Lin/Snyder, *Principles of Parallel Programming*, Figure 5.7 Corrected and a compatible Figure 5.8 [Contributed by Edin Hodzic.]

```
int   operandArray[N];  // input
int   Result[N];        // output

tally nodeval'[P];      // tree synchronization and communication
tally ltally[P];        // left tally

forall ( index in ( 0 .. P-1 ) )
{
    int size           = mySize( operandArray[], 0 );
    int myData[size]   = localize( operandArray[] );
    int myResult[size] = localize( Result[] );

    tally myTally;
    tally ptally;

    // compute the local tally into myTally
    myTally = init( );
    for ( i = 0; i < size; i++ )
    {
        myTally = accum( myTally,
                         myData[i],
                         localToGlobal( myData, i, 0 ) );
    }
    nodeval'[index] = myTally;

    // propagate partial tallys up the tree all the way to the root;
    // remember the left tally
    int stride = 1;
    while ( stride < P )
    {
       if ( index % ( 2 * stride ) == 0 )
       {
          ltally[index+stride] = nodeval'[index];
          nodeval'[index] = combine( ltally[index+stride],
                                     nodeval'[index+stride] );
          stride = 2 * stride;
       }
       else
       {
          break;
       }
    }

    // root's parent value is the init tally
    if ( index == 0 )
    {
       dummy = nodeval'[0];  // just empty the FE variable
       nodeval'[0] = init( );
    }

    // propagete parent value through the tree to the leaf nodes
    stride = P/2;
    while ( stride >= 1 )
    {
       if ( index % ( stride * 2 ) == 0 )
       {
           ptally = nodeval'[index];
           nodeval'[index] = ptally; // left child's parent value
           nodeval'[index+stride] =  // right child's parent value
```

```
            combine( ptally, ltally[index+stride] );
        }
        stride = stride / 2;
    }

    // leaf node, generate result for all local elements
    ptally = nodeval'[index];  // get the parent value
    for ( i = 0; i < size; i++ )
    {
        ptally = accum( ptally, myData[i], localToGlobal( myData, i, 0 ) );
        myResult[i] = scanGen( ptally,
                               myData[i],
                               localToGlobal( myData, i, 0 ) );
    }
}
```

Compatible Figure 5.8 for sum scan.

```
//
// Using generalized scan to solve the prefix sum.
//

type int tally;

tally init( )
{
    return 0;
}

tally accum( tally t, int elem, int i )
{
    return t + elem;
}

tally combine( tally left, tally right )
{
    return left + right;
}

int scanGen( tally t, int elem, int i )
{
    return t;
}
```