Chapter-8
Parallel patterns: prefix sum
An introduction to work efficiency in parallel algorithms

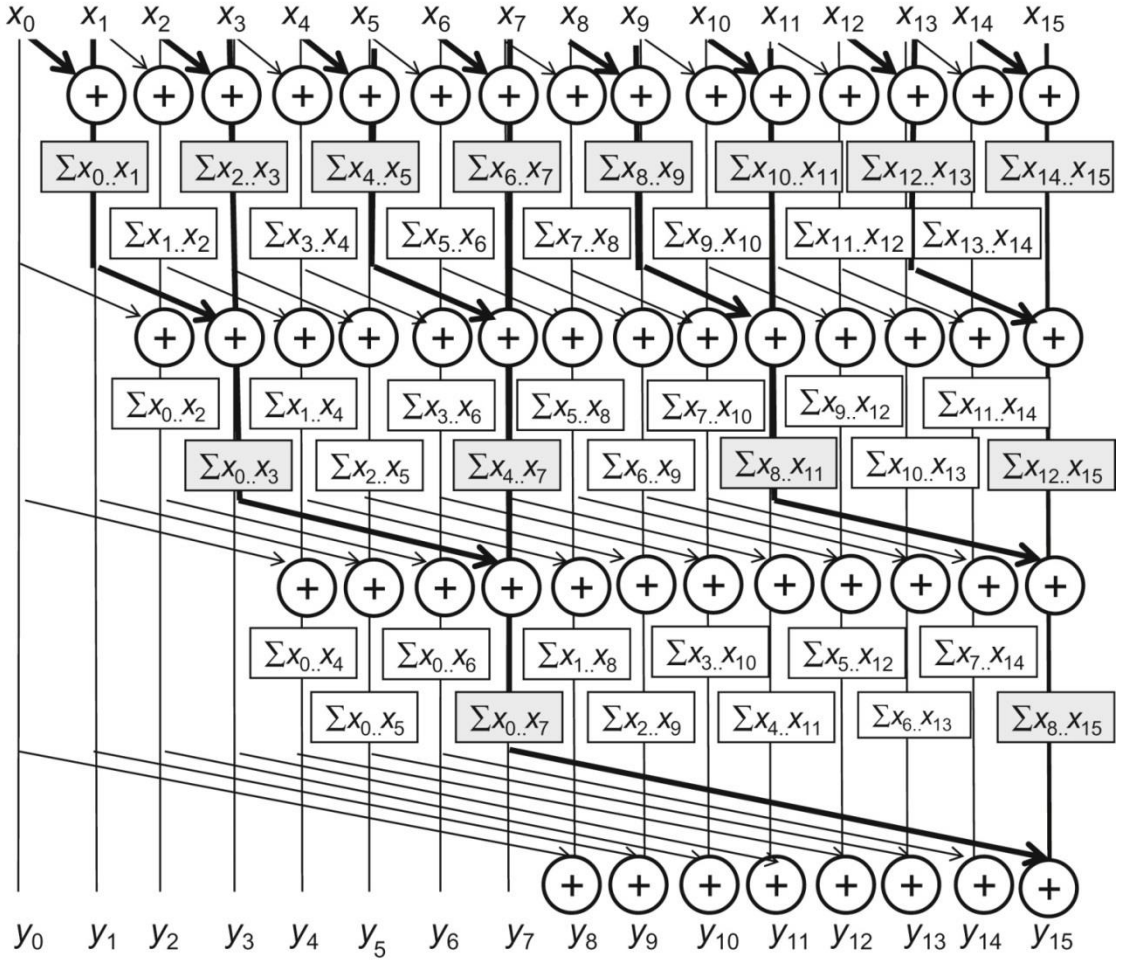

FIGURE 8.1: A parallel inclusive scan algorithm based on Kogge–Stone adder design.

```
__global__ void Kogge-Stone_scan_kernel(float *X, float *Y,
  int InputSize) {

  __shared__ float XY[SECTION_SIZE];

  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < InputSize) {
    XY[threadIdx.x] = X[i];
  }

  // the code below performs iterative scan on XY
  for (unsigned int stride = 1; stride < blockDim.x; stride *= 2) {
    __syncthreads();
    if (threadIdx.x >= stride)XY[threadIdx.x] += XY[threadIdx.x-stride];
  }

  Y[i] = XY[threadIdx.x];

}
```

FIGURE 8.2: A Kogge–Stone kernel for inclusive scan.

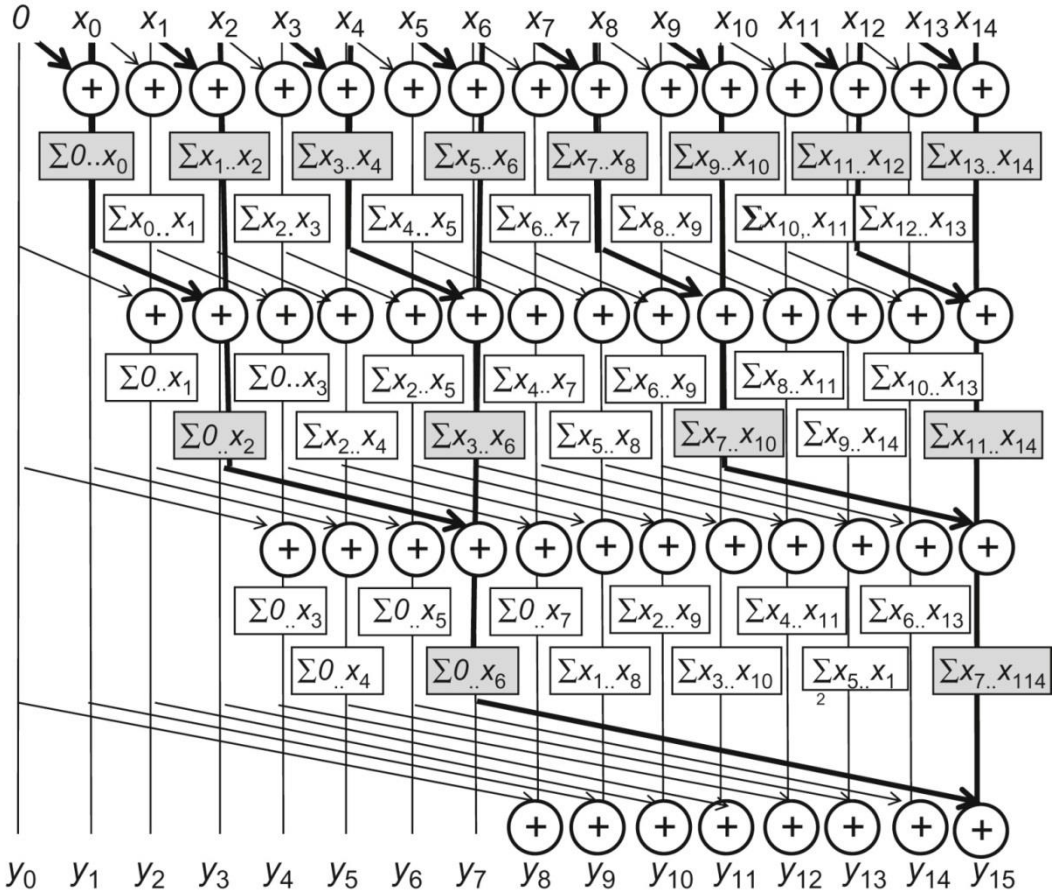

FIGURE 8.3: A parallel exclusive scan algorithm based on Kogge–Stone adder design.

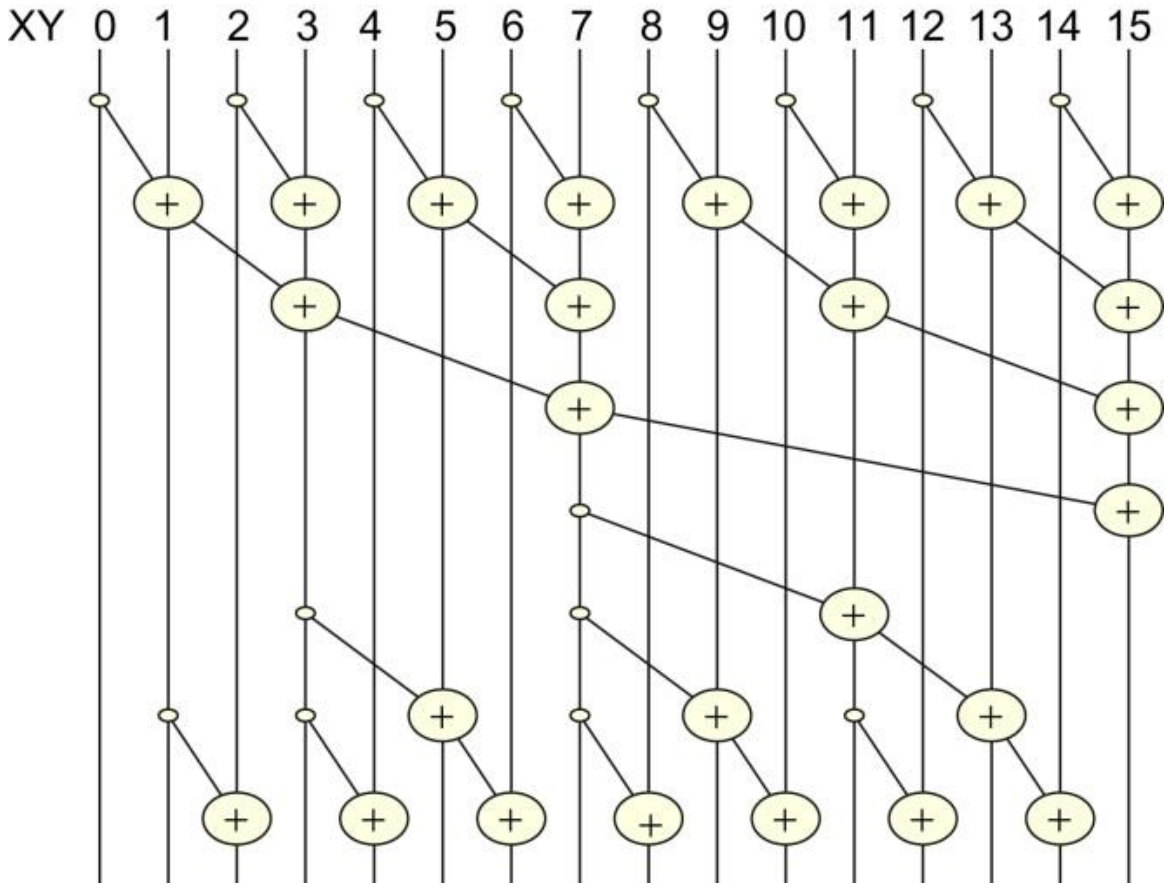

FIGURE 8.4: A parallel inclusive scan algorithm based on the Brent–Kung adder design.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_0	$x_0.x_1$	x_2	$x_0.x_3$	x_4	$x_4.x_5$	x_6	$x_0.x_7$	x_8	$x_8..x_9$	x_{10}	$x_8.x_{11}$	x_{12}	$x_{12}..x_{13}$	x_{14}	$x_0.x_{15}$
											$x_0..x_{11}$				
					$x_0..x_5$				$x_0..x_9$				$x_0..x_{13}$		

FIGURE 8.5: Partial sums available in each XY element after the reduction tree phase.

```
__global__ void Brent_Kung_scan_kernel(float *X, float *Y,
  int InputSize) {

  __shared__ float XY[SECTION_SIZE];
  int i = 2*blockIdx.x*blockDim.x + threadIdx.x;
  if (i < InputSize) XY[threadIdx.x] = X[i];
  if (i+blockDim.x < InputSize) XY[threadIdx.x+blockDim.x] = X[i+blockDim.x];

  for (unsigned int stride = 1; stride <= blockDim.x; stride *= 2) {
    __syncthreads();
    int index = (threadIdx.x+1) * 2* stride -1;
    if (index < SECTION_SIZE) {
      XY[index] += XY[index - stride];
    }
  }

  for (int stride = SECTION_SIZE/4; stride > 0; stride /= 2) {
    __syncthreads();
    int index = (threadIdx.x+1)*stride*2 - 1;
    if(index + stride < SECTION_SIZE) {
      XY[index + stride] += XY[index];
    }
  }

  __syncthreads();
  if (i < InputSize) Y[i] = XY[threadIdx.x];
  if (i+blockDim.x < InputSize) Y[i+blockDim.x] = XY[threadIdx.x+blockDim.x];
}
```

FIGURE 8.6: A Brent–Kung kernel for inclusive scan.

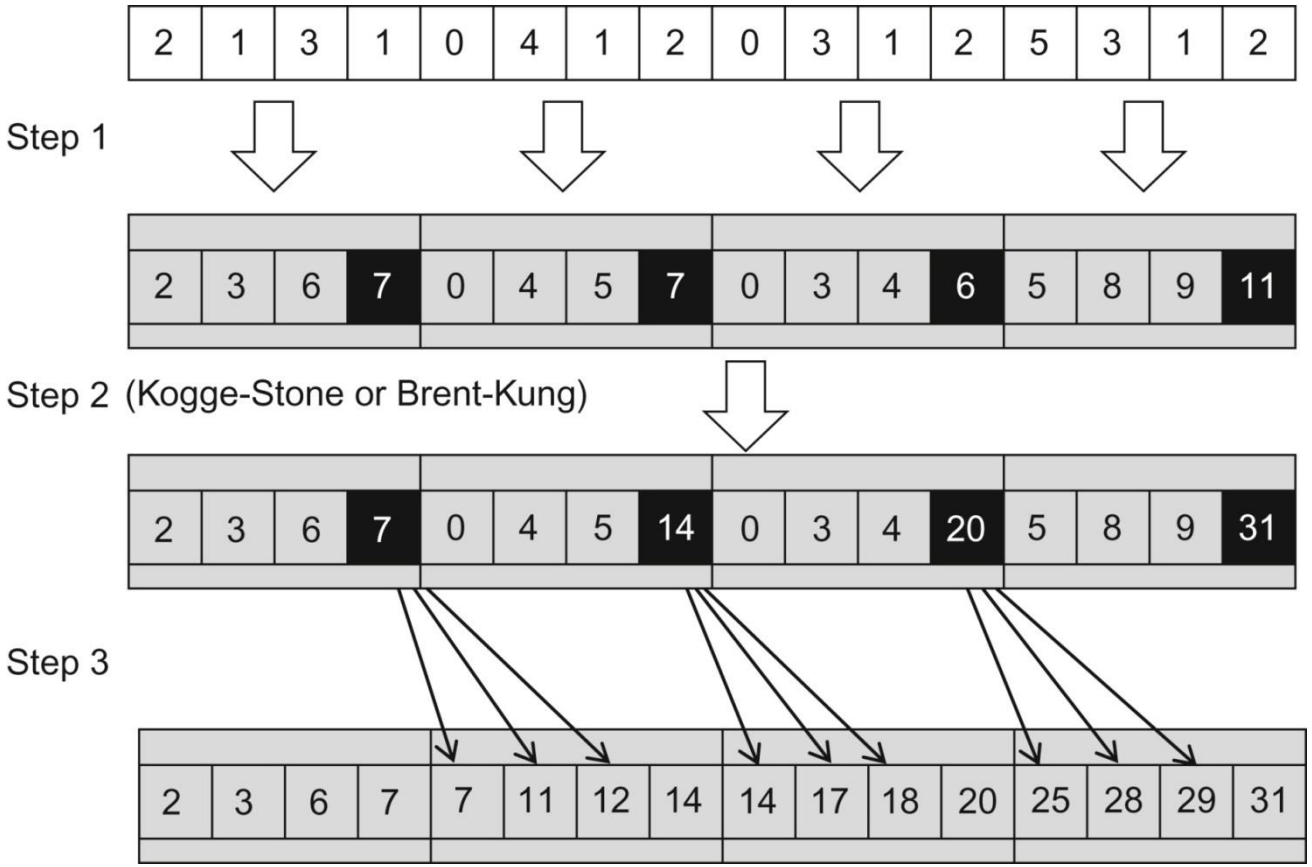

FIGURE 8.7: Three-phase parallel scan for higher work efficiency and speed.

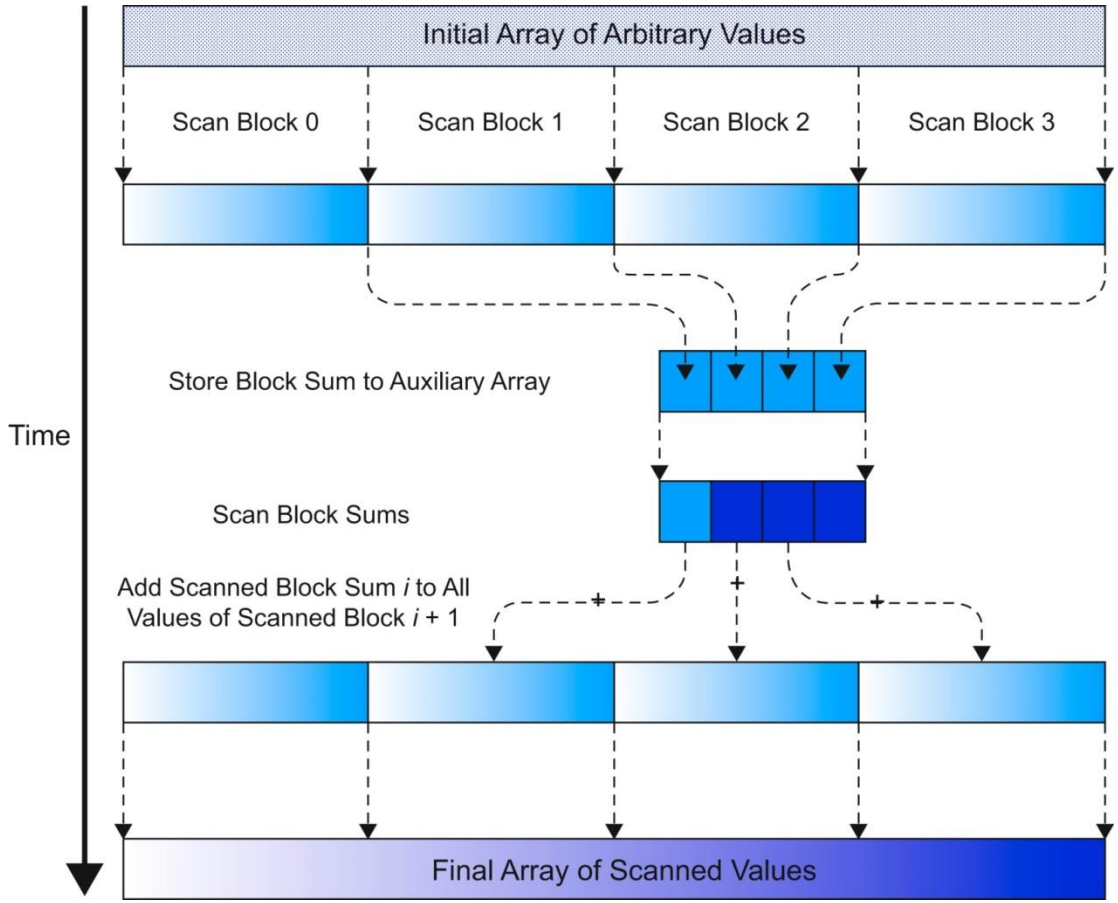

FIGURE 8.8: A hierarchical scan for arbitrary length inputs.

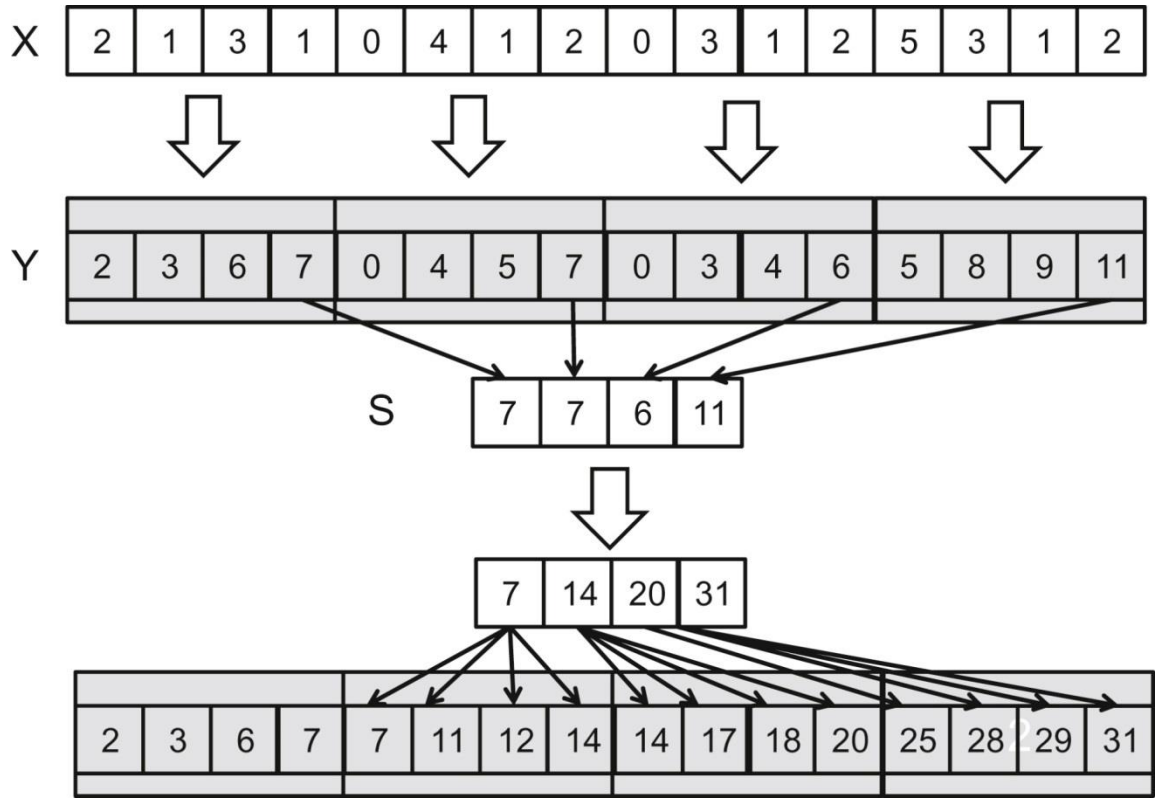

FIGURE 8.9: An example of hierarchical scan.