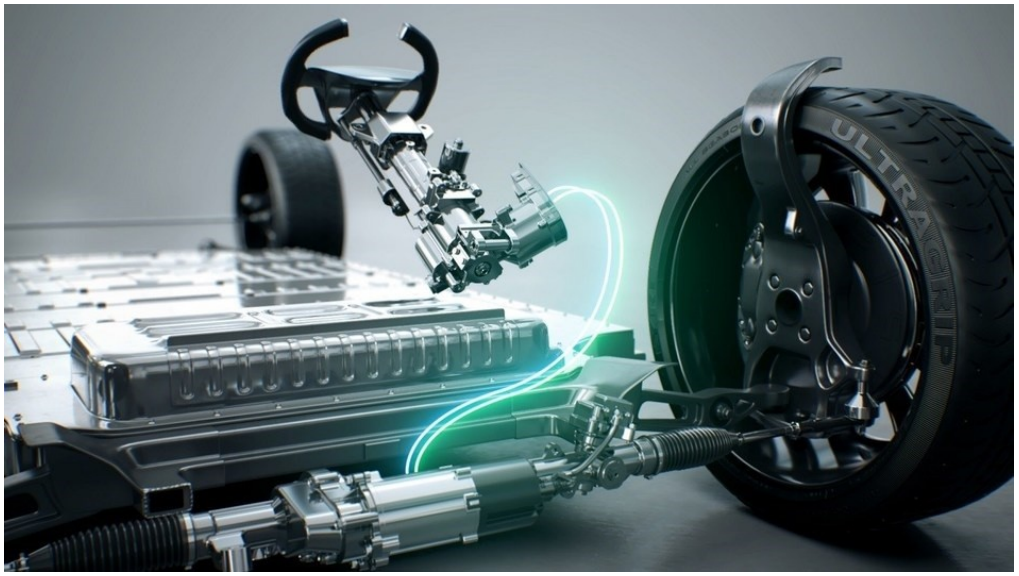
TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Betriebssysteme

Optimizing Software-based Soft-Error Detector Configurations

Robin Thunig, Michael Lenz, Peter Ulbrich, Horst Schirmeier

Motivation



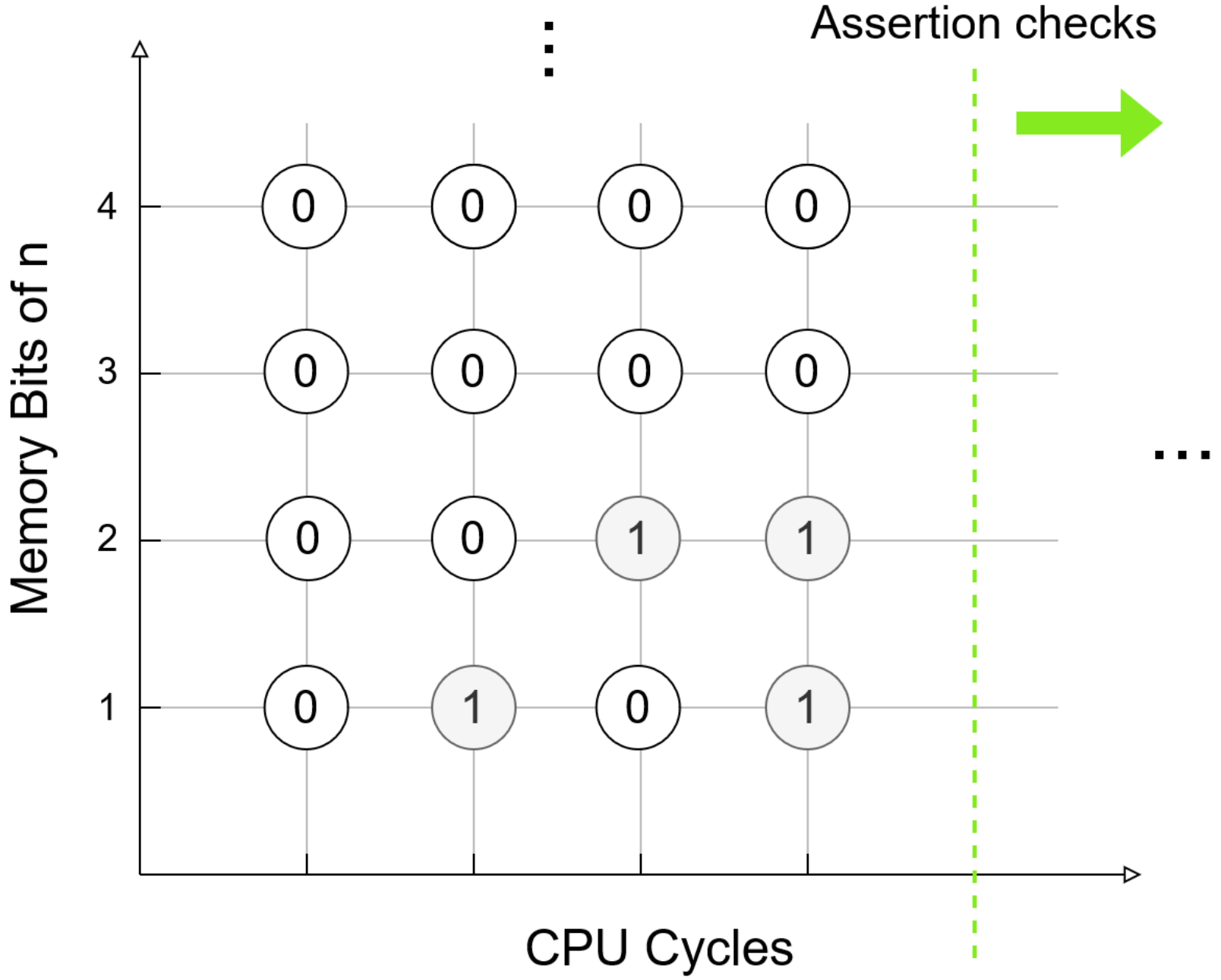
- **Hardware is effected by cosmic rays**
  - Isolation not always possible
  - Redundant Hardware is expensive

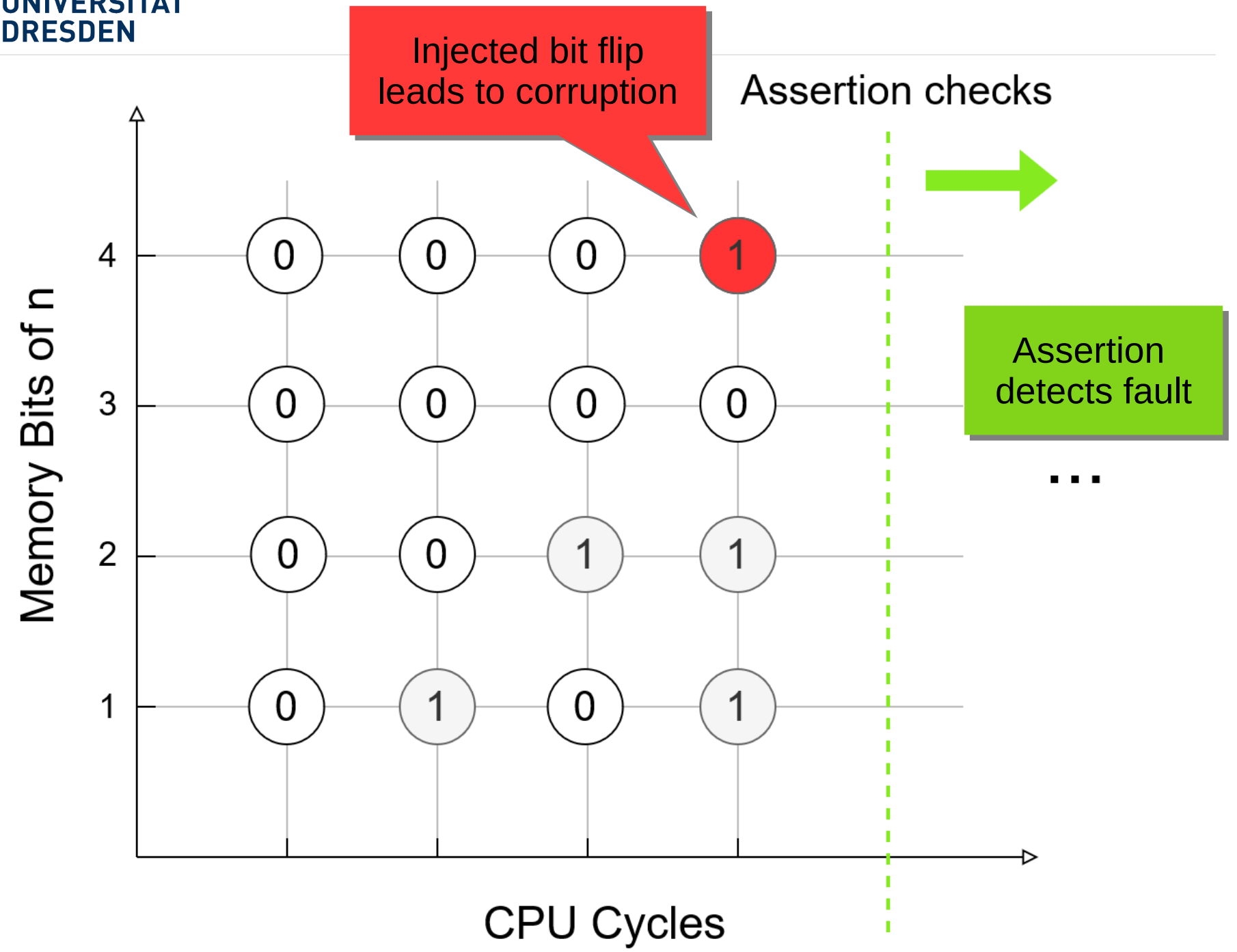
➔ **Software-based hardware fault tolerance**

Assertions for Fault Tolerance

- **Assertion triggers if defined fault state occurs**
  - Assertions (usually) deactivated in production
- **Improve fault tolerance with existing assertions**
- **High number of assertions in most operating systems**
  - e.g. tens of thousands for the Linux Kernel

```
static int dbFindLeaf(dmtree_t * tp, int l2nb, int *leafidx)
{
        int ti, n = 0, k, x = 0;

        if (l2nb > tp->dmt_stree[ROOT])
        return -ENOSPC;

        for (k = le32_to_cpu(tp->dmt_height), ti = 1;
        k > 0; k--, ti = ((ti + n) << 2) + 1) {

                for (x = ti, n = 0; n < 4; n++) {
                        if (l2nb <= tp->dmt_stree[x + n])
                        break;
                }

                assert(n < 4);
        }

        return (0);
}
```


Assertion checks
Memory Bits of n
4
3
2
1
0 0 0 0
0 0 0 0
0 0 1 1
0 1 0 1
...
CPU Cycles


Injected bit flip leads to corruption
Assertion checks
Assertion detects fault
...
Memory Bits of n
4
3
2
1
0 0 0 1
0 0 0 0
0 0 1 1
0 1 0 1
CPU Cycles

```
void quicksort(char data[], int begin, int end) {
    assert1(data);
    if (end > begin) {
        int pivot = begin; int l = begin + 1; int r = end;
        while(l < r) {
            if (data[l] <= data[pivot]) { l += 1; }
            else if (data[r] > data[pivot]) { r -= 1; }
            else {
                swap(data+l, data+r);
                assert2(data[l] <= data[pivot] && data[pivot] <= data[r]);
            }
            assert3(l <= r);
        } l -= 1;
        assert4(data[l] <= data[pivot]);
        swap(data+pivot, data+l);
        quicksort(data, begin, l); quicksort(data, r, end);
    }
}

void sort(char data[], input_data_length) {
    quicksort(input_data, 0, input_data_length - 1);
    assert5(is_sorted(input_data, input_data_length));
}
```

Are all assertions positive for fault tolerance?

assert1	assert2	assert3	assert4	assert5	SDC count
0	0	0	0	1	667 792
1	0	0	0	1	688 507
0	1	0	0	1	697 033
0	0	0	1	1	701 957
0	0	1	0	1	703 802
⋮					
1	1	1	1	1	755 252
⋮					
0	0	0	0	0	1 539 162

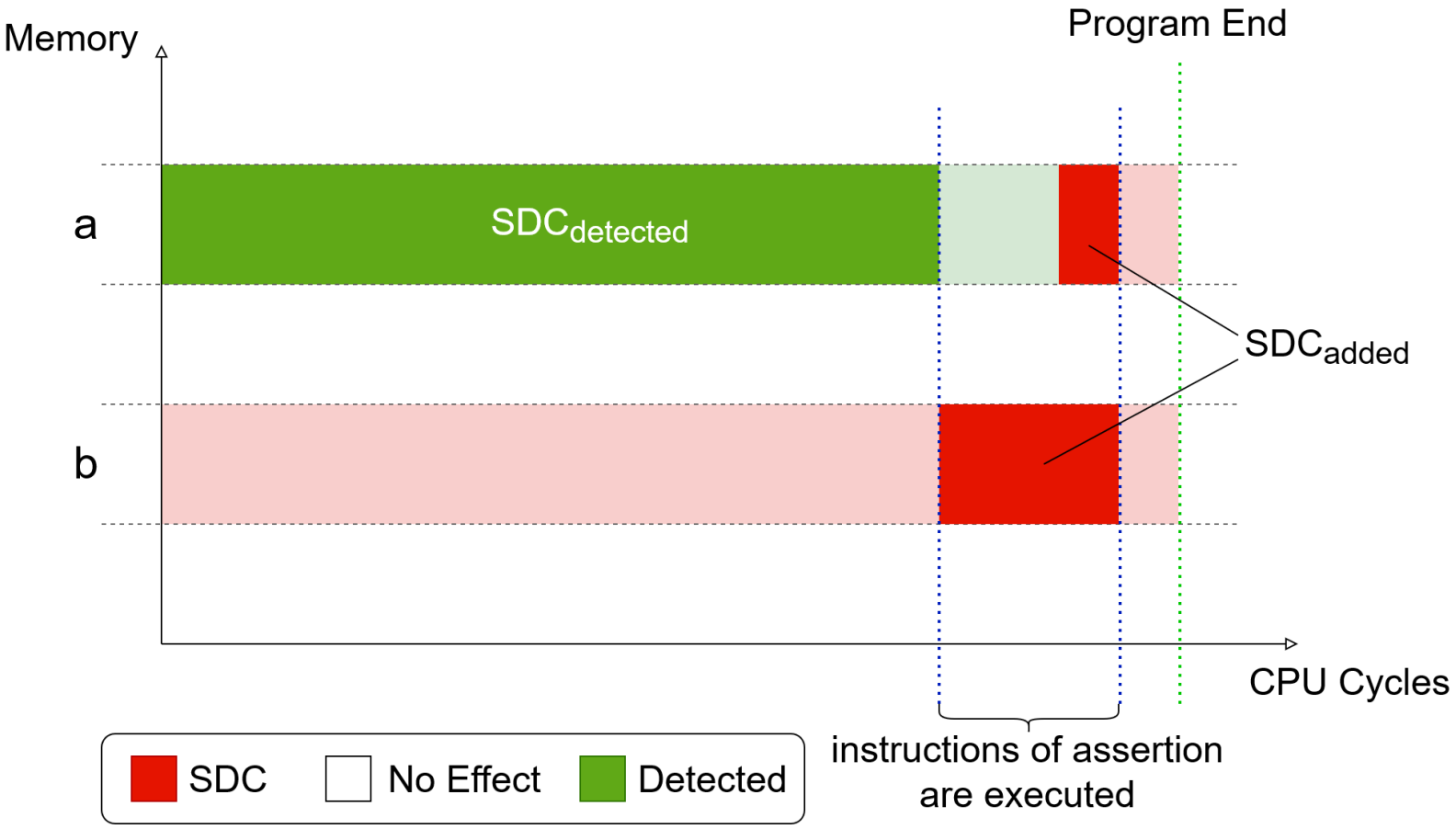
11%

56%

```
void quicksort(char data[], int begin, int end) {
    assert1(data);
    if (end > begin) {
        int pivot = begin; int l = begin + 1; int r = end;
        while(l < r) {
            if (data[l] <= data[pivot]) { l += 1; }
            else if (data[r] > data[pivot]) { r -= 1; }
            else {
                swap(data+l, data+r);
                assert2(data[l] <= data[pivot] && data[pivot] <= data[r]);
            }
            assert3(l <= r);
        } l -= 1;
        assert4(data[l] <= data[pivot]);
        swap(data+pivot, data+l);
        quicksort(data, begin, l); quicksort(data, r, end);
    }
}

void sort(char data[], input_data_length) {
    quicksort(input_data, 0, input_data_length - 1);
    assert5(is_sorted(input_data, input_data_length));
}
```

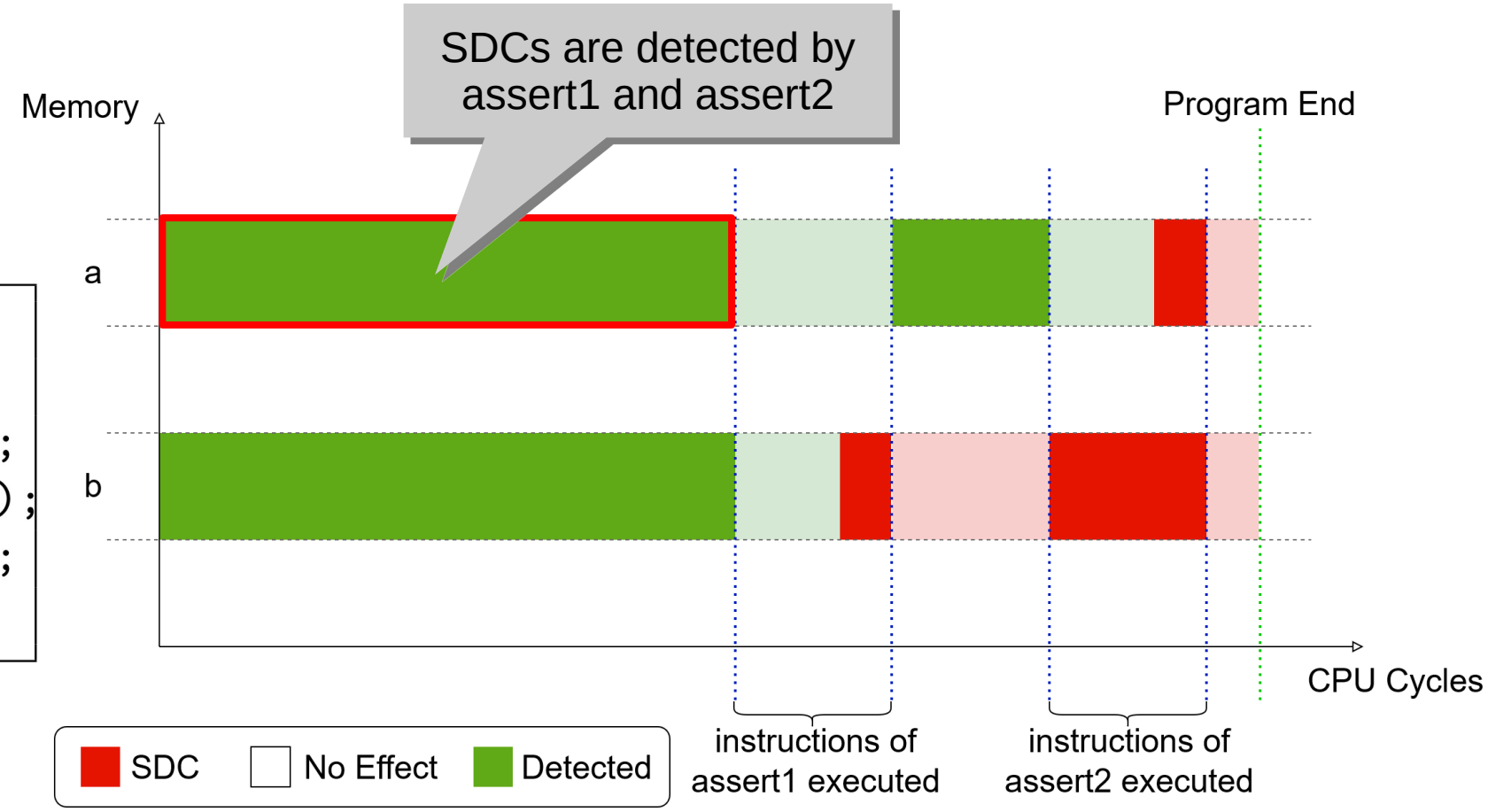
Why are certain assertions bad for fault tolerance?

```
bool a = true;
bool b = false;
someWorkload();
assert(a);
```



- **Runtime of assertion increases attack surface**
- **Total number of reduced SDCs is determined by $SDC_{detected} - SDC_{added}$**

```
bool a = true;
bool b = true;
someWorkload();
assert1(a && b);
someWorkload();
assert2(a);
```

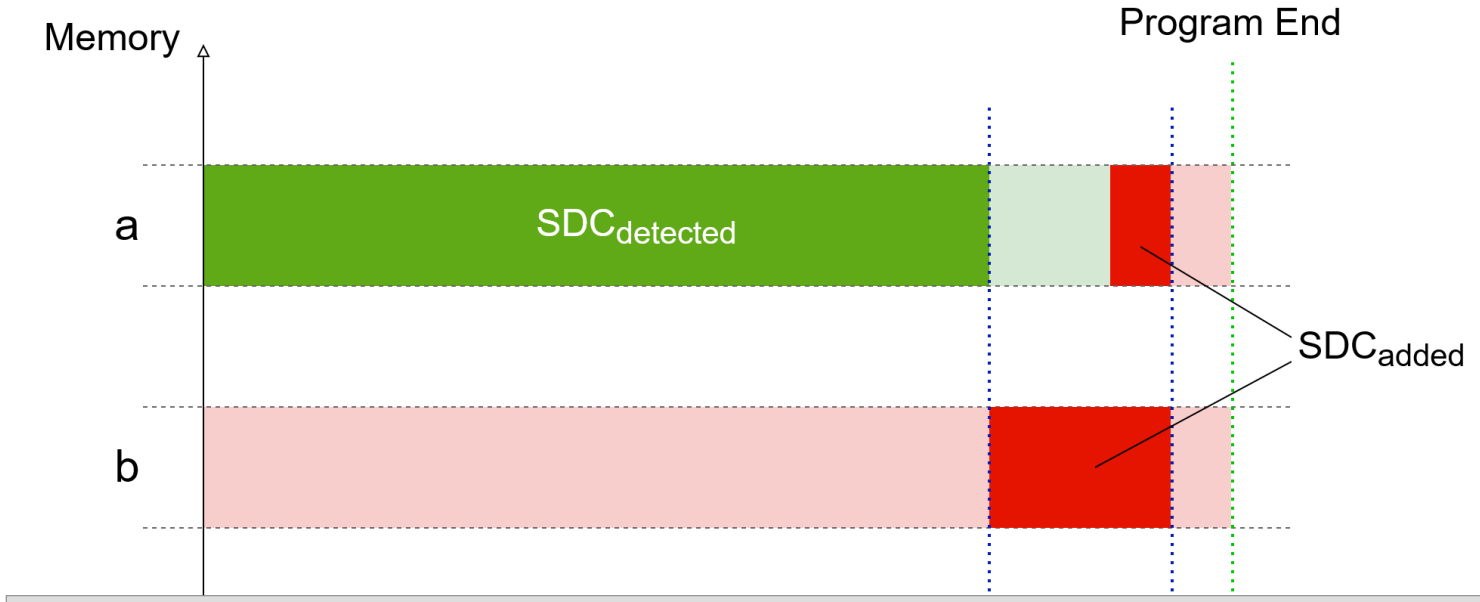


- **assert1 and assert2 detects partly the same SDCs**
- **Assertions are not independent from each other**

How to choose the optimal selection out of N assertions?

Finding an Optimal Configuration

- **Executing all 2^N configurations practically not possible**
- **Calculate SDCs of configuration from fault injection campaign with all assertions active**



$$SDC_{assertion\text{-}inactive} = SDC_{all\text{-}assertions\text{-}active} + SDC_{detected} - SDC_{added}$$

assert1	assert2	assert3	assert4	assert5	SDC real	SDC synthetic	error
0	0	0	0	1	667 792	671 174	0.50%
1	0	0	0	1	688 507	680 590	−1.16%
0	1	0	0	1	697 033	696 201	−0.12%
0	0	0	1	1	701 957	693 018	−1.29%
0	0	1	0	1	703 802	698 904	−0.70%
				⋮			
1	1	1	1	1	755 252	755 252	0.00%
				⋮			
0	0	0	0	0	1 539 162	1 569 759	1.95%

Finding an Optimal Configuration

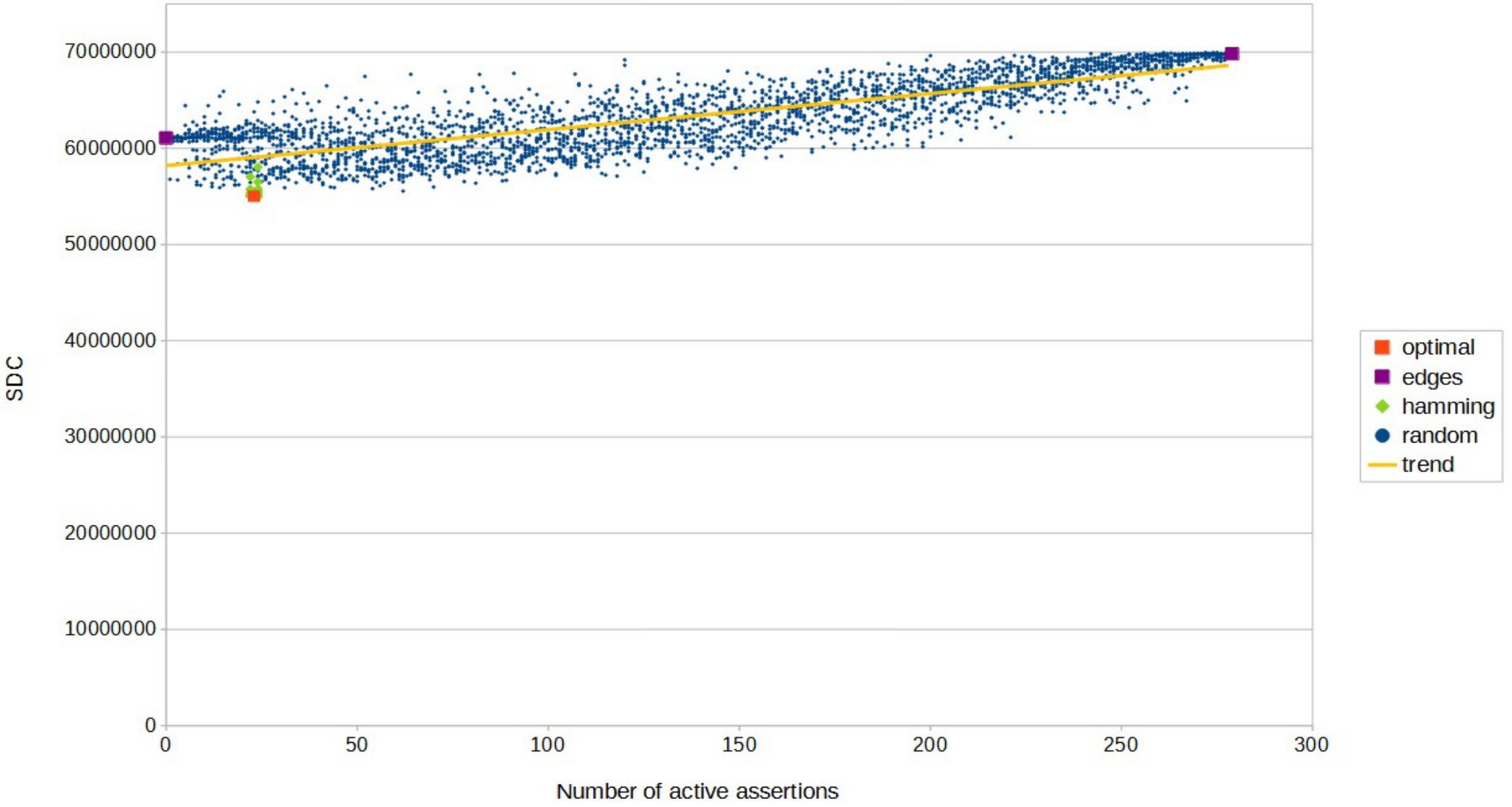
- **Calculating all 2^N configurations may be still not possible**

- **Integer Linear Programming could be the salvation**
  - Minimize cost function under linear constraints
  - Addition and subtraction of detected and added SDCs
  - Existing ILP solver like glpsol or gurobi can be used

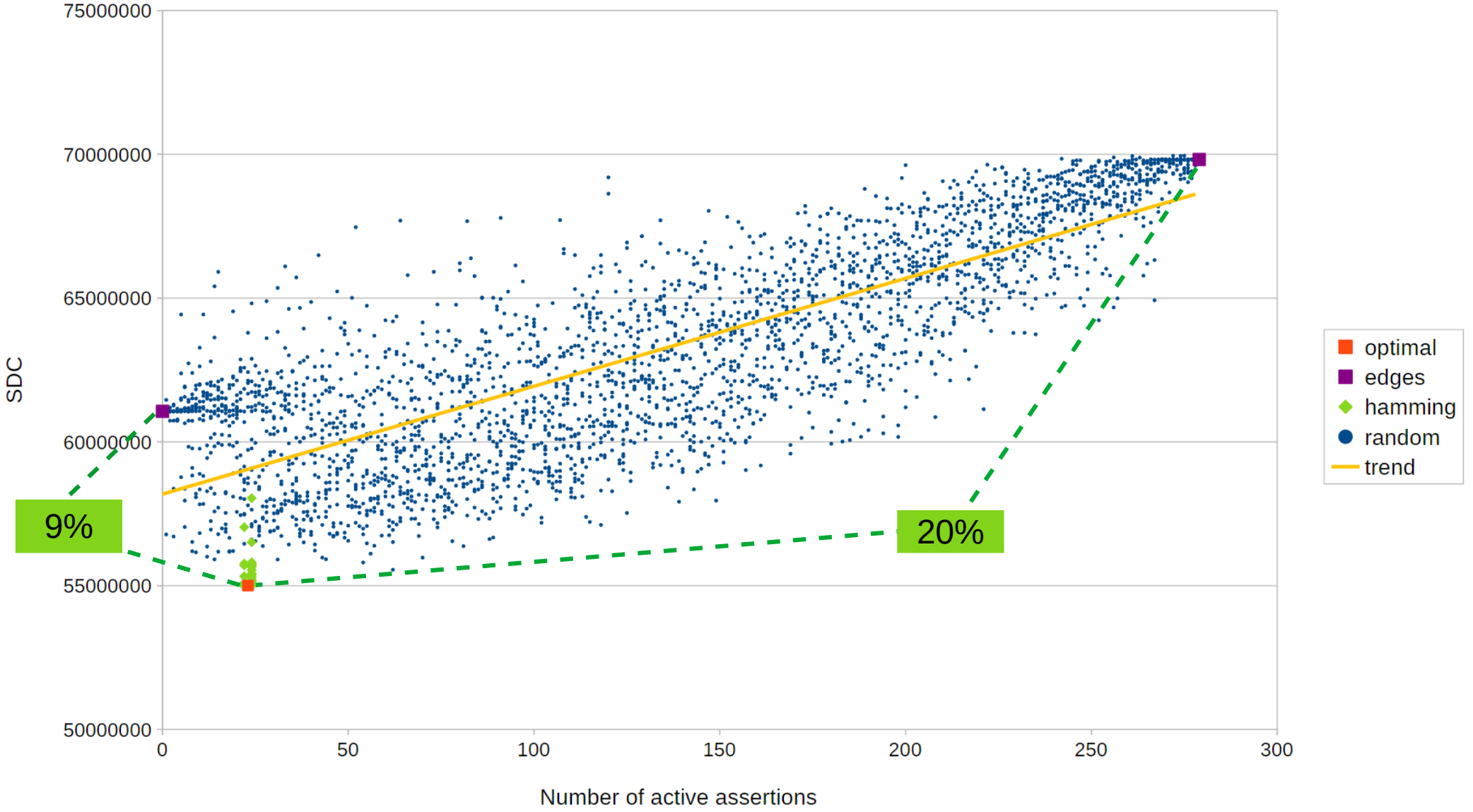
Evaluation

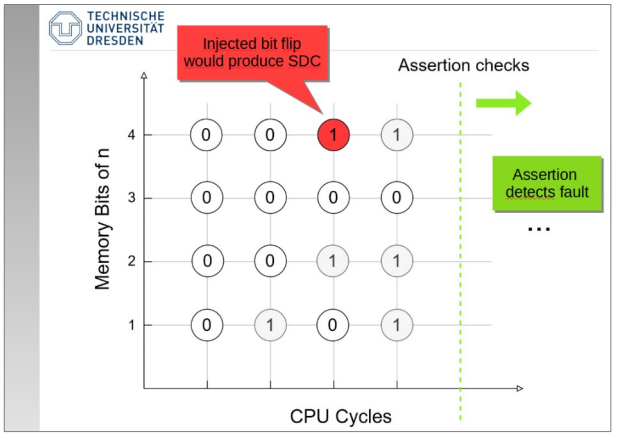
- **FreeRTOS**
  - 224 assertions in code
  - Demo with mutex and queue
  - 60 assertions executed
- **eCos**
  - 4489 assertions
  - Provided test suit
  - Custom test with various kernel functions
  - Up to 279 assertions executed
- **Focus on eCos due to significantly more assertions**

eCos custom test

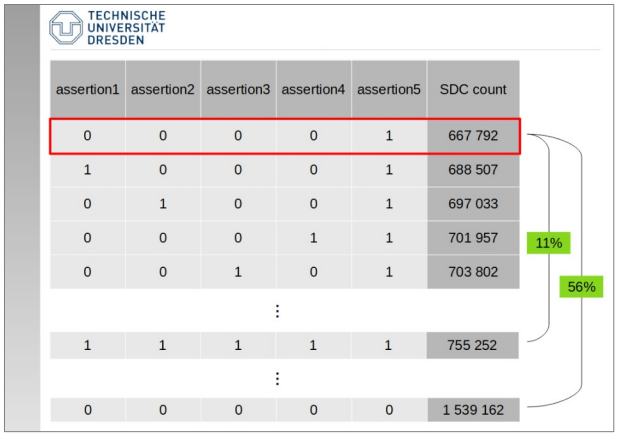


eCos custom test





Assertions can detect faults

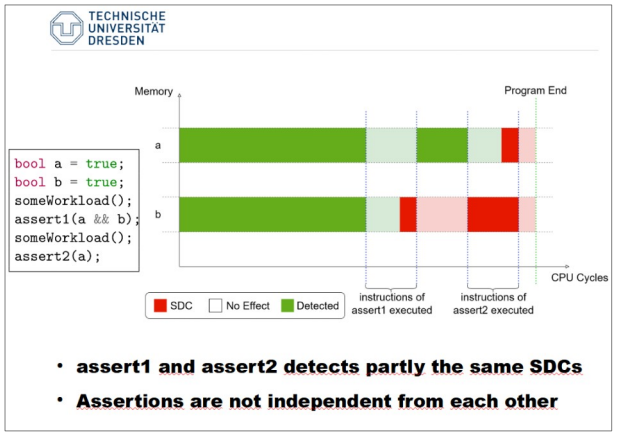


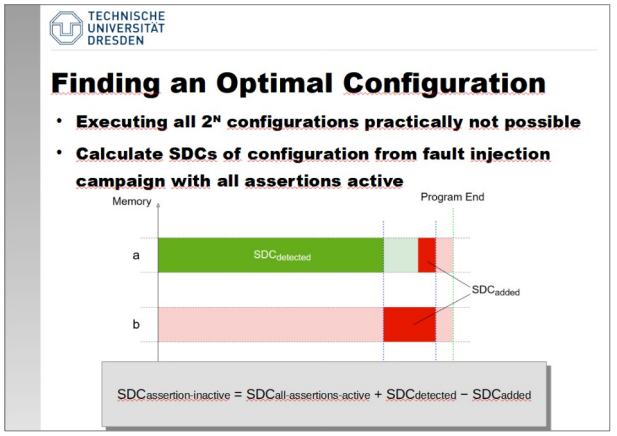
assertion1	assertion2	assertion3	assertion4	assertion5	SDC count
0	0	0	0	1	667 792
1	0	0	0	1	688 507
0	1	0	0	1	697 033
0	0	0	1	1	701 957
0	0	1	0	1	703 802
⋮					
1	1	1	1	1	755 252
⋮					
0	0	0	0	0	1 539 162

11%
56%

Only certain assertions good for fault tolerance

Key Takeaways





- Runtime of assertion increases attack surface
- Assertions are dependent on each other

- Configurations can be calculated

➔ ILP can be created