A linear approach to complex constraint structures

From 17 hours to 90 seconds…

Elihai Maicas

Intel

June 1st, 2016

SNUG Israel

Agenda

Background

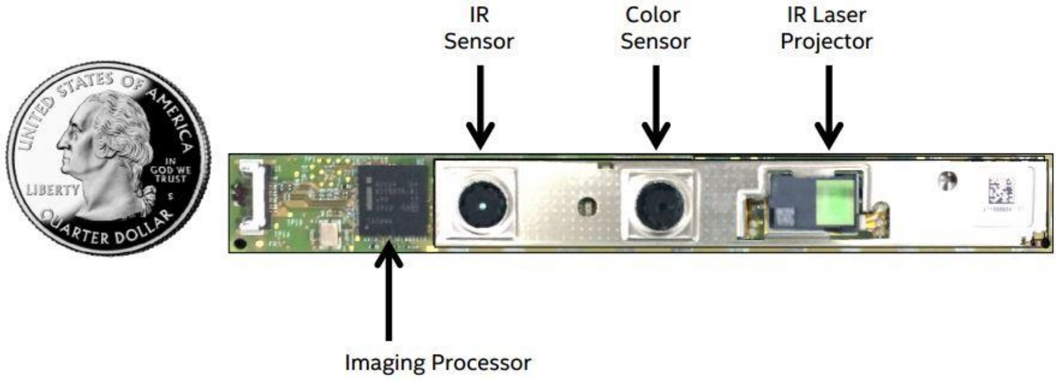
The problem

Offered solutions

Summary

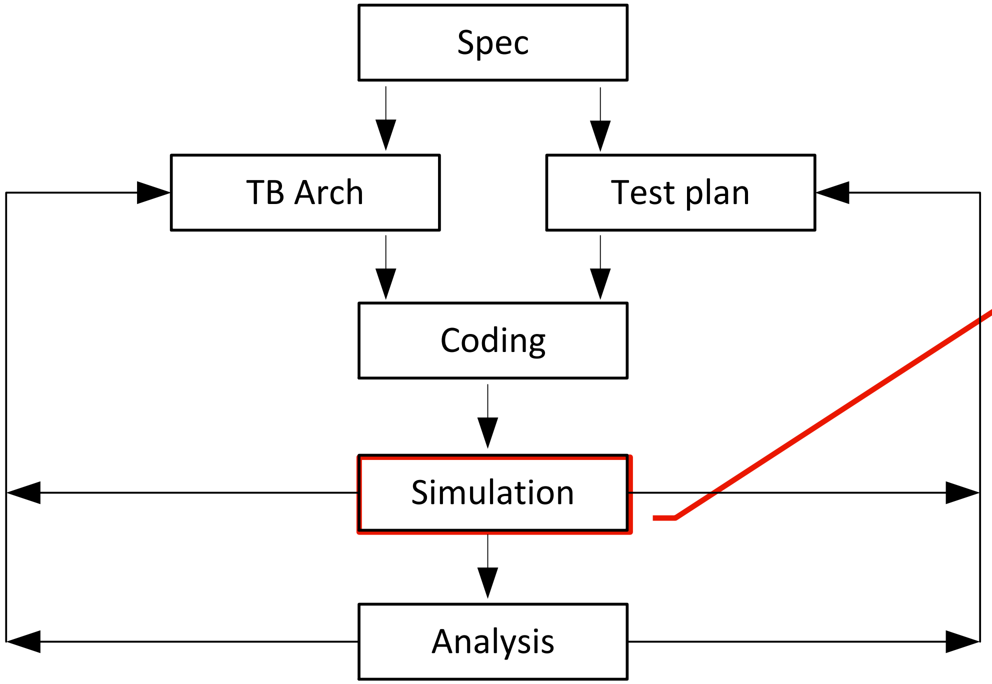
Background

- Intel® RealSense™ camera fits remarkable technology into a small package. There are three cameras that act like one - a 1080p HD camera, an infrared camera, and an infrared laser projector - they “see” like the human eye to sense depth and track human motion



Background

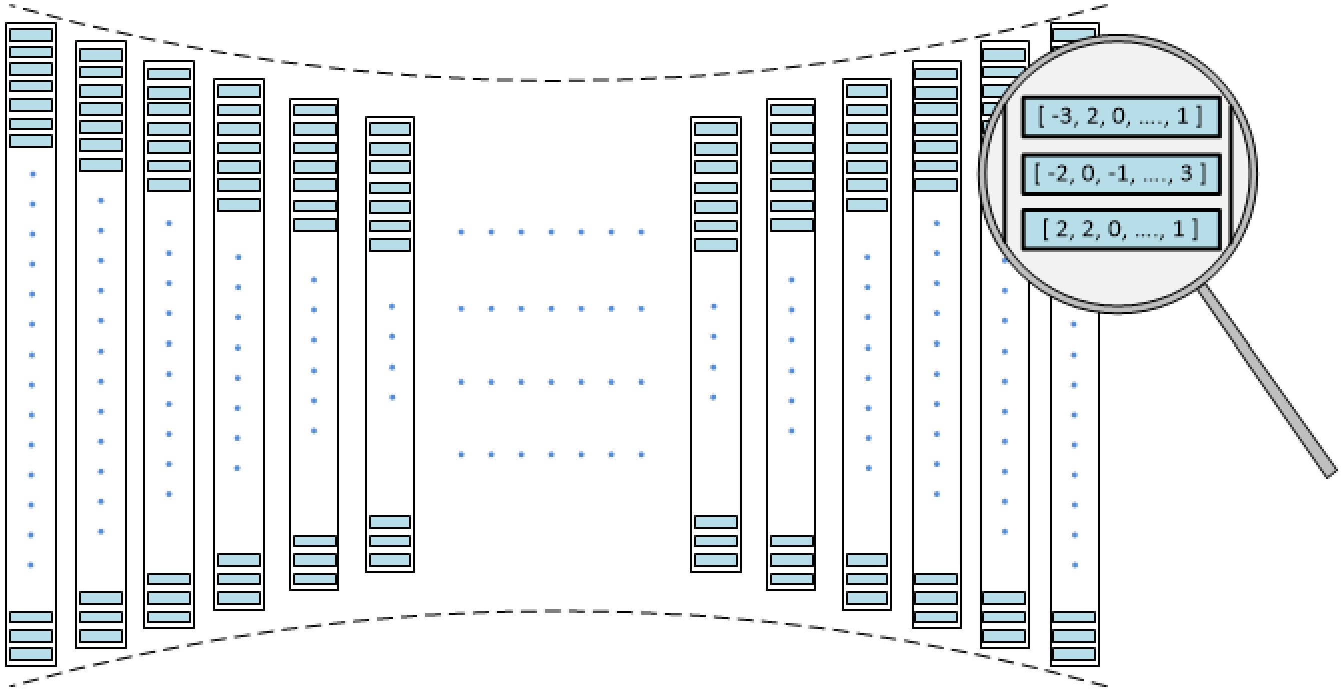
- General work flow (verification is done at the unit level):



- Randomize configuration
- Init DUT
- Randomize and start stimuli

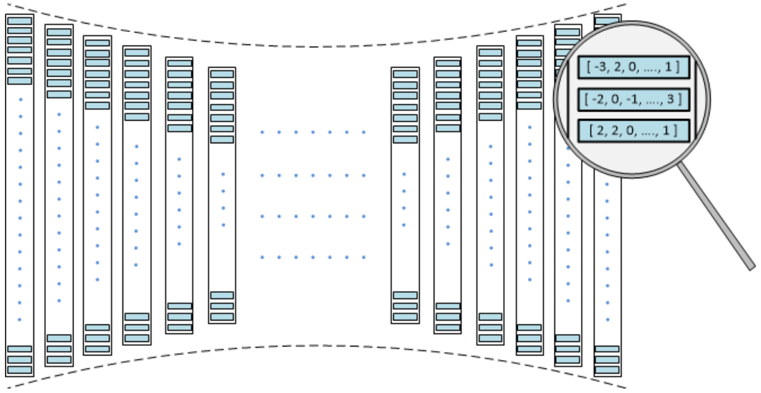
Background

- Unit level verification
  - Randomization of complicated data structure is needed



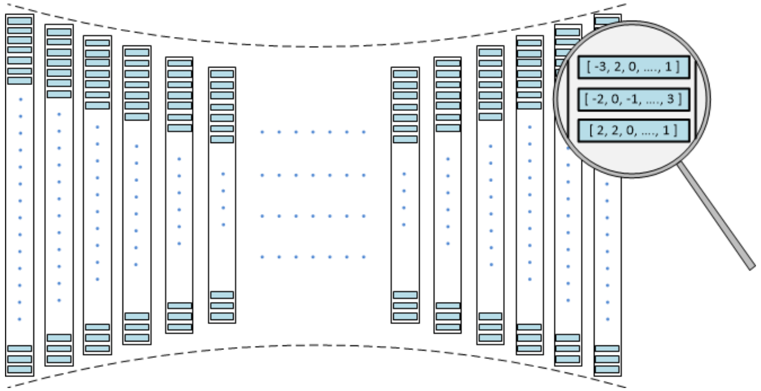
Background

- Complex relation between elements
  - The frame is an array of columns (up to 1920)
  - Each column is an array of “samples” (up to 5000, varied per column; light blue rectangles)
  - Each sample is an array of 16 attributes (array sum is limited)
  - Each of the 16 attributes in a sample is an integer in the range [-4:3]
  - More…



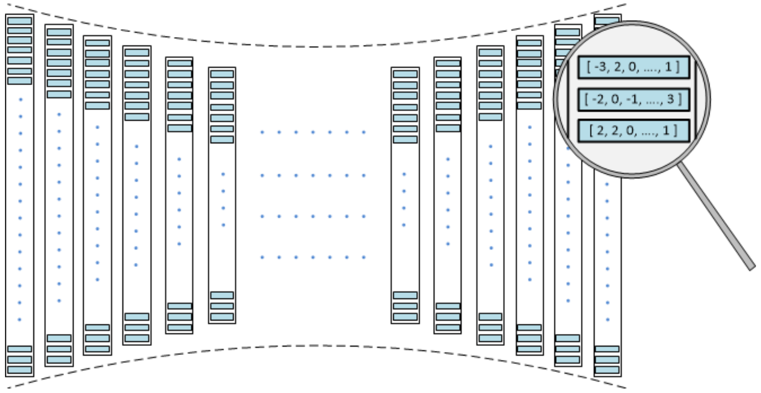
The Problem

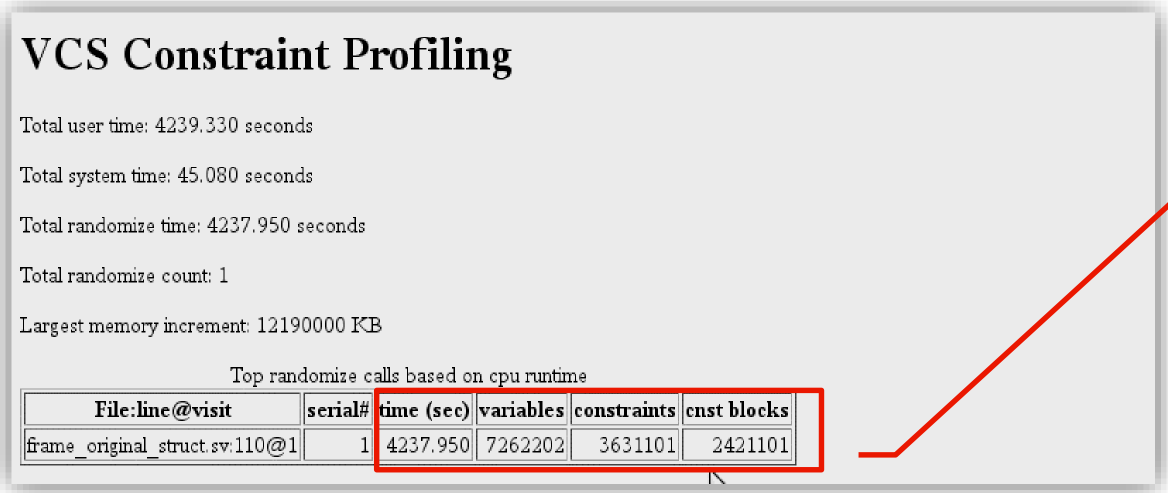
- Frame randomization time is too long!
  - Randomization of a full frame 1920x5000 (1920 columns, each with 5000 samples) took 17(!!) hours
  - In this presentation, we will demonstrate the measurements and analysis of a simpler testcase: 1100x1100 frame (1100 columns, each with 1100 samples



The Problem

- Problem analysis
  - We used the VCS Simulation Profiler to find the root cause of the problem
  - Constraint profiling report snippet for 1100x1100 frame





VCS Constraint Profiling

Total user time: 4239.330 seconds

Total system time: 45.080 seconds

Total randomize time: 4237.950 seconds

Total randomize count: 1

Largest memory increment: 12190000 KB

Top randomize calls based on cpu runtime

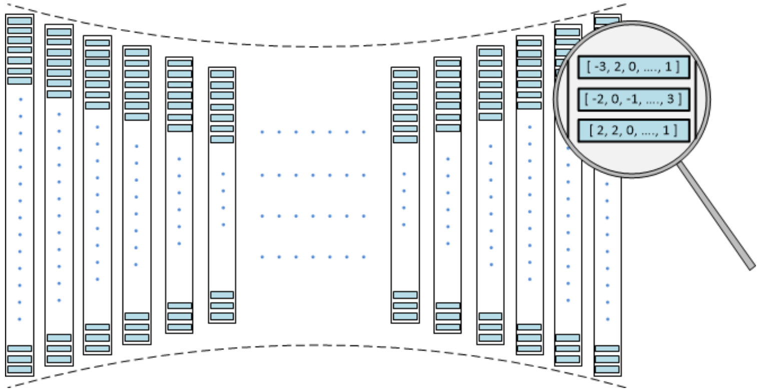
File:line@visit	serial#	time (sec)	variables	constraints	cnst blocks
frame_original_struct.sv:110@1	1	4237.950	7262202	3631101	2421101

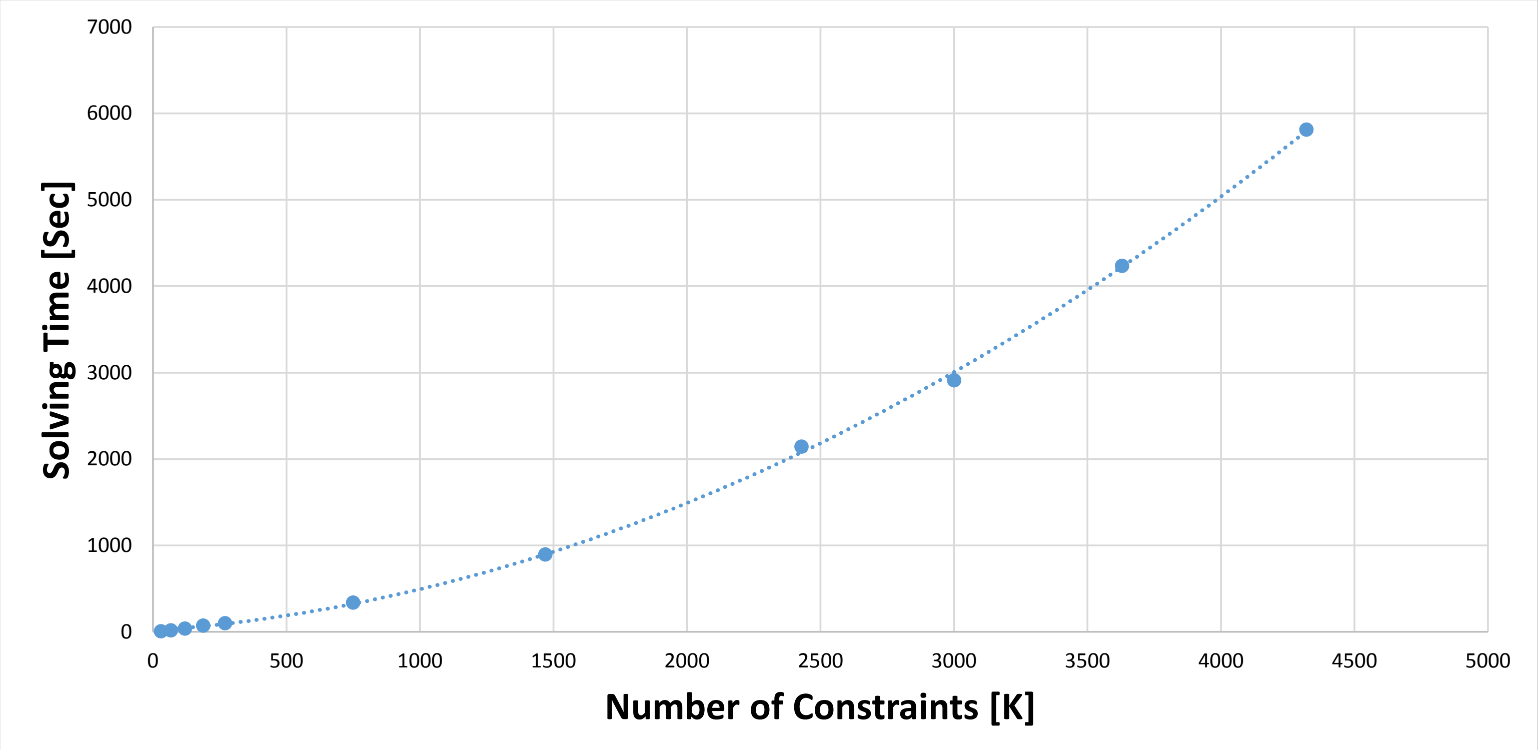
Total randomization time: 71 min.

Number of constraints: > 3.6M

The Problem

- Exponential relation between the number of constraints and the time and memory consumption of the simulation





The Solutions

- Decreasing the number of constraints
  - Technique 1: Variable type matching - do we really need i(n)t?
  - Technique 2: Declarative to sequential - porting to post_randomize
  - Technique 3: Divide and conquer - on the fly randomization

The Solutions

- Technique 1: Variable type matching – do we really need i(n)t?

  – Straightforward approach

```
rand int attr[16];

constraint attr_span_c {
  foreach (attr[ii]) {
    attr[ii] inside {[-4:3]};
  }
}
```

The Solutions

- Technique 1: Variable type matching – do we really need i(n)t?

  - Straightforward approach

```
rand int attr[16];

constraint attr_span_c {
  foreach (attr[ii]) {
    attr[ii] inside {[-4:3]};
  }
}
```

  - Optimized code

```
rand bit signed [2:0] attr[16];
```

  - A 3-bit signed packed array values span the range [-4:3] (2’s complement)
  - No further constraint is needed

The Solutions

- Technique 1: Variable type matching – do we really need i(n)t?
  - Profiler results (1100x1100)

Attribute. var type	Number of Constraints	Avg. solving time [sec]
Int	3,631,101	4,237 (71 min.)
Singed packed array	1,211,101	**1,270 (21.1 min.)**

  - Total improvement so far: **70%**

⚠ Asymmetric span may require additional constraint or other adjustments

The Solutions

- Technique 2: Declarative to sequential - porting to post_randomize
  - Straightforward approach

```
// define new data type, to be used in casting
typedef bit signed [7:0] signed_8_bit_t;

rand bit signed [2:0] attr[16];

constraint attr_sum_c {
  attr.sum() with (signed_8_bit_t'(item)) == 1;
}
```

The Solutions

- Technique 2: Declarative to sequential - porting to post_randomize
  - Optimized code

```
function void sample::post_randomize();

  // adjust array values to fit target sum
  if (diff_sum > 8'sb0) begin

    while (1) begin
      if (attr[idx]>-3'sh4) begin
        attr[idx] -= 3'sb1;
        cntr      += 8'sb1;
      end
      if (cntr == diff_sum) break;
      idx = (idx+7)%16;
    end

  // symmetric operation for (diff_sum < 8'sb0)
```

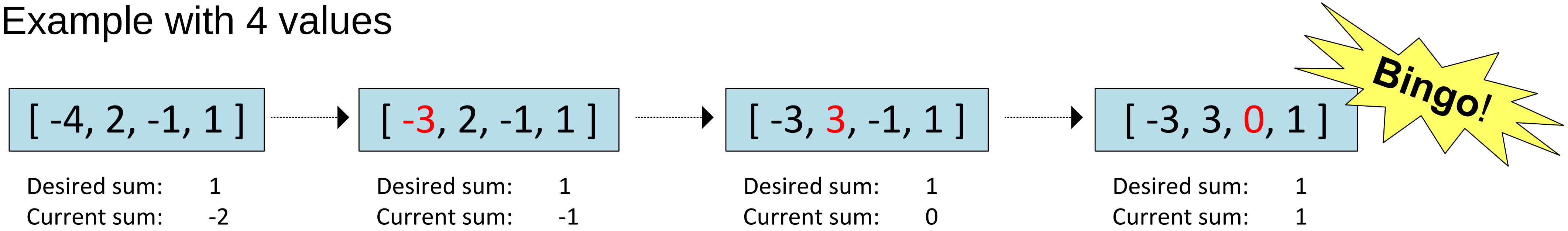
The Solutions

- Technique 2: Declarative to sequential - porting to post_randomize

  – Optimized code

    - Remove the sum() constraint
    - Let the constraint solver assign 16 random attributes
    - Iterate through the array and adjusted its values, so its sum will match the desired value

  – Example with 4 values



[-4, 2, -1, 1]	[-3, 2, -1, 1]	[-3, 3, -1, 1]	[-3, 3, 0, 1]
Desired sum: 1	Desired sum: 1	Desired sum: 1	Desired sum: 1
Current sum: -2	Current sum: -1	Current sum: 0	Current sum: 1

The Solutions

- Technique 2: Declarative to sequential - porting to post_randomize
  - Profiler results (1100x1100)

Sum Method	Number of Constraints	Avg. solving time [sec]
Constraint	1,211,101	1,270 (21.1 min.)
Post randomize	1,101	**9**

  - Total improvement so far: **99.8%**
  - Similar statistical attributes

⚠ Other constraints (dist, implication) needs different treatment

The Solutions

- Technique 3: Divide and conquer - on the fly randomization

  – Straightforward approach

```
// create the frame
m_frame = new;

// randomize and send to the driver
m_frame.randomize();

// send frame to the driver
```

The Solutions

- Technique 3: Divide and conquer - on the fly randomization

  – Straightforward approach

  ```
  // create the frame
  m_frame = new;

  // randomize and send to the driver
  m_frame.randomize();

  // send frame to the driver
  ```

  – Optimized code

  ```
  // create the frame
  m_frame = new;

  foreach (m_frame.column_arr[ii]) begin
    m_frame.column_arr[ii].randomize();

    // send column to the driver
  end
  ```

The Solutions

- Technique 3: Divide and conquer - on the fly randomization

  – Profiler results (1100x1100)

Sum Method	Number of Constraints	Avg. solving time [sec]
Constraint	1,101	9
Post randomize	1	**2**

  – Total improvement so far: **99.95%**

Summary

- When encounter performance issues - use the profiler to pinpoint the root cause

- Optimize complex constraint structure:
  - Variable type matching
  - Declarative to sequential
  - Divide and conquer

- Final results:
  - Full frame 17 hours to 90 seconds
  - Testcase 71 minutes to 2 seconds

Thank You

Backup