For instance, let’s assume that we are going to represent the 4-bit numbers in the following fashion

This means the multiplicand is (22 A + B) and the multiplier is (22 C + D). Thus, the complete product would look as follows –

(22 A + B) (22 C + D) = (24 (A.B) + 22 (B.C + A.D) + B.D)

The entire product is now computed based on either temporal or spatial or spatio-temporal fusion architectures. So, for any 6-bit number that could be represented as follows –

Here, U, V and W are the bit-bricks of the multiplicand and X, Y and Z are the same for the multiplier. Thus, the entire product is now computed as –

**(24 U + 22 V + W) (24 X + 22 Y + Z) = (28 (U.X) + 26 (U.Y+V.X) + 24 (U.Z+W.X+V.Y) + 22 (W.Y + V.Z) + W.Z**

Splitting the 6-bit multiplication into three 2-bit computations, we get the following three numbers

1. The first one is the multiplication of -
2. The second one is the multiplication of –
3. The third one is the multiplication of –

The new computation has been divided into the following components –

**24 (24 (U.X) + 22 (U.Y+V.X) + (V.Y))** **+**

**22** [**22** **(U.Z + W.X)]** **+**

**22 [(W.Y + V.Z)]** + **W.Z**

The first partial sum in the entire computation could be computed from the partial products present in 1. The second and the third computations could be implemented in the other bit-bricks available within the architecture. The entire diagram for the updated architecture looks as follows –

**WZ**

**VZ**

**00**

**VZ**

**VX**

**UX**

**VY**

**UY**

**00**

**00**

**WX**

**WY**

**00**

**00**

**00**

**00**

**Figure 1: Scheduling of bits to different bit bricks for 6b x 6b multiplication**

For the other configurations such as 6b x 2b, 6b x 4b and 6b x 10b, we get the other diagrams depicted in figures 2 through 4.

* For the 6b x 2b case, the computation looks like – **[24 (U.X) + 22 (V.X) + W.X]**

**WX**

**VX**

**00**

**00**

**00**

**00**

**00**

**00**

**UX**

**00**

**00**

**00**

**00**

**00**

**00**

**00**

**Figure 2: Scheduling of bits to different bit bricks for 6b x 2b multiplication**

As it can be seen from figures 1 and 2, there are a lot of resources that are under-utilized when we are performing 6b x 6b or 6b x 2b multiplications. This could be utilized in a more efficient manner and that’s the focus of our work. Thus, we wish to efficiently utilize the green boxes by filling it with data values corresponding to other fusion units.  
  
  
For the **6b x 4b** case, the computation looks like – **22 [24 (U.X) + 22 (U.Y+V.X) + (V.Y + W.X)] + (W.Y)**

**WY**

**VY**

**00**

**UY**

**00**

**00**

**00**

**UX**

**VX**

**WX**

**00**

**00**

**00**

**00**

**00**

**00**

**Figure 3:** **Scheduling of bits to different bit bricks for 6b x 4b multiplication**

**Report guidelines**

How did you get the delay numbers

which component is taking how much time

how do you integrate the 1 simulated layer with the non-simulated layers (keras)

no need of entire time accurate simulator

just report the cycles needed for each combination of input and weight (2,4,6,8)

give out additions in terms of hardware over existing bitfusion