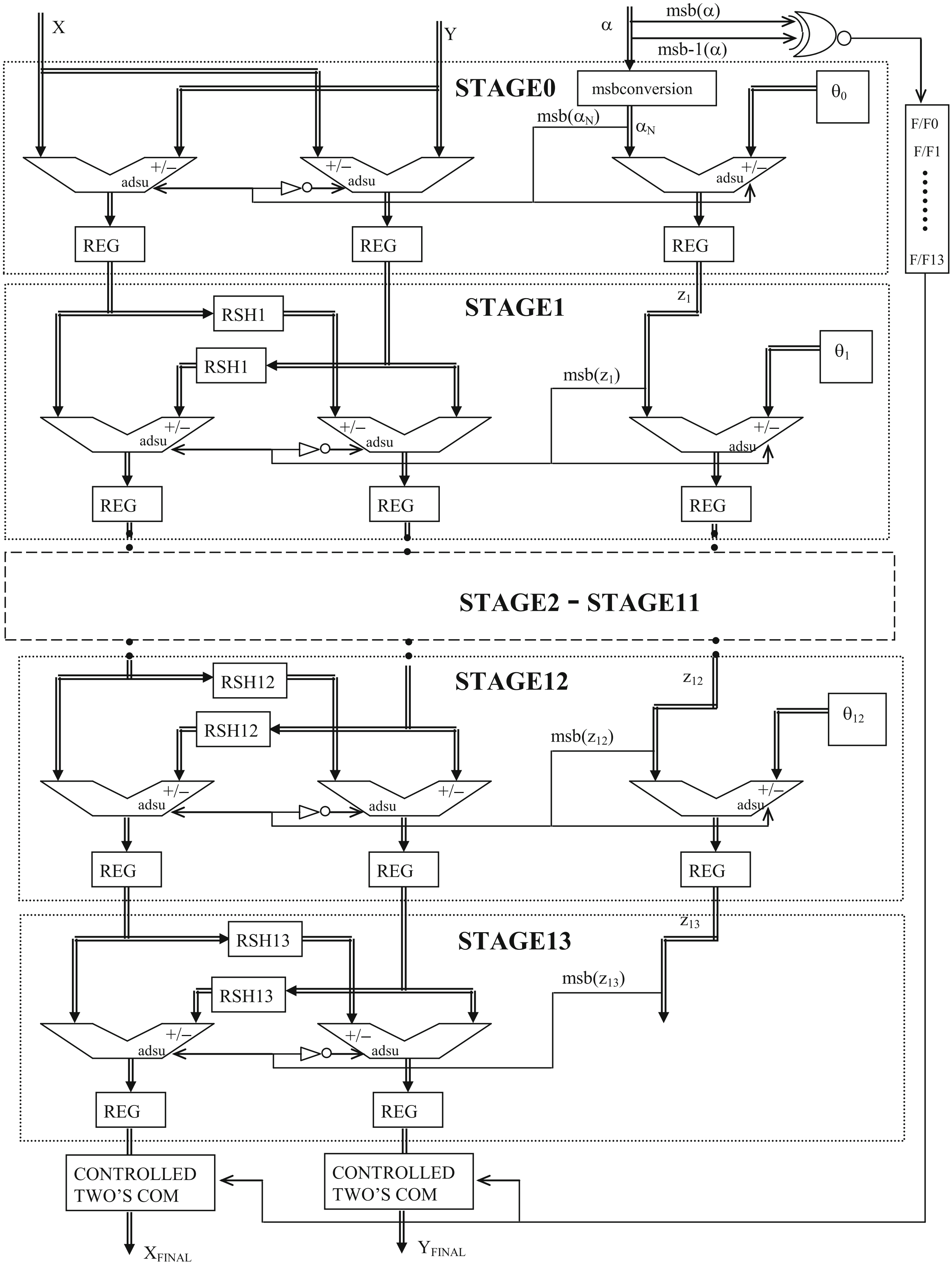

Figure 4 Pipelined CORDIC architecture.

The output is then subtracted from the desired response to generate the error signal as in Eq. 10b. The weight block is a dual port RAM along with two address generators, where initially first set of weight values is written. When weight from a particular location, say $(n-i)$ location, is read out for multiplication to generate $y(n)$, at the same time weight updating operation is going on at first by reading weight from $(n-i-k)$ location, keeping an offset from the other read operation, and updated weight after operation as per Eq. 10c is written back to the same $(n-i-k)$ location. Thus two weights are simultaneously accessed from two different address locations, with a fixed offset between them. The error signal is fed to another pipelined multiplier where complex conjugate operation is done. The other input to the pipelined multiplier is running DFT output after passing through a proper synchronization delay. The output of the conjugate multiplier is multiplied by stepsize μ' through a variable right shifter as μ' can be expressed here in the form 2^{-i}. Then it is added with previous weight, thus performing operation as per Eq. 10c to generate updated weight. The decision block is generating final output by taking decision as per structure of constellation points. Here the primary clock speed is $f_{max}=1/T_{delay(adder)}$ and the operations such as pipelined multiplications, accumulation operations to get $y(n)$, weight updating, running DFT internal operations are by primary clock and for N transform length the maximum sampling clock speed is f_{max}/N. The adaptation delay corresponds here to a maximum of two sampling clocks.

The main processing unit in this architecture is the running DFT block of transform length N that equals 32. Here the updating operation for a particular frequency component, as shown in Eq. 11, can be achieved through only an addition, a subtraction and a complex rotation amounting to $2\pi k/N$ for the k-th frequency component. Now the complex rotation is performed with the help of pipelined CORDIC module whose functionality is described in Section 3.A. The architecture of running DFT, shown in Fig. 3, processes real and imaginary data in two similar sections each comprising of two RAM blocks, adders, subtractors, scale blocks, two data registers, and pipelined CORDIC block, which is

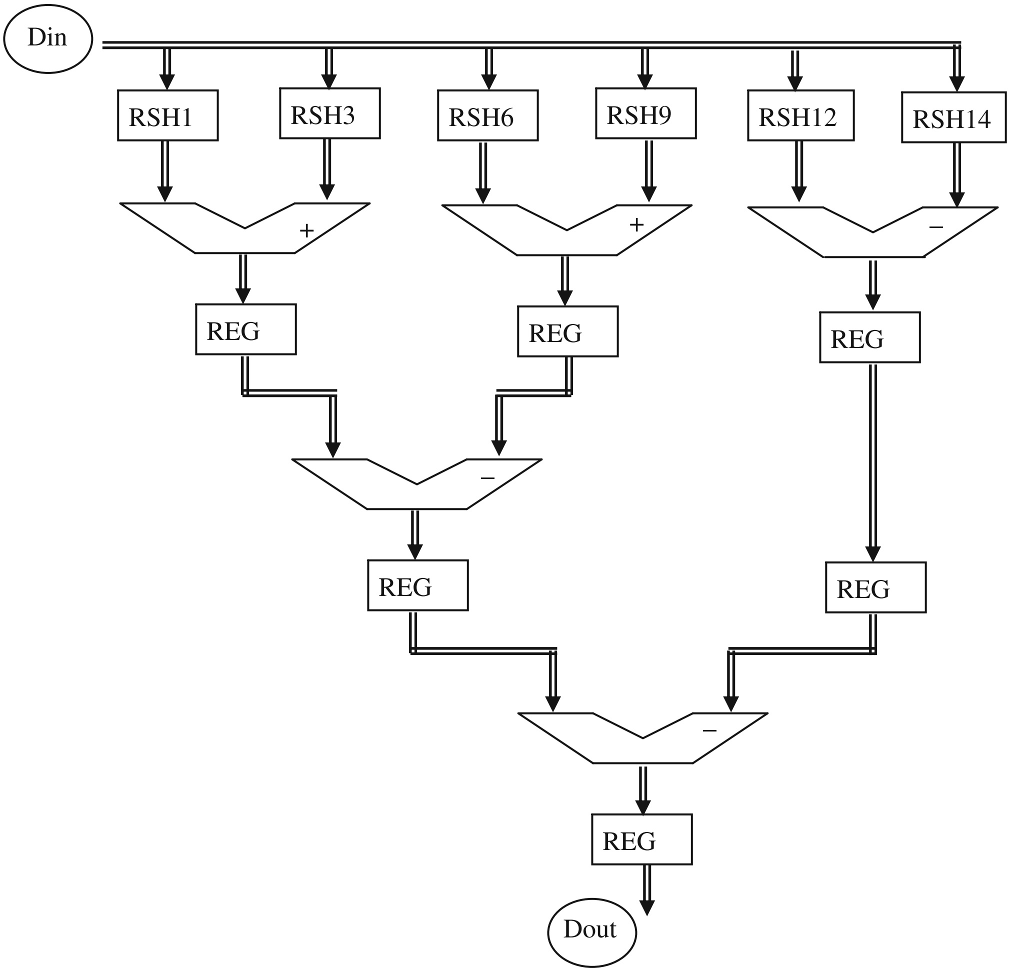

Figure 5 Scale block architecture.

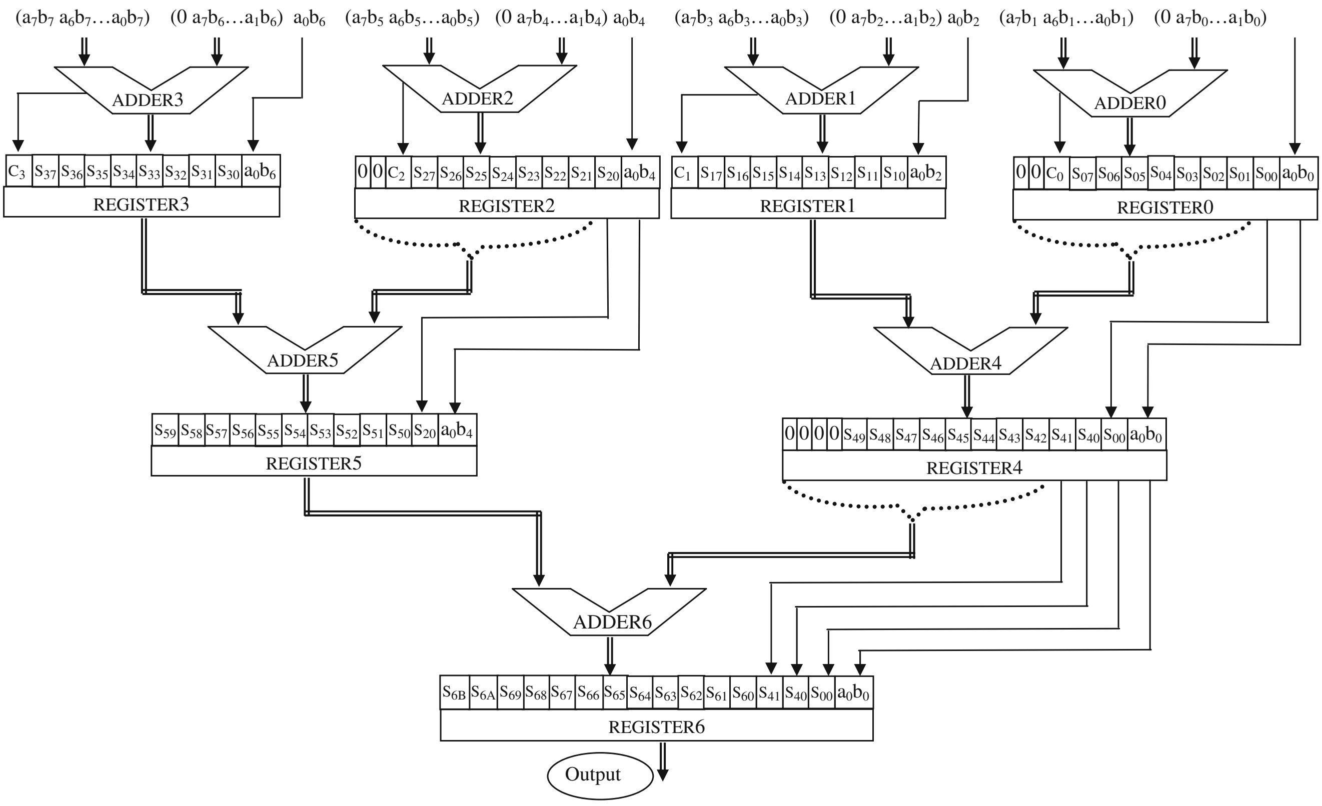
$(a_7b_7\ a_6b_7\ldots a_0b_7)$
$(0\ a_7b_6\ldots a_1b_6)$
a_0b_6
$(a_7b_5\ a_6b_5\ldots a_0b_5)$
$(0\ a_7b_4\ldots a_1b_4)$
a_0b_4
$(a_7b_3\ a_6b_3\ldots a_0b_3)$
$(0\ a_7b_2\ldots a_1b_2)$
a_0b_2
$(a_7b_1\ a_6b_1\ldots a_0b_1)$
$(0\ a_7b_0\ldots a_1b_0)$
ADDER3
ADDER2
ADDER1
ADDER0
$C_3\ S_{37}\ S_{36}\ S_{35}\ S_{34}\ S_{33}\ S_{32}\ S_{31}\ S_{30}\ a_0b_6$
REGISTER3
$0\ 0\ C_2\ S_{27}\ S_{26}\ S_{25}\ S_{24}\ S_{23}\ S_{22}\ S_{21}\ S_{20}\ a_0b_4$
REGISTER2
$C_1\ S_{17}\ S_{16}\ S_{15}\ S_{14}\ S_{13}\ S_{12}\ S_{11}\ S_{10}\ a_0b_2$
REGISTER1
$0\ 0\ C_0\ S_{07}\ S_{06}\ S_{05}\ S_{04}\ S_{03}\ S_{02}\ S_{01}\ S_{00}\ a_0b_0$
REGISTER0
ADDER5
ADDER4
$S_{59}\ S_{58}\ S_{57}\ S_{56}\ S_{55}\ S_{54}\ S_{53}\ S_{52}\ S_{51}\ S_{50}\ S_{20}\ a_0b_4$
REGISTER5
$0\ 0\ 0\ 0\ S_{49}\ S_{48}\ S_{47}\ S_{46}\ S_{45}\ S_{44}\ S_{43}\ S_{42}\ S_{41}\ S_{40}\ S_{00}\ a_0b_0$
REGISTER4
ADDER6
$S_{6B}\ S_{6A}\ S_{69}\ S_{68}\ S_{67}\ S_{66}\ S_{65}\ S_{64}\ S_{63}\ S_{62}\ S_{61}\ S_{60}\ S_{41}\ S_{40}\ S_{00}\ a_0b_0$
REGISTER6
Output

Figure 6 Pipelined multiplier architecture (8 bit unsigned binary).

common for both the sections. Here the primary clock speed is *N* times the sample clock speed so that on arrival of each sample, all *N* frequency components can be computed. The input RAM blocks (1RX and 1RY) are single port RAM, which can store *N* words of 16-bit length. The data is written in the input RAM and initially all the flip-flops and counters are cleared to zero. The general scheme for subtraction of (*i*–*N*)-th data from *i*-th data is as follows. Since the RAM can store *N* data, so the address of writing *i*-th data is the same where previously (*i*–*N*)-th data is written. For a given address, first the previously stored data is latched in data register (1DX(Y)) then the new data is written in the same location in the RAM. The latched data is subtracted from the new data by subtractor to obtain ($R(i+1)-R(i-N+1)$). For each input arrived at sample clock rate (fixed *i*), *N* different rotation angles (different *k* values) are fed to the CORDIC block successively at primary clock rate so that rotation corresponding to all *N* angles is achieved and *N* updated frequency components (real and imaginary) are available at the output in successive clock cycles after the latency period of the CORDIC processor. After the rotation operation, CORDIC outputs (real & imaginary) are fed to the scale blocks (*X* and *Y*) for multiplying it by a factor $\prod_{i=0}^{15} Cos\ \theta i = 0.607252935$, to get the actual value of the

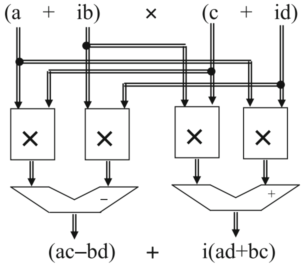

Figure 7 Complex multiplier architecture.

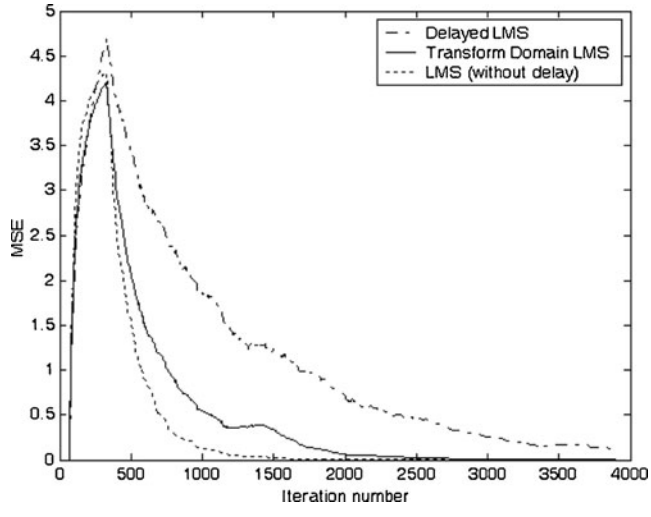

Figure 8 MSE convergence rate for different types of equalizer.

Table 1 Hardware utilization of proposed architecture.

N tap equalizer	16 bit adder	16 bit register	RAM (16 bit)	No. of blocks
CORDIC	43	43	–	1
Scale	76	78	–	2
Complex multiplier	18	18	–	2
Weight block	1	1	*N* word	1
Data RAM	–	–	*N* word	5

rotated output. After the scaling operation is done, the outputs are written in the dual port RAM blocks (2RX and 2RY). The data written corresponding to a particular frequency is to be read out again after the arrival of the next input data during the computation of the corresponding frequency component. This is done by keeping an offset between read and write addresses of the dual port RAM to cover the latency of the scale and CORDIC block. RAM block outputs are fed to adder, the other input of which is the new subtracted data ($R(i+1)$ – $\mathrm{R}(i–N+1)$). This subtracted data remaining same as one of the input of the adder, whereas the other inputs are the *N* frequency components corresponding to the previous input. The outputs of adder are fed back to the CORDIC block for the next set of rotation operations. The clock in the input RAM is the sample clock, which is obtained by a division of *N* of the primary clock, which is used in the other portion of the running DFT block.

3.1 Pipelined CORDIC Architecture

The CORDIC operation is done here through a pipelined structure, as shown in Fig. 4. There are three inputs at a time, real data (*X*), imaginary data (*Y*) and target angle (α). The total rotation is achieved here by decomposing the target angle α in elementary angles $\tan^{-1}(2^{-i})$, *i* starting from 0 to 15, and also in clockwise or anticlockwise direction for a particular *i*, as required. But as per normalized angle representation scheme [7], value of $\tan^{-1}(2^{-13})$ is equal to weight of LSB only, so rotation angle is decomposed up to this value. For rotation of each of these 14 angles, we have to perform three addition/subtraction operations along with right shifting (in two cases) as per Eq. 13. Since the amount of shift is fixed for a particular stage, it can be realized by bus cross connection only, without deploying additional hardware. The operation proceeds through stages in a pipelined structure, each stage performing a defined amount of rotation by taking data set from its previous stage, and new data set is entering in very first stage in each clock cycle. However, at the time of loading the target angle in the very first stage, the corresponding register copies MSB−1 bit as the MSB instead of the original value of MSB which has got an weightage equals to $-\pi$. This is done so as to keep the target angle α within the range $[-\pi/2, \pi/2]$. If the target angle is beyond that limit, i.e. if it lies either in the second or in the third quadrant, the sign changing of the output variables *X* and *Y* are to be carried out to incorporate the reflections about the axes, as described in [7]. The controlled sign change for the two outputs are done through controlled two's complementers. The pipe fills in 15 clock cycles, which defines its latency and then we get one set of outputs (*X* and *Y*) in every clock cycle, which defines its throughput. The maximum combinatorial delay in this CORDIC block for 16 bit internal arithmetic is delay of a 16 bit adder, which also defines its maximum clock speed as $f_{max}=1/T_{delay(adder)}$

3.2 Scale Block Architecture

The scale block is designed for a fixed amount of multiplication of the input and it is also implemented in a pipelined structure. Let us consider the structure of the scale block, shown in Fig. 5, which multiplies the input by the factor 0.607252935. So here the relation between the input and the output is as follows: $\mathrm{dout}=\mathrm{din}\times 0.607252935=\mathrm{din}.(2^{-1}+2^{-3}-2^{-6}-2^{-9}-2^{-12}+2^{-14})$, assuming 16 bit accuracy. Here the addition/subtraction operation is done in three stages. In the first stage, input din is right shifted by appropriate amounts (through RSH blocks by bus cross connection only), and then necessary additions/subtractions are carried out in this stage and also in the second and third stage, to get final output. This scale block is a pipeline structure and the operation proceeds here stage by stage in successive clock cycles. So the latency of the scale block is 3 clock cycles and throughput is one output per clock cycle. The maximum combinatorial delay in this scale block for 16 bit internal arithmetic is delay of a 16 bit adder which also defines its maximum clock speed as $f_{max}=1/T_{delay(adder)}$.

Table 2 Comparison between various architectures.

N tap equalizer	DFT_DLMS [9]	FD LMS [10]	Proposed
Adaptation delay	$(2N+1)\mathrm{T}_{\mathrm{sample}}$	$(N-1)\mathrm{T}_{\mathrm{sample}}$	$2\mathrm{T}_{\mathrm{sample}}=2N\mathrm{T}_{\mathrm{primary}}$
Critical period	$\mathrm{T}_{\mathrm{mult}}$	$\mathrm{T}_{\mathrm{mult}}+\mathrm{T}_{\mathrm{add}}$	$\mathrm{T}_{\mathrm{add}}$
No. of multipliers	$5N-2$	$16N$	8
No. of adders	$5N+1$	–	58