International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Improved FIR Filter using Schönhage-Strassen Algorithm based Multipliers

S. Gayathri

PG Scholar, Department of Electronics and Communication,Sri Manakula Vinayakar Engineering College, Puducherry, India
gayathri5gayathri[at]gmail.com

Abstract: ***When it comes to very large integers; the traditional naive algorithm for multiplication does not work well. This leads to very bad performance of the circuit in which the multiplier is implemented. FIR filter is a basic digital filter that mostly consists of multiplication and addition blocks. Multipliersalso mark their significance in many DSP based processors. FIR is one of the most basic fundamental blocks of the DSP processors.SSA is an practical and much understandable algorithm for this purpose. This algorithm increases the speed of the filter without affecting any other major constraints.***

Keywords: Schönhage strassen algorithm, FIRfilter, NTT

1. Introduction

Multipliers are the fundamental blocks of many large circuits. They find their importance in many DSP processors and other important electronic circuits. Adders and multipliers are periodically improved to match with day to day advancements in the electronic circuits. Reduced area, high speed and very low power consumption are basic constraints that are to be considered. This can be achievedby improving the basic circuits like multipliers.SSA is much practical that can meet out the needs.

2. Basic Mathematical Techniques Involved in SSA

This session gives detail understanding of SSA and other mathematical techniques that are involved in performing SSA based multiplication.

Pre requisites of SSA:
Before performing SSA we need to know about some basic mathematical concepts that are used in SSA to produce intermediate results. Numerical Theoretical Transform is one such concept.

Chinese reminder theorem is also a part of this concept. A cumulative bunch of these concepts and algorithms help us understand the SSA easily. DFT is a familiar concept. NTT is the process that generalises DFT. Apart from this NTT involves Montgomery reductions which can be alternatively replacedby the Barrett reductions to increase the speed of the modular arithmetic that is used in NTT. Montgomery reduction allows us to simplify the modular multiplication process. By this we perform modulo multiplication in the absence of heavy modulo operations. This is based on extended Euclidean algorithm. Extended Euclidean algorithm as the name suggests is the extension of the Euclidean algorithm. This states that Ax+By = gcd(A,B). Barrett reduction on the other hand is an algorithm that is designed especially for reduction.

Chinese Remainder Theorem:
There is always a value 'x' that satisfies the convergence. Let's take two co primes, that istwo numbers that have only one number 1 as their common divisor. In CRT we can represent a value say 'x' as

X≡ a1 (mod m(i))
X≡ a2 (mod m(j)) Where, m (i) and m(j) are co primes.

Consider an multiplication operation 17*37

Number	Mod7	Mod11	Mod 13
17	3	6	4
37	2	4	1
629	6	2	4

Figure 2.2.1: Modulo multiplication

Numerical Theoretical transform:
NTT is based on Fourier Transform. This is a generalization method. Consider a sequence ofnon- zero values. Now let us consider the procedure to perform NTT on the integers.

- Consider X= (6,0,10,7,2)
- No. Of elements in the sequence= 5
- Select a prime M for modulo multiplication. Maximum value =10;therefore M=11
- Select a prime value N such thatN=kn+1 and also N≥M

N= k+1
N=11, 13,15,17,19...
Say N= 11 = (11-1)/5 = 2
Say N=13 = (13-1)/5 =2.4(decimal value) Say N=15 = (15-1)/5 =2.6(decimal value) Say N=17 = (17-1)/5 =3.4(decimal value) Say N=19 = (19-1)/5 =3.8(decimal value) Only for 11 we get a non decimal value. Wecan use any number like 11 that does not produce a decimal value.

Assume N= 11
Therefore 5k =10
k=2

- Select a generator for this group. A generator 'g' should be GƆ=g^k

g=6
g^k= 6^2 = 36/11 =3mod 11

Y(0)=X(0)GƆ^(0*0)+X(1)GƆ^(0*1)+X(2)GƆ^(
0*2)+X(3)GƆ^(0*3)+X(4)GƆ^(0*4)
=6(1)+(0)*(6)+(10)*(6*2)+(7)*(6*3)+(2)*(6*4)
=300/11 =3mod 11

Y(1)=X(0)GƆ^(1*0)+X(1)GƆ^(1*1)+X(2)GƆ^(
1*2)+X(3)GƆ^(1*3)+X(4)GƆ^(1*4)
=6+90+189+162
=447/11 =7mod 11

Y(2)=X(0)GƆ^(2*0)+X(1)GƆ^(2*1)+X(2)GƆ^(
2*2)+X(3)GƆ^(2*3)+X(4)GƆ^(2*4)
=6+0+810+5103+13122
=19041/11 =0mod 11

Y(3)=X(0)GƆ^(3*0)+X(1)GƆ^(3*1)+X(2)GƆ^(
3*2)+X(3)GƆ^(3*3)+X(4)GƆ^(3*4)
=6+0+7290+137781+1062882
=1207959/11 =5mod 11

Y(4)=X(0)GƆ^(4*0)+X(1)GƆ^(4*1)+X(2)GƆ^(
4*2)+X(3)GƆ^(4*3)+X(4)GƆ^(4*4)
=6+0+65610+3720087+86093442
=89879145/11 =4mod 11

For Y (0) use addition modulo and for the others we use multiplication modulo. We take GƆ^1 = 3mod 11. This is because another modulo operation is going to be performed with modulo 11 for the answer.

- Inverse of NTT

This is the inverse operation of NTT. HereGƆ^ (-1) = 3^ (n-1)= 81/11 = 4mod 11
Consider Y= (3, 7, 0, 5, 4) from NTTUse the same values of k, N, GƆ
X(0)=Y(0)GƆ^(0*0)+Y(1)GƆ^(0*1)+Y(2)GƆ^(
0*2)+Y(3)GƆ^(0*3)+Y(4)GƆ^(0*4)
=3+7+0+5+4 = 19/11 =8 mod 11
X(1)=Y(0)GƆ^(1*0)+Y(1)GƆ^(1*1)+Y(2)GƆ^(
1*2)+Y(3)GƆ^(1*3)+Y(4)GƆ^(1*4)
=1375= 1375/11 =0 mod 11
X(2)=Y(0)GƆ^(2*0)+Y(1)GƆ^(2*1)+Y(2)GƆ^(
2*2)+Y(3)GƆ^(2*3)+Y(4)GƆ^(2*4)
=282739 /11 =6 mod 11
X(0)=Y(0)GƆ^(0*0)+Y(1)GƆ^(0*1)+Y(2)GƆ^(
0*2)+Y(3)GƆ^(0*3)+Y(4)GƆ^(0*4)
=68420035/11 =2 mod 11
X(0)=Y(0)GƆ^(0*0)+Y(1)GƆ^(0*1)+Y(2)GƆ^(
0*2)+Y(3)GƆ^(0*3)+Y(4)GƆ^(0*4)
=17263757049/11 =10 mod 11
X (n) = (8, 0, 6, 2, 10)

Calculate n^ (p-1); where p= set of uniqueprime factors
5^ (p-1) = 5^4=625/11=9

Therefore (8,0,6,2,10)*9 =(6,0,10,7,2)mod11

NTT can be performed based on FFT and thepseudo code is given below

```
fft(signal, b, omega, M):
    if b == 1:
        return [signal[0]]
    signal_even = [signal[0], signal[2], ..., signal[b - 2]]
    signal_odd  = [signal[1], signal[3], ..., signal[b - 1]]
    fft_even = fft(signal_even, b / 2, omega^2, M)
    fft_odd  = fft(signal_odd,  b / 2, omega^2, M)
    res = array(b)
    for k in [0, b/2):
        # assume we've precomputed all omega^k so it's free to use here
        prod = omega^k * fft_odd[k]
        res[k] = fft_even[k] + prod (mod M)
        res[k + b/2] = fft_even[k] - prod (mod M)
    return res
```

Figure 2.3.1: Pseudo code for NTT

3. Algorithm

Schönhage – Strassen Multiplicationalgorithm
There are various algorithms that can be used to implement multipliers. But mostly they are under research. The algorithms like Furer and De Et Al are advanced multiplication algorithms. The range of bits for which they outperform the SSA is not yet vivid. So the most practical algorithm for the existing software packages is SSA. This algorithm requires notable amount of memory for storage. The time complexity of SSA is calculated as O (n.log n.log log n) for multiplying two integers of bit size n.

Schönhage – Strassen Algorithm

Input : Any two integers (say) 'X" and "Y"
Output: Product of X and Y:Z

1. Calculate NTT of the given inputs "X"and "Y"
2. Bitwise multiply the output of theNTT
3. Z[i]NTT(A)[i]*NTT(B)[i]
4. Perform INTT for Z
5. Z' = INTT (Z)
6. Now the carries need to be accumulated
7. Say, Z[i] greater than or equal to R
8. Then Z[i+1] =Z[i+1]+[Z[i]/R]
9. Then Z[i]=Z(mod R)
10. Return value of R

The value of "X" and "Y" are padded with 0sso as they are of the base value 2.These integers are split into B number of bit of length L. Then the NTT and then inverse NTT is performed on these values.

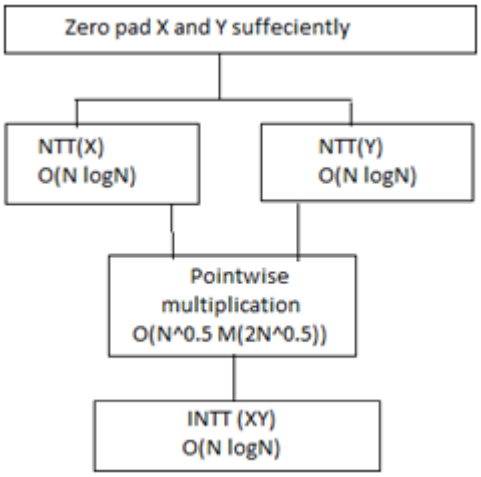

Figure 3.1.1: Time complexity calculation forNTT

This algorithm is used for large operant. By this we can achieve low area, reduced power consumption and achieve high speed. By the end of the process the result is obtained in O(Nlog N log log N) time .

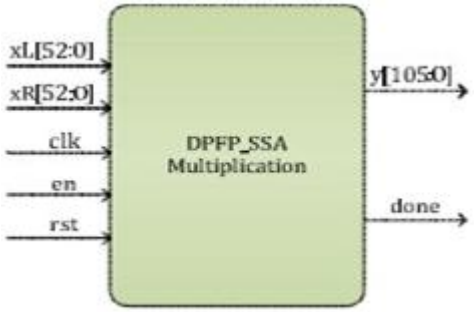

Figure 3.1.2: RTL of 64 bit SSA basedmultiplier

4. Implementation of SSA in FIR Filter

Finite Impulse Response Filter:
FIR filters are one of the most important digital filters that are used for finite response. The outputs of these filters depend on present as well as past values of the given input. This filter is non recursive by nature. These filters are built using three blocks. They are Adders, Multipliers and Delay elements. Considering these three blocks: adders perform binary addition using a half adder or the full adders. Delay elements are chosen as per need. Multipliers have longer operation time. Hence the delays in these filters are majorly caused by these multipliers.

SSA based FIR filter PrototypeImplementation:
With increase in technology, large operations with larger value of input bits are required. This means that we need to use adders and multiplier blocks that can perform their operation on big integer values without compromising with the performance and speed. Hence we create a prototype of FIR filters using the SSA based multipliers. This can significantly reduce the operation time ofthe FIR filters.

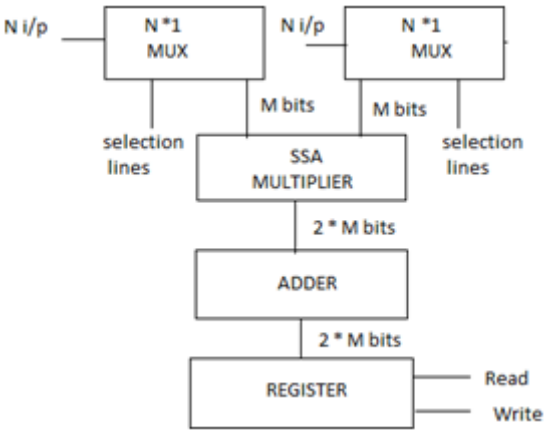

Figure 4.2.1: Prototype of SSA multiplierbased FIR filter

5. Conclusion

Thus an FIR filter that meets required speed without compromising the performance and accuracy to certain level is developed.SSA is one such algorithm that can be used to improve the performance of the circuit and thus improving the overall time complexity of any application. SSA also finds its applications in many other electronic components that needto perform with input integers with large number of bits.

References

[1] Soniya,Suresh Kumar, " A Review of Different Types of Multipliers and Multiplier – Accumulator Unit" International journsl of Emerging Trends and Technologies in Computer science (ISSN 2278-6856)
[2] B.Srikanth,M.Siva Kumar,J.V.R.Ravindra,K.Hari Kishore, "Double Precession Floating Point Multiplier using Schönhage – Strassen Algorithm used for FPGA Accelerator" International Journal of Emerging trends in Engineering Research(ISSN 2347-3983)
[3] Kevin Millar,Marcin Lukowiak, Stanislaw Radziszowski, "Design of a flexible Schönhage – Strassen FFT Polynomial Multiplier with High – Level Synthesis to Accelerate HE in the Cloud" 2019 International Conference on ReConFigurable Computing and FPGA(ReConFig),2019,pp.1-5,doi:10.1109/ReconFig48160.2019.8994790, I EEE Xplore July 26,2020
[4] Tsz-Wo Sze, "Schönhage – Strassen Algorithm with Map reduce for Multiplying Terabit Integers,"SNC'11: Proceedings of the 2011 International Workshop on Symbolic Numeric Computation June 2012 pages 54-62,http://doi.org/10.1145/2331684.2331693
[5] K.Kawamura M.Yangisawa and N.Togawa, "A loop structure optimization targeting high –level synthesis of fast numeric theoretical transform" in 2018 19th international symposium on Quality Electronic Design(ISQED), March 2018,pp.106-111
[6] Schönhage and V.Strassien, "Schnelle multiplication grober zachen,"Computing vol.7.no.3-4,pp.281-292,sep1971
[7] A.S "Asymtotically fast algorithms for the numerical multiplication and division of polynomials with complex co-effecient,"in Computer Algebra, J.Calmet Ed.Berlin.Heidelberg: Spinger Berlin Heidelberg,

1982,pp.3-15
[8] P.Gaudry,A.Kruppa and P.Zimmermann,"A GMP-based implementation of Schönhage – Strassen large integer multiplication algorithm," in Proceedings of the 2007International Symposium on Symbolicand Algebraic Computation,pages167-174,ACM,2007
[9] Iffat Fatima, "Analysis of multiplier in VLSI" Journal of Global Research in Computer Science.
[10] U.Meyer-Baese,G.Botella,D.E.T.Romero and Martin Kumm, "Optimization of high speed pipelining in FPGA- based FIR filter design using generic Algorithm," Proc.of SPIE,Vol.8401,2012