```
__global__
void plus_reduce(int *input, unsigned int N, int *total)
{
    unsigned int tid = threadIdx.x;
    unsigned int i   = blockIdx.x*blockDim.x + threadIdx.x;

    // Each block loads its elements into shared memory, padding
    // with 0 if N is not a multiple of blocksize
    __shared__ int x[blocksize];
    x[tid] = (i<N) ? input[i] : 0;
    __syncthreads();

    // Every thread now holds 1 input value in x[]
    //
    // Build summation tree over elements.
    for(int s=blockDim.x/2; s>0; s=s/2)
    {
        if(tid < s)  x[tid] += x[tid + s];
        __syncthreads();
    }

    // Thread 0 now holds the sum of all input values
    // to this block. Have it add that sum to the running total
    if( tid == 0 )  atomicAdd(total, x[tid]);
}
```

FIGURE C.8.9 CUDA implementation of plus-reduction.