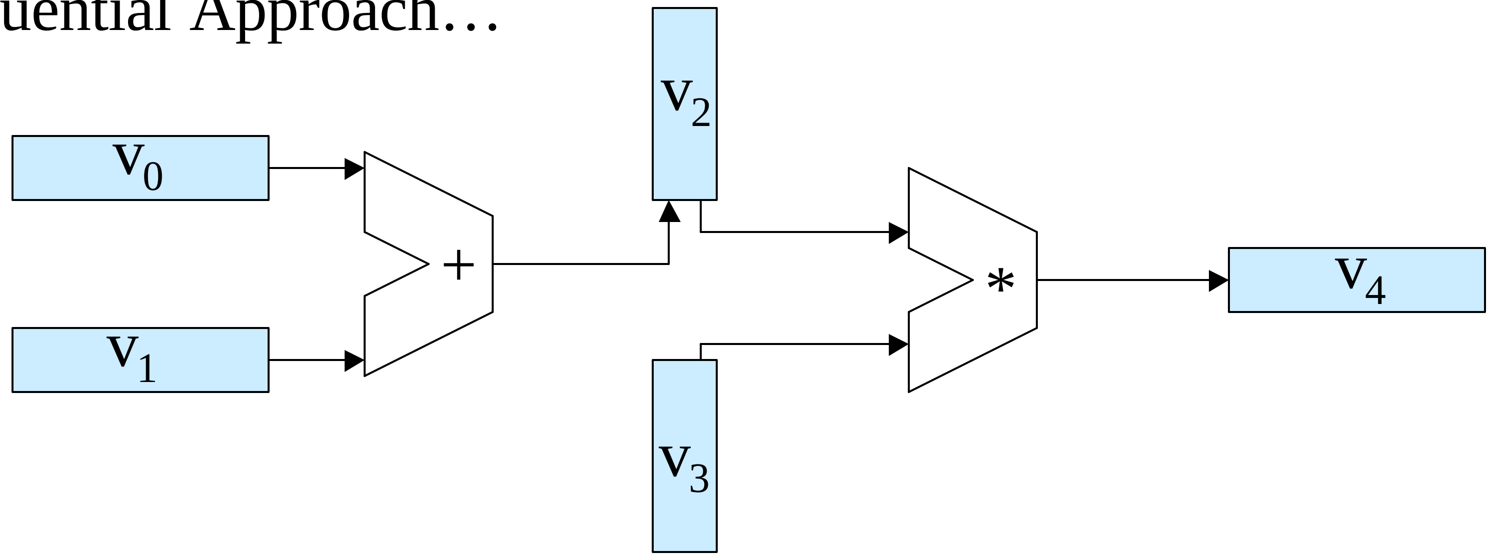
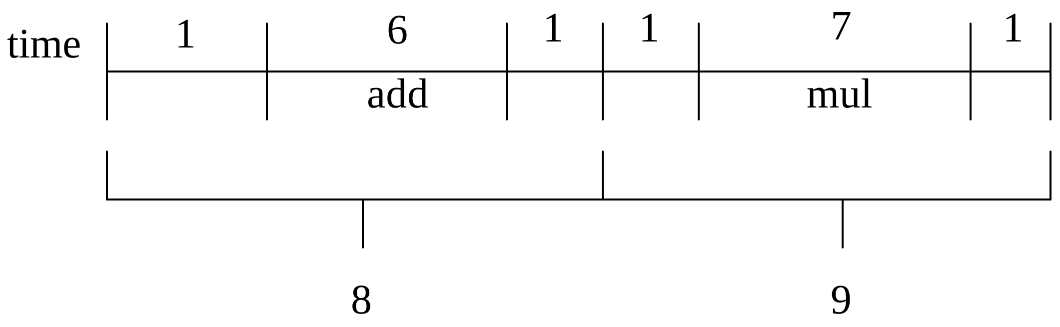
More Chaining and Storing Matrixes

February 16th 2004

Dusty Price

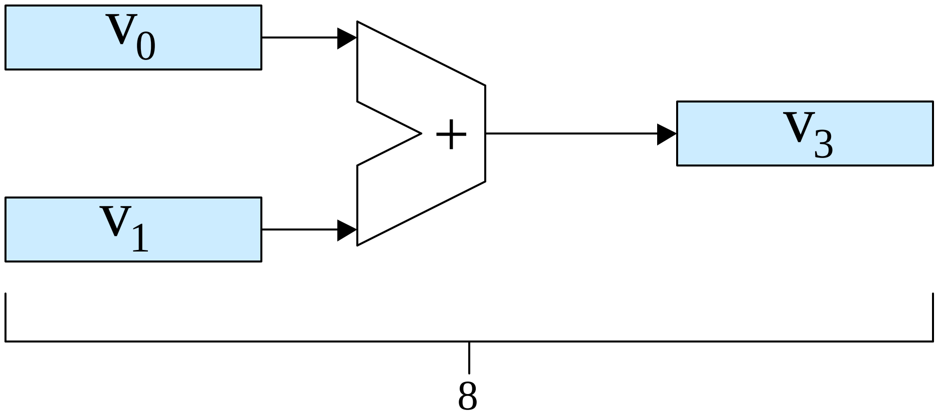
Sequential Approach…



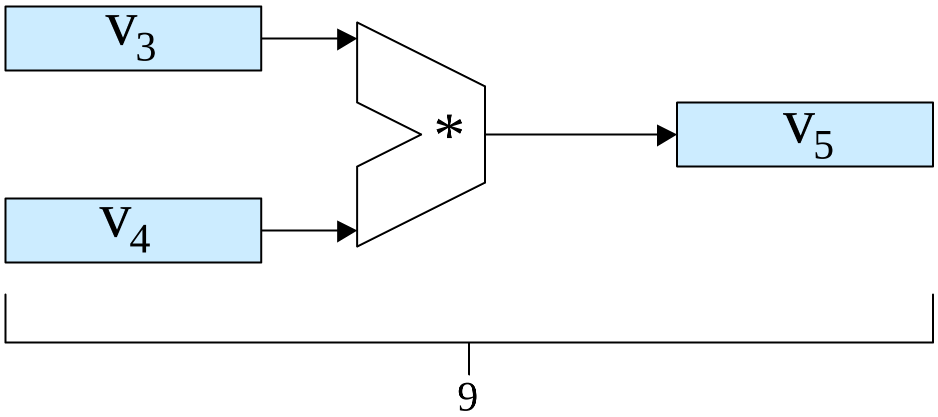


64 Elements in sequence: $T_s = 64 * (8 + 9) = 1088$

Using Pipeline Approach…



Using pipelining it takes 8 units of time to fill pipeline and produce first result, each unit of time after that produces another result
$T_{p+} = 8 + 63$

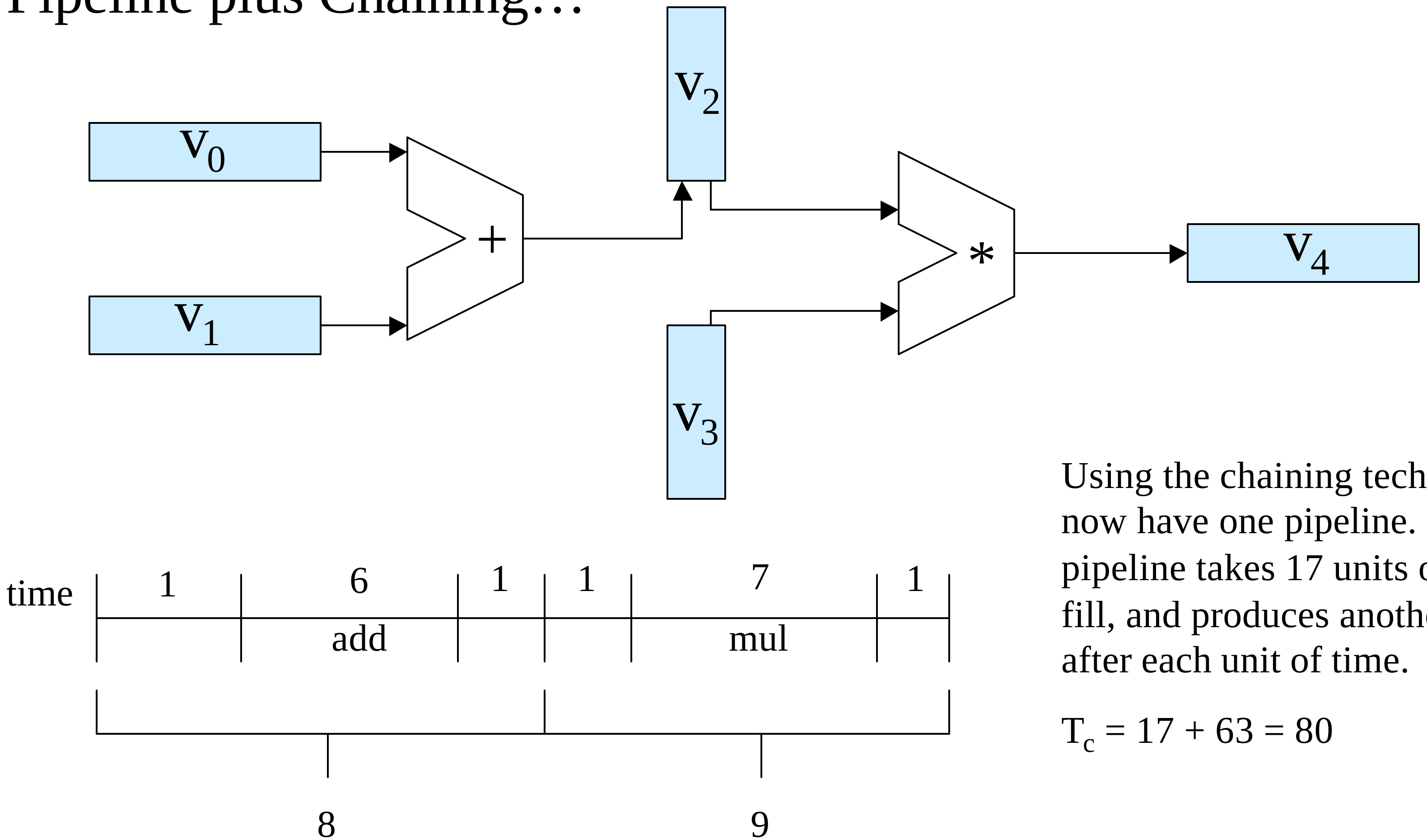


The multiplication pipeline takes 9 units of time to fill, and produces another result after each additional unit of time

$T_{p*} = 9 + 63$

The combination of the two $T_p = T_{p+} + T_{p*} = 8 + 63 + 9 + 63 = 143$

Pipeline plus Chaining…



Using the chaining technique, we now have one pipeline. This new pipeline takes 17 units of time to fill, and produces another result after each unit of time.

$T_c = 17 + 63 = 80$

Operation using Chaining $T_c = 17 + 63 = 80$

Review of time differences in the three approaches…

Sequential: $T_s = 17 * 64 = \boxed{1088}$

Pipelining: $T_p = 8 + 63 + 9 + 63 = \boxed{143}$

Chaining: $T_c = 17 + 63 = \boxed{80}$

Storing Matrixes in a SISD Architecture w/ Memory Interleaving…

Matrix

A_{11}	A_{12}	A_{13}	A_{14}
A_{21}	A_{22}	A_{23}	A_{24}
A_{31}	A_{32}	A_{33}	A_{34}
A_{41}	A_{42}	A_{43}	A_{44}

4 Memory Modules

M_1	M_2	M_3	M_4
A_{11}	A_{21}	A_{31}	A_{41}
A_{12}	A_{22}	A_{32}	A_{42}
A_{13}	A_{23}	A_{33}	A_{43}
A_{14}	A_{24}	A_{34}	A_{44}

One column of the matrix can be accessed at one time.

Storing the Matrix by Column…

Matrix

A_{11}	A_{12}	A_{13}	A_{14}
A_{21}	A_{22}	A_{23}	A_{24}
A_{31}	A_{32}	A_{33}	A_{34}
A_{41}	A_{42}	A_{43}	A_{44}

4 Memory Modules

M_1	M_2	M_3	M_4
A_{11}	A_{12}	A_{13}	A_{14}
A_{21}	A_{22}	A_{23}	A_{24}
A_{31}	A_{32}	A_{33}	A_{34}
A_{41}	A_{42}	A_{43}	A_{44}

One Row can be accessed at a time with this storage technique.

Sometimes we need to access both rows and columns fast…

Matrix

A_{11}	A_{12}	A_{13}	A_{14}
A_{21}	A_{22}	A_{23}	A_{24}
A_{31}	A_{32}	A_{33}	A_{34}
A_{41}	A_{42}	A_{43}	A_{44}

4 Memory Modules

M_1	M_2	M_3	M_4
A_{11}	A_{12}	A_{13}	A_{14}
A_{24}	A_{21}	A_{22}	A_{23}
A_{33}	A_{34}	A_{31}	A_{32}
A_{42}	A_{43}	A_{44}	A_{41}

By using a skewed matrix representation, we can now access each row at a time, as well as access each column at a time.

Sometimes we need access to the main diagonal as well as rows and columns…

Matrix

A_{11}	A_{12}	A_{13}	A_{14}
A_{21}	A_{22}	A_{23}	A_{24}
A_{31}	A_{32}	A_{33}	A_{34}
A_{41}	A_{42}	A_{43}	A_{44}

5 Memory Modules

M_1	M_2	M_3	M_4	M_5
A_{11}	A_{12}	A_{13}	A_{14}	
	A_{21}	A_{22}	A_{23}	A_{24}
A_{34}		A_{31}	A_{32}	A_{33}
A_{43}	A_{44}		A_{41}	A_{42}

At the cost of adding another memory module and wasted space, we can now access the matrix by row, column, and main diagonal.