In this thesis, the proposed idea is about hardware implementation of hyperbolic tangent activation function using a LUT based solution where the input is in floating point format. The suggestion robust the implementation of non-linear activation functions. This thesis introduces unique way for hardware implementation of hyperbolic tangent by approximate function in MATLAB, which reduces the hardware cost while implementation. Furthermore, input data points are used as keys or addresses for memory. Design options with different properties settings submit variant delays and accuracy and eventually the best and optimized one will be selected.

There are two approximation methods in this section, Lookup Table (LUT) approximation and Range Addressable Lookup Tables (RALUT). Lookup table approach is approximated by several points which are distributed uniformly and stored in memory tables. RALUTs are very similar to LUTs except some differences. In the LUTs every point is stored in the tables and has link to one address while in RALUTs every point has link to the range of addresses. This grouping different points to one address reduce implementation cost, because there are many points which can be assigned to one output. In this example the bound of x-axis is between -4 and 4 for hyperbolic tangent and as it is obvious points between (-4, -2) and (2, 4) have been determined almost to one output, somehow points in (-4, -2) are -1 and data points in (2, 4) are equal to 1. Therefore, the point between (-2, 2) are selected to implement.

There is FunctionApproximation package in MATLAB R2020a which defines the function to approximate, or the lookup table to optimize. After problem definition by FunctionApproximation using the solve method can generate a FunctionApproximation.LUTSolution object that includes the approximation.

The FunctionToApproximate property is set to 'Tanh' which is hyperbolic tangent. It is possible to specify a math function, a function handle, or a block. There are several properties which must be assigned as InputTypes (data type for inputs), OutputType (desired data type for outputs), InputLowerBounds and InputUpperBounds are assigned by the bound of the function which is here (-4, 4)

Inputtype = numerictype (1,16,15)

Numerictype (s, w, f) creates a fixed-point with binary point scaling, s shows a signed property value, w is word length of, and f is fraction length.

BreakpointSpecification is spacing of breakpoint data, has three values as ExplicitValues, EvenSpacing and EvenPow2Spacing. In the explicit value, breakpoints can be closer together for some input ranges and farther apart in others. In the evenspacing, breakpoints are evenly spaced throughout and in the EvenPow2Spacing lookup table breakpoints apply power-of-two spacing. This breakpoint specification enhances the fastest execution speed because a bit shift can replace the position search.

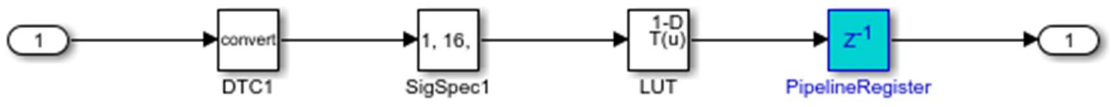
In this approach interpolation=flat has been used because of it elevates speed dramatically and delay will be about 1.5 ns which is acceptable result. Flat interpolation is used when an input falls between breakpoint values, the lookup table interpolates the output value using neighboring breakpoints and flat returns the output value corresponding to the breakpoint which is immediately less than the input one. If there is no breakpoint below the input value, so, it returns the breakpoint nearest the input value.

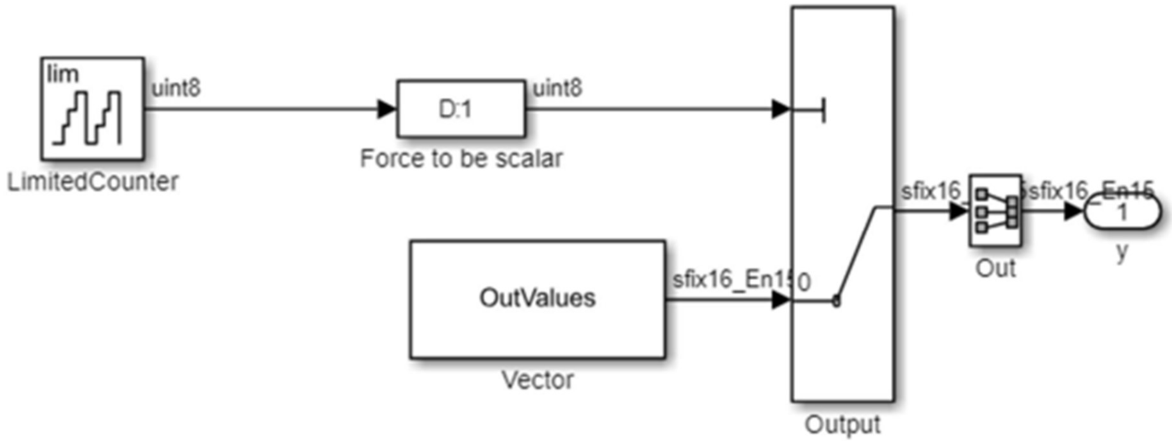
SaturateToOutputType can saturate the output of a function to approximate to the range of output type.

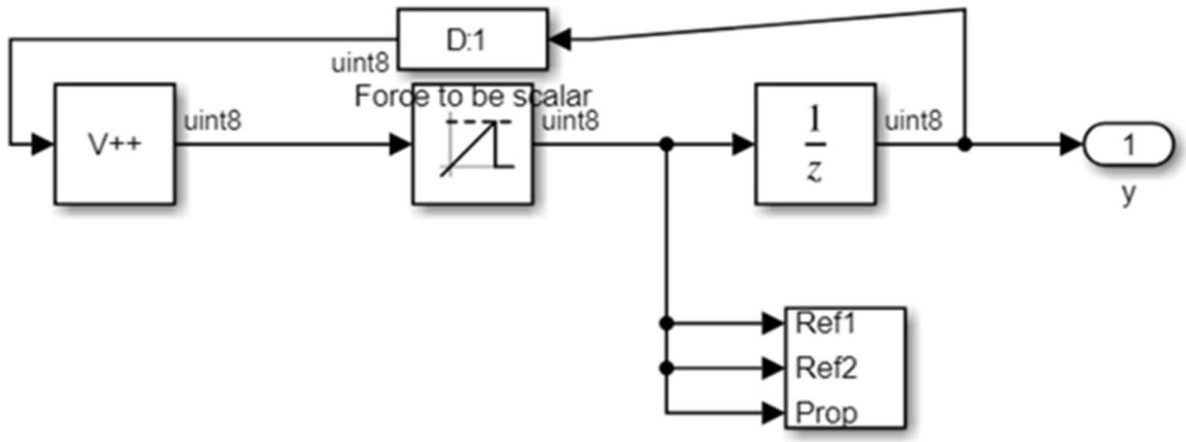
WordLengths specifies the word lengths, in bits, which can be used in the lookup table approximate based on user intended hardware. The word lengths must be between 1 and 128.

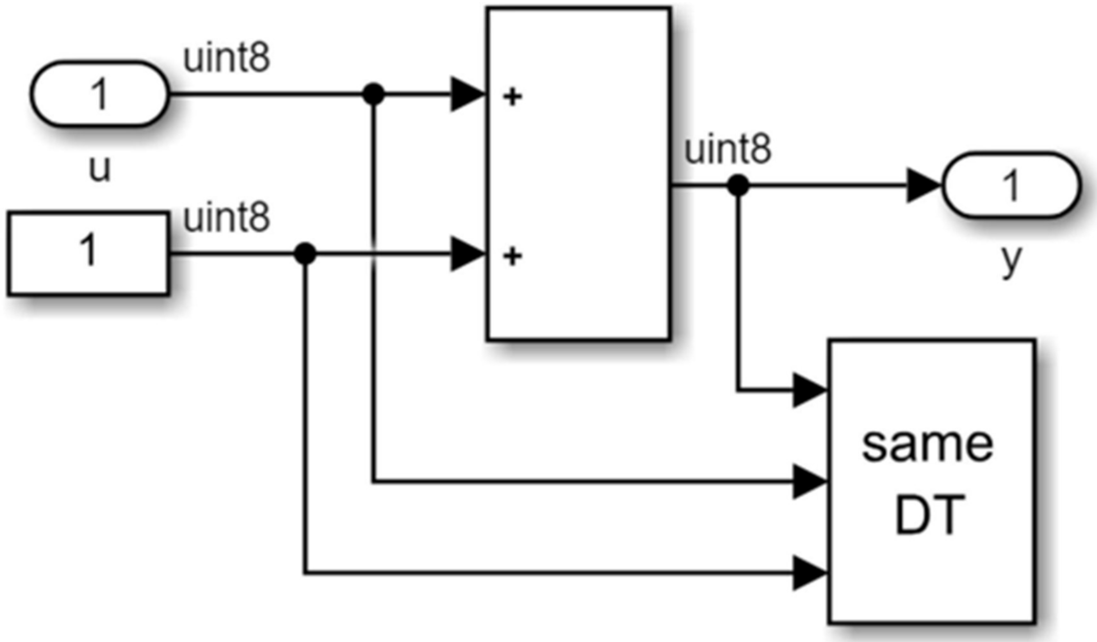
Architecture	Resources	Delay
LUT		2.5 ns
RALUT		1.5 ns

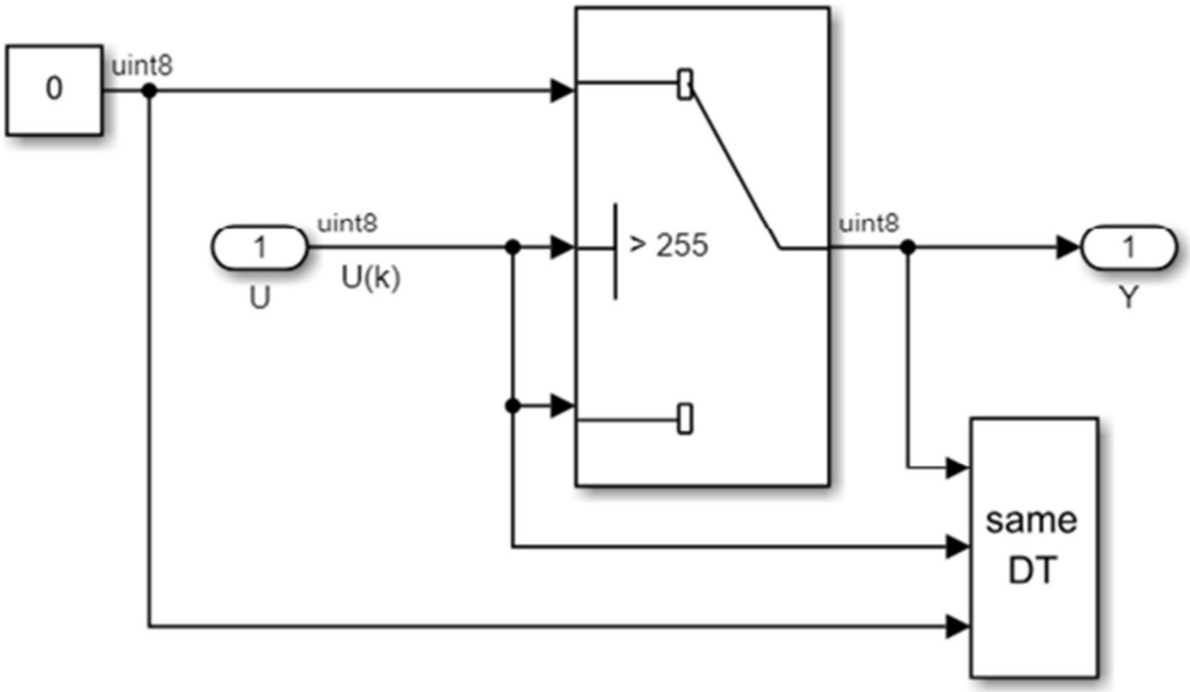
The extracted block diagrams are as follow:






D:1
uint8
V++
uint8
Force to be scalar
uint8
1/z
uint8
1
y
Ref1
Ref2
Prop


1
u
uint8
1
uint8
+
+
uint8
1
y
same DT


0
uint8
1
U
uint8
U(k)
> 255
uint8
1
Y
same
DT