9.5 Alternative Parallel Approaches

- Some people argue that real breakthroughs in computational power-- breakthroughs that will enable us to solve today's intractable problems-- will occur only by abandoning the von Neumann model.
- Numerous efforts are now underway to devise systems that could change the way that we think about computers and computation.
- In this section, we will look at three of these: dataflow computing, neural networks, and systolic processing.

9.5.1 Dataflow Computing

- Von Neumann machines exhibit sequential control flow: A linear stream of instructions is fetched from memory, and they act upon data.
- Program flow changes under the direction of branching instructions.
- In dataflow computing, program control is directly controlled by data dependencies.
- There is no program counter or shared storage.
- A *data flow* graph represents the computation flow in a dataflow computer.
- Its nodes contain the instructions and its arcs indicate the data dependencies.

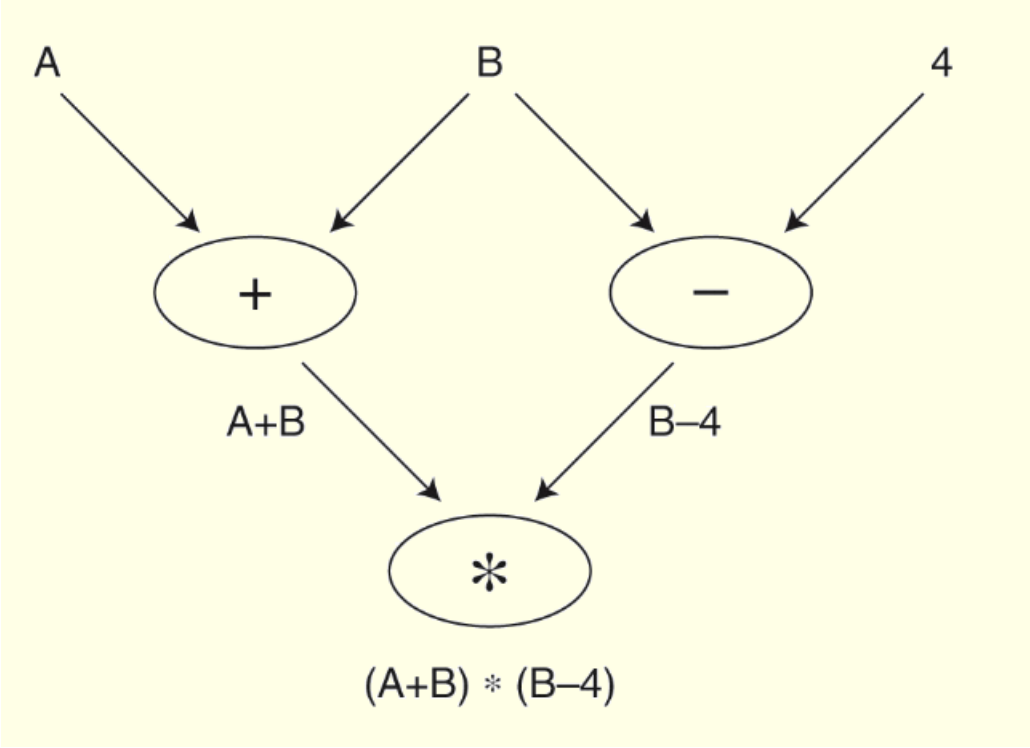

Figure 9.11 Dataflow Graph Computing
$N = (A + B) * (B - 4)$

- Data flows continuously and is available to multiple instructions simultaneously

- When a node has all of the data tokens it needs, it fires, performing the required operation, and consuming the token.

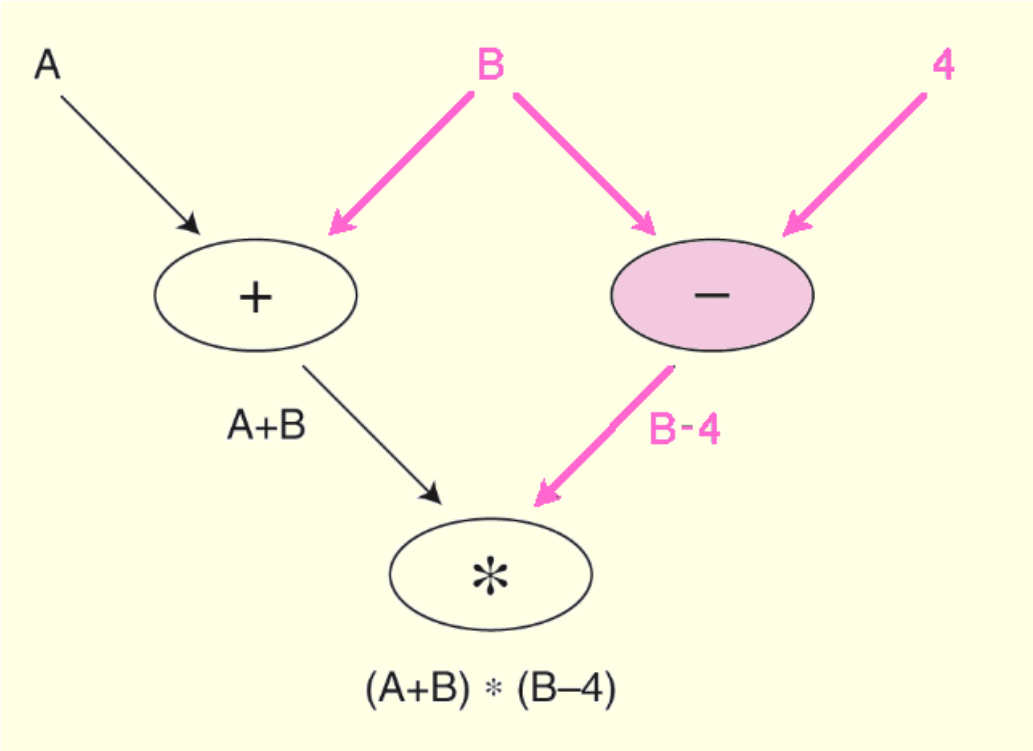

Figure 9.11.1 Dataflow Graph Computing – A Node Firing

- The result is placed on an output arc.

- A dataflow program to calculate $n!$ and its corresponding graph are shown below.

```
(  initial j <- n; k<-1;
   while j > 1 do
      new k <- k*j;
      new j <- j-1;
   return k)
```

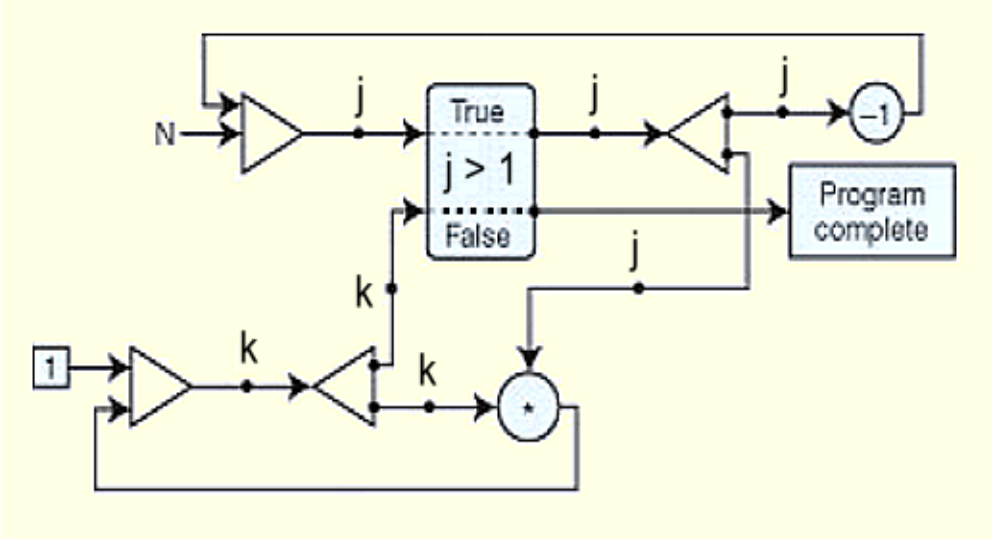

Figure 9.12 Dataflow Graph Corresponding to the Program to Calculate $n!$

- The architecture of a dataflow computer consists of processing elements that communicate with one another.
- Each processing element has an enabling unit that sequentially accepts tokens and stores them in memory.
- If the node to which this token is addressed fires, the input tokens are extracted from memory and are combined with the node itself to form an executable packet.
- Using the executable packet, the processing element's *functional unit* computes any output values and combines them with destination addresses to form more tokens.
- The tokens are then sent back to the enabling unit, optionally enabling other nodes.
- Because dataflow machines are data driven, multiprocessor dataflow architectures are not subject to the cache coherency and contention problems that plague other multiprocessor systems.

9.5.2 Neural Networks

- Neural network computers consist of a large number of simple processing elements that individually solve a small piece of a much larger problem.
- They are particularly useful in dynamic situations that are an accumulation of previous behavior, and where an exact algorithmic solution cannot be formulated.
- Like their biological analogues, neural networks can deal with imprecise, probabilistic information, and allow for adaptive interactions.
- Neural network processing elements (PEs) multiply a set of input values by an adaptable set of weights to yield a single output value.
- The computation carried out by each PE is simplistic-- almost trivial-- when compared to a traditional microprocessor. Their power lies in their massively parallel architecture and their ability to adapt to the dynamics of the problem space.
- Neural networks learn from their environments. A built-in *learning algorithm* directs this process.
- The simplest neural net PE is the *perceptron*.
- Perceptrons are trainable neurons. A perceptron produces a Boolean output based upon the values that it receives from several inputs.

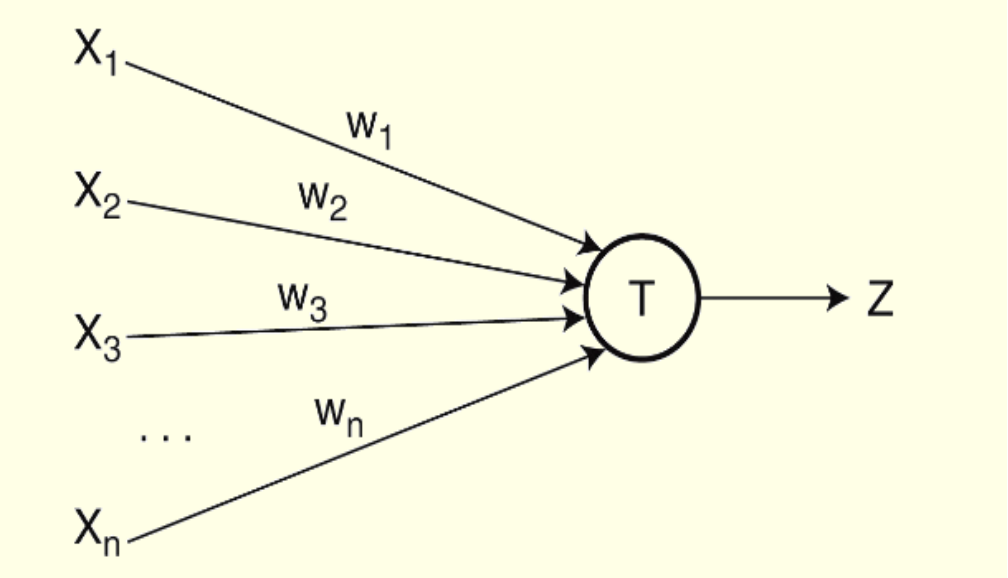

Figure 9.13 A Perceptron

- Perceptrons are trainable because the threshold and input weights are modifiable.

- In this example, the output Z is true (1) if the net input, $w_1x_1 + w_2x_2 + \ldots + w_nx_n$is greater than the threshold T.

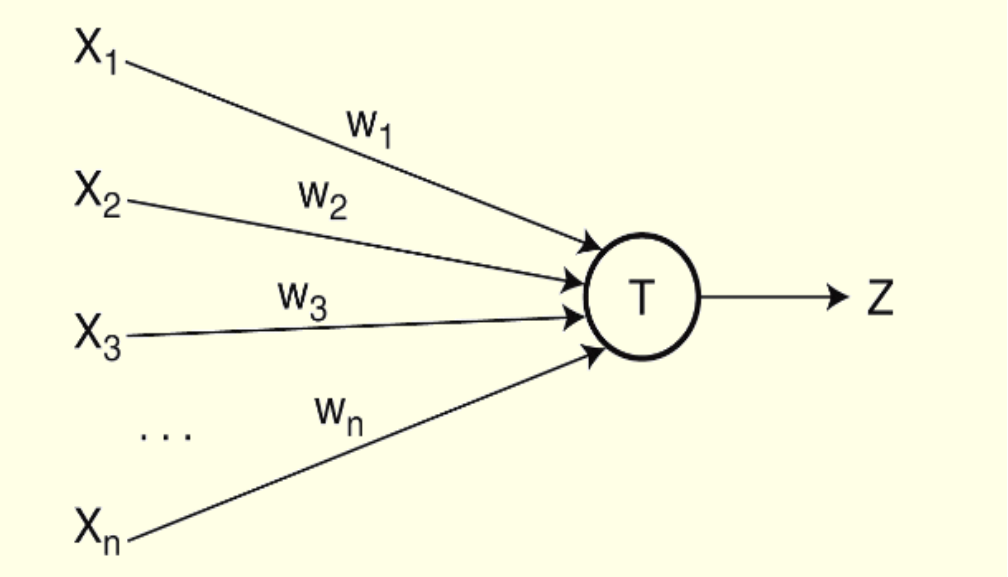

Figure 9.13.1 Training a Perceptron

- Perceptrons are trained by use of supervised or unsupervised learning.
- Supervised learning assumes prior knowledge of correct results which are fed to the neural net during the training phase. If the output is incorrect, the network modifies the input weights to produce correct results.
- Unsupervised learning does not provide correct results during training. The network adapts solely in response to inputs, learning to recognize patterns and structure in the input sets.
- The biggest problem with neural nets is that when they consist of more than 10 or 20 neurons, it is impossible to understand how the net is arriving at its results. They can derive meaning from data that are too complex to be analyzed by people.
    - The U.S. military once used a neural net to try to locate camouflaged tanks in a series of photographs. It turned out that the nets were basing their decisions on the cloud cover instead of the presence or absence of the tanks.
- Despite early setbacks, neural nets are gaining credibility in sales forecasting, data validation, and facial recognition.

9.5.3 Systolic Arrays

- Where neural nets are a model of biological neurons, systolic array computers are a model of how blood flows through a biological heart.
- Systolic arrays, a variation of SIMD computers, have simple processors that process data by circulating it through vector pipelines.

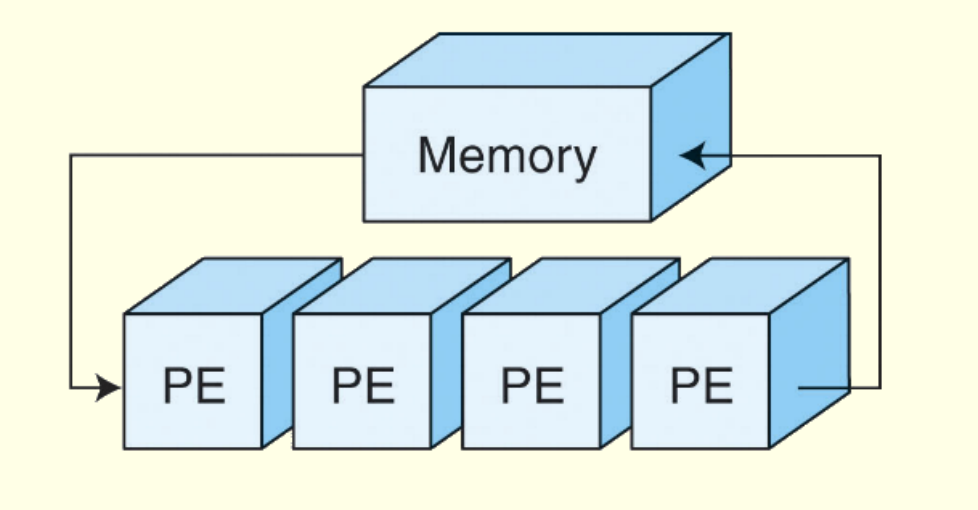

Figure 9.14 b) A Systolic Array Processor

- Systolic arrays can sustain great throughout because they employ a high degree of parallelism.
- Connections are short, and the design is simple and scalable. They are robust, efficient, and cheap to produce. They are, however, highly specialized and limited as to the types of problems they can solve.
- They are useful for solving repetitive problems that lend themselves to parallel solutions using a large number of simple processing elements.
    - Examples include sorting, image processing, and Fourier transformations.
- Example – evaluate a polynomial using Horner's rule
- $y = a_0 + a_1x + a_2x^2 + \cdots + a_kx^k$
- $y = ((((a_kx + a_{k-1}) \times x + a_{k-2}) \times x + a_{k-3}) \times x \cdots a_1) \times x + a_0$

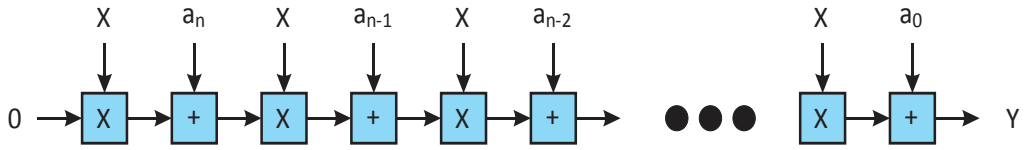

Figure 9.15 Using a Systolic Array to Evaluate a Polynomial