Compare timing performance between the InTime Default recipe and a Placement Seed Sweep

01 Executive Summary

This whitepaper compares the effectiveness of two timing optimization methods: The InTime Default recipe provided by the InTime FPGA design optimization tool and another one commonly known as a “Seed Sweep”.

InTime Default is a machine learning approach that finds good synthesis and place-and-route setting combinations for a design. It shares data insights across different designs and produces predictable effects.

A Seed Sweep varies the Quartus Fitter SEED value which affects the initial placement of a design. Changing the seed modifies the conditions of a design at the start of place-and-route and leads to fluctuations in the Fitter results. This is a well-known approach used by design teams worldwide to optimize their designs. However, the effects are random and there is no “golden” seed value that applies to all designs.

The experiment described here is performed using a Stratix V design and an Arria 10 design, each compiling 200 data points for the InTime Default recipe and 200 data points for a Seed Sweep for each design. The best Worst Slack (WS), Total Negative Slack (TNS) , Fmax and runtimes are then compared.

For the Stratix V design, InTime Default improved the Worst Slack by 57.38% while Seed Sweep only had 13.46% improvement.

For the Arria 10 design, InTime Default recipe improved the Worst Slack by 34.98% while Seed Sweep only had 11.44% improvement.

In summary, the InTime Default recipe performed better than a Seed Sweep with respect to timing performance.

02 Design Details

The 2 designs used for this experiment is listed below.

Table 2.1: Stratix V design

Design Info	
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Standard Edition
Revision Name	jesdsv
Top-level Entity Name	jesdrx
Family	**Stratix V**
Device	**5SGSMD3H3F35I4**
Timing Info	
Clock name	rxlink_clk
Corner	Slow 850mV -40C Model
Worst Slack , WS (ns)	-0.765
Total Negative Slack, TNS (ns)	-897.292
Fmax (MHz)	306.28
Utilization Info	
Logic utilization (in ALMs)	42,553 / 89,000 (48 %)
Total registers	47330
Total pins	138 / 544 (25 %)
Total virtual pins	6,540
Total block memory bits	81,920 / 14,090,240 (< 1 %)
Total DSP Blocks	0 / 600 (0 %)
Total PLLs	0 / 52 (0 %)
Total DLLs	0 / 4 (0 %)

Table 2.2: Arria 10 design

Design Info	
Quartus Prime Version	17.1.0 Build 240 10/25/2017 SJ Pro Edition
Revision Name	jesda 10
Top-level Entity Name	jesdrx
Family	**Arria 10**
Device	**10AX027E4F27E3LG**

Timing Info	
Clock name	rxlink_clk
Corner	Slow 900mV 100C Model
Worst Slack , WS (ns)	-0.769
Total Negative Slack, TNS (ns)	-977.742
Fmax (MHz)	305.90
Utilization Info	
Logic utilization (in ALMs)	52,860 / 101,620 (52 %)
Total registers	56480
Total pins	66 / 296 (22 %)
Total virtual pins	8,262
Total block memory bits	102,400 / 15,360,000 (< 1 %)
Total DSP Blocks	0 / 830 (0 %)
Total HSSI RX channels	0 / 12 (0 %)
Total HSSI TX channels	0 / 12 (0 %)
Total PLLs	0 / 32 (0 %)

03 Test Procedures

The experiment was performed following below steps:

1. Open Stratix V design in InTime.
2. Run InTime Default recipe for 10 rounds. Each round will run 20 compilations.
3. Run Seed Sweep for 200 compilations, each with a different seed value.
4. Compare the Worst Slack (WS), Total Negative Slack (TNS) and Fmax.
5. Repeat Steps 1 to 4 for the Arria 10 design.

04 Results

Table 4.1 and 4.2 below show the timing comparison between Stratix V and Arria 10 designs using WS and TNS.

Table 4.1: Worst Slack, WS comparison

Best Worst Slack, WS (ns)					
	Original (ns)	Seed Sweep		InTime Default	
		Slack (ns)	Improvement (%)	Slack (ns)	Improvement (%)
Stratix V design	-0.765	-0.662	13.46	-0.326	57.38
Arria 10 design	-0.769	-0.681	11.44	-0.5	34.98

Table 4.2: Total Negative Slack, TNS comparison

Best Total Negative Slack, TNS (ns)					
	Original (ns)	Seed Sweep		InTime Default	
		Slack (ns)	Improvement (%)	Slack (ns)	Improvement (%)
Stratix V design	-897.292	-745.228	16.95	-39.736	95.57
Arria 10 design	-977.742	-635.613	34.99	-75.44	92.28

Table 4.3: Fmax comparison

Fmax (MHz)					
	Original (MHz)	Seed Sweep		InTime Default	
		Fmax (MHz)	Improvement (%)	Fmax (MHz)	Improvement (%)
Stratix V design	306.28	316.25	3.26	353.89	15.54
Arria 10 design	305.90	314.37	2.76	333.33	8.97

05 Optimization Process and Run Time

Running a Seed Sweep is a simple process of specifying “Fitter Seed” values from 1 to 200. The default value is 1. The InTime Default recipe involves running 10 rounds of 20 compilations each. Each round starts after the previous round has ended. Therefore, there are differences in the overall runtime of the two approaches.

Table 5.1 below shows design runtime differences. It is worth noting that the average runtime for InTime Default is higher than that for Seed Sweep. In addition, there are larger variations in the compilation times for InTime Default as the settings attempted are more varied than seeds. The total runtime is also longer for InTime Default due to the need for analyzing results at the end of each round to generate settings for the next.

Table 5.1: Run Time comparison based on different concurrent runs and rounds

Run Time (h)						
	Original Run Time (h)	Concurrent Runs	Seed Sweep (1 round of 200 compilations)		InTime Default (10 rounds of 20 compilations)	
			Total Run Time (h)	Avg Run Time (h)	Total Run Time (h)	Avg Run Time (h)
Stratix V design	0.5	5	26	0.6	37	0.7
Arria 10 design	0.5	3	37	0.8	38	1.0

05.1 Stratix V design results

Here are the results for the Stratix V design.

05.1.1 Seed Sweep

Figure 5.1.1: TNS(ns) values for 200 seeds

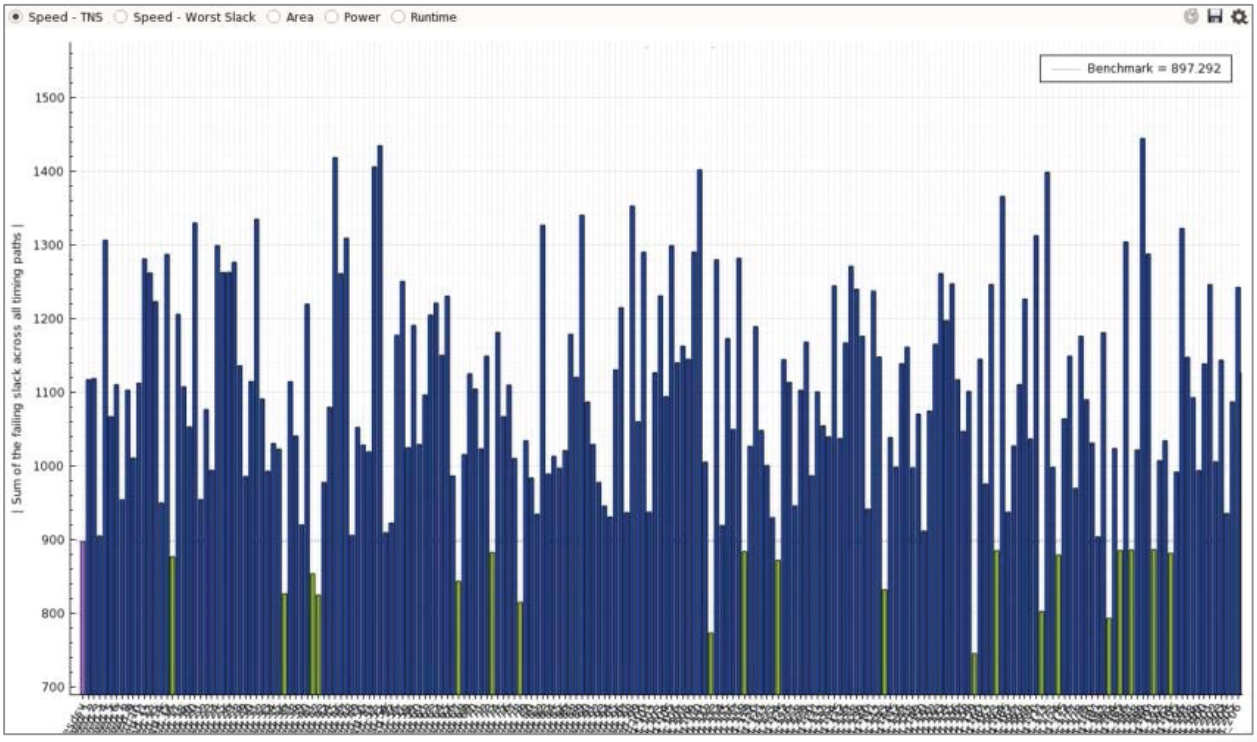
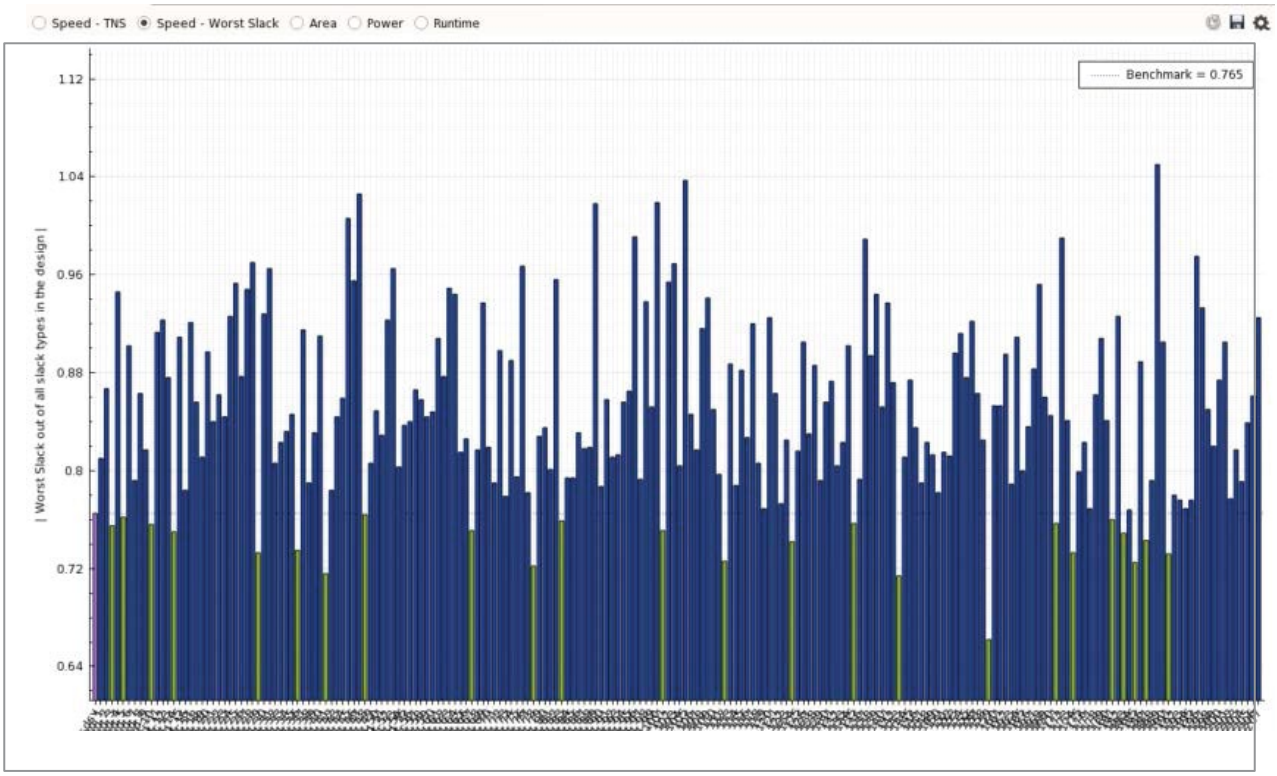


Figure 5.1.2: WS(ns) values for 200 seeds



05.1.2 InTime Default

Figure 5.1.3: TNS values for InTime Default

(Note: The Y-axis is in a logarithmic scale due to larger fluctuations)

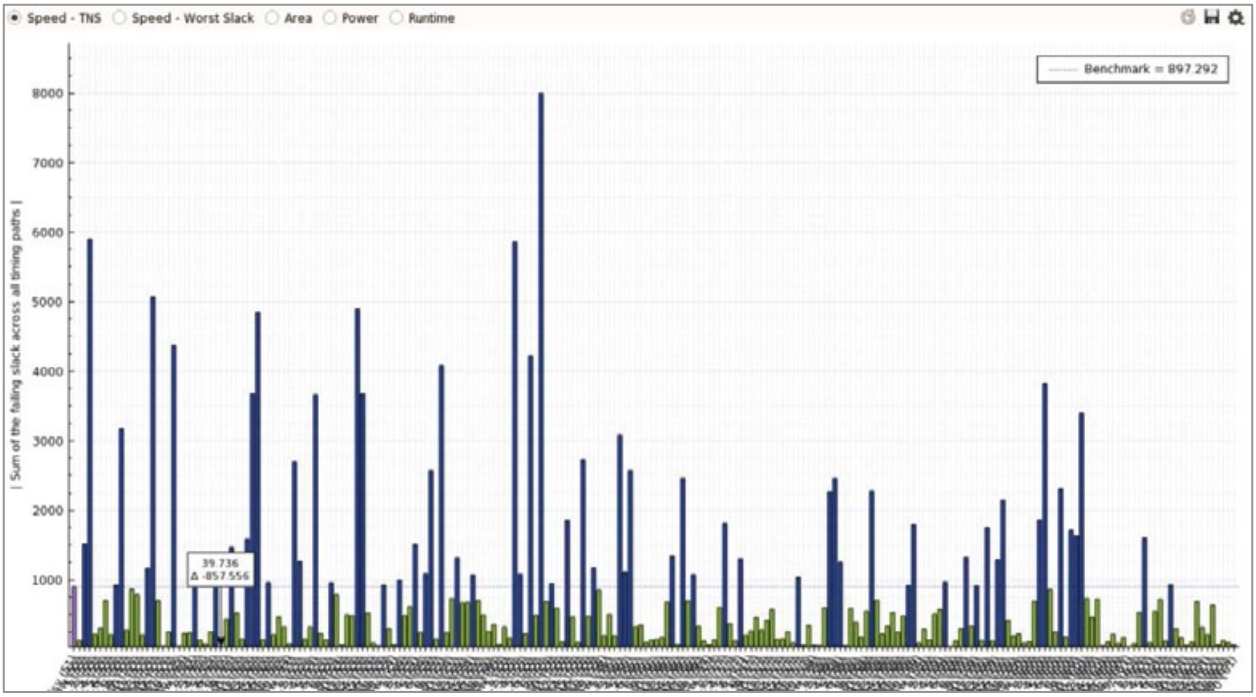
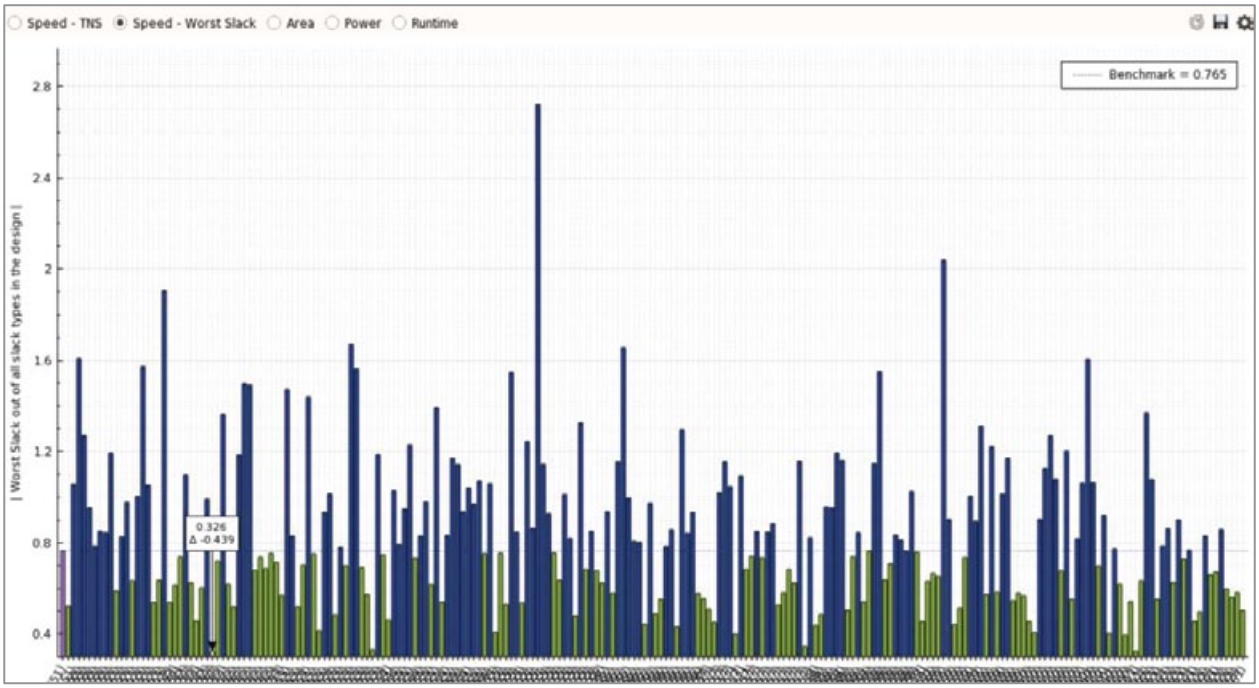


Figure 5.1.4: WS values for InTime Default



The chart below shows how the results improved across rounds. The X-axis displays job numbers. Each column represents the timing results for a job. The green line is the best result in each job and the red line is the worst.

Figure 5.1.5: TNS values for InTime Default

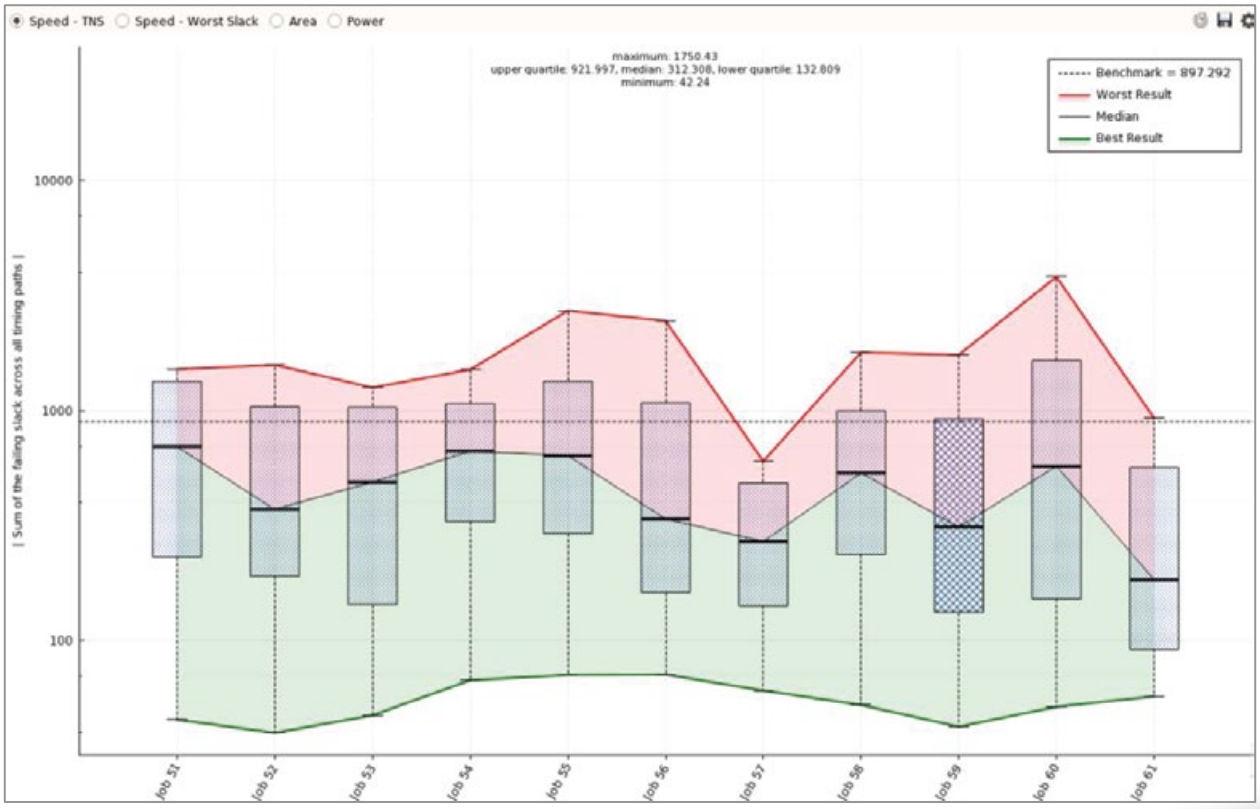


Figure 5.1.6: WS values for InTime Default

05.2 Arria 10 design results

Here are the results for the Arria 10 design.

05.2.1 Seed Sweep

The following chart shows the results for 1 to 200 seed values. Green represents results that are better than the original result and blue indicates results that are worse.

Figure 5.2.1: TNS (ns) values for 200 seeds

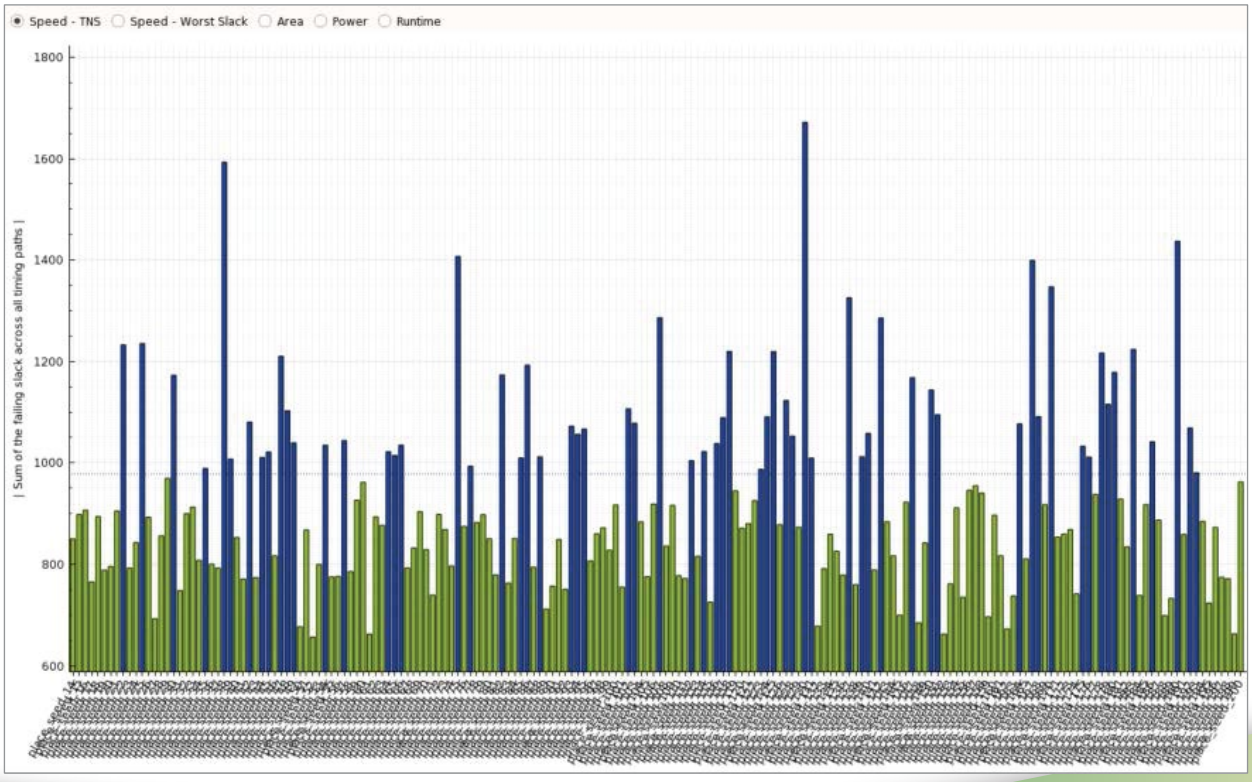
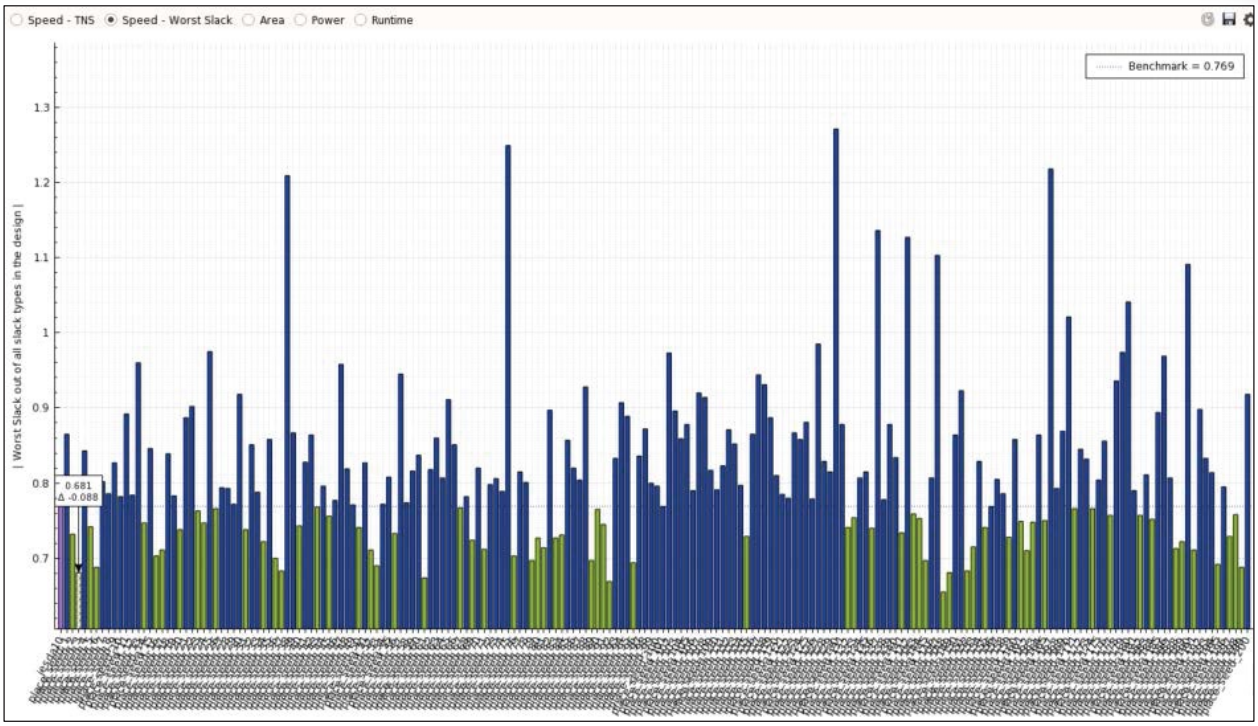


Figure 5.2.2: WS(ns) values for 200 seeds



05.1.2 InTime Default

InTime runs 20 compilations each time for 10 rounds. When each round completes, InTime analyzes and learns from the data to generate the parameters for the next round.

The following charts show the results for 200 compilations of InTime. Note that the Y-axis is in a logarithmic scale due to larger fluctuations in timing results.

Figure 5.2.3: TNS values for InTime Default

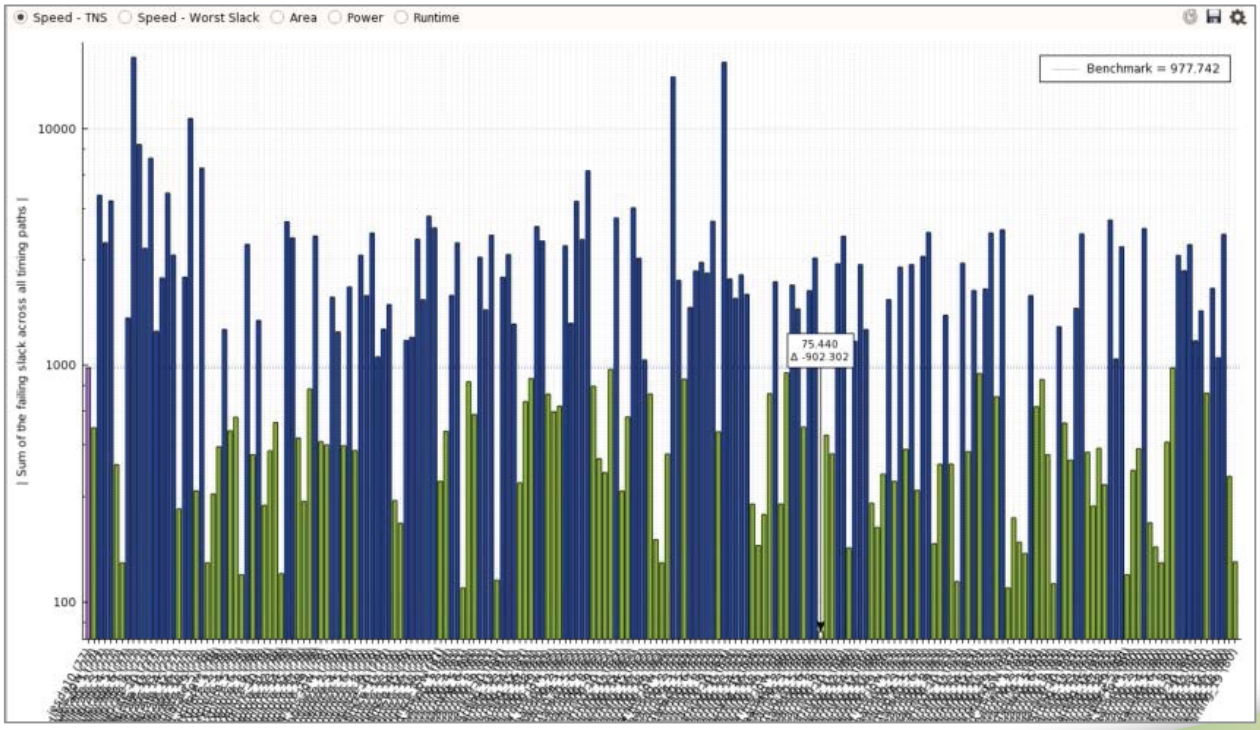
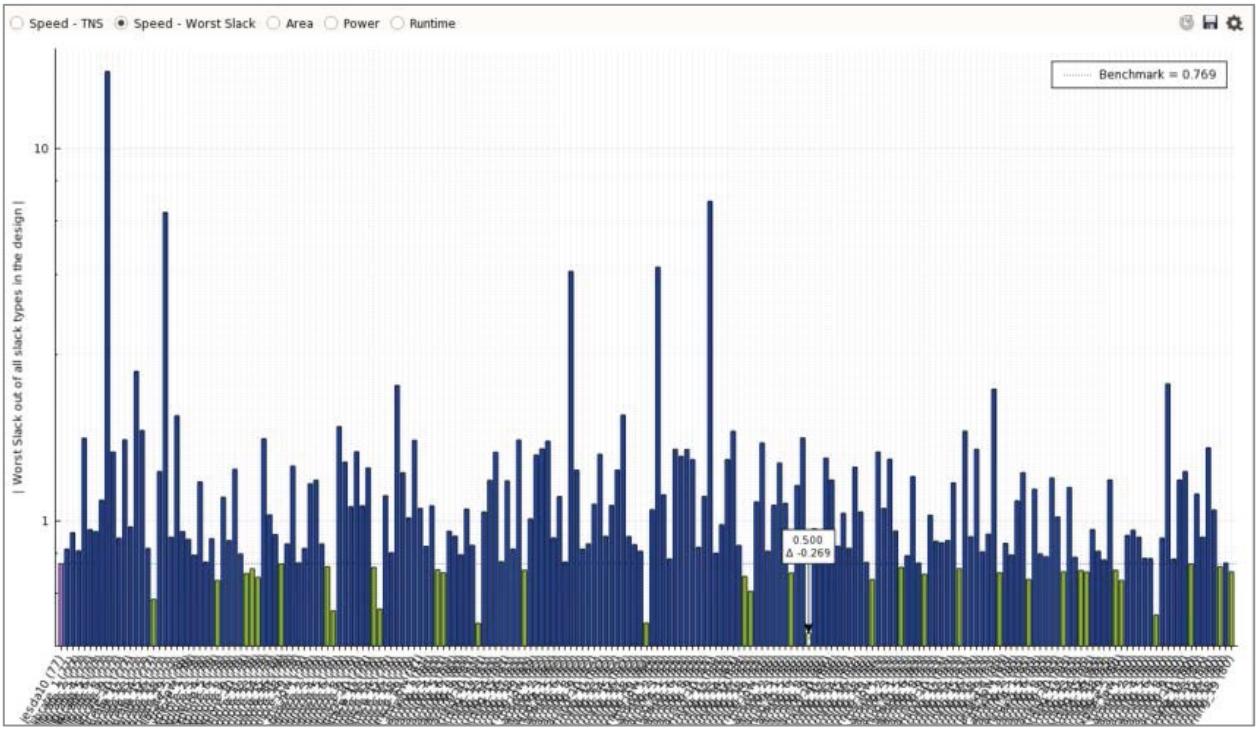


Figure 5.2.4: WS values for InTime Default



The chart below shows how the results improve across rounds. The X-axis displays job numbers. Each column represents a job. The green line is the best result in each job and the red line is the worst.

Figure 5.2.5: TNS improvements for InTime Default (10 rounds)

Figure 5.2.6: WS values for InTime Default (10 rounds)

06 Conclusion

InTime is more effective at delivering higher timing performance. In the same number of compilations runs (200), InTime Default is able to produce 30% to 50% performance improvements as compared to only 11% to 14% gains for a Seed Sweep.

Furthermore, there are even greater performance improvements if both methods are used in conjunction – namely, by doing a Seed Sweep based on good InTime Default results. This approach is used by the InTime tool to enhance timing performance and accelerate time-to-market. The key is to first get a sufficiently good result before using seeds to get one over the finishing line.

To read more about InTime and its capabilities, please go to https://www.plunify.com/en/intime/