

CUDA-based Program

```
#include <cuda.h>
#include <stdio.h>
#define N 2

__global__ void definitions(int* A) {
    atomicAdd(A,10);
}

int main (){
    int a = 5;
    int *dev_a;
    cudaMalloc((void**) &dev_a, sizeof(int));
    cudaMemcpy(dev_a, &a,sizeof(int),cudaMemcpyHostToDevice);
    ESBMC_verify_kernel(definitions,1,N,dev_a);
    cudaMemcpy(&a,dev_a,sizeof(int),cudaMemcpyDeviceToHost);
    assert(a==25);
    cudaFree(dev_a);
    return 0;
}
```

Change the
default kernel call

Correspondent
Operational Model

CUDA Operational Model

```
cudaError_t cudaMalloc(void ** devPtr, size_t size) {
    cudaError_t tmp;

    __ESBMC_assert(size > 0,
        "Size to be allocated must be greater than zero");

    *devPtr = malloc(size);
    if(*devPtr==NULL){
        tmp = CUDA_ERROR_OUT_OF_MEMORY; exit(1);
    } else {
        tmp = CUDA_SUCCESS;
    }

    __ESBMC_assert(tmp == CUDA_SUCCESS,
        "Memory was not allocated");

    lastError = tmp;
    return lastError;
}
```

Precondition

Simulate behaviour

Postcondition