arithmetic as it would only speed up the non-critical multiplier path.

Implementation of division and of early termination in multiplication are for future investigation. A serial multiplier can be enhanced with early termination functionality by adding extra leading-digit detection and barrel-shift capability [14].

The Multiplier/CORDIC module described could be integrated to a standard CPU by an on-chip bus. A system architecture including a small coordinate RAM would allow single-chip hardware support for OHR signal processing in a PDA-type design. Programmability could be implemented either by a program decoder or ROM microcode.

6. CONCLUSION

A Multiplier/CORDIC architecture has been shown to offer an improvement on the general-purpose signal processing utility of a CORDIC unit, while simultaneously giving greater functionality than a MAC unit. This represents a more attractive platform for OHR pre-processing operations.

ACKNOWLEDGEMENT

The authors acknowledge their indebtedness to the UNIPEN group at NICI, University of Nijmegen [6], and to Silicon and Software Systems (S3), Dublin. This work was supported under the Forbairt Strategic Research Programme as ST/98/023.

7. REFERENCES

[1] C. Tappert, C.-Y. Suen and T. Wakahara, "The State of the Art in On-line Handwriting Recognition," *IEEE Trans PAMI*, vol. 12, pp. 787-808, 1990.

[2] W. Guerfali and R. Plamondon, "Normalizing and Restoring On-line Handwriting," *Pattern Recognition*, vol. 26, pp. 419-431, 1993.

[3] P. Morasso, L. Barberis, S. Pagliano and D. Vergano, "Recognition experiments of cursive dynamic handwriting with self-organizing networks," *Pattern Recognition*, vol. 26, pp. 451-460, 1993.

[4] L. Schomaker, "Using stroke- or character-based self-organizing maps in the recognition of on-line, connected cursive script," *Pattern Recognition*, vol. 26, pp. 443-450, 1993.

[5] J. Hu, M. K. Brown and W. Turin, "Invariant features for HMM based handwriting recognition," pp. 588-593, *Proc. ICIAP*, 1995.

[6] http://hwr.nici.kun.nl/unipen/

[7] J. Volder, "The CORDIC Trigonometric Computing Technique" *IRE Trans Electronic Computers*, vol. 8, pp. 330-334, 1959.

[8] J. Walther, "A Unified Algorithm for Elementary Functions" *Spring Joint Computer Conf. Proc.* vol. 2, pp. 927-930, 1971.

[9] I. Koren, *"Computer Arithmetic Algorithms,"* Prentice-Hall, 1993.

[10] S. Wang and V. Piuri. "A Unified View of CORDIC Processor Design," *Application Specific Processors*, ed. E Swartzlander, pp. 121-160, 1996.

[11] D. Timmerman, B. Rix, H. Hahn and B. Hosticka, "A CMOS Floating-Point Vector-Arithmetic Unit," *IEEE J-SSC*, vol. 29, pp. 634-639, 1994.

[12] S. Wang, V. Piuri and E. Swartzlander, "Hybrid CORDIC Algorithms," *IEEE Transactions on Computers*, vol. 46, pp. 1202-1207, 1997.

[13] D. König and J. Böhme. "Optimizing the CORDIC Algorithm for Processors with Pipeline Architecture," *Signal Processing V : Theories and Applications*, ed. L Torres et al., pp. 1391-1394, 1990.

[14] US Patent #5557563, Advanced RISC Machines Ltd., 1996.