Original CUDA Program

```
#include <cuda.h>
#define tid threadIdx.x
#define N 1024

__device__ inline void inlined(int *A, int offset) {
    int temp = A[tid + offset];
    A[tid] += temp;
}

__global__ void inline_test(int *A, int offset) {
    inlined(A, offset);
}

int main( ) {
    int *a;
    int *dev_a;
    int size = N*sizeof(int);
    cudaMalloc((void**)&dev_a, size);
    a = (int*)malloc(N*size);
    for (int i = 0; i < N; i++)
        a[i] = i;
    cudaMemcpy(dev_a,a,size, cudaMemcpyHostToDevice);

    inline_test<<<1,N>>>(dev_a, 2);

    cudaMemcpy(a,dev_a,size,cudaMemcpyDeviceToHost);
    free(a);
    cudaFree(dev_a);
    return 0;
}
```

Change the default kernel call

Modified CUDA Program

```
#include <cuda.h>
#define tid threadIdx.x
#define N 1024

__device__ inline void inlined(int *A, int offset) {
    int temp = A[tid + offset];
    A[tid] += temp;
}

__global__ void inline_test(int *A, int offset) {
    inlined(A, offset);
}

int main( ) {
    int *a;
    int *dev_a;
    int size = N*sizeof(int);
    cudaMalloc((void**)&dev_a, size);
    a = (int*)malloc(N*size);
    for (int i = 0; i < N; i++)
        a[i] = i;
    cudaMemcpy(dev_a,a,size, cudaMemcpyHostToDevice);

    ESBMC_verify_kernel_intt(inline_test, 1, N, dev_a, 2);

    cudaMemcpy(a,dev_a,size,cudaMemcpyDeviceToHost);
    free(a);
    cudaFree(dev_a);
    return 0;
}
```