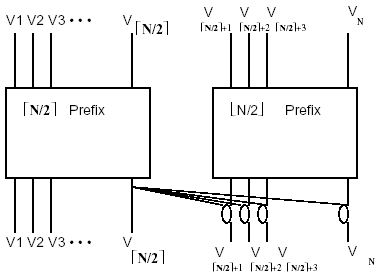
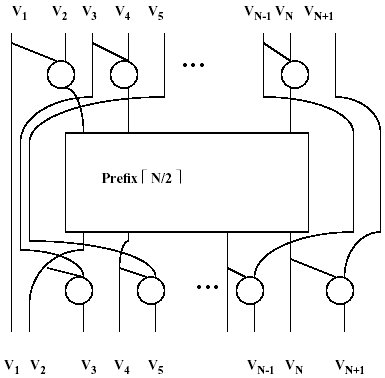
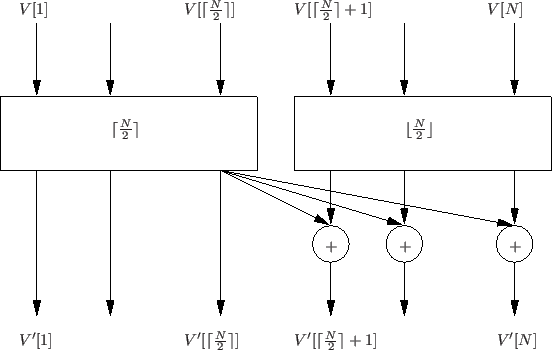
**Team Assignment 1**

1. There are two advantage features that can be used in multiple specialized arithmetic units in a single CPU. One feature is lookahead which is prefetching instructions into an instruction buffer. Score boarding, which is the other feature, determines which instructions can be issued concurrently without a conflict. In score boarding, there are three common order conflicts can be resolved as follow: same D unit or result register conflict, incomplete instruction conflict and incomplete operation conflict.

2. Since a simply sated problem may have several different algorithm, we can use the divide and conquer technique which is a general technique for constructing non-trivial parallel algorithms. It splits a problem into 2 smaller problems so that make the large problem to smaller ones. Then continue splitting recursively until problems are small enough to solve easily and quickly. There are several divide and conquer parallel prefix algorithms: upper/lower, odd/even and Ladner and Fischer's parallel prefix algorithm.

The upper/lower algorithm uses a divide and conquer approach to recursively divide the problem into smaller sub-problems. The results of the sub-problems are then combined to give the final output. The upper/lower algorithm divides the lower half of the elements from the upper half of the elements. The prefix sum is independently computed on the lower and upper halves by recursively applying the divide and conquer. The output from the highest element of the lower half is added to each output of the upper half. The size is N/2, the depth is .

The odd/even parallel prefix algorithm also uses a divide and conquer approach. The algorithm divides *N* inputs into groups whose indices are odd and even, respectively. The recursion continues to halve the number of inputs until reaching a base case. Recursive application of odd/even construction continues until a prefix of 2 inuts is reached. The size is 2N- -2 , and depth is 2 -2.

Ladner and Fischer's parallel prefix algorithm is a more sophisticated parallel prefix algorithm and uses both the upper/lower and even/odd constructions. The odd/even construction is used to construct (N) that produced from (N/2). The upper/lower construction is used to construct (N) that produced from (N/2) and (N/2). The depth of P0(N) = and the size of (N) ≤ 4N.   
If *N= 2k*, then *S0(N) = 4N - F(2 + k) - 2F(3 + k) + 1* and *S1(N) = 3N – F(1 + k) – 2F(2 + k)*. This algorithm allows tradeoffs between depth and size.

To illustrate the size and depth differences of upper/lower and odd/even algorithm, we need to introduce the parameter of sequential algorithm. For upper/lower algorithm, the size is N/2, the depth is . For odd/even algorithm, the size is 2N- -2, and depth is 2 -2. As to sequential algorithm, its size and depth are both N-1. After comparison, we conclude that sequential algorithm has smallest size and biggest depth. If there are unlimited processors, we can choose upper/lower algorithm gain minimum depth. We can also choose odd/even algorithm to keep small size and shallow depth if the processor are limited. Odd/even is about twice as deep as upper/lower.

3. In evaluation of Arithmetic Expressions, for any number P of processors, the time to evaluate an expression in N atoms satisfies: ≥ ⌈. What’s more, to reduce the height of an expression tree, we can use associative, commutative, and distributive laws. Those laws is powerful to change transfer expressions into more parallel forms.

4. The goal of parallel processing is to enhance performance which is related to speedup and efficiency of algorithms. The performance will be affected by the algorithms, programming languages and architectures. We could characterize the parallel performance by Amdahl’s Law. The speedup with *P* processors is *Sp = T1 / Tp*, and efficiency is *Ep = Sp / P*. Where *Tp* is the time to perform a computation with *P* processors. In Amdahl’s Law, we know the total time with *P* processors is *T(P) = S + Q/P*, where *S* is the time doing the sequential part of the work, *Q* is the time to do the parallel part of the work sequentially. The fraction is *f = S / (S + Q).* So when running in one processor, *T1 = T1(S) + T1(Q),* that is *T1 = f T1+(1-f)T1*. When running in *P* processors, *Tp = f T1 + (1 - f)T1 / P*. So the speedup is *S(P) = T1 / Tp = 1 / (f + (1 - f) / P)*, efficiency is *E(P) = Sp / P = 1 / (1 + f(P - 1))*. For example, if we have 50% efficiency,   
that is *1 / (1 + f(P - 1)) = 1 / 2*, then *f = 1 / (P-1).*

**Team Assignment 2-1**

**Report on “Design Of A Optimized Parallel Array Multiplier  
Using Parallel Prefix Adder [1]”**

In DSP processors, multiplication is the basic block, and for years the computational complexities have gradually increased. This means it requires a parallel array multiplier which can achieve high execution speed or meet the performance demands. In this paper, they introduced a new design of Braun Multiplier and it used a very fast parallel prefix adder.

Multiplication is an important for the scientific computations and we can get it by a repeated addition n of n bits. Parallel multipliers has a three important criteria that is the chip area, speed of computation and power dissipation. Nowadays, high speed parallel multipliers are used in reduced instruction set computers, digital signal processing and graphic accelerator. Parallel prefix adder is the most flexible and widely used for binary addition and best suited for VLSI implementation. In this paper, they are proposed a new design of Braun Multiplier with very fast parallel prefix adder (Brent Kung adder).

The Brent Kung Adder is divided three separate stages, generate/propagate generation, the dot (.) operation, and sum generation. In the Brent Kung approach, they designed the computation graph for are optimization. In this paper, they experimented their new Braun Multiplier which is designed for two inputs ‘A’ and ‘B’ each of width 4 bits. With this results, they showed that the proposed new design of Braun Multiplier using Brent Kung adder would provide the lesser area in effectively.

**Reference**

[1] K. KalaiKaviya, D.P. Balasubramanian, and S. Tamilselvan, “Design Of A Optimized parallel Array Multiplier Using Parallel Prefix Adder”, *In Proc. I.J. Engineering and Manufacturing*, 2013, 2, 40-50

**Team Assignment 2-2**

**Report on “Constructing Zero-deficiency Parallel Prefix Adder of   
Minimum Depth [1]”**

In general, parallel prefix adder is a technique for speeding up binary addition. This paper proposed a new architecture of zero-deficiency prefix adder which has minimal depth. And then they designed a 64-bit prefix adder Z64 and compare it against several classical prefix adders of the same bit width in terms of area and delay using logical effort method.

In [electronics](https://en.wikipedia.org/wiki/Electronics), an adder is a [digital circuit](https://en.wikipedia.org/wiki/Digital_circuit) that performs [addition](https://en.wikipedia.org/wiki/Addition) of numbers. The most fundamental module in computer arithmetic design is binary adder. For instance, carry-skip adder, the carry-select adder, and the carry-lookahead adder are all classic efficient adders. These adders represent unique area-time tradeoffs in the design space. As to parallel prefix adder, it represents a class of general adder structure that exhibits flexible area-time tradeoffs for adder design. The computation nodes are arranged in levels that represent the signal timing in the unit delay timing model of parallel prefix adders.

Snir’s zero-deficiency theorem: + ≥ 2n - 2, and are the size and the depth of an n-bit prefix adder C(n) respectively. When + = 2n – 2, it can be defined as def(C(n)) = + − 2n + 2 = 0 which we call zero-deficiency.

This paper presented a zero-deficiency prefix adder which is a new architecture and has minimal depth for the given depth. And we can show the extreme of linear depth-size tradeoff of prefix adder with the *Z(d)* adder. And also Z64 is fast because it achieves a reasonable tradeoff between logic depth, fan-out and lateral wire connection. It also has the smallest area because of zero-deficiency. And this paper remains the future work for analyzing the power efficiency of *Z(d)* adder.

**Reference**:

[1] Haikun Zhu, Chung-Kuan Cheng, Ronald Graham, “Constructing Zero-deficiency Parallel Prefix Adder of Minimum Depth”, *In Proc. Special issue: Design Automation Conference, 2005.* Pages 883 - 888