Fall 2010

Efficient modular arithmetic units for low power cryptographic applications

Rajashekhar Reddy Modugu

Recommended Citation

Modugu, Rajashekhar Reddy, "Efficient modular arithmetic units for low power cryptographic applications" (2010). *Masters Theses*. 6645.
https://scholarsmine.mst.edu/masters_theses/6645

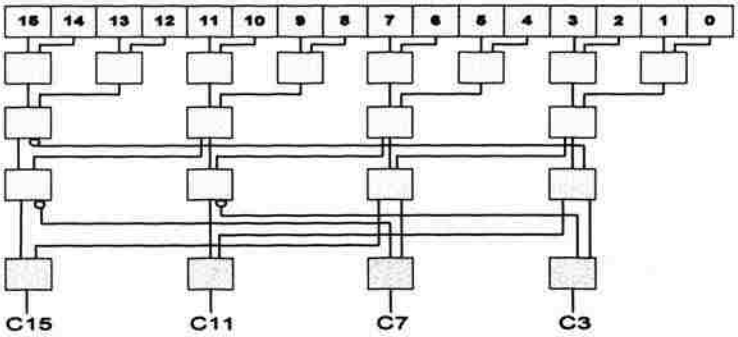


Fig. 6 Inverted EAC adder implemented using sparse tree structure

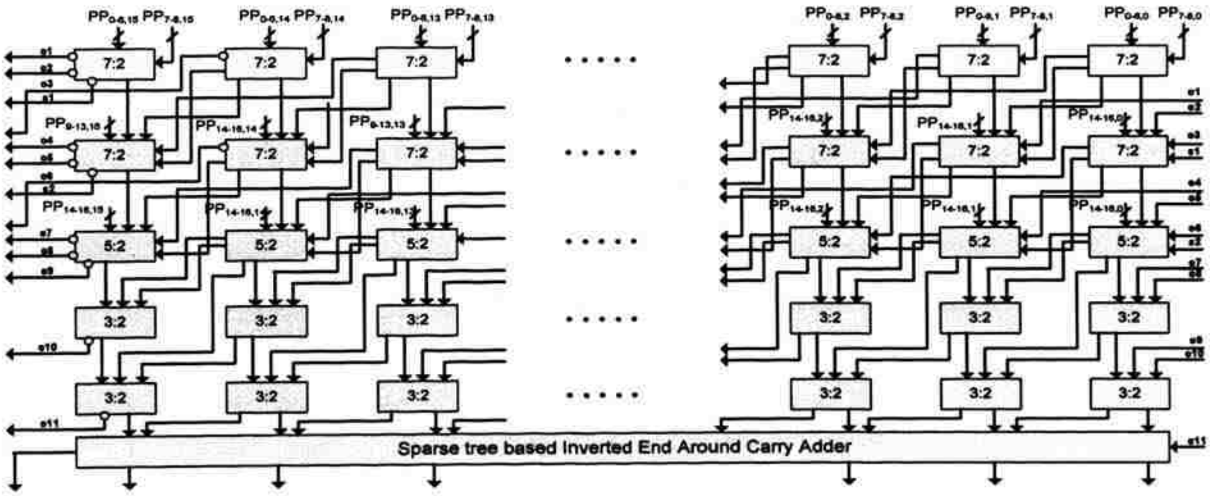
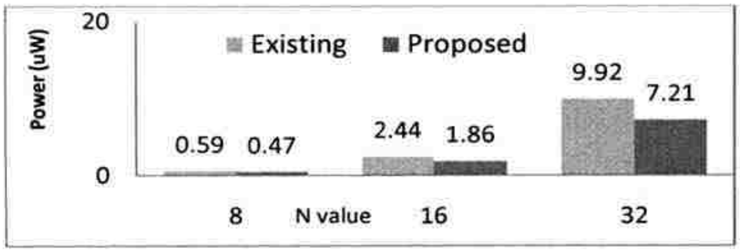
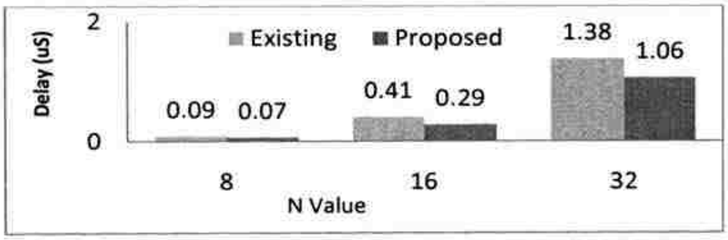


Fig. 7 Proposed implementation of the mod $2^{16}+1$ multiplier using efficient compressor

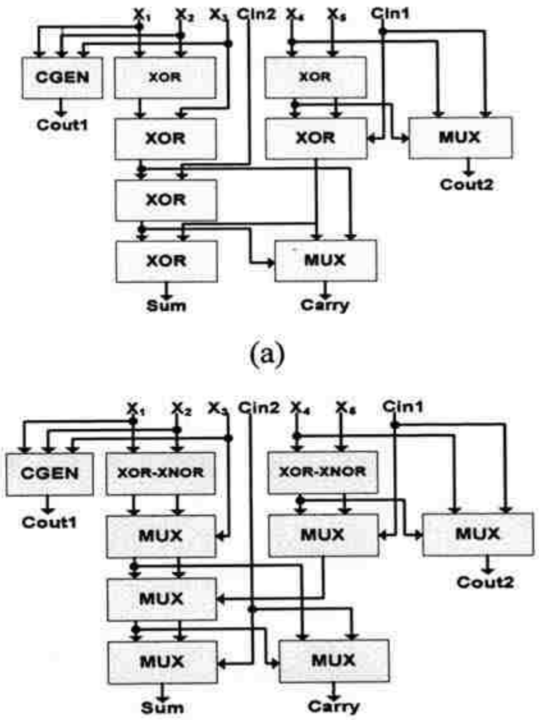


(a)



(b)

Fig. 8 (a) Power (b) Delay comparisons of existing and proposed multipliers



(a)

(b)

Fig.2 5:2 compressors; (a) Existing design (b) New design

compared with the existing prefix tree based adders [14]. Hence this sparse tree can be used to design Inverted-End-Around-Carry adder.

The newly designed Inverted-End-Around-Carry adder using sparse tree adder structure is used in the final stage addition of the modulo multiplier is shown in Fig. 3. The proposed implementation of the modulo $2^{16}+1$ multiplier for IDEA cipher is shown in Fig. 3 and $R_{16}R_{15}.....R_2R_1R_0$ represents the final product of the modulo $2^{16}+1$ multiplier.

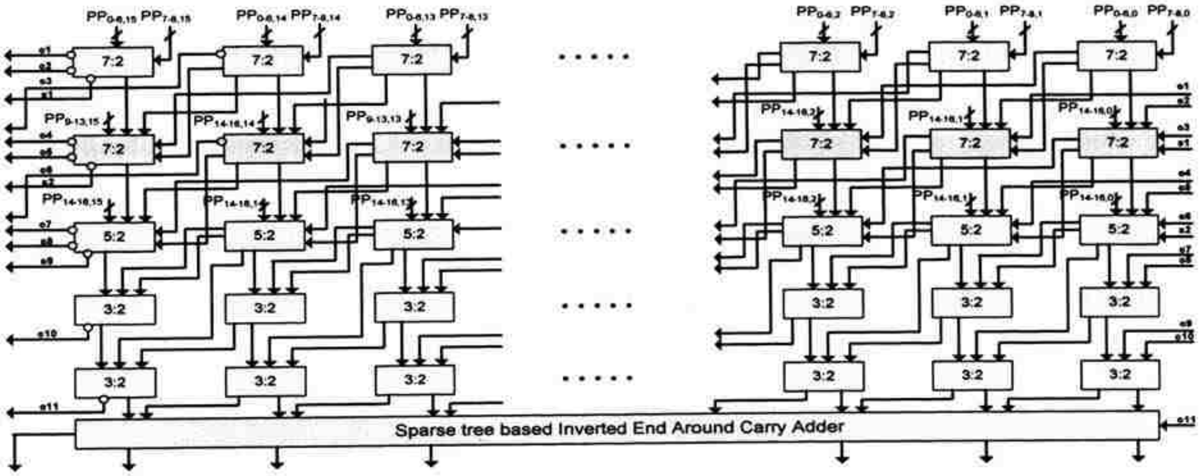


Fig. 3 Proposed implementation of the mod $2^{16}+1$ multiplier using efficient compressor

4. NOVEL IMPLEMENTATION OF INTERNATIONAL DATA ENCRYPTION ALGORITHM (IDEA) USING MODULO 2^N+1

The modulo 2^n+1 computation is an integral part of the International Data Encryption Algorithm (IDEA) where n = 16 [1, 4, 15]. Three major operations that decide the overall delay and performance of IDEA cipher are:

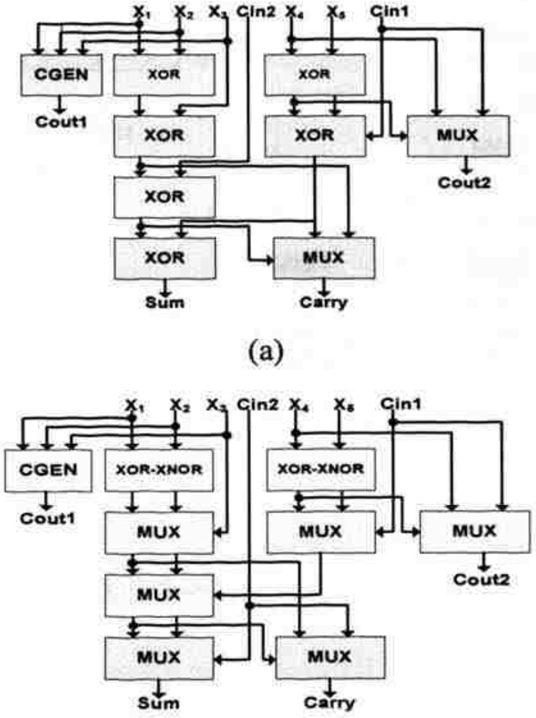
1) modulo 2^{16} addition,

2) bitwise-XOR and 3) modulo $2^{16} + 1$ multiplication.

As the first two operations take less time and are easy to implement, the delay and power efficiency of the entire IDEA cipher depends significantly on the modulo $2^{16}+1$ multiplication operation. Hence, the IDEA cipher is implemented using the proposed modulo multiplier and compared with the existing implementations. To encrypt a data block using IDEA cipher, the data should be processed through three modulo multiplication operations in a single round and the manipulated data again should pass through seven such rounds iteratively and a final output transformation to produce the final encrypted output. The IDEA cipher takes 64-bit input data and produces a 64-bit cipher text with a 128-bit key. The encryption and decryption algorithms in IDEA are almost identical except they utilize two different sets of sub key generated by the same key with different processes.

The IDEA encryption and decryption processes consist of eight rounds of data manipulation using sub keys and a final output transformation stage. In this cipher, all the operations are carried out on 16-bit sub-blocks. In the encryption process, the input data block of 64-bits is divided into 4 sub blocks of 16-bits each (X_1, X2, X3, X4). 52 sub keys for the encryption process are generated from the original 128-bit key by shifting a part of it. Out of the 52 sub keys, six different sub keys (i.e. $Z^{(r)}{}_1$;$Z^{(r)}{}_2$;$Z^{(r)}{}_3$,$Z^{(r)}{}_4$,$Z^{(r)}{}_5$ and $Z^{(r)}{}_6$, where r is the round number) are used for each round and the remaining 4 sub keys are used in the final output transformation stage. The 16-bit outputs at each round are represented as $Y^{(r)}{}_1$, $Y^{(r)}{}_2$, $Y^{(r)}{}_3$, $Y^{(r)}{}_4$ and W_1,W_2,W_3,W_4 are the outputs of the final output stage transformation. The 52 subkeys used for the decryption process are obtained using a different algorithm [17]. As shown in Fig. 4, the critical path consists of three modulo $2^{16}+1$ multiplication operations, two modulo 2^{16} addition operations and two 16-bit XOR

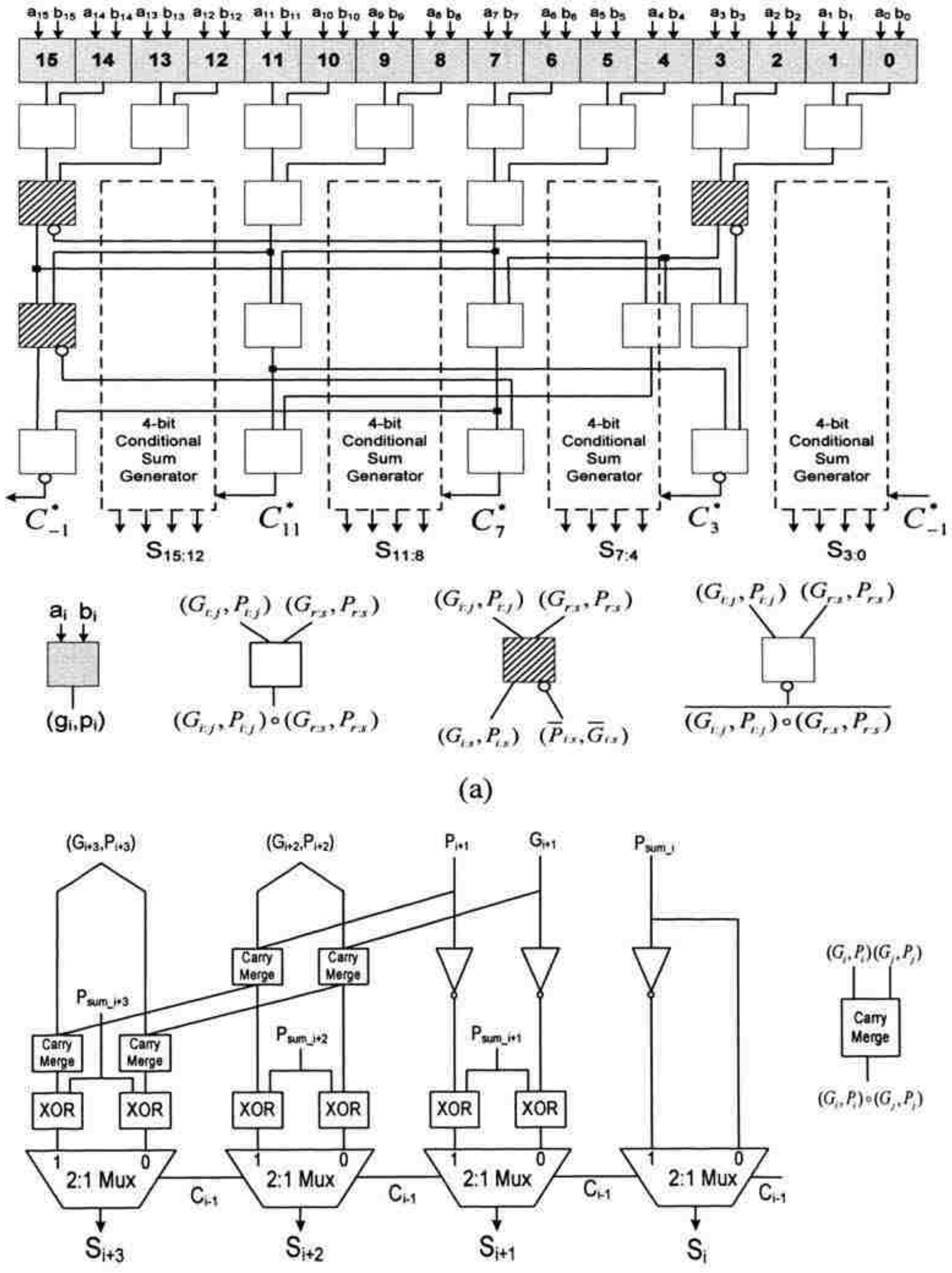
checking modulo multipliers. The proposed MUX-based design of the 7:2 compressor is shown in Fig. 4. CGEN block used in Fig. 3, Fig. 4 can be obtained from the equation $Cout1 = (x1 + x2) \cdot x3 + x1 \cdot x2$.



(a)

(b)

Fig. 3 5:2 compressors; (a) Existing design (b) New design



(a)

(b)

Fig. 5 (a) 16-bit sparse tree based Inverted EAC adder (b) 4-bit conditional sum generator

compressors are used instead of full adders in each column of the carry save adder network. This selection of the compressors is based on the input width. For a particular input width, several compressor networks are possible. The best possible compressor network consists of compressors with high order such as 7:2 and 5:2 compressors. The sum and carry vectors generated by the partial products reduction module have to be added in the final stage addition module. A part of the correction factor 1 left behind is used in the final stage addition to take advantage of the following equation.

$$|S + C + 1|_{2^n+1} = \left|S + C + \overline{Cout}\right|_{2^n} \qquad (7)$$

From (7), observe that the inverted carry out of the addition of Sum and Carry vectors has to be fed back. Hence, adding a constant '1' to the Sum and Carry vectors results in Inverted End Around Carry (EAC) modulo 2*n* addition which has regular VLSI implementation. This inverted EAC is efficiently designed using the sparse tree adder network, which has less interstage wiring and less cell density. A novel modulo $2^{16}+1$ multiplier, which uses efficient compressors in the partial products reduction module is shown in Fig. 6.

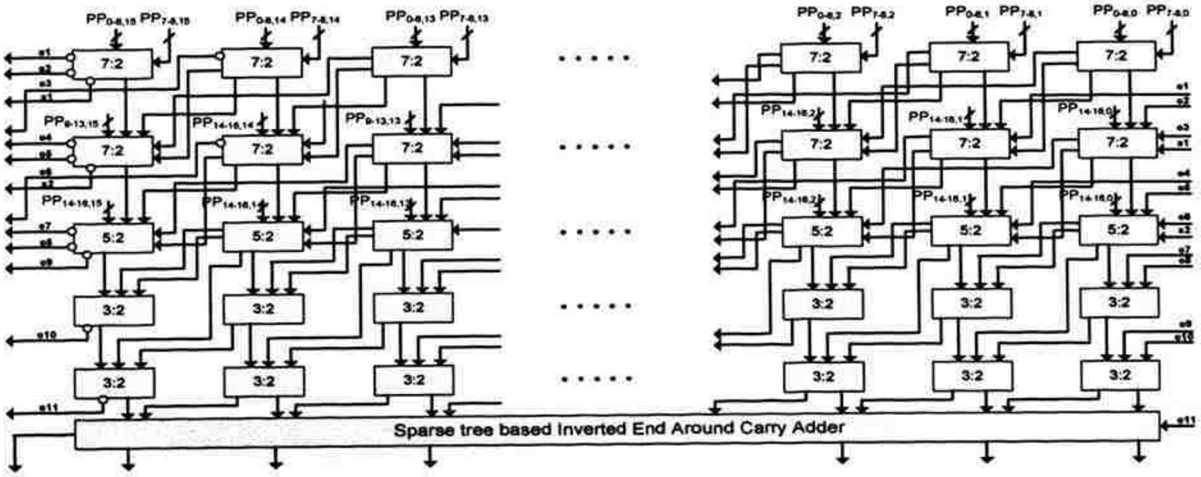


Fig. 6 Hardware implementation of the modulo $2^{16} + 1$ multiplier