Low Delay-Power Product CMOS Design Using One-Hot Residue Coding

William A. Chren, Jr., Member IEEE
NASA Lewis Research Center
Grand Valley State University

***Abstract:* CMOS implementations of arithmetic units for One-Hot Residue encoded operands are presented. They are shown to reduce the delay-power product of conventional, fully-encoded designs by more than 85%, as exemplified by the design of a direct digital frequency synthesizer for frequency-hopped spread spectrum communication systems. The reduction is attributable to the one-hot representation, which decreases the activity factors of the signals and the number of circuit and critical path transistors.**

I. Introduction

The contributions of this paper are twofold. First, CMOS arithmetic circuits for One-Hot Residue (OHR) operands are presented. OHR is an encoding technique for Residue Number System (RNS) operands in which the residue digits are one-hot encoded. The simplified structure and lower activity factors of these circuits yield superior delay-power products when compared with fully-encoded units. The circuits presented include an adder/subtracter, multiplier and scaler.

Second, the delay-power benefits are demonstrated for a direct digital frequency synthesizer (DDFS) design which exhibits a delay-power product that is at least 85% less than that of a recently developed, fully-encoded design called the High-Agility Direct Synthesizer (HADS).

This paper is organized as follows. Section II is a presentation of background information on the RNS, One-Hot Residue (OHR) encoding and DDFS. Section III presents the OHR arithmetic circuits and the example synthesizer design. Section IV presents its delay-power product estimate and compares it with that of the HADS. Section V contains our conclusions.

II. Background

The RNS represents an integer X as the vector of its residues modulo a fixed, specially chosen set of integers called moduli. Letting m_i denote the ith modulus, this is depicted as $X \Leftrightarrow (x_1, x_2, \ldots, x_n)$, where $x_i = |X|_{m_i} \triangleq X$ modulo m_i. The operations of addition, subtraction and multiplication are performed in "digit-parallel" fashion, modulo m_i. If operands X and Y have residue representations $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$, and $Z \triangleq X * Y$ where $*$ represents any of the operations, then we have $Z \Leftrightarrow (|x_1 * y_1|_{m_1}, |x_2 * y_2|_{m_2}, \ldots, |x_n * y_n|_{m_n})$. Because these operations are done in parallel, modulo small integers m_i, they can be performed quickly. The division operation, however, is relatively slow [1].

A useful property of prime moduli is that they possess at least one primitive root. A primitive root is an integer α whose successive powers equal the nonzero integers modulo m_i. This allows the use of logarithm-like operations for multiplication. If modulus m_i has primitive root α, then for any $1 \le x \le m_i - 1$, $x = \alpha^k$ modulo m_i for some $0 \le k \le m_i - 2$. It is then said that x has index k. Multiplication can be performed by addition modulo m_i -1 of indices. Section III shows a simple way of finding the index of a residue digit in the OHR representation.

A residue operation called Scaling is useful for frequency synthesis and is the equivalent of "right-shifting" (truncation) in the binary number system. Scaling by modulus m_i consists of a subtraction of x_i and multiplication by m_i^{-1} in each modulus except the ith, that is $\left\lfloor \frac{X}{m_i} \right\rfloor \Leftrightarrow (|m_i^{-1}(x_1 - x_i)|_{m_1}, |m_i^{-1}(x_2 - x_i)|_{m_2}, \ldots, |m_i^{-1}(x_n - x_i)|_{m_n})$. Scaling by a product of moduli is done by successive application of the above formula. The result is encoded only in those moduli not used for scaling. An operation called Base Extension [2] is used to restore the lost digits. It consists of successive scalings by each of the remaining moduli (with the lost digits initialized to zero), followed by a final multiplication by the additive inverse of the product of these moduli. Base Extension is not needed for frequency synthesis, and will not be discussed further. Note that the Scaling operation requires some method of converting a residue digit from one modulus to another. It will be seen in Section III that the OHR encoding provides a simple and fast way of doing this conversion.

The OHR representation is the representation of RNS operands using one-hot encoding. It allows implementation of arithmetic circuits with lower delay-power product than conventional RNS designs employing positional binary codes. These benefits result because all major operations (addition, subtraction, multiplication, modulus conversion, scaling and index computation) are performed using transposition of lines and barrel shifters. Fewer transistors are used than in gate-intensive architectures, leading to lower power and shorter critical path length. Furthermore, OHR circuits have a lower (and constant) activity factor, and possess a critical path delay which does not depend on the size of the operands.

The OHR representation for the ith residue digit x_i is depicted in Figure 1. Only the single line corresponding to the digit value is asserted at any time.

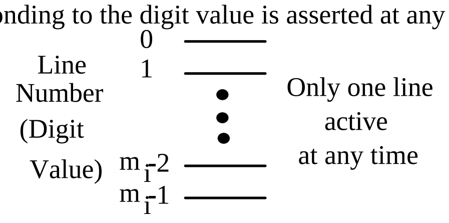


Figure 1: One-Hot Residue Representation for Digit x_i

Direct Digital Frequency Synthesis

Direct Digital Frequency Synthesis (DDFS) is a method of sinusoidal signal generation that yields frequencies of high-precision and resolution with frequency switching that is fast and phase continuous. Most DDFS systems use the Sine Table Lookup Method [3], wherein the output is generated by periodically accessing a ROM in which are stored samples of a single period of a sine wave. The samples are converted to analog by a digital-to-analog converter (DAC). The ROM addresses are computed using a Phase Accumulator, which generates successive multiples of an externally-supplied frequency setting word denoted by k.

The architecture of an RNS-based pipelined synthesizer called HADS (High-Agility Direct Synthesizer) [4] is shown in Figure 2. It will be the benchmark for comparison with the OHR design. The residue-encoded setting word, k, is the step size through the Sample ROM and thus establishes the output frequency. Small (large) k values produce low (high) output frequencies. The Phase Accumulator forms the multiples of k (modulo the product of the moduli) by successive addition. Typical widths of accumulator output are at least 32 bits and therefore must be truncated. Truncation is performed by the Scaler, which computes the Scaling operation on the accumulator output. The AI units and the SI ("sign invert") input of the DAC allow exploitation of the sine quarter-wave symmetry to reduce Sample ROM size. The AI units compute the additive inverse of their inputs when the next-msb is asserted. The SI input of the DAC performs a sign inversion of the DAC output when it is asserted.

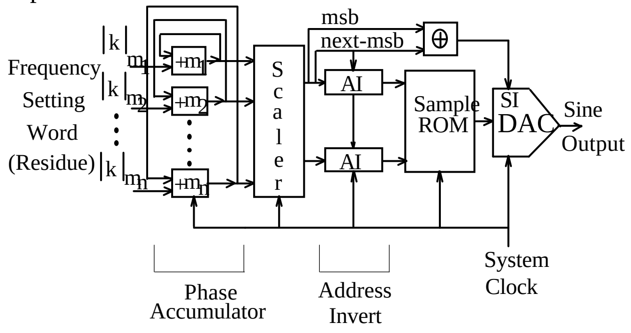


Figure 2: High-Agility Direct Synthesizer (HADS)

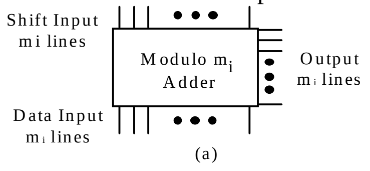
III: Arithmetic Circuits for the One-Hot Residue Encoding

The OHR arithmetic circuits are barrel shifter-based because for many moduli the basic operations of addition, subtraction and multiplication in the OHR representation can be performed by cyclic rotations. A low delay-power product results from the circuits economy of transistors and low activity factors.

OHR Arithmetic Circuits

For one-hot encoded operands, modulo m_i addition is a cyclic permutation and can be implemented using a barrel shifter. The barrel shifter can be built using pass transistors or transmission gates. Figure 3a shows the adder symbol and 3b shows the internal architecture. For ease of routability pass transistors are preferred over transmission gates [6].

Subtraction is implemented by transposing the subtrahend wires to generate its additive inverse modulo mi. A dedicated subtracter would hardwire the transposition on the subtrahend input of the barrel shifter.



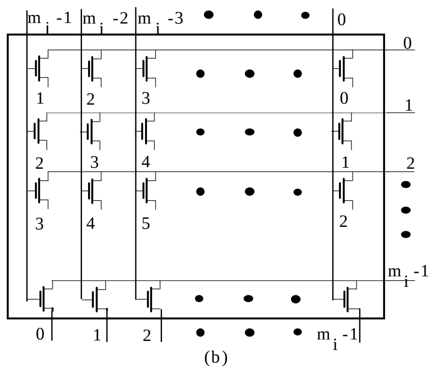


Figure 3: OHR Adder: (a) Symbol (b) Architecture

A combined adder /subtracter unit would employ a multiplexer to selectively apply the inverse. The architecture of such a unit will be obvious after the discussion of multiplication.

Multiplication by a constant modulo m_i can also be done by wire transposition. For non-constant multiplication the architecture is dependent on the modulus. Multipliers for moduli of the form $2, 4, p^e, 2p^e$ (p an odd prime) possess primitive roots [7] and admit the simple and regular multiplier structure shown in Figure 4. It

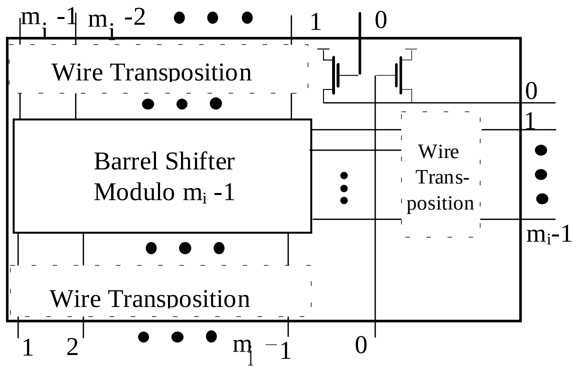


Figure 4: Internal architecture of modulo m_i multiplier

consists of wire transpositions on all ports with a barrel shifter core. The transpositions perform index conversion. The barrel shifter computes the index sum modulo mi-1.

Multipliers for moduli other than these are implemented as crossbar switches. The lack of transpositions enhances their speed but makes them more difficult to route. Their structure is identical to Figure 4 except for the lack of port transpositions and the substitution of a crossbar switch for the barrel shifter.

Modulus conversion of an operand from m_i to m_j can be performed simply and easily. If $m_i < m_j$ the operation is trivial, consisting of "wire padding" with extra lines. Otherwise, a wire corresponding to a value $\alpha \geq m_j$ is applied to the gate of a pass transistor whose source is tied high, as shown for the zero inputs in Figure 4. The drain is tied to the output line corresponding to the value $\alpha \bmod m_j$.

Scaling consists of a subtraction by a modulus-converted operand, followed by a constant postmultiplication (see Section II). This can be implemented by a barrel shifter with modulus conversion on the subtrahend and a wire transposition on the output.

OHR-based Frequency Synthesizer

The OHR synthesizer is the HADS shown in Figure 2 with the Phase Accumulator (PA), Scaler, Address Invert (AI) Units and Sample ROM modified for the OHR encoding. See Figure 5. It includes an Encoder unit

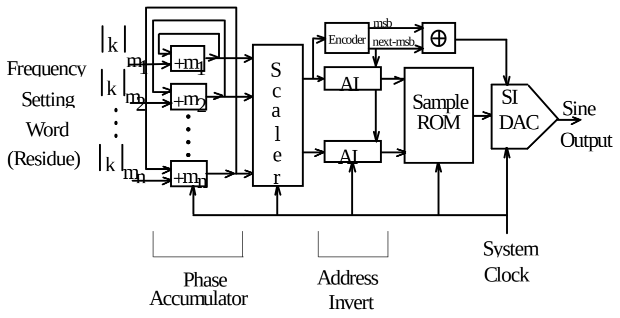


Figure 5: OHR Synthesizer

which generates the msb and next-msb for the XOR from the even modulus of the Scaler output.

A modulo m_i adder in the PA is shown in Figure 6a, and consists of a single edge-triggered register with OHR adder feedback. Each register element is implemented with two level-sensitive latches [8], each consisting of a two-input inverting mux with an output inverter (see Figure 6b).

The Scaler performs truncation using a memory-oriented approach as shown in Figure 7a. It is implemented with one ROM per PA modulus and a binary tree of OHR adders, each r moduli wide. ROMi converts the output of the ith PA adder to its truncated value in r-digit OHR code. The adders accumulate these results to yield a phase value with reduced resolution which is input to the AI Units. The value of r is 2 in Figures 2 and 5. This value is advantageous, as will be seen below in the Sample ROM discussion. ROMi and the adder outputs are pipelined with the register element shown in Figure 6b.

Figure 7b shows the ROMi word architecture [8].

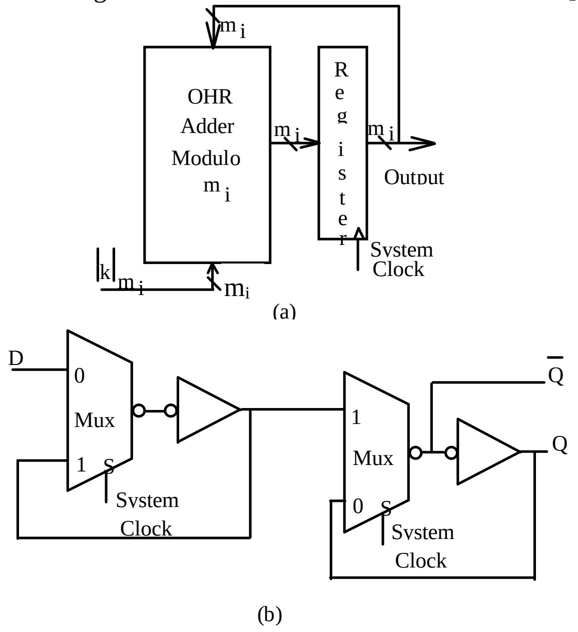


Figure 6: (a) PA adder (b) register element

Pull-down transistors are located only at the 0-bit positions. The bit lines are precharged during high system clock. During low clock the word line and bit lines are asserted. Output data is active low and is converted to active high by using the inverted output of the register element in Figure 6b. Note that each word line drives only two transistor gates (due to the OHR encoding).

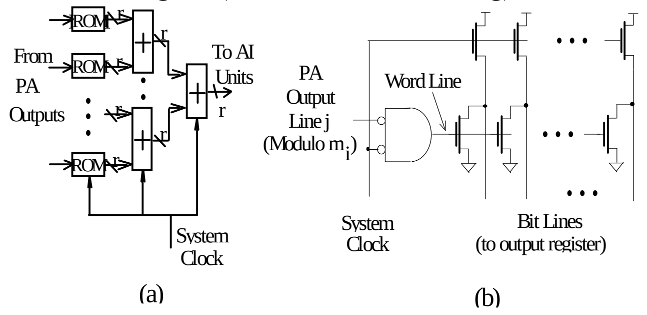


Figure 7: (a) Scaler and (b) ROMi Word Architectures

An AI Unit selectively computes the additive inverse based on the value of the next-msb (generated by the Encoder unit). It is implemented as shown in Figure 8, using a 2-to-1 MUX and wire transposition. In the figure, m_j is one of the r=2 Scaler output moduli.

The Sample ROM uses the Figure 7b word architecture in standard fashion, as shown in Figure 9. The figure depicts a m1m2 X 1 bit plane. A number p of these share the word and bit addressing circuitry and are connected in parallel to form the m1m2 X p Sample ROM, where p is the DAC width. Moduli m1 and m2 comprise the word and bit select lines, respectively. Corresponding bit lines from each word are tied together and provide the data input for the Sense Amplifier/Output Buffers (SA/OBs). Bit selection is accomplished by using the bit select lines to control the output enables of the SA/OBs, whose outputs are tied to the output bus. The SA/OBs are high gain, single-ended inverters with transmission gate output control.

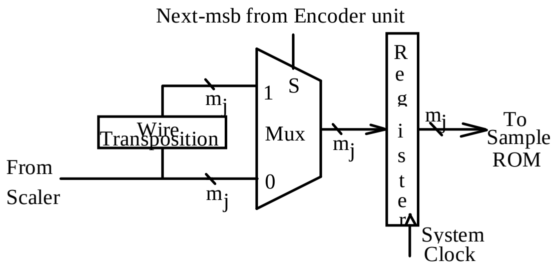


Figure 8: AI Unit Architecture

The Encoder unit is implemented with NOR combinational logic in the straightforward manner. The DAC is a standard high speed twos-complement type. The SI (sign invert) input performs the output sign

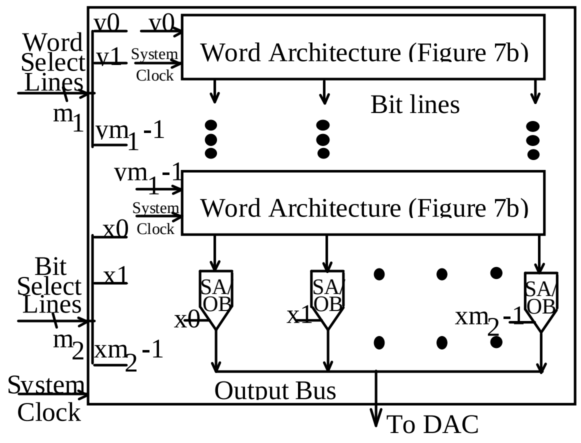


Figure 9: Sample ROM Architecture

complementation after the conversion to analog, where it can be done more quickly.

IV. OHR Synthesizer Delay-power Estimate

This section contains a derivation of a delay-power estimate for both the OHR synthesizer and the HADS. Delay is estimated by the number of critical path transistors. Power is estimated by the number of transistors needed for implementation of each subsystem, multiplied by an “activity” factor. This factor expresses the average fraction of transistors that switch per clock transition [9]. DAC delays and powers will not be estimated because of their difficulty and because they are the same for both designs.

HADS Delay Estimate

The critical path length of the HADS measured in number of transistors is given by the sum of the middle column of entries in Table 1[4], [5]. We have used the fact that a two-input NOR has two transistors of delay [8], and (as will be shown for the OHR synthesizer) the critical path length for the Scaler ROMs is 3 transistors.

HADS Power Estimate

The power can be estimated by adding the number of transistors in the PA, Scaler, AI Units and ROM, and scaling by the activity factor f_H. These estimates are given in the top half of Table 2. We have assumed that on average half of all transistors switch on any clock transition, so that $f_H = .5$. The XOR gate is negligible. We will assume that r=2, that the Scaler moduli are equal (with value m_{out}) and that one of these output moduli is a power of two. The number of pipeline registers will be estimated using a “pipeline intensity” factor d with units of stages/bit of ripple carry addition [5].

We have assumed that the PA employs Shanbhag adders [10]. The Scaler consists of n ROMs of size m_i X w, where $w = 2\log_2 m_{out}$ and m_{out} is the size of the two output moduli. The AI Units consist of constant-operand subtracters controllable by the next-msb of Scaler output. The Sample ROM is of size m_{out}^2 X p, where p is the resolution of the DAC.

OHR Synthesizer Delay Estimate

The critical path delay D_{OHR} of the OHR Synthesizer can be found by adding the delays of the PA, Scaler, Encoder, AI Units and Sample ROM. The results are presented in the last column of Table 1. The PA delay consists of 1 transistor for the barrel shifter and 4 for the pipeline register element. The Scaler path is equal to 3 (two for the word line NOR gate and one for the bit pull down) transistors for the ROM, 4 for the register element at the ROM output (2 each for master and slave), and 5 for each adder in the tree depth (1 for the barrel shifter and 4 for the output register element).

The Encoder requires three $\frac{m_{out}}{4}$-input and two two-input NORs. We have used two-input NORs to implement the multi-input gates and a register element. The AI Units have one transistor of delay through the mux transmission gate and four through the pipeline register element (see Figure 8). The Sample ROM path equals 3 for the word architecture blocks (Figure 9), 3 for the SA/OB and 4 for the pipeline register element.

OHR Synthesizer Power Estimate

The power consumption of each subsystem is found in the same way as for the HADS, and the results are presented in the bottom half of Table 2. As before, we will assume that r=2, one modulus is a power of two and they are approximately equal with value m_{out}.

The PA is implemented with modulo m_i adders and register elements (see Figure 6). f_{O_i} and $f_{O_{out}}$ denote the utilization factors of the ith input and output moduli circuitry. ROMi in the Scaler can be shown to have power $(6f_{O_i} m_i + 38f_{O_{out}} m_{out})$, and the total adder power can be shown to be $(n-1)(2m_{out}^2 + 36m_{out})f_{O_{out}}$, where the number of adders is n-1. Each SA/OB in the Sample ROM requires four transistors, p is the output width in bits (each of which is registered), and the utilization factor of the output bits is .5. The Encoder power was estimated assuming as before that the multi-input NORs are implemented in binary-tree fashion with two-input gates of four transistors each. The sum accounts for the register transistors, (18 per registered bit). There are $\lceil \log_2 m_{out} - 2 \rceil$ levels of pipeline registers with $\left\lceil \frac{m_{out}}{2^{i+2}} \right\rceil$ registers at the ith level, $1 \le i \le \lceil \log_2 m_{out} - 2 \rceil$.

Tables 1 and 2 compare the critical path delays

Table 1: Critical Path Delay (transistors)

Subsystem	**HADS**	**OHR**
Phase Accumulator	$14 + 8\lceil \log_2 m_{max} \rceil$	5
Scaler	$3+(14+8\log_2 m_{out})\lceil \log_2 n \rceil$	$7 + 5\lceil \log_2 n \rceil$
AI Units	$7 + 4\log_2 m_{out}$	5
Sample ROM	$2\lceil \log_2(1+\log_2 m_{out}) \rceil + 2\lceil \log_2 \log_2 m_{out} \rceil + 2\lceil \log_2 m_{out} \rceil$	10
Encoder		$2\left\lceil \log_2 \frac{m_{out}}{4} \right\rceil + 6$

and powers of the HADS and OHR architectures. The percent reduction of the product of the critical path delay and power estimates (henceforth known as the "delay-power product") for the OHR synthesizer below that of the HADS, for the three modulus sets M1=[128,127,113,109,107,103,101,97,89,83], M2=[32,31, 29,27,25,23,19,17,13,11] and M3=[29,23,19,17,13,11,7,5, 3,2] is plotted in Figure 10. We have used $f_H = .5$, $f_{O_i} = \frac{2}{m_i}$ and $f_{O_{out}} = \frac{2}{m_{out}}$ because half (two) of the bits change, on average, every clock cycle for binary-encoded (OHR-encoded) operands. Other parameters used in the figure are d=.5 and p=12, which correspond to 2 bits per pipeline stage and 12 bits of amplitude resolution on the output. It can be seen from the figure that the OHR synthesizer has adelay-power product which is reduced by at least 85% below that of the HADS. Furthermore, the reduction is inversely related to the size of m_{out}. Other data (not shown) indicate that the percent reduction in delay is independent of modulus size, and furthermore, changes in d, p and n have very little effect on the path-power product.

V. Conclusion

The One-Hot Residue (OHR) number system encodes the residue digits in one-hot form. It exhibits a significantly reduced delay-power product below that of fully-encoded residue systems. The arithmetic units for the number system are fast and simple and possess exceptional routing regularity. Use of the circuits is exemplified in the design of a direct digital frequency synthesizer for frequency-hopped spread spectrum communication systems. The delay-power product for this system is reduced by at least 85% below that of a recently proposed conventional residue-based design.

Table 2: Power Estimates (transistors)

Subsystem	HADS
Phase Accumulator	$nf_H \sum_{i=1}^{n} (10d \lceil \log_2 m_i \rceil^2 + 54 \lceil \log_2 m_i \rceil)$
Scaler	$f_H \sum_{i=1}^{n} \left[4\sqrt{m_i} (\log_2 \sqrt{m_i} - 1) + 2m_i \log_2 m_{out} + 8 \log_2 m_{out} (\sqrt{m_i} \log_2 \sqrt{m_i} + \sqrt{m_i} - 1) \right]$
AI Units	$P_{AI} = 52f_H \lceil \log_2 m_{out} \rceil + 20df_H \lceil \log_2 m_{out} \rceil^2 + 52f_H (\lceil \log_2 m_{out} \rceil - 2) + 20df_H (\lceil \log_2 m_{out} \rceil - 2)^2$
Sample ROM	$4f_H m_{out} (\log_2 m_{out} - 1) + pf_H m_{out}^2 + 4pf_H (m_{out} \log_2 m_{out} + m_{out} - 1)$
Encoder	0

Subsystem	OHR
Phase Accumulator	$\sum_{i=1}^{n} f_{O_i} (m_i^2 + 18m_i)$
Scaler	$\sum_{i=1}^{n} (6f_{O_i} m_i + 38f_{O_{out}} m_{out}) + f_{O_{out}} (n-1)(2m_{out}^2 + 36m_{out})$
AI Units	$44f_{O_{out}} m_{out}$
Sample ROM	$p(9f_{O_{out}} m_{out} + .5m_{out}^2 + 9)$
Encoder	$f_{O_{out}} \left(3m_{out} - 4 + \sum_{i=1}^{\lceil \log_2 m_{out} - 2 \rceil} 18 \left\lceil \frac{m_{out}}{2^{i+2}} \right\rceil \right)$

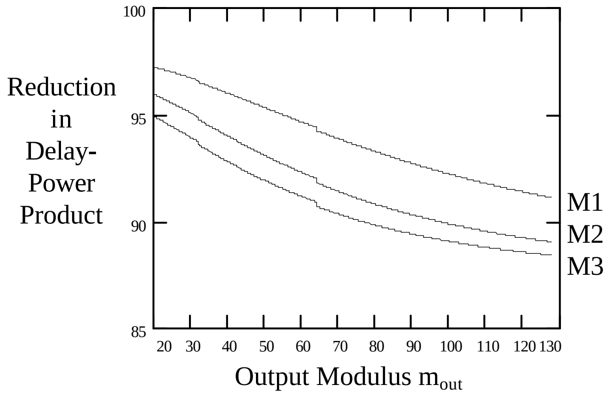


Figure 10: OHR Delay-Power Product Reduction

Acknowledgment

The author is grateful to Mr. William Ivancic of the NASA Lewis Research Center, Space Communications /Electronics Division for supporting this research.

References

[1] Chren, W.A. Jr., *A New Residue Number System Division Algorithm*, Computers and Mathematics With Applications, vol. 19, no. 7, pp. 13-29.

[2]. N.S. Szabo, R.I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*, NY: McGraw-Hill, 1967, pp. 147-151.

[3] J. Tierney, C.M. Rader, B. Gold, "A Digital Frequency Synthesizer", *IEEE Transactions on Audio and Electroacoustics*, AU-19, no. 1, pp. 48-56, March, 1971.

[4] W. A. Chren, Jr., "Area and Latency Improvements for Direct Digital Synthesis Using the Residue Number System", *Proceedings of the 37th Midwest Symposium on Circuits and Systems*, Lafayette, LA, 1994

[5] W. A. Chren, Jr., "*RNS-Based Enhancements for Direct Digital Frequency Synthesis*", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, in press.

[6] Wolf, W., *Modern VLSI Design: A Systems Approach*, Englewood Cliffs, NJ: PTR Prentice Hall, 1994, p. 222-223.

[7] Niven, I., Zuckerman, H.S., *An Introduction to the Theory of Numbers*, New York: Wiley, 3rd Ed., 1972.

[8]. Weste, N.H.E., Eshraghian, K., *Principles of CMOS VLSI Design, A Systems Perspective,* Reading, MA: Addison Wesley, 2nd Ed., 1993.

[9] Bakoglu, H.B., *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, 1990, p. 440.

[10] N.R. Shanbhag, R.E. Siferd, "A Single-Chip Pipelined 2-D FIR Filter Using Residue Arithmetic", *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 5, pp. 796-805, May, 1991.