

APPLICATION NOTE 4399

Modulo Exponentiation Timing with the DS5250 Microcontroller

By: Conrad Schlundt

Abstract: The DS5250 high-speed, secure microcontroller features a MAA (Modular Arithmetic Accelerator). This application note explains the configuration of the MAA for exponentiation, discusses the trade offs in execution time, and shows typical execution times.

Introduction

Modulo exponentiation is used in many cryptographic algorithms. Anyone implementing one of these algorithms must know approximately how long the operation will take. This application note describes how modulo exponentiation is done on the [DS5250](#) high-speed, secure microcontroller. It lists typical times to run various expressions, and describes the code flow to obtain the timings.

Basic MAA Operation

Modulo exponentiation is the function, $(\text{base}^{\text{exponent}}) \bmod \text{modulus}$. For example, $(2^9 \bmod 10)$ is equal to $(512 \bmod 10)$ which is equal to 2. The answer is always a number between 0 and modulus-1.

The MAA (Modulo Arithmetic Accelerator) on the DS5250 always uses MAA register "a" for the base, MAA register "e" for the exponent, and MAA register "m" for the modulus. The MAA register "b" is initialized to 1 before the operation, and contains the result after the operation. The MAA Size registers (MAS1 and MAS0 at A2h and A1h) tell the MAA the maximum number of bits in these registers. The m register must have the highest bit set to work. The size register can have values from 2 to 4096.

The Modular Arithmetic Accelerator Control register (MACT at A3h) contains the bits used to control the operation of the MAA. The Calculation Configuration bits (CLC1 and CLC0 of the MACT register) determine which of four operations is to be executed. The operations can be modulo multiply; modulo square; modulo square and multiply; and the operation discussed here, modulo exponentiation.

Modulo exponentiation is calculated with repeated squares and multiplies. The square operation is done for every bit in the exponent. The multiply operation only has to be done when the corresponding bit in the exponent is set. **Figure 1** gives the pseudo code for the modulo exponentiation operation. The Optimized Calculation Control bit (OCALC of the MACT register) determines if a multiply operation is done for every bit. When the OCALC bit is enabled, every time a 1 bit is found in the exponent, a multiply operation is done. When the OCALC bit is disabled, a multiply is done for every bit, zero or 1, in the exponent, thus giving us similar time calculations for any specific modulus size. All private key calculations should be done with OCALC=0 (disabled) along with running from the ring (RNGSEL=1) to avoid timing attacks.

The MAA can run using the system clock, or it can be run from the ring. The MAA runs at half the system clock speed when this option is selected. Thus, for a 22.1MHz crystal, the MAA will run at 10.05MHz. It will take the same time to execute the same values when running from the system clock. When the MAA is run from the ring, the execution time can vary depending on voltage, temperature, and the inherent speed of the ring which will vary from part to part. The MAA runs at the full speed of the ring. In the typical data presented in **Tables 1** and **2**, the ring is running near 22Mhz. Ring Oscillator Select (RNGSEL) of the MACT register controls which clock is used for the modulo exponentiation operation.

Typical MAA Timings

The timings that have been collected are broken into two groups. The first group looks at large numbers in each of the modulus, base and exponent. The second group looks at the timing when using a small exponent that only has 2 bits set, specifically, the value 10001h. This number is sometimes used as a public exponent in the RSA algorithm. In each group, there are two halves. The first half has optimization enabled (OCALC=1) and the second half has optimization disabled. Within each half, typical timing values are listed for different clock sources. These timings are all displayed in millisecond (ms) units.

The typical timing values given in the table are the average of ten different calculations using random values in each of the registers. The modulus is random up to the most significant digit, which is always a 1. In general, about half the bits are set in each of the parameters.

The timing of each calculation was measured using Timer 0 as a divide-by-12 clock. When the 16-bit Timer 0 rolls over, an interrupt occurs and a 1 is added to the six external count bytes. At the end of the calculation the timer is stopped, and the external count bytes and the 16-bit timer count are displayed as a 64-bit number that gives the length of the calculation. A 22.1MHz oscillator will

have a 543ns resolution per timer tic. The resolution is 1.085µs at 11.0592MHz. **Figure 2** has the pseudo code for timing a MAA calculation.

Table 1. Modulo Exponentiation Times in Milliseconds

(a, e, and m are random values)

Modulus Size	Optimization ON Clock Source			Optimization OFF Clock Source		
	Ring	22.1MHz Osc	11.1MHz Osc	Ring	22.1MHz Osc	11.1MHz Osc
256	12.38	26.28	51.44	16.33	34.79	69.55
512	74.98	155.43	312.06	98.18	208.79	416.91
768	225.44	468.50	943.04	296.10	626.89	1,252.23
1024	507.39	1,050.53	2,079.01	664.20	1,397.87	2,793.32
1280	958.41	1,967.81	3,922.17	1,248.33	2,629.90	5,258.52
1536	1,611.08	3,321.94	6,623.29	2,112.68	4,421.99	8,833.31
1792	2,520.53	5,176.46	10,311.88	3,295.64	6,889.75	13,771.52
2048	3,729.76	7,573.35	15,199.66	4,863.27	10,143.31	20,249.51
2304	5,251.26	10,773.81	21,372.70	6,852.96	14,276.87	28,532.62
2560	7,159.86	14,557.57	29,079.79	9,328.25	19,392.38	38,761.51
2816	9,434.47	19,216.24	38,474.44	12,334.35	25,636.24	51,189.86
3072	12,152.62	24,807.55	49,631.36	15,930.13	33,070.91	66,018.62
3328	15,360.16	31,377.07	62,436.28	20,147.92	41,818.90	83,544.01
3584	19,138.10	38,988.81	78,039.69	25,073.03	51,951.35	103,848.07
3840	23,445.08	47,678.86	95,490.03	30,691.85	63,689.30	127,205.55
4096	28,327.98	57,649.65	115,295.25	37,128.98	76,965.83	153,828.69

Table 2. Modulo Exponentiation Times in Milliseconds

(e = 10001h; a and m are random values)

Modulus Size	Optimization ON Clock Source			Optimization OFF Clock Source		
	Ring	22.1MHz Osc	11.1MHz Osc	Ring	22.1MHz Osc	11.1MHz Osc
256	0.65	1.35	2.70	15.87	32.62	65.15
512	1.87	3.88	7.72	98.02	200.88	401.50
768	3.71	7.66	15.29	294.26	611.73	1,222.39
1024	6.16	12.70	25.35	660.95	1,371.87	2,741.38
1280	9.20	18.97	37.89	1,248.98	2,587.99	5,171.69
1536	12.88	26.49	52.93	2,110.76	4,366.96	8,726.72
1792	17.16	35.27	70.55	3,297.84	6,815.56	13,619.78
2048	22.03	45.33	90.51	4,862.39	10,040.36	20,064.18
2304	27.55	56.60	113.06	6,856.06	14,148.38	28,273.26
2560	33.67	69.14	138.26	9,332.14	19,246.16	38,460.11
2816	40.41	82.91	165.70	12,342.92	25,440.42	50,838.52
3072	47.74	97.92	195.79	15,933.52	32,838.19	65,621.43
3328	55.70	114.25	228.36	20,158.79	41,545.91	83,022.64
3584	64.28	131.83	263.28	25,083.32	51,670.49	103,254.99
3840	73.45	150.57	300.69	30,747.58	63,318.76	126,532.11
4096	83.27	170.62	340.98	37,183.65	76,597.28	153,067.16

```

Modulo_exponentiation(base, exponent, modulus)
  result = 1;
  while (exponent > 0) {
    lowbit = exponent mod 2      <- get low bit of exponent
    if (lowbit == 1) {          <- if low bit of exponent is set
      result = (result*base) mod modulus  <- multiply it
    }
    base = (base*base) mod modulus      <- square the base
    exponent = exponent / 2      <- shift the exponent right
  }
  return result;

```

Figure 1. Pseudo code for modular exponentiation.

- Initialize Timer 0 to be a divide-by-12 counter with interrupts on rollover.
- Clear Timer 0 and the external count bytes.
- Set the size registers (MAS1 and MAS0) for the number of bits to be calculated.
- Clear all of MAA RAM.
- Fill MAA Register A with a random number up to the number of bits specified.
- Fill MAA Register E with a random number up to the number of bits specified or the value of 10001h for the small exponent tests.
- Fill MAA Register M with a random number up to the number of bits specified.
- Set the bit in the MAA Register that corresponds to the size.
- Write a 1 to the least significant byte of MAA register B.
- Configure the OALC, RNGSEL, and Calculation Configuration bits in the MACT register.
- Start the MAA calculation.
- Start Timer 0. (On every Timer 0 interrupt, increment the external count bytes.)
- Wait until the calculation is done.
- Stop Timer 0.
- Display the external count bytes and the value of Timer 0 giving a 64-bit value.

Figure 2. Pseudo code for timing a modular exponentiation calculation.

Application note 4399: www.maxim-ic.com/an4399

More Information

For technical support: www.maxim-ic.com/support

For samples: www.maxim-ic.com/samples

Other questions and comments: www.maxim-ic.com/contact

Automatic Updates

Would you like to be automatically notified when new application notes are published in your areas of interest? [Sign up for EE-Mail™](#).

Related Parts

DS5250: [QuickView](#) -- [Abridged Data Sheet](#)

AN4399, AN 4399, APP4399, Appnote4399, Appnote 4399

Copyright © by Maxim Integrated Products

Additional legal notices: www.maxim-ic.com/legal