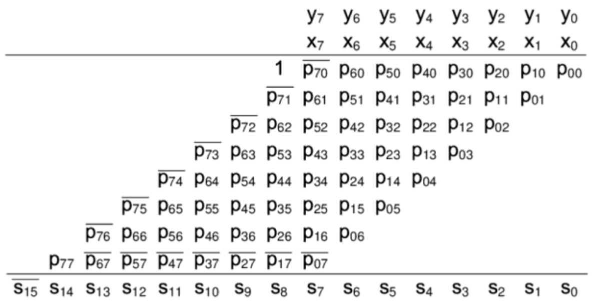
1. Designing a PE containing 9 approximate Baugh-Wooly multipliers (when the 5 least significant are truncated).



y7 y6 y5 y4 y3 y2 y1 y0
x7 x6 x5 x4 x3 x2 x1 x0
1 p70 p60 p50 p40 p30 p20 p10 p00
p71 p61 p51 p41 p31 p21 p11 p01
p72 p62 p52 p42 p32 p22 p12 p02
p73 p63 p53 p43 p33 p23 p13 p03
p74 p64 p54 p44 p34 p24 p14 p04
p75 p65 p55 p45 p35 p25 p15 p05
p76 p66 p56 p46 p36 p26 p16 p06
p77 p67 p57 p47 p37 p27 p17 p07
s15 s14 s13 s12 s11 s10 s9 s8 s7 s6 s5 s4 s3 s2 s1 s0

Figure 1-Baugh-Wooly multiplier architecture

Multiply-accumulate operation is widely used in DSP and image processing, and Deep Learning applications. The multiplier shown in Figure 1 is an 8-bit signed multiplier considered for inference of DNNs. **To apply approximation you have to truncate the 5 least significant columns**. In this project, you will perform the computations of a few fully connected layers of a DNN when both the input feature maps and filter weights are quantized to 8-bit. Then you have to calculate and report the Normalized Mean Error Distance (NMED) of these layers compared to when the exact multipliers are used. You can calculate the NMED of each layer using the equation provided in (1). Also, you have to report the area, and power consumption of the approximate Baugh-Wooly multiplier and compare it with the exact one. The 8-bit quantized format of input feature maps will be provided. To have the final results of convolution, you can simply sign extend the multiplication results and add them using behavioral language (the considered bit-width of the total partial sum is 24 bits).

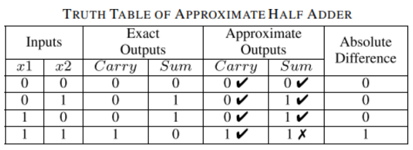
$$E_{MAC} = \sum_{l=1}^{c} \sum_{i=1}^{w} \sum_{j=1}^{h} Approx.(I_{lij} \times K_{lij}) - (I_{lij} \times K_{lij}) \qquad (1)$$

$$NMED = \frac{Mean(|E_{MAC}|)}{MAX(\sum_{l=1}^{c} \sum_{i=1}^{w} \sum_{j=1}^{h} I_{lij} \times K_{lij})}$$

The details of Baugh-Wooly multiplier can be found in the following paper:

Mohanty, P., 2013. An Efficient Baugh-Wooley Architecture for Signed & Unsigned Fast Multiplication.*NIET Journal of Engineering & Technology*,*1*(2).

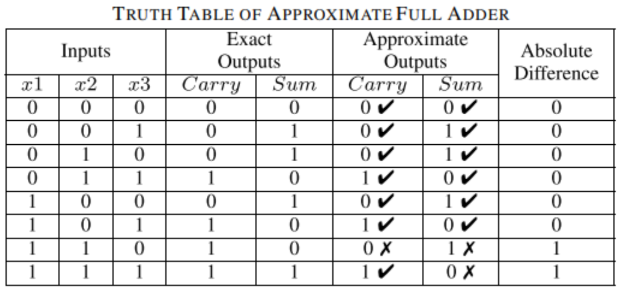
2. Designing a PE containing 9 Baugh-Wooly multipliers (when approximate Half Adders and Full Adders are used).



TRUTH TABLE OF APPROXIMATE HALF ADDER

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0 ✔	0 ✔	0
0	1	0	1	0 ✔	1 ✔	0
1	0	0	1	0 ✔	1 ✔	0
1	1	1	0	1 ✔	1 ✗	1

Figure 2- Approximate Half Adder architecture



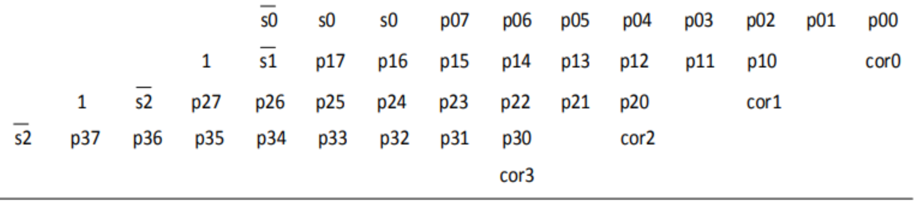
TRUTH TABLE OF APPROXIMATE FULL ADDER

Inputs			Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$x3$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0	0 ✔	0 ✔	0
0	0	1	0	1	0 ✔	1 ✔	0
0	1	0	0	1	0 ✔	1 ✔	0
0	1	1	1	0	1 ✔	0 ✔	0
1	0	0	0	1	0 ✔	1 ✔	0
1	0	1	1	0	1 ✔	0 ✔	0
1	1	0	1	0	0 ✗	1 ✗	1
1	1	1	1	1	1 ✔	0 ✗	1

Figure 3- Approximate Full adder architecture

Multiply-accumulate operation is widely used in DSP and image processing, and Deep Learning applications. The multiplier shown in Figure 1 is an 8-bit signed multiplier considered for inference of DNNs. In this project, you will perform the computations of a few fully connected layers of a DNN when both the input feature maps and filter weights are quantized to 8-bit. **You have to use approximate Half Adders and Full Adders to perform the accumulation of the 6 least significant columns**. **The truth table of these units is provided in Figures 2 and 3.** Then you have to calculate the Normalized Mean Error Distance (NMED) of these layers compared to when the exact multiplies are used. You can calculate the NMED using the equation provided in (1) for each layer. Also, you have to report the area, and power consumption of the approximate Baugh-Wooly multiplier and compare it with the exact one. The 8-bit quantized format of input feature maps will be provided. To have the final results of convolution, you can simply sign extend the multiplication results and add them using behavioral language (the considered bit-width of the total partial sum is 24 bits).
The details of approximate Half Adder and Full Adder can be found in the following paper:
Venkatachalam, S. and Ko, S.B., 2017. Design of power and area efficient approximate multipliers. IEEE Transactions on Very Large Scale Integration (VLSI) Systems,25(5), pp.1782-1786.

3. Designing a PE containing 9 approximate radix-4 8-bit Booth multipliers (When the partial product in the 5 least columns are replaced by '1')



				$\overline{s0}$	s0	s0	p07	p06	p05	p04	p03	p02	p01	p00
			1	$\overline{s1}$	p17	p16	p15	p14	p13	p12	p11	p10		cor0
	1	$\overline{s2}$	p27	p26	p25	p24	p23	p22	p21	p20		cor1		
$\overline{s2}$	p37	p36	p35	p34	p33	p32	p31	p30		cor2				
								cor3						

Figure 4- Internal architecture of an exact radix-4 8-bit Booth multiplier

Multiply-accumulate operation is widely used in DSP and image processing, and Deep Learning applications. The multiplier shown in Figure 4 is an 8-bit Booth multiplier considered for inference of DNNs. To apply the approximation the partial product in the 5 least significant columns is replaced with '1'. In this project, you will perform the computations of a few fully connected layers of a DNN when both the input feature maps and filter weights are quantized to 8-bit**.** Then you calculate the Normalized Mean Error Distance (NMED) of these layers compared to when the exact multipliers are used. You can calculate the NMED using the equation provided in (1) for each layer. Also, you have to report the area, and power consumption of the Booth multiplier and compare it with exact one. The

8-bit quantized format of input feature maps will be provided. To have the final results of convolution, you can simply sign extend the multiplication results and add them using behavioral language (the considered bit-width of the total partial sum is 24 bits).

4. Designing a PE containing 9 approximate radix-4 8-bit Booth multipliers (when approximate Half Adder and Full adder are used).
   Multiply-accumulate operation is widely used in DSP and image processing, and Deep Learning applications. The multiplier shown in Figure 4 is an 8-bit Booth multiplier considered for inference of DNNs. In this project, you will perform the computations of a few fully connected layers of a DNN when both the input feature maps and filter weights are quantized to 8-bit. **You have to use approximate Half Adders and Full Adders to perform the accumulation of the 5 least significant columns**. **The truth table of these units is provided in Figures 2 and 3.** Then you calculate the Normalized Mean Error Distance (NMED) of these layers compared to when exact multipliers are used. You can calculate the NMED using the equation provided in (1) for each layer. Also, you have to report the area, and power consumption of the approximate Booth multiplier and compare it with exact one. The 8-bit quantized format of input feature maps will be provided. To have the final results of convolution, you can simply sign extend the multiplication results and add them using behavioral language(the considered bit-width of the total partial sum is 24 bits).

5. Designing a PE containing 9 approximate Booth multipliers (when approximate partial product generators are used).

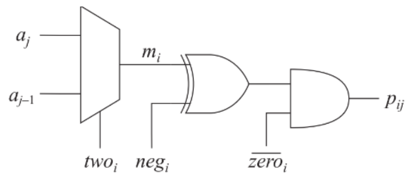


Figure 5- Exact Radix-4 partial product generator

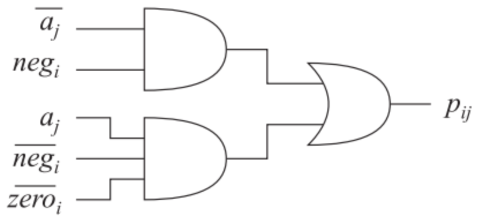


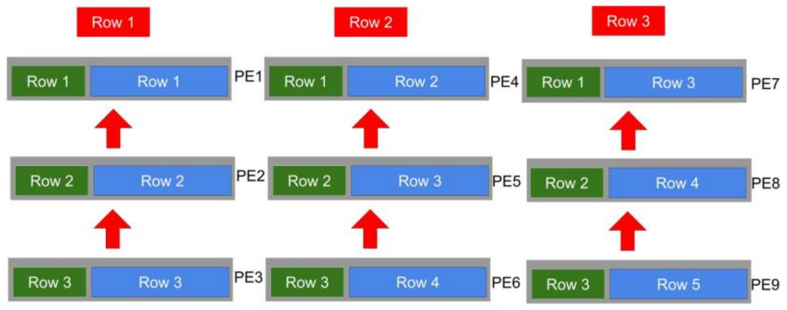
Figure 6- Inexact radix-4 partial product generator

Multiply-accumulate operation is widely used in DSP and image processing, and Deep Learning applications. The multiplier shown in Figure 4 is an 8-bit Booth multiplier considered for inference of DNNs. In this project, you will perform the computations of a few fully connected layers of a DNN when both the input feature maps and filter weights are quantized to 8-bit. **You have to use an approximate partial product generator in the 5 least significant columns.** Then you calculate the Normalized Mean Error Distance (NMED) of these layers compared to when the exact multiplier is used. You can calculate the NMED using the equation provided in (1) for each layer. Also, you have to report the area, and power consumption of the approximate Booth multiplier and compare it with exact one. The 8-bit quantized format of input feature maps will be provided. To have the final results of convolution, you can simply sign extend the multiplication results and add them using behavioral language.

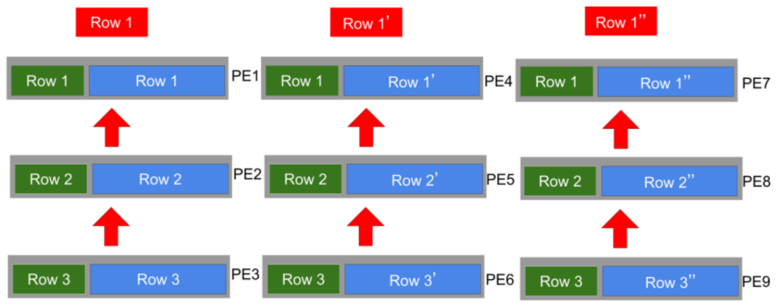
The details of the approximate partial product generator in Radix-4 Booth algorithm can be found in the following paper:

Venkatachalam, S., Adams, E., Lee, H.J. and Ko, S.B., 2019. Design and analysis of area and power-efficient approximate booth multipliers. *IEEE Transactions on Computers*, *68*(11), pp.1697-1703.

6. Row Stationery (RS) dataflow (graduate)



(a)



(b)

Figure 7- the dataflow when (a) all the data types are reused (RS dataflow), and (b) when input feature maps are not reused locally. Filter rows are distinguished by green color, ifmaps are distinguished by blue color, and partial sum are distinguished by red color.

Local reuse of different data types is a common method for decreasing energy consumption while performing inference of DNNs. In Figure 7, each PE has 1 MAC unit and performs the computations of each row containing 3 elements. The data flow is shown in Figure 7 (a) is called RS data flow. In this project, you will perform the computations of a few fully connected layers of a DNN when both the input feature maps and filter weights are quantized to 8-bit. As shown in Figure 7, you will perform the computations of 3 output features concurrently. You can calculate the number of memory accesses compared to dataflow shown in figure 7 (b). Furthermore, you have to report the energy consumption when RS dataflow is implemented and when input feature maps are not used locally. In addition, you have to report the area, and power consumption of the design. The 8-bit quantized

format of input feature maps will be provided. To have the final result of convolution, you can simply sign extend the multiplication results and add them using behavioral language.
The details of RS dataflow can be found in the following paper:
Chen, Y.H., Krishna, T., Emer, J.S. and Sze, V., 2016. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, *52*(1), pp.127-138.

7. Fast convolution/dot-product with Winograd algorithm (graduate)
Convolution is the basic arithmetic operation of a deep convolutional neural network. In a deep convolutional neural network, the convolution is a dot-product operation. For example, we have a 4x4 input image, a 3x3 filter. We can generate a 2x2 output, as shown below.

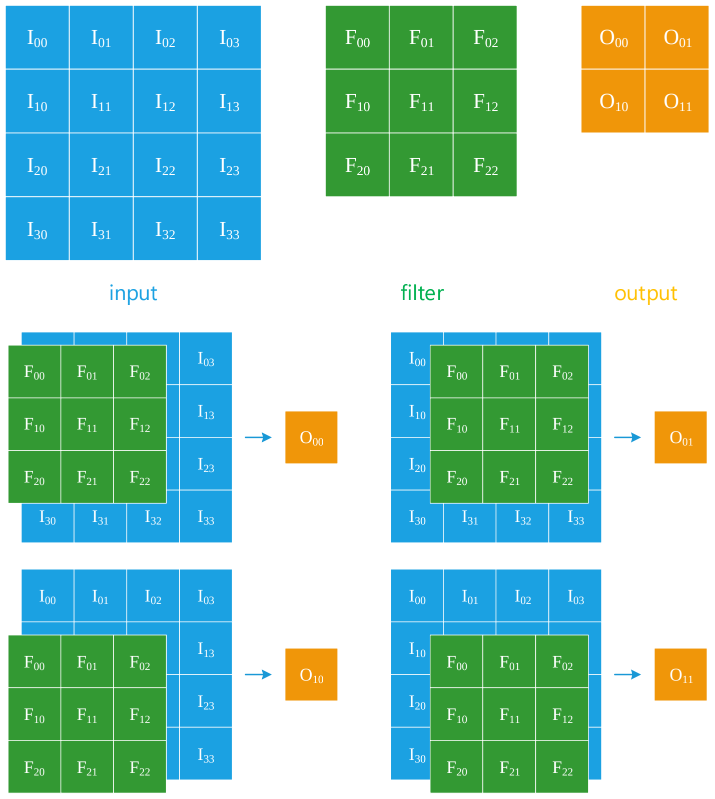

I_{00} I_{01} I_{02} I_{03}
I_{10} I_{11} I_{12} I_{13}
I_{20} I_{21} I_{22} I_{23}
I_{30} I_{31} I_{32} I_{33}
F_{00} F_{01} F_{02}
F_{10} F_{11} F_{12}
F_{20} F_{21} F_{22}
O_{00} O_{01}
O_{10} O_{11}
input
filter
output

Figure 8-Convolution for a 3x3 filter and 4x4 input feature map

To generate one output, the filter and part of the input are multiplied element-wise and the 9 products are then added together. The filter then slides through the whole input to generate all outputs. As we see, to generate a 2x2 output, we need 36 multiplication operations.

Winograd algorithm is basically used to reduce the number of multiplications. For the same operation, Winograd algorithm first converts input matrix and filter matrix to 4x4 transformed matrix, respectively. Then the two transformed matrices are multiplied element-wise. The 4x4 result matrix is then transformed back to 2x2 matrix which is the final result of the convolution operation. Therefore, in Winograd algorithm, you only need to perform 16 multiplications but with the overhead of transformations (addition and shifting).

The details of Winograd algorithm can be found in the following paper:
Andrew Lavin, Scott Gray, "Fast Algorithms for Convolutional Neural Networks," *arXiv preprint*, arXiv:1509.09308v2, Nov 2015.

You can implement the row-wise F(2x2, 3x3) algorithm in hardware and compare the results with the conventional convolution algorithm. (Hint: the transformation can be implemented as shifting and addition instead of multiplication and addition. You can also use format F(2,3) of Winograd for implementation)

8. Fused approximate Booth Multiply-Accumulate (MAC) unit
It is possible to fuse multiplication and accumulation operations. Figure 9 shows the internal architecture of a simple MAC unit. In more detail, the result of the multiplication can be in a fused format, accumulate with the previous result, and perform the final addition. Assume you need to round the final result to save in storage. The multiplier shown in Figure 4 is an 8-bit Booth multiplier considered for inference of DNNs. **In this project to apply approximation, correction terms are removed**. Furthermore, In this project, you will perform the computations of a few fully connected layers of a DNN when both the input feature maps and filter weights are quantized to 8-bit. Then you calculate the Normalized Mean Error Distance (NMED) of these layers compared when exact multiplier is used. You can calculate the NMED using the equation provided in (1) for each layer. Also, you have to report the area, and power consumption of the approximate Booth multiplier and compare it when you use the exact Booth multiplier and a separate adder. The 8-bit quantized format of input feature maps and the Matlab output results will be provided. Compare the speed, area, power, and NMED with separate exact Booth multiplier and adder implementation.

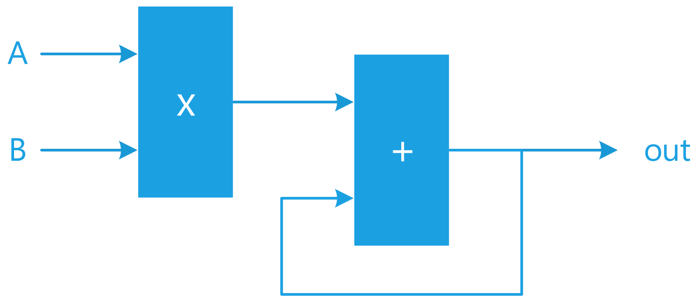


Figure 9- A simple MAC unit

9. Approximate fixed-point sum-of-product unit
To improve the performance, more parallelism is preferred. The sum-of-product unit can perform multiple multiplications in parallel and then accumulate and sum these products. Figure 10 shows the internal architecture of a circuit that performs the sum of product for rows containing 3 elements. In this project, only 1 of the multipliers is in exact format (shown in Figure 1) and the other 2 are in the approximate format (multiplier in project 2). In this project, you will perform the computations of a few fully connected layers of a DNN when both the input feature maps and filter weights are quantized to 8-bit. Then you calculate the Normalized Mean Error Distance (NMED) of these layers compared to when the exact multipliers are used. You can calculate the NMED using the equation provided in (1)

for each layer. Also, you have to report the area, and power consumption of the approximate multipliers (multiplier in project 2) and compare it when you only use Baugh-Wooly multiplier (shown in Figure 1). The 8-bit quantized format of input feature maps will be provided.

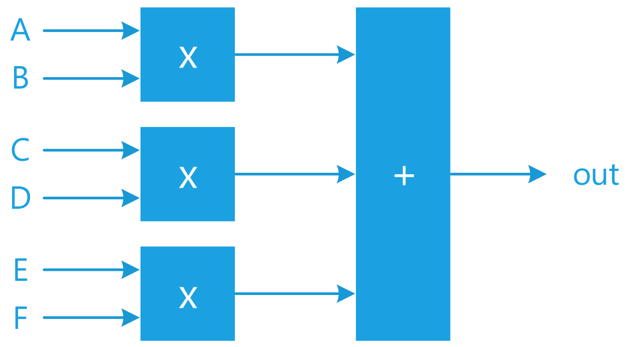


Figure 10- A circuit that performs sum of product for rows containing 3 elements

10. Fast Fixed-Point sum-of-product unit

    To improve the performance, more parallelism is preferred. The sum-of-product unit can perform multiple multiplications in parallel and then accumulate and sum these products. The result of each multiplication must be in carry-save format. The diagram of a 3-way sum-of-product unit is shown in Figure 10. In this project, you will perform the computations of a few fully connected layers of a DNN when both the input feature maps and filter weights are quantized to 8-bit. Then you calculate the Normalized Mean Error Distance (NMED) of these layers compared to when exact multiplier is used. You can calculate the NMED using the equation provided in (1) for each layer. The 8-bit quantized format of input feature maps will be provided. Compare the area, power, and, speed with when the multiplication result is not in carry-save format.