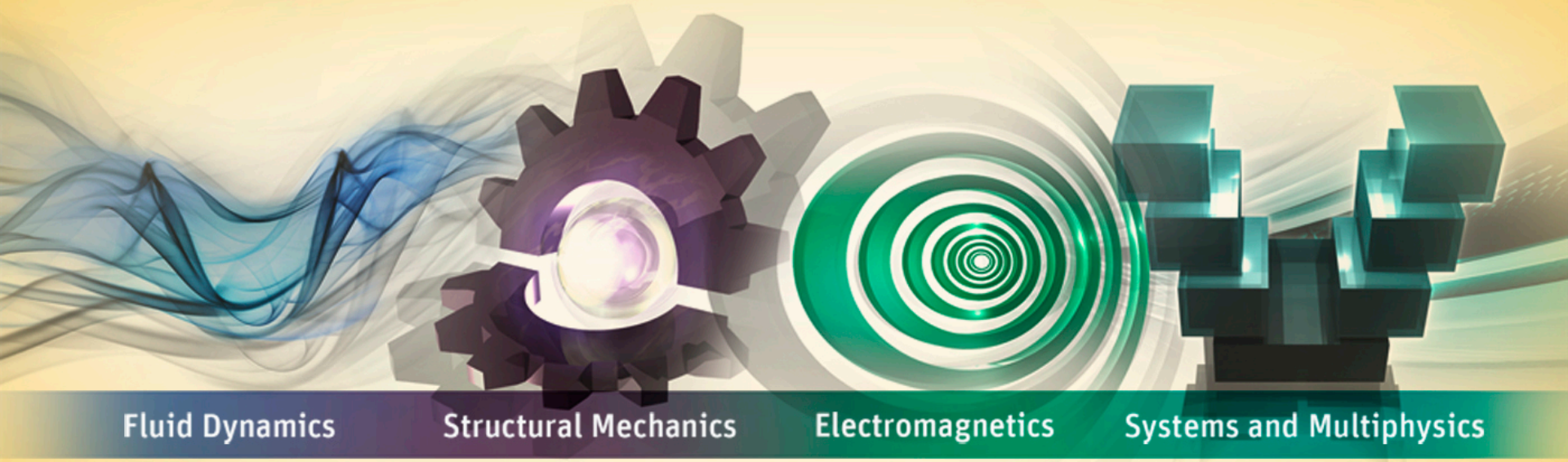
ANSYS

Realize Your Product Promise ®

SCADE Training



SCADE Suite Timing And Stack Optimizers

ESTEREL Technologies

Agenda

- Introduction to Timing and Stack Optimizers (TSO)
- Product overview
- Using TSO

INTRODUCTION

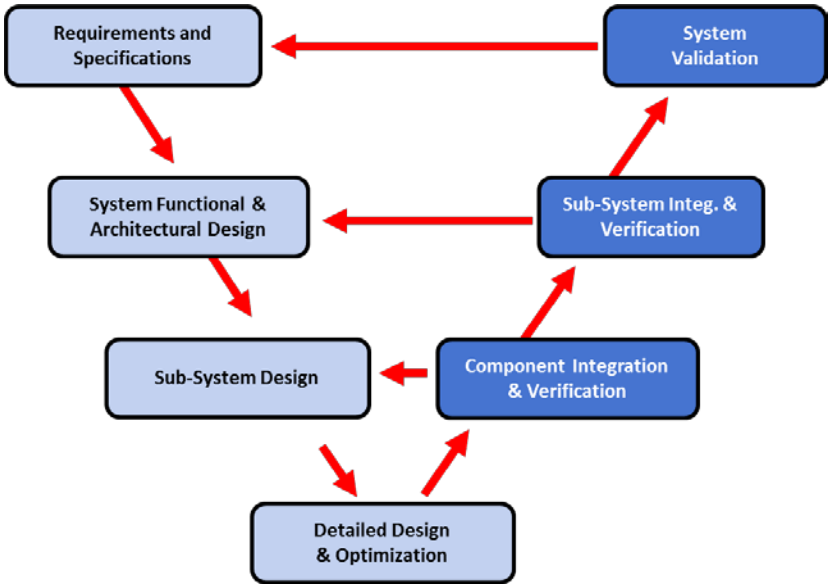
Timing and Stack Optimizers (TSO)

- **Computed for all possible inputs, produces an upper bound on the …**
  - **Execution time**
  - **Stack usage**

- **Noticeable features:**
  - Performs **static program analysis**
    - No need to write tests
  - **Lightweight tool configuration**
    - Uses a typical hardware (HW) model based on PPC 603e
    - No need for a precise target description

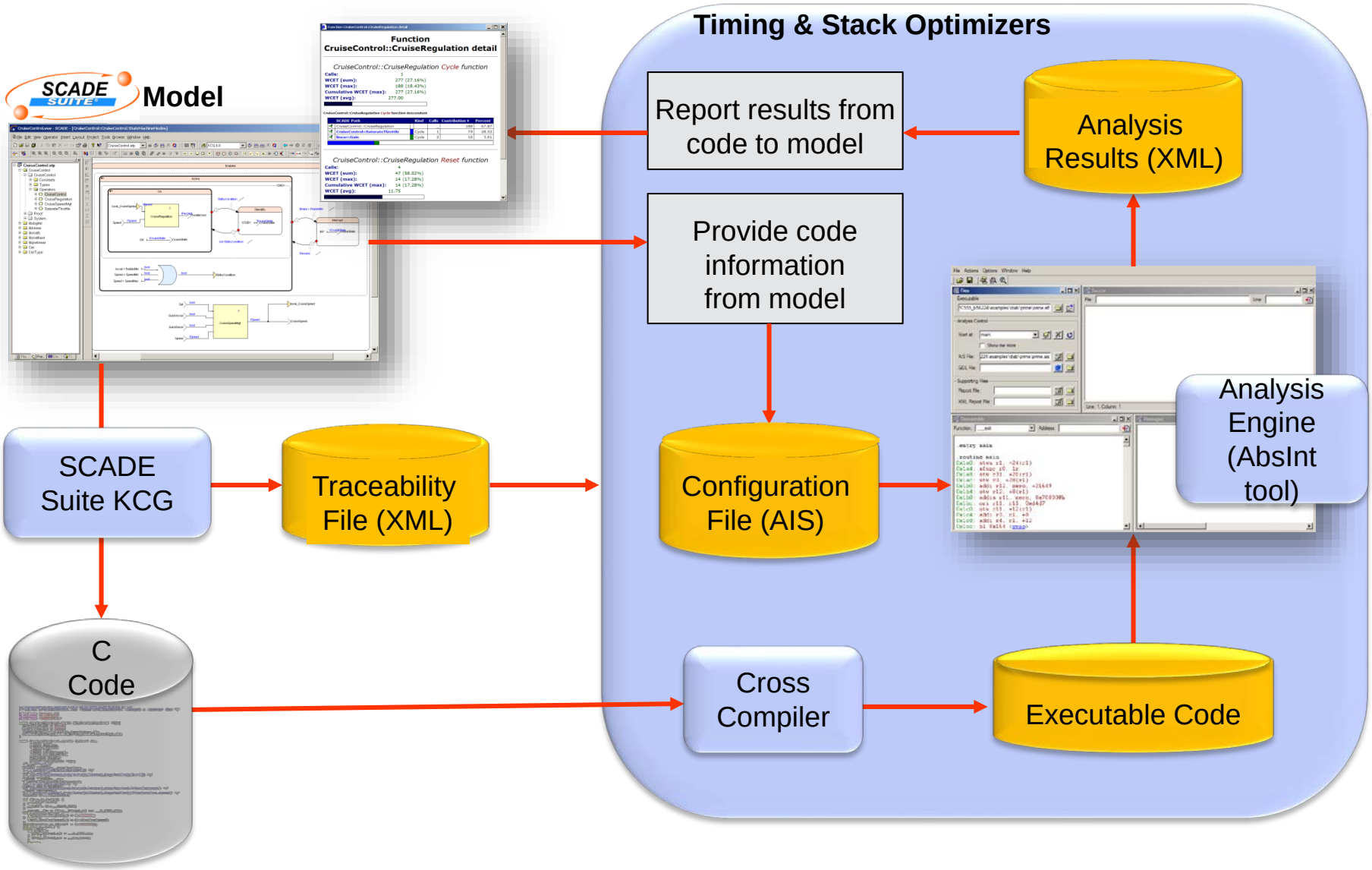
Timing and Stack Optimizers (TSO)

- **Debug tool to be used on the left hand side of the V-cycle**
  - Interactive mode to find timing bottlenecks and stack fillers
  - Design support tool
- **Allows to compare the performance of different designs**
- **Tool Qualification not required**



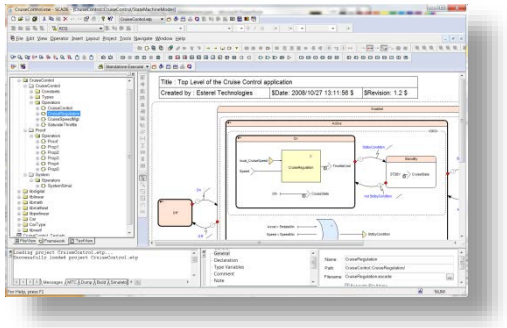
PRODUCT OVERVIEW

TSO Tool Architecture

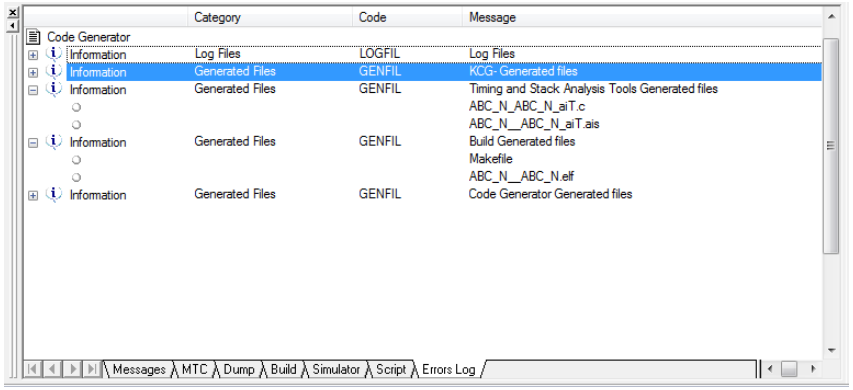


TSO Use Flow

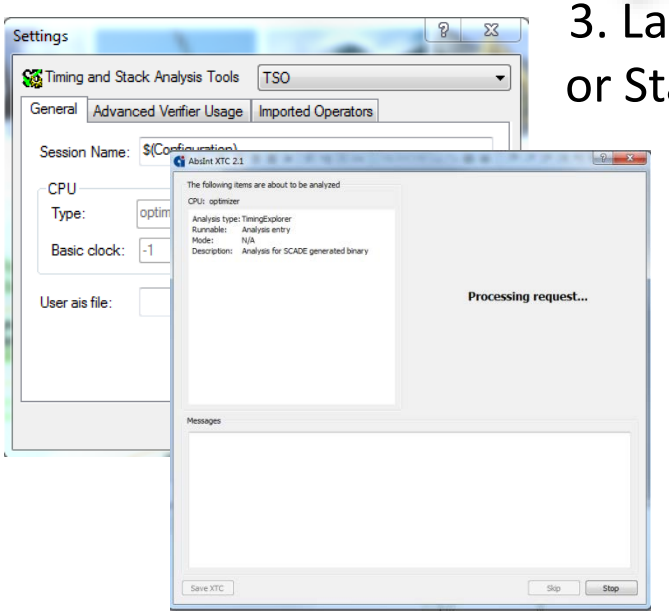
1. Create a TSO configuration

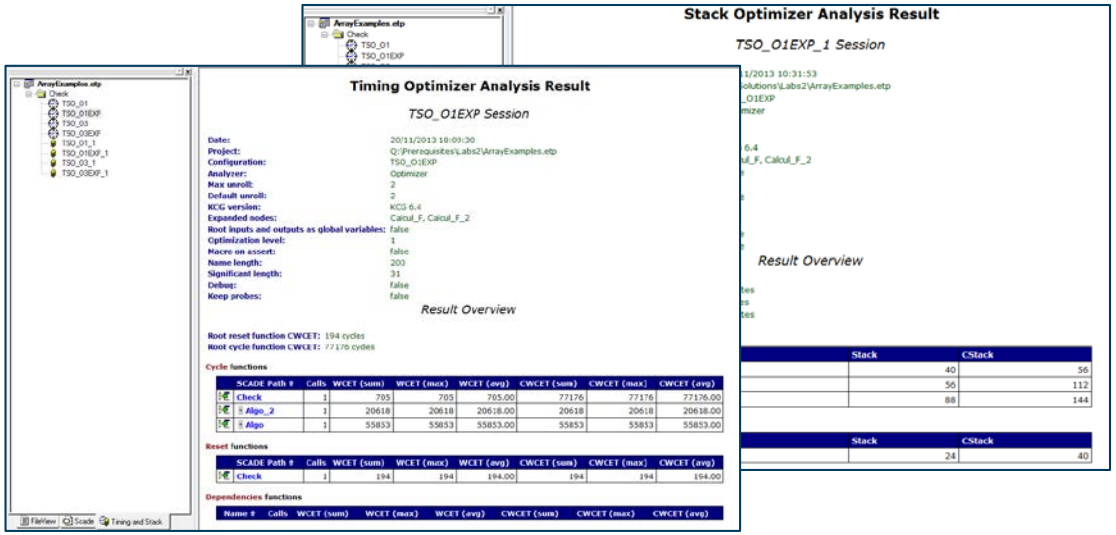


2. Build Code



3. Launch Timing or Stack Analyser





4. TSO Results Reports

Product Overview

- **Framework combining**
  - Two analyzers
    - Timing computation
    - Stack use
  - Mechanisms for reporting their results
    - Textually (XML files)
    - Graphically in the SCADE Editor
      - ***Timing and Stack tab*** to manage Timing and Stack sessions
      - Displaying Timing or Stack analysis results of a selected session from the *Timing and Stack* tab

Product Overview

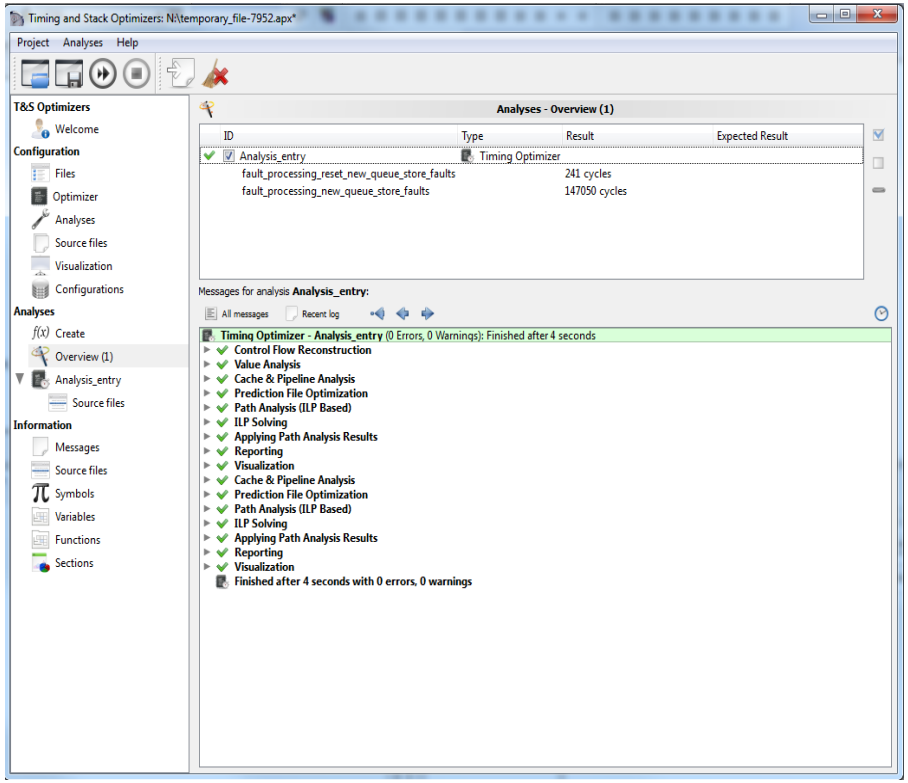
- **Timing Optimizer:**
  - Estimates upper bounds for the **Worst-Case Execution Time (WCET)** of an application or parts of the application
  - Why watching WCET upper bound from TSO is useful?
    - Gives WCET estimates early in the design phase
    - Helps supervising the evolution of WCET all along the design phase

    Note: partial estimations can be used as inputs for an overall timing analysis

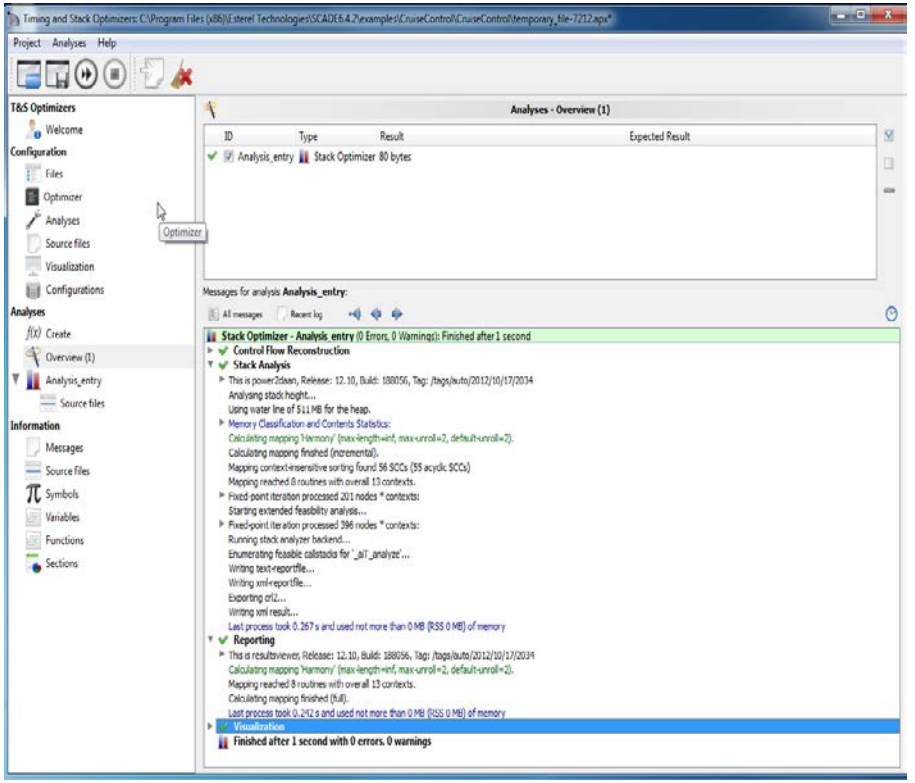
- **Stack Optimizer:**
  - Determines **upper bounds of the filling up of stacks** either in
    - Single routines or
    - The whole application
  - Why watching stacks upper bounds from TSO is useful?
    - Verifies the absence of stack overflow for a given setting of stack parameters
    - Gives a gauge to size the stacks

Product Overview

- **Interactive mode (AbsInt tool)**



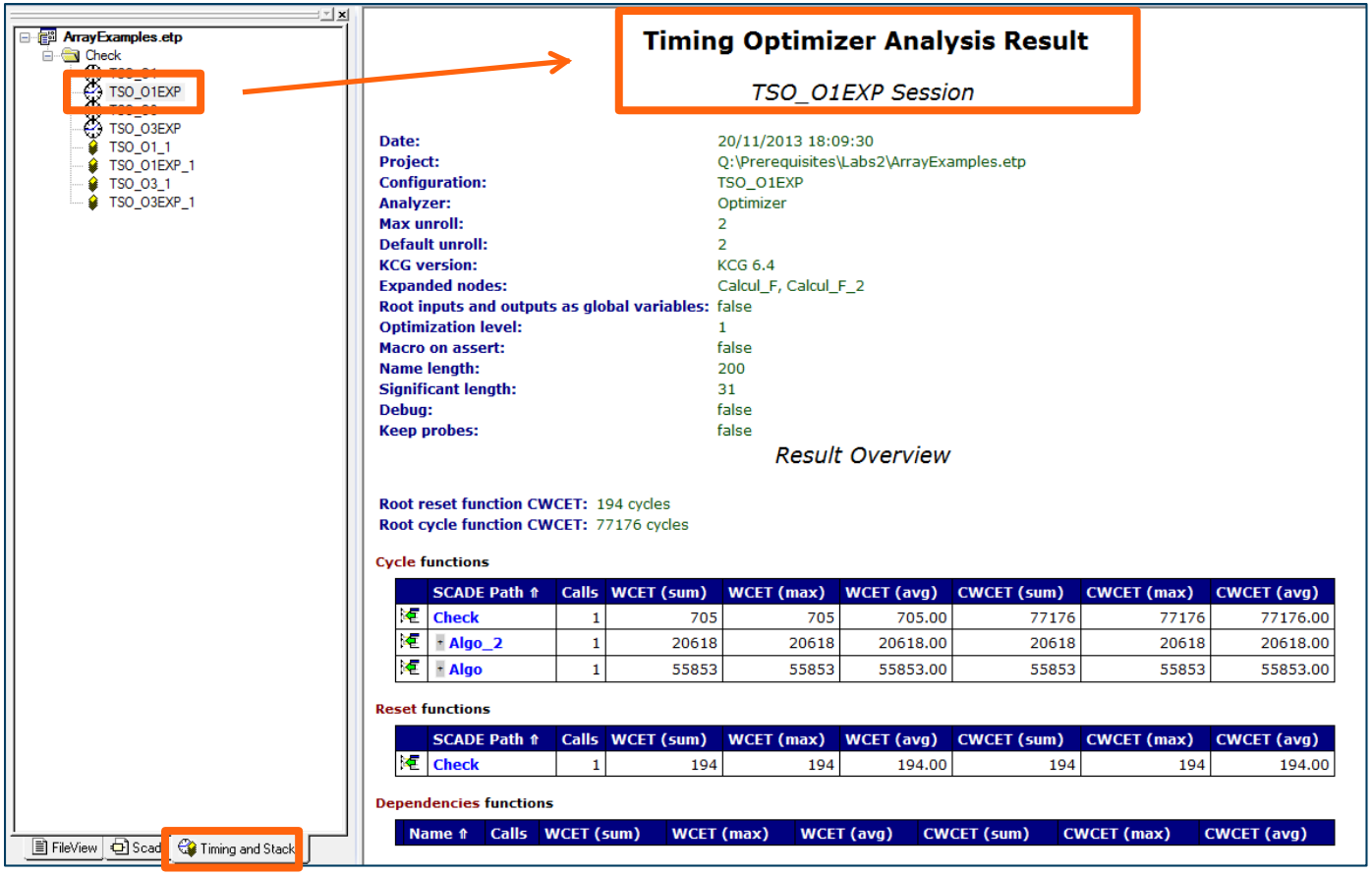
Timing



Stack

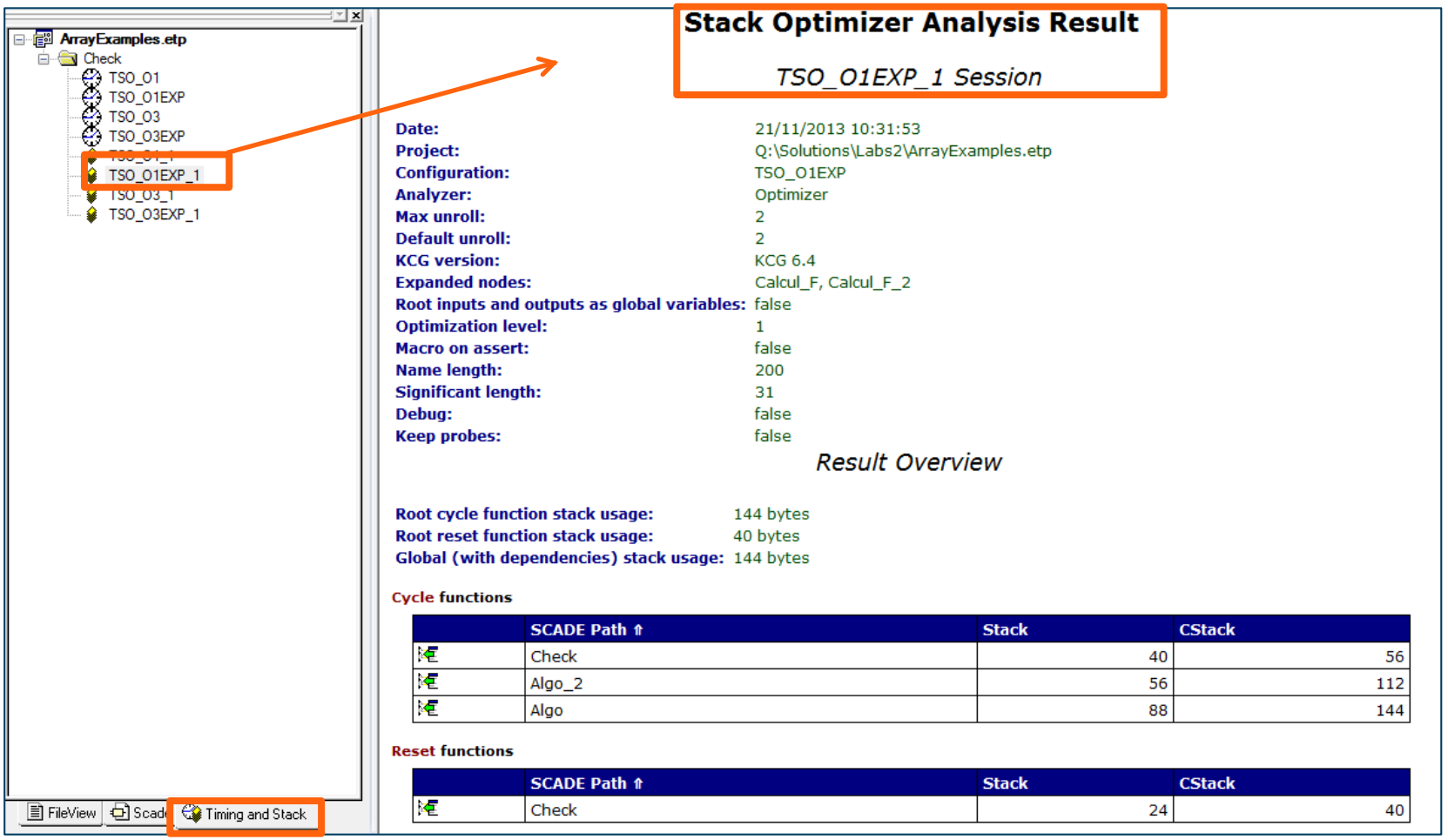
Product Overview

- **Results reporting**
  - Textually in .xml files
  - Graphically in the SCADE Editor



Timing Analysis

Product Overview



Stack Analysis

USING TSO

Using TSO: Configure The Tool

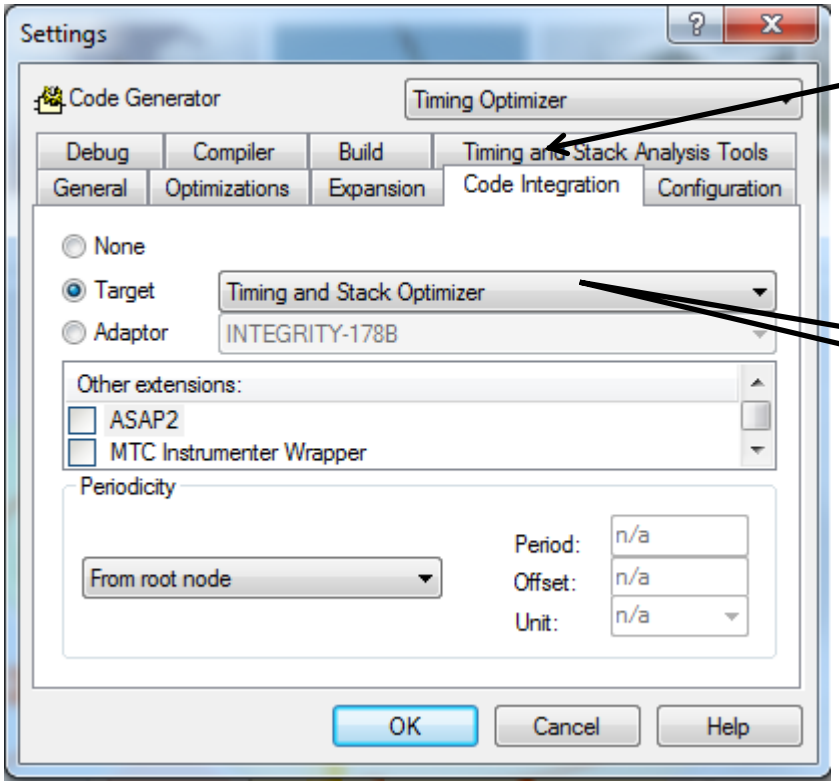
- **Click on the Settings icon of the Code Generator toolbar to display available settings**

- **To run Time or Stack Analysis on SCADE Suite models, it is recommended to use a specific configuration**
  - Create a new configuration from one of the list of available configurations as KCG

Tip: click on the top menu of the Workspace and check the Code Generator toolbar to display it

Using TSO: Configure The Tool

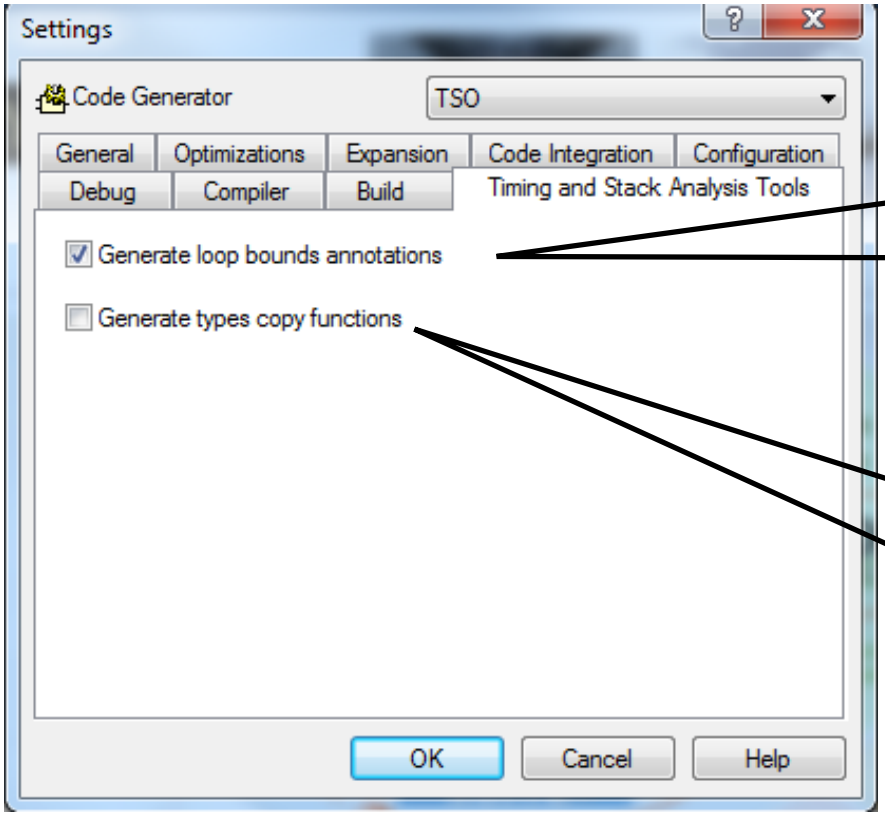
- **In the Code Integration tab**
  - Set the Target radio button and set *Timing and Stack Optimizer*



- Once this target is selected, "Timing and Stack Analysis Tools" tab is inserted

Select "Timing and Stack Optimizer" target

Using TSO: Configure The Tool

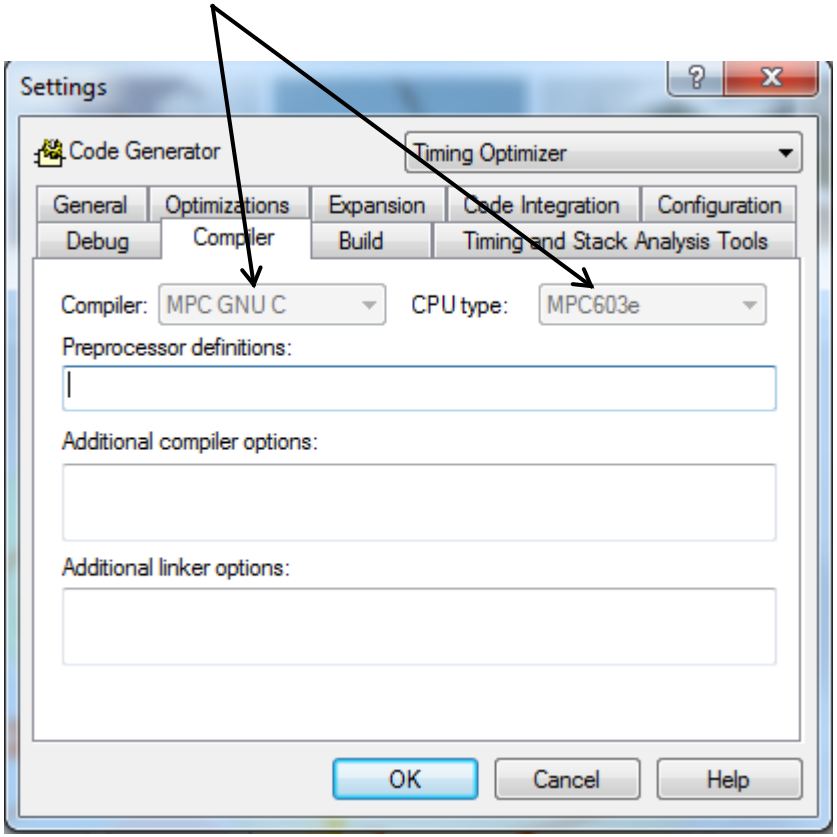


Uncheck "**Generate loop bounds annotations**" to suppress loop bounds annotations in the TSO engine configuration file (checked by default)

Check "**Generate type copy functions**" to generate copy functions field by field for complex types instead of using the _kcg_assign macro file (unchecked by default)

Using TSO: Configure The Tool

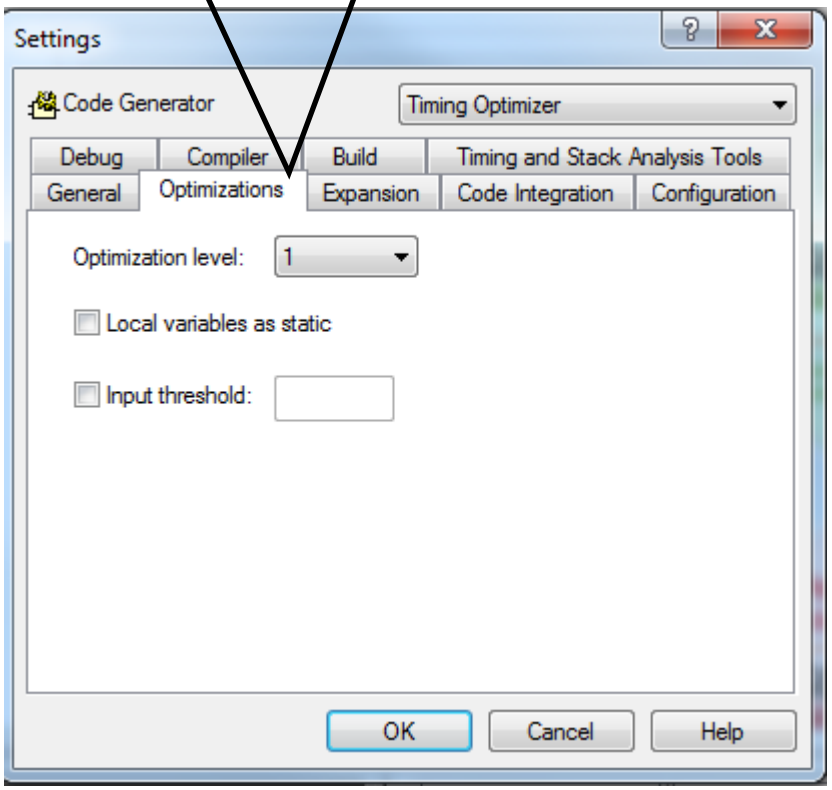
- **Standard Cross-Compiler and standard CPU are automatically chosen**



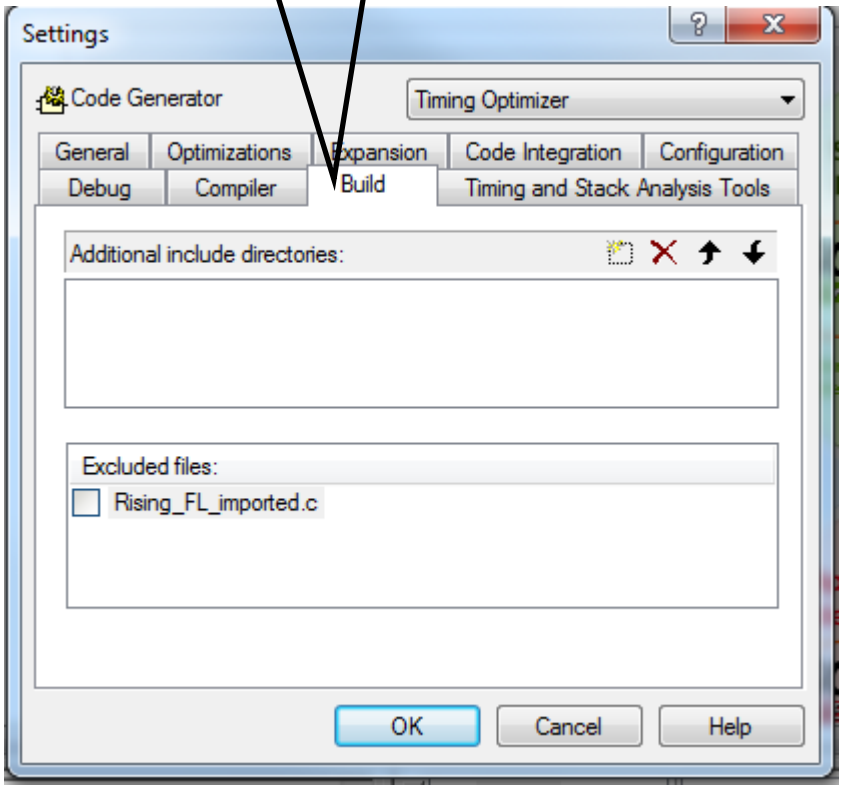
Using TSO: Configure The Tool

- **Reminders:**

Set Code generation optimizations in the Optimizations tab

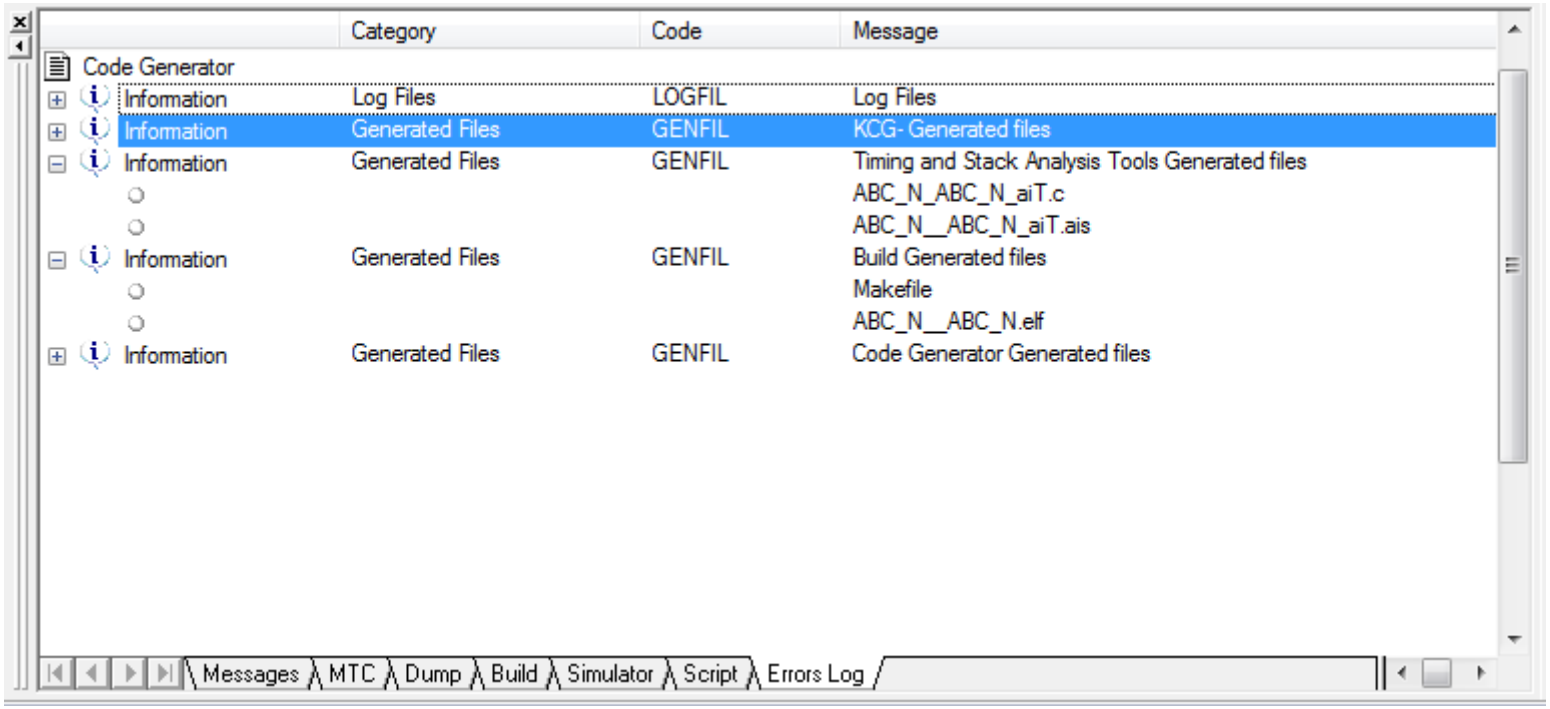


Set Build options in the Build tab



Using TSO: Generate The Executable

- **Click on the Build icon of the Code Generator toolbar**
  - To generate C code and
  - To produce an executable with the cross-compiler



Using TSO: Call The Tool

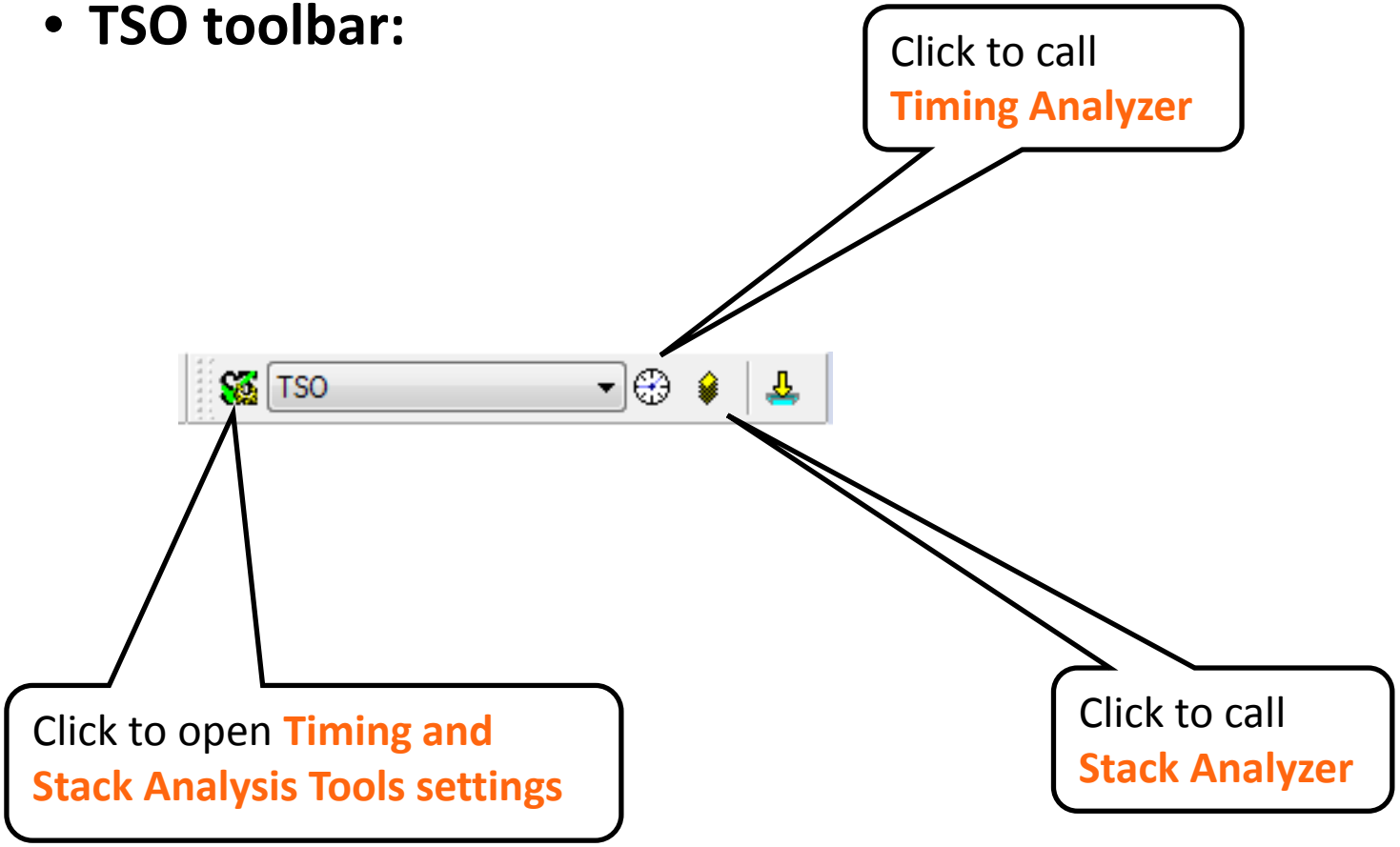
- **Use the dedicated toolbar in SCADE Suite to launch TSO tool**

Note: before building the TSO code, the toolbar is not active

Tip: click on the top menu of the Workspace and check the Timing and Stack Analysis Tools toolbar to display it

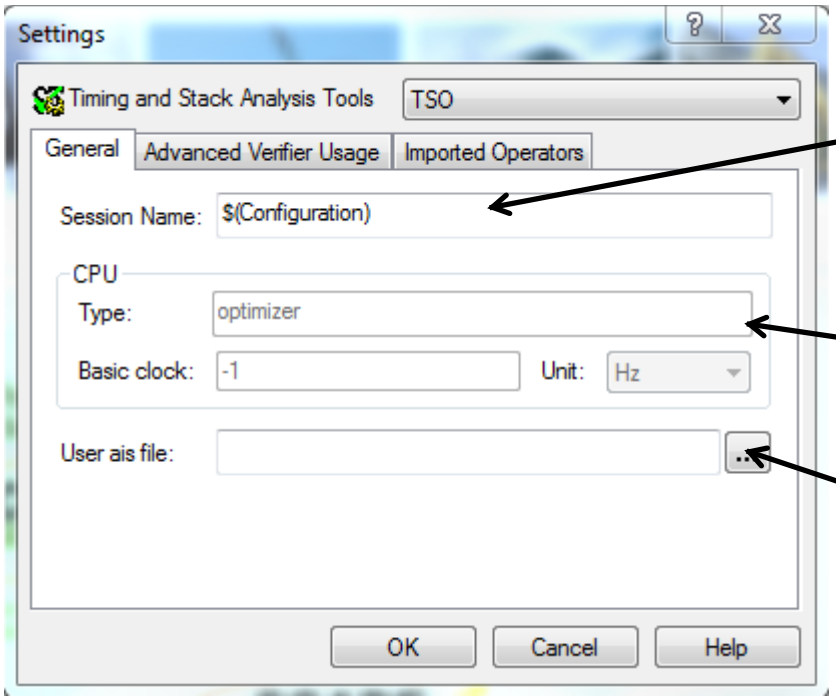
Using TSO: Call The Tool

- **TSO toolbar:**



Using TSO: Call The Tool

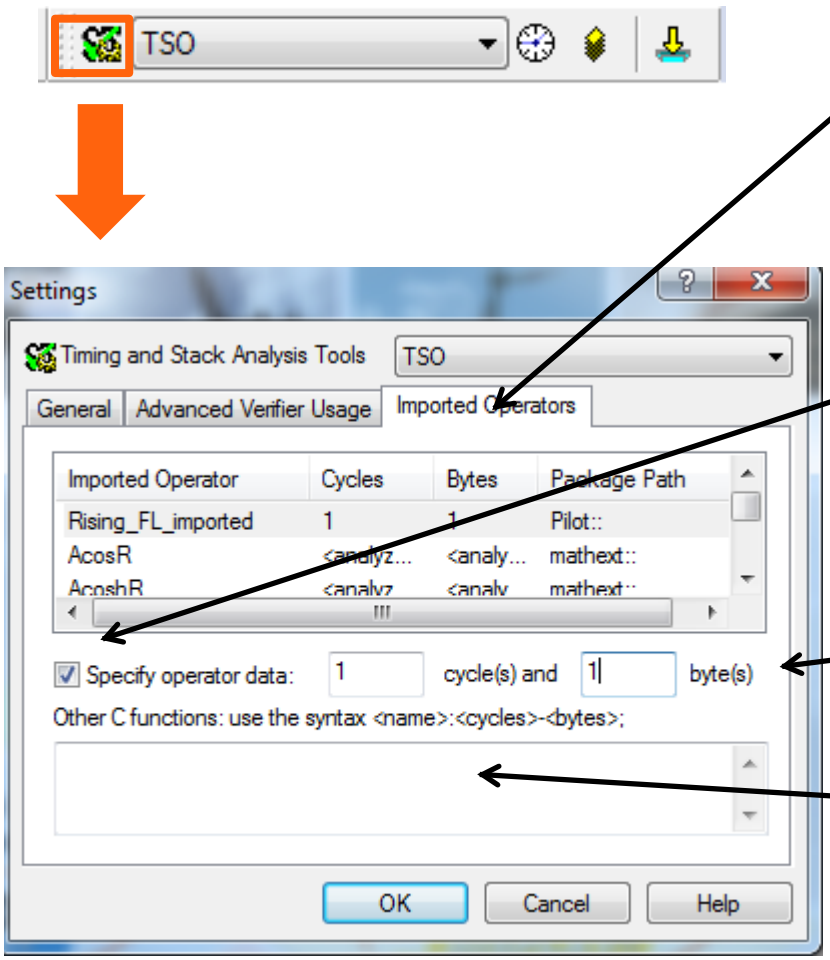
- **TSO settings:**



- These settings can be modified after having built the code: they only affect the Analyzers
- Change the name of the Analyzer session from the **Session Name** field
  - Useful when the session name must be different from the active configuration name
- CPU options are frozen
- Set an user ais file pathname from **User AIS file** field if need be to customize the behavior of timing analysis with specific annotations

Using TSO: Call The Tool

- **TSO settings:**



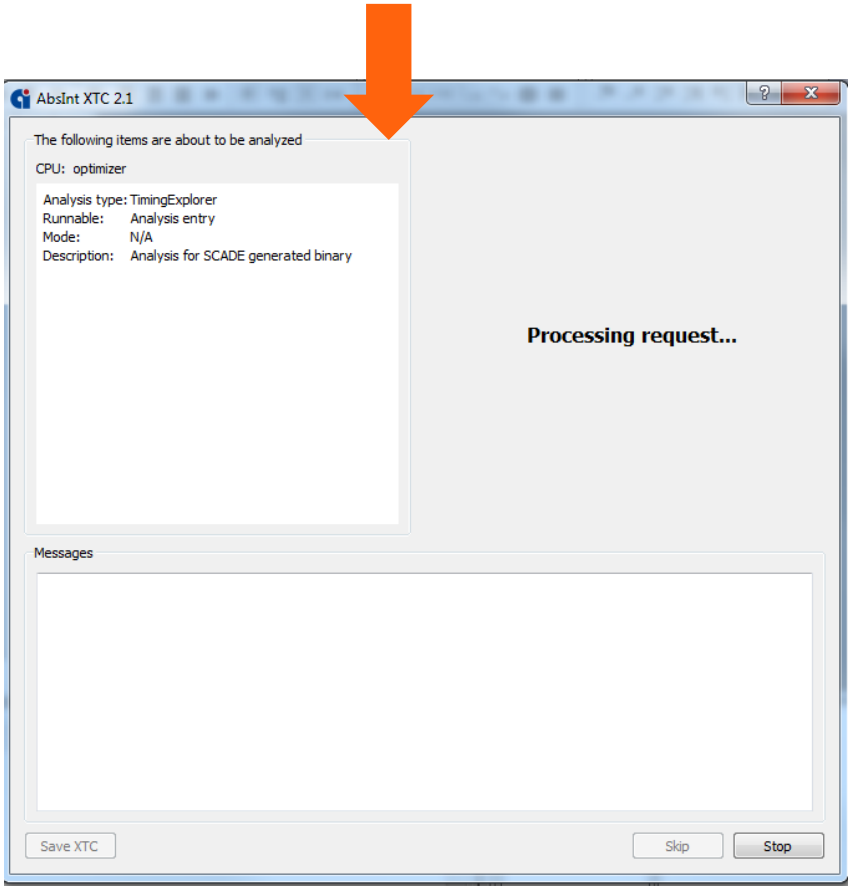
- Can specify the time cycle and the stack usage for any imported operator defined in SCADE Suite models from the Imported Operators tab

- For each operator you select in the given list of imported operators:
  1. Check Specify operator data field to enable the specification of time cycle and stack usage values or if the selected operator must not be analyzed (set 0 in data)
  2. Specify a time cycle value in **cycle(s)** and a stack usage value in **byte(s)** field
  3. Specify other C functions that are not present in the model as follows: `<functionname>:<nb cycles>-<stack size>`

Using TSO: Call The Tool

- **TSO toolbar:**

Click **"Analyze Timing"**



To launch timing analysis

Using TSO: Results Report

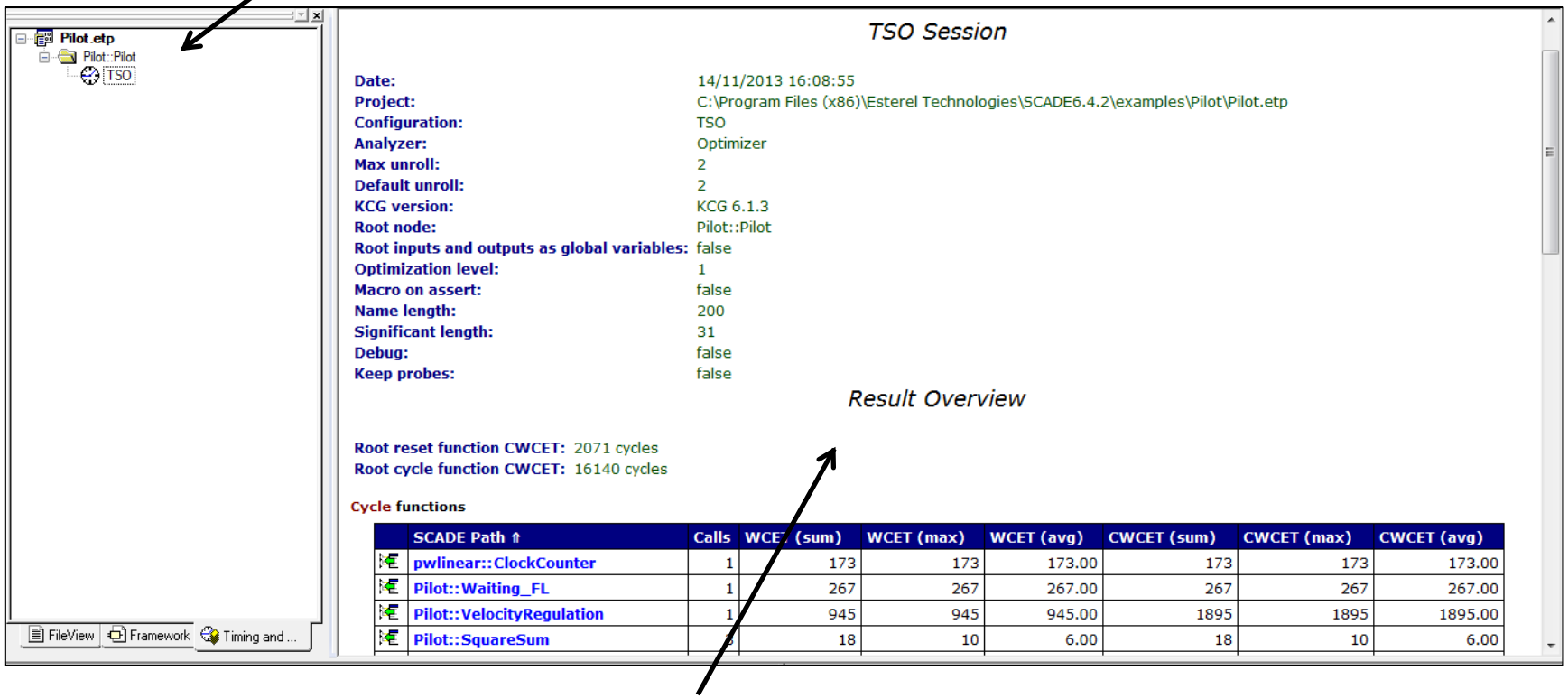
- **Abbreviations**

Abbreviations

Acronym	Definition
WCET (sum)	WCET contribution of the function without descendants, i.e. sum of the function WCETs computed for each context
WCET (max)	WCET of the function without descendants, i.e. max of the function WCETs computed for each context
WCET (avg)	WCET average of the function without descendants, i.e. average of the function WCETs computed for each context
CWCET (sum)	Cumulative WCET contribution of the function with descendants
CWCET (max)	Cumulative WCET of the function with descendants
CWCET (avg)	Cumulative WCET average of the function with descendants
Stack	Stack usage of the function
CStack	Cumulative Stack usage of the function with its parents

Using TSO: Timing Analyzer Report

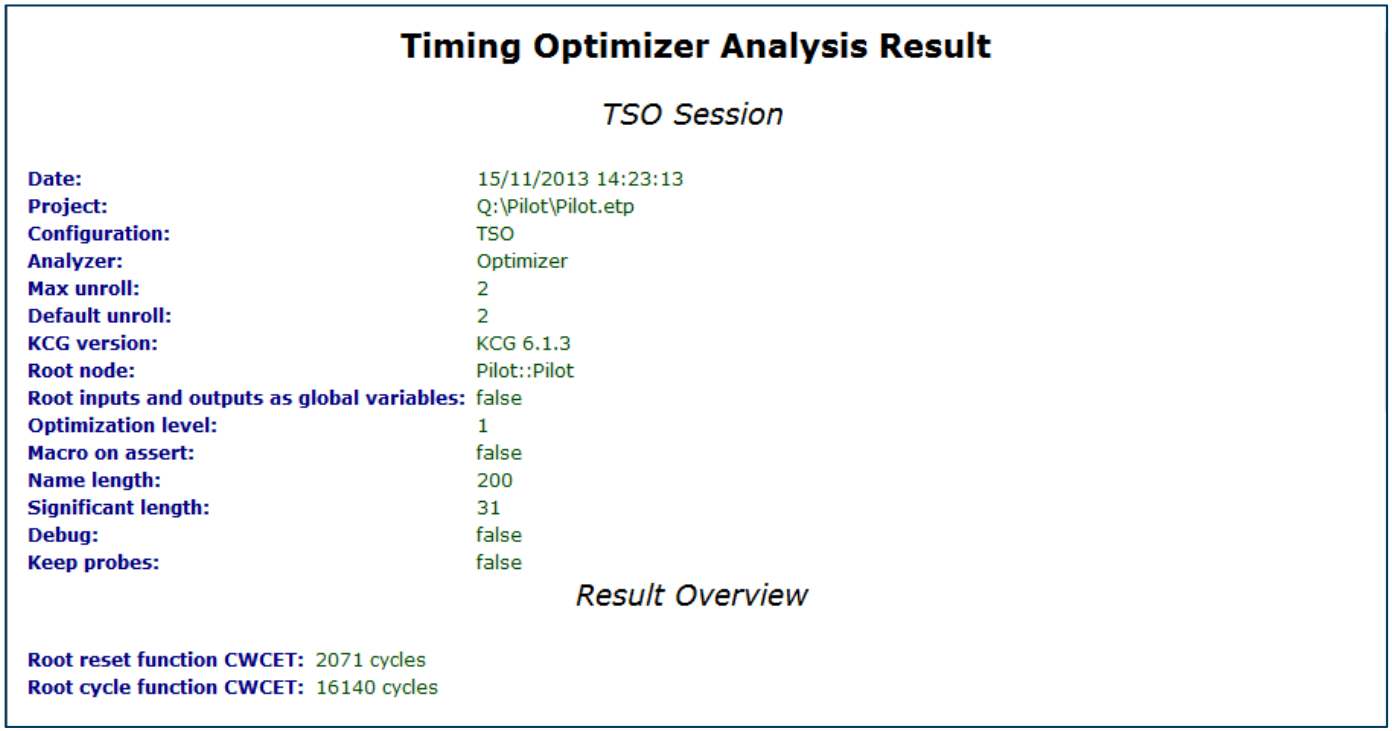
- **Once the timing analysis is completed**
  - The current session is stored in the **Timing and Stack** tab



  - The session results view is automatically displayed

Using TSO: Timing Analyzer Report

- **The following information is available for each session at the top level of the Results view:**
  - A title specifies if it is a Timing Optimizer report and a subtitle identifies a session name based on the current configuration
  - A summary of the code generation options
  - The results overview



Timing Optimizer Analysis Result

TSO Session

Date:	15/11/2013 14:23:13
Project:	Q:\Pilot\Pilot.etp
Configuration:	TSO
Analyzer:	Optimizer
Max unroll:	2
Default unroll:	2
KCG version:	KCG 6.1.3
Root node:	Pilot::Pilot
Root inputs and outputs as global variables:	false
Optimization level:	1
Macro on assert:	false
Name length:	200
Significant length:	31
Debug:	false
Keep probes:	false

Result Overview

Root reset function CWCET: 2071 cycles
Root cycle function CWCET: 16140 cycles

Using TSO: Timing Analyzer Report

- **Results table for the Cycle functions (root operator and descendants)**
  - Contains results for all non-expanded nodes

Cycle functions

SCADE Path ⇑	Calls	WCET (sum)	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
pwlinear::ClockCounter	1	173	173	173.00	173	173	173.00
Pilot::Waiting_FL	1	267	267	267.00	267	267	267.00
Pilot::VelocityRegulation	1	945	945	945.00	1895	1895	1895.00
Pilot::SquareSum	3	18	10	6.00	18	10	6.00
Pilot::Rising_FL_imported	0	0	0	-1.#J	0	0	-1.#J
Pilot::Pilot	1	1072	1072	1072.00	16140	16140	16140.00
Pilot::MvtCalculus	1	761	761	761.00	1543	1543	1543.00
Pilot::Module	1	476	476	476.00	6115	6115	6115.00
Pilot::GetMvtMode	1	256	256	256.00	256	256	256.00
Pilot::DetermineVelocity	1	1029	1029	1029.00	7613	7613	7613.00
Pilot::DeterminePosition	1	340	340	340.00	820	820	820.00
Pilot::DeterminePointPosition	3	480	173	160.00	480	173	160.00
Pilot::Derivate	3	469	172	156.33	469	172	156.33
Pilot::CorrectVelocityAxes	3	950	360	316.67	950	360	316.67
Pilot::Command	1	804	804	804.00	4242	4242	4242.00
Pilot::CheckTelemetry	1	440	440	440.00	950	950	950.00
Pilot::CheckPosition	1	1187	1187	1187.00	10570	10570	10570.00
Pilot::CheckPointPosition	3	337	113	112.33	337	113	112.33
Pilot::Approach_FL_textual	1	515	515	515.00	515	515	515.00

Using TSO: Timing Analyzer Report

- **Results table for the Reset functions**
  - Contains results for all non-expanded nodes

Reset functions

SCADE Path ⇑	Calls	WCET (sum)	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
pwlinear::ClockCounter	1	58	58	58.00	58	58	58.00
Pilot::VelocityRegulation	1	367	367	367.00	533	533	533.00
Pilot::Pilot	1	328	328	328.00	2071	2071	2071.00
Pilot::DetermineVelocity	1	315	315	315.00	475	475	475.00
Pilot::Derivate	3	160	55	53.33	160	55	53.33
Pilot::CorrectVelocityAxes	3	166	58	55.33	166	58	55.33
Pilot::Command	1	176	176	176.00	709	709	709.00
Pilot::CheckTelemetry	1	175	175	175.00	233	233	233.00
Pilot::CheckPosition	1	326	326	326.00	1034	1034	1034.00

Using TSO: Timing Analyzer Report

- **Results table for Dependencies functions**
  - Contains results for all non-Scade functions that are generated and called by the SCADE Suite-generated code
  - Dependencies are calls to external code
  - Appears only if external code is used in the model

Dependencies functions

	Name ⇑	Calls	WCET (sum)	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
	sqrt	1	1338	1338	1338.00	5621	5621	5621.00
	numtest	3	1989	670	663.00	1989	670	663.00
	ldexp	1	1096	1096	1096.00	2108	2108	2108.00
	frexp	1	514	514	514.00	1526	1526	1526.00
	__errno	2	684	342	342.00	684	342	342.00

Using TSO: Timing Analyzer Report

- **Sort the Cycle functions table by WCET to pinpoint the "worst offenders", candidates for model optimization**

Result Overview

Root reset function CWCET: 2071 cycles
Root cycle function CWCET: 16140 cycles

Click column header for sorting

Cycle functions

SCADE Path ⇑	Calls	WCET (sum)	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
pwlinear::ClockCounter	1	173	173	173.00	173	173	173.00
Pilot::Waiting_FL	1	267	267	267.00	267	267	267.00
Pilot::VelocityRegulation	1	945	945	945.00	1895	1895	1895.00
Pilot::SquareSum	3	18	10	6.00	18	10	6.00
Pilot::Rising_FL_imported	0	0	0	-1.#J	0	0	-1.#J
Pilot::Pilot	1	1072	1072	1072.00	16140	16140	16140.00
Pilot::MvtCalculus	1	761	761	761.00	1543	1543	1543.00
Pilot::Module	1	476	476	476.00	6115	6115	6115.00
Pilot::GetMvtMode	1	256	256	256.00	256	256	256.00
Pilot::DetermineVelocity	1	1029	1029	1029.00	7613	7613	7613.00
Pilot::DeterminePosition	1	340	340	340.00	820	820	820.00
Pilot::DeterminePointPosition	3	480	173	160.00	480	173	160.00
Pilot::Derivate	3	469	172	156.33	469	172	156.33
Pilot::CorrectVelocityAxes	3	950	360	316.67	950	360	316.67
Pilot::Command	1	804	804	804.00	4242	4242	4242.00
Pilot::CheckTelemetry	1	440	440	440.00	950	950	950.00
Pilot::CheckPosition	1	1187	1187	1187.00	10570	10570	10570.00
Pilot::CheckPointPosition	3	337	113	112.33	337	113	112.33
Pilot::Approach_FL_textual	1	515	515	515.00	515	515	515.00

Using TSO: Timing Analyzer Report

- "Worst offenders" at the top level

Result Overview

Root reset function CWCET: 2071 cycles
Root cycle function CWCET: 16140 cycles

Cycle functions

	SCADE Path	Calls	WCET (sum) ⇑	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
	Pilot::CheckPosition	1	1187	1187	1187.00	10570	10570	10570.00
	Pilot::Pilot	1	1072	1072	1072.00	16140	16140	16140.00
	Pilot::DetermineVelocity	1	1029	1029	1029.00	7613	7613	7613.00
	Pilot::CorrectVelocityAxes	3	950	360	316.67	950	360	316.67
	Pilot::VelocityRegulation	1	945	945	945.00	1895	1895	1895.00
	Pilot::Command	1	804	804	804.00	4242	4242	4242.00
	Pilot::MvtCalculus	1	761	761	761.00	1543	1543	1543.00
	Pilot::Approach_FL_textual	1	515	515	515.00	515	515	515.00
	Pilot::DeterminePointPosition	3	480	173	160.00	480	173	160.00
	Pilot::Module	1	476	476	476.00	6115	6115	6115.00
	Pilot::Derivate	3	469	172	156.33	469	172	156.33
	Pilot::CheckTelemetry	1	440	440	440.00	950	950	950.00
	Pilot::DeterminePosition	1	340	340	340.00	820	820	820.00
	Pilot::CheckPointPosition	3	337	113	112.33	337	113	112.33
	Pilot::Waiting_FL	1	267	267	267.00	267	267	267.00
	Pilot::GetMvtMode	1	256	256	256.00	256	256	256.00
	pwlinear::ClockCounter	1	173	173	173.00	173	173	173.00
	Pilot::SquareSum	3	18	10	6.00	18	10	6.00
	Pilot::Rising_FL_imported	0	0	0	-1.#J	0	0	-1.#J

Using TSO: Timing Analyzer Report

- "Worst offenders" at the low level

Result Overview

Root reset function CWCET: 2071 cycles
Root cycle function CWCET: 16140 cycles

Cycle functions

SCADE Path	Calls	WCET (sum) ⇓	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
Pilot::Rising_FL_imported	0	0	0	-1.#J	0	0	-1.#J
Pilot::SquareSum	3	18	10	6.00	18	10	6.00
pwlinear::ClockCounter	1	173	173	173.00	173	173	173.00
Pilot::GetMvtMode	1	256	256	256.00	256	256	256.00
Pilot::Waiting_FL	1	267	267	267.00	267	267	267.00
Pilot::CheckPointPosition	3	337	113	112.33	337	113	112.33
Pilot::DeterminePosition	1	340	340	340.00	820	820	820.00
Pilot::CheckTelemetry	1	440	440	440.00	950	950	950.00
Pilot::Derivate	3	469	172	156.33	469	172	156.33
Pilot::Module	1	476	476	476.00	6115	6115	6115.00
Pilot::DeterminePointPosition	3	480	173	160.00	480	173	160.00
Pilot::Approach_FL_textual	1	515	515	515.00	515	515	515.00
Pilot::MvtCalculus	1	761	761	761.00	1543	1543	1543.00
Pilot::Command	1	804	804	804.00	4242	4242	4242.00
Pilot::VelocityRegulation	1	945	945	945.00	1895	1895	1895.00
Pilot::CorrectVelocityAxes	3	950	360	316.67	950	360	316.67
Pilot::DetermineVelocity	1	1029	1029	1029.00	7613	7613	7613.00
Pilot::Pilot	1	1072	1072	1072.00	16140	16140	16140.00
Pilot::CheckPosition	1	1187	1187	1187.00	10570	10570	10570.00

Using TSO: Timing Analyzer Report

- Sort the Cycle functions table by Calls to see the “the best candidates” for in-lining

Cycle functions

	SCADE Path	Calls ⇑	WCET (sum)	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
	Pilot::CorrectVelocityAxes	3	950	360	316.67	950	360	316.67
	Pilot::SquareSum	3	18	10	6.00	18	10	6.00
	Pilot::Derivate	3	469	172	156.33	469	172	156.33
	Pilot::CheckPointPosition	3	337	113	112.33	337	113	112.33
	Pilot::DeterminePointPosition	3	480	173	160.00	480	173	160.00
	pwlinear::ClockCounter	1	173	173	173.00	173	173	173.00
	Pilot::Approach_FL_textual	1	515	515	515.00	515	515	515.00
	Pilot::Module	1	476	476	476.00	6115	6115	6115.00
	Pilot::DeterminePosition	1	340	340	340.00	820	820	820.00
	Pilot::VelocityRegulation	1	945	945	945.00	1895	1895	1895.00
	Pilot::DetermineVelocity	1	1029	1029	1029.00	7613	7613	7613.00
	Pilot::CheckTelemetry	1	440	440	440.00	950	950	950.00
	Pilot::Pilot	1	1072	1072	1072.00	16140	16140	16140.00
	Pilot::GetMvtMode	1	256	256	256.00	256	256	256.00
	Pilot::MvtCalculus	1	761	761	761.00	1543	1543	1543.00
	Pilot::Command	1	804	804	804.00	4242	4242	4242.00
	Pilot::CheckPosition	1	1187	1187	1187.00	10570	10570	10570.00
	Pilot::Waiting_FL	1	267	267	267.00	267	267	267.00
	Pilot::Rising_FL_imported	0	0	0	-1.#J	0	0	-1.#J

Using TSO: Timing Analyzer Report

- To display detailed results for a Cycle function

Result Overview

Root reset function CWCET: 2071 cycles
Root cycle function CWCET: 16140 cycles

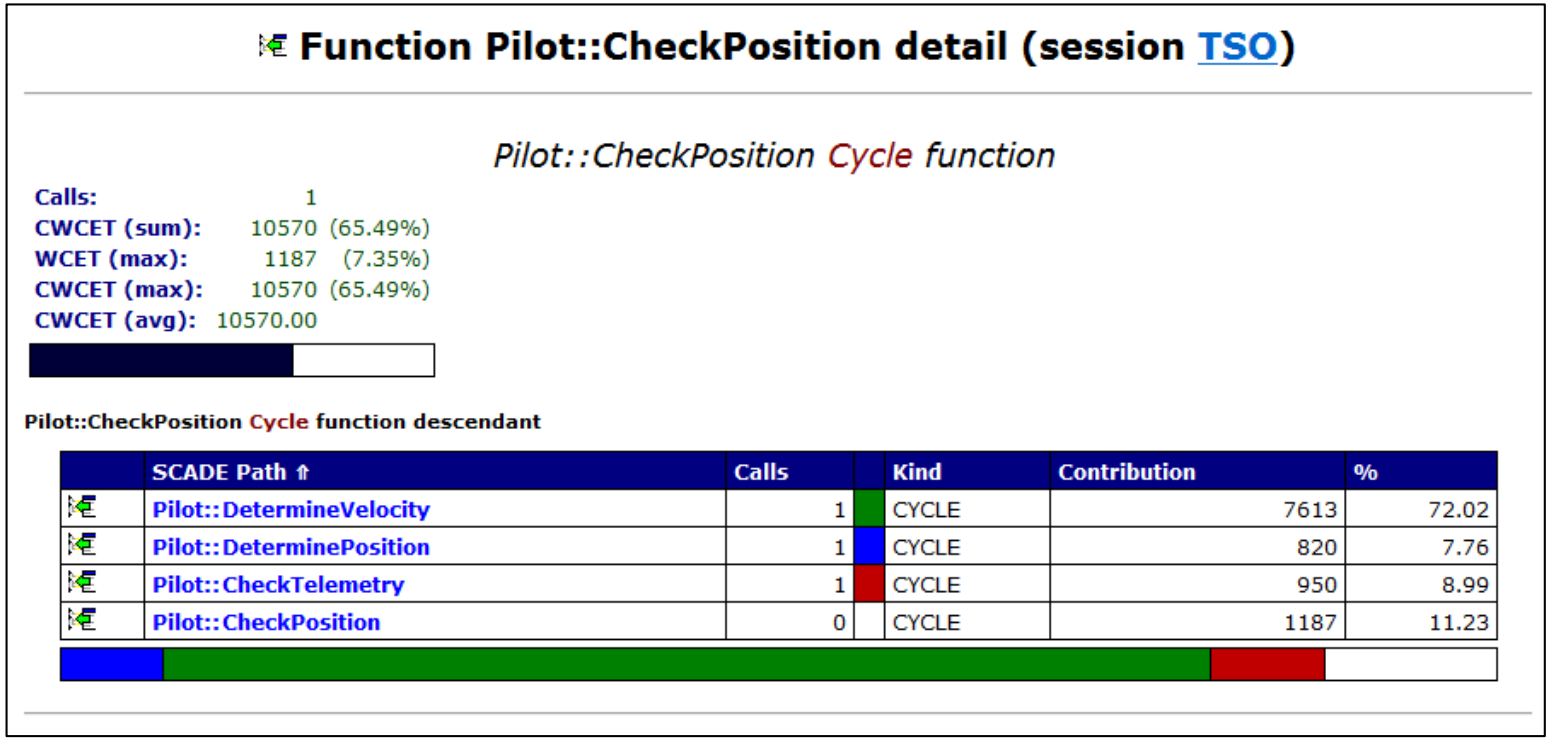
Cycle functions

SCADE Path	Calls	WCET (sum) ⇓	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
Pilot::Rising_FL_imported	0	0	0	-1.#J	0	0	-1.#J
Pilot::SquareSum	3	18	10	6.00	18	10	6.00
pwlinear::ClockCounter	1	173	173	173.00	173	173	173.00
Pilot::GetMvtMode	1	256	256	256.00	256	256	256.00
Pilot::Waiting_FL	1	267	267	267.00	267	267	267.00
Pilot::CheckPointPosition	3	337	113	112.33	337	113	112.33
Pilot::DeterminePosition	1	340	340	340.00	820	820	820.00
Pilot::CheckTelemetry	1	440	440	440.00	950	950	950.00
Pilot::Derivate	3	469	172	156.33	469	172	156.33
Pilot::Module	1	476	476	476.00	6115	6115	6115.00
Pilot::DeterminePointPosition	3	480	173	160.00	480	173	160.00
Pilot::Approach_FL_textual	1	515	515	515.00	515	515	515.00
Pilot::MvtCalculus	1	761	761	761.00	1543	1543	1543.00
Pilot::Command	1	804	804	804.00	4242	4242	4242.00
Pilot::VelocityRegulation	1	945	945	945.00	1895	1895	1895.00
Pilot::CorrectVelocityAxes	3	950	360	316.67	950	360	316.67
Pilot::DetermineVelocity	1	1029	1029	1029.00	7613	7613	7613.00
Pilot::Pilot	1	1072	1072	1072.00	16140	16140	16140.00
Pilot::CheckPosition	1	1187	1187	1187.00	10570	10570	10570.00

Click on a function name

Using TSO: Timing Analyzer Report

- **Detailed results**
  - A summary of the Timing Analyser results of the session
  - A horizontal stacked chart table shows the contribution of each non-expanded descendant (using a different color for each node)
  - Access from the SCADE Path to descendants detailed results



Function Pilot::CheckPosition detail (session TSO)

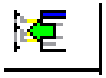
Pilot::CheckPosition Cycle function

Calls:	1
CWCET (sum):	10570 (65.49%)
WCET (max):	1187 (7.35%)
CWCET (max):	10570 (65.49%)
CWCET (avg):	10570.00

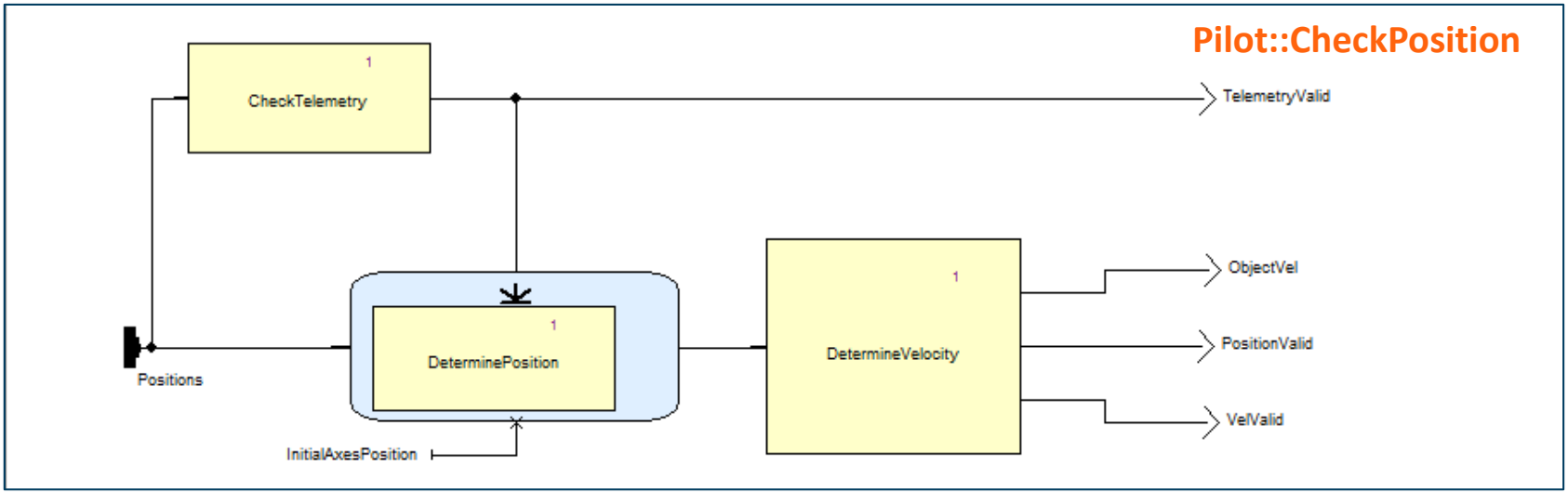
Pilot::CheckPosition Cycle function descendant

	SCADE Path	Calls		Kind	Contribution	%
	Pilot::DetermineVelocity	1		CYCLE	7613	72.02
	Pilot::DeterminePosition	1		CYCLE	820	7.76
	Pilot::CheckTelemetry	1		CYCLE	950	8.99
	Pilot::CheckPosition	0		CYCLE	1187	11.23

Using TSO: Timing Analyzer Report

- **At any row of the table, click on  , to locate the corresponding function in the SCADE Suite model**
  - **Network view** of the selected node is opened

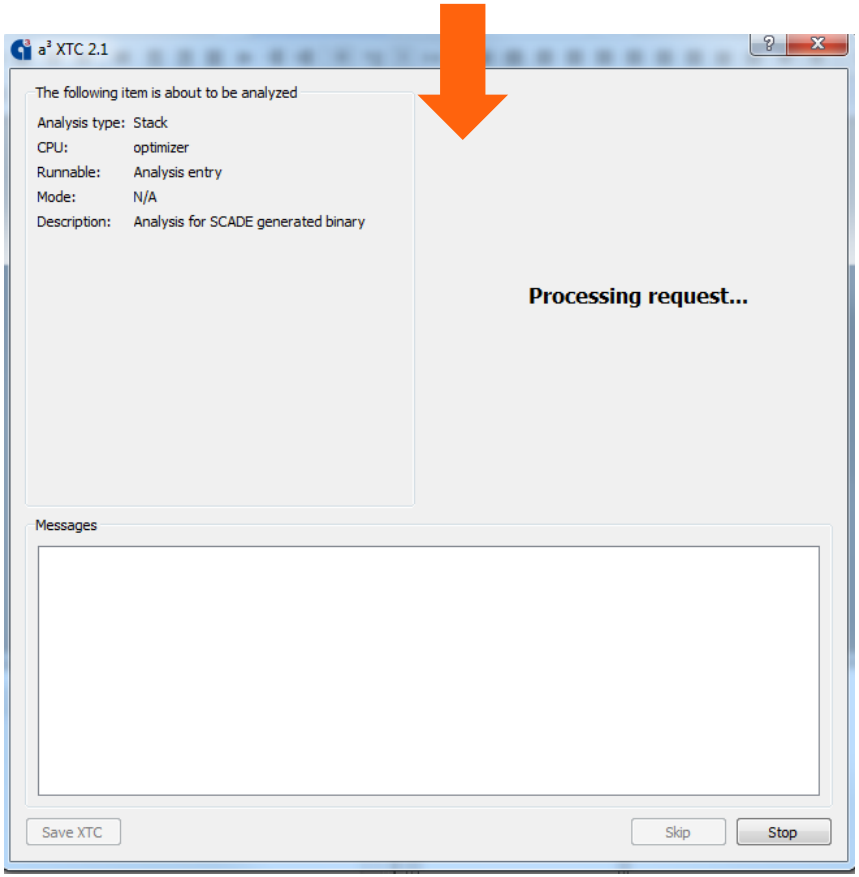
	SCADE Path	Calls	WCET (sum) ⇑	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
	Pilot::CheckPosition	1	1187	1187	1187.00	10570	10570	10570.00
	Pilot::Pilot	1	1072	1072	1072.00	16140	16140	16140.00
	Pilot::DetermineVelocity	1	1029	1029	1029.00	7613	7613	7613.00



Using TSO: Call The Tool

- **TSO toolbar:**

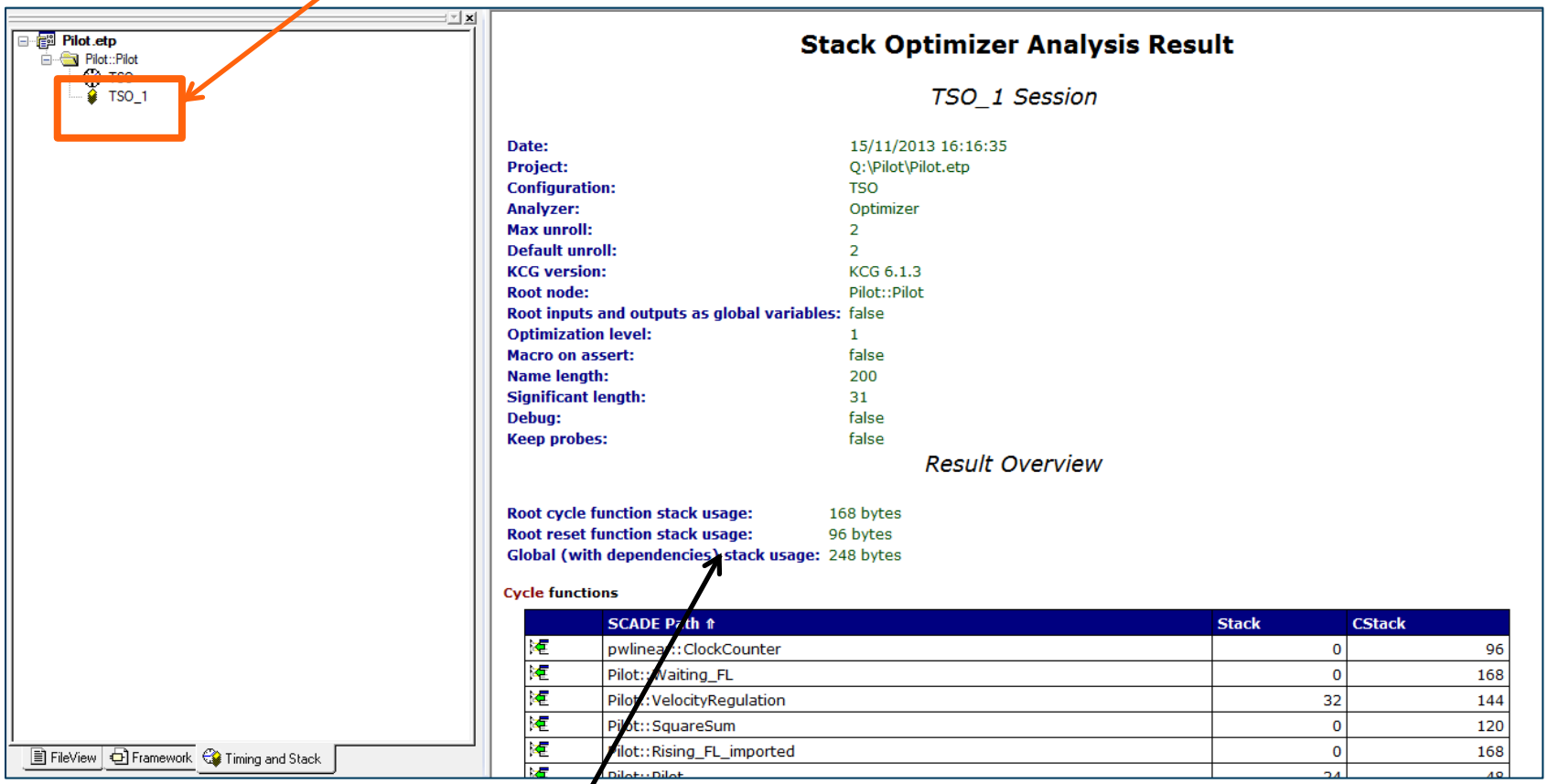
Click "**Analyze Stack**"



To launch stack use analysis

Using TSO: Stack Analyzer Report

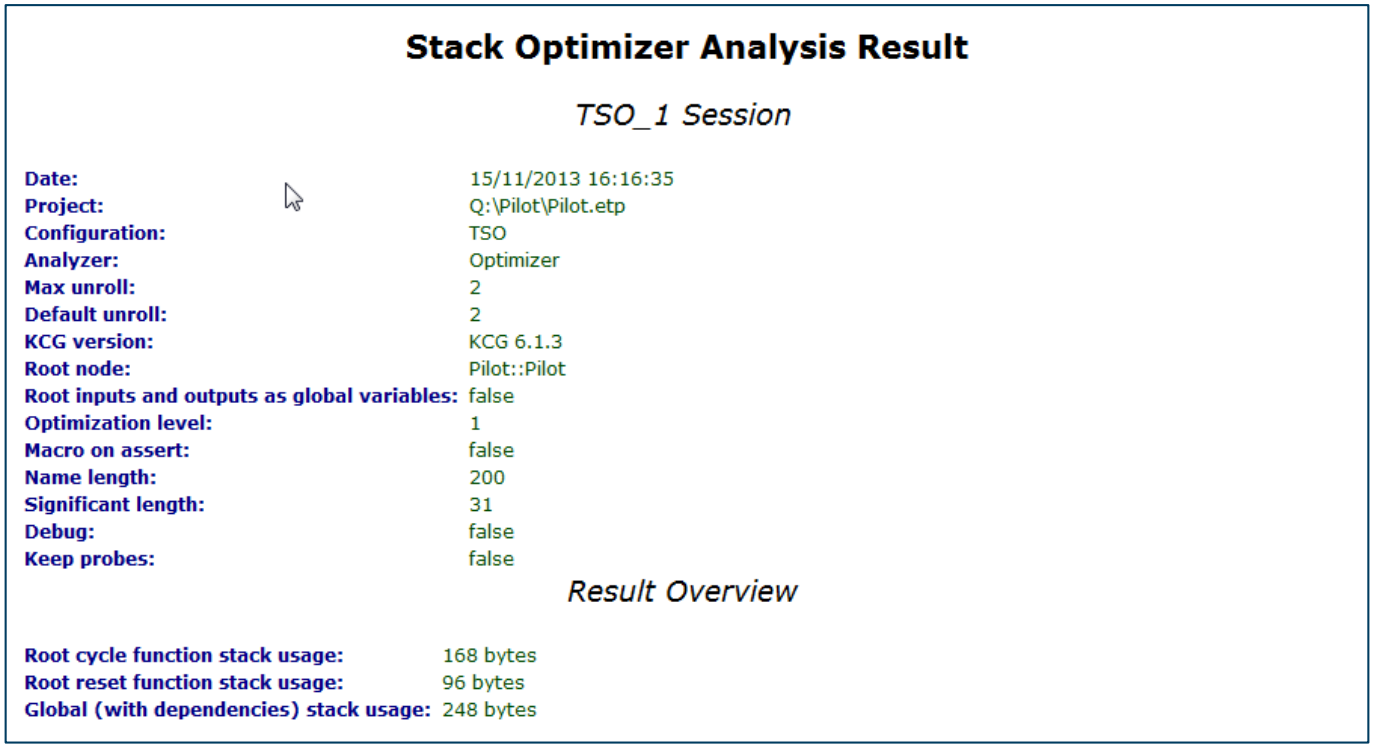
- **Once the stack analysis is completed**
  - The current session is stored in the **Timing and Stack** tab



  - The session results view is automatically displayed

Using TSO: Stack Analyzer Report

- **The following information is available for each session at the top level of the Results view:**
  - A title specifies if it is a Stack Optimizer report and a subtitle identifies a session name based on the current configuration
  - A summary of the code generation options
  - The Result Overview



Stack Optimizer Analysis Result

TSO_1 Session

Date:	15/11/2013 16:16:35
Project:	Q:\Pilot\Pilot.etp
Configuration:	TSO
Analyzer:	Optimizer
Max unroll:	2
Default unroll:	2
KCG version:	KCG 6.1.3
Root node:	Pilot::Pilot
Root inputs and outputs as global variables:	false
Optimization level:	1
Macro on assert:	false
Name length:	200
Significant length:	31
Debug:	false
Keep probes:	false

Result Overview

Root cycle function stack usage:	168 bytes
Root reset function stack usage:	96 bytes
Global (with dependencies) stack usage:	248 bytes

Using TSO: Stack Analyzer Report

- **Results table for the Cycle functions**
  - Contains results for all non-expanded nodes

Cycle functions

	SCADE Path ⇑	Stack	CStack
	pwlinear::ClockCounter	0	96
	Pilot::Waiting_FL	0	168
	Pilot::VelocityRegulation	32	144
	Pilot::SquareSum	0	120
	Pilot::Rising_FL_imported	0	168
	Pilot::Pilot	24	48
	Pilot::MvtCalculus	56	168
	Pilot::Module	24	120
	Pilot::GetMvtMode	0	48
	Pilot::DetermineVelocity	24	96
	Pilot::DeterminePosition	24	96
	Pilot::DeterminePointPosition	0	96
	Pilot::Derivate	0	96
	Pilot::CorrectVelocityAxes	0	144
	Pilot::Command	64	112
	Pilot::CheckTelemetry	24	96
	Pilot::CheckPosition	24	72
	Pilot::CheckPointPosition	0	96
	Pilot::Approach_FL_textual	0	168

Using TSO: Stack Analyzer Report

- **Results table for the Reset functions**
  - Contains results for all non-expanded nodes

Reset functions

	SCADE Path ⇑	Stack	CStack
	pwlinear::ClockCounter	0	80
	Pilot::VelocityRegulation	24	80
	Pilot::Pilot	24	48
	Pilot::DetermineVelocity	24	96
	Pilot::Derivate	0	96
	Pilot::CorrectVelocityAxes	0	80
	Pilot::Command	8	56
	Pilot::CheckTelemetry	8	80
	Pilot::CheckPosition	24	72

Using TSO: Stack Analyzer Report

- **Results table for Dependencies functions**
  - Contains results for all non-Scade functions that are generated and called by the SCADE Suite-generated code
  - Dependencies are calls to external code
  - Appears only if external code is used in the model

Dependencies functions

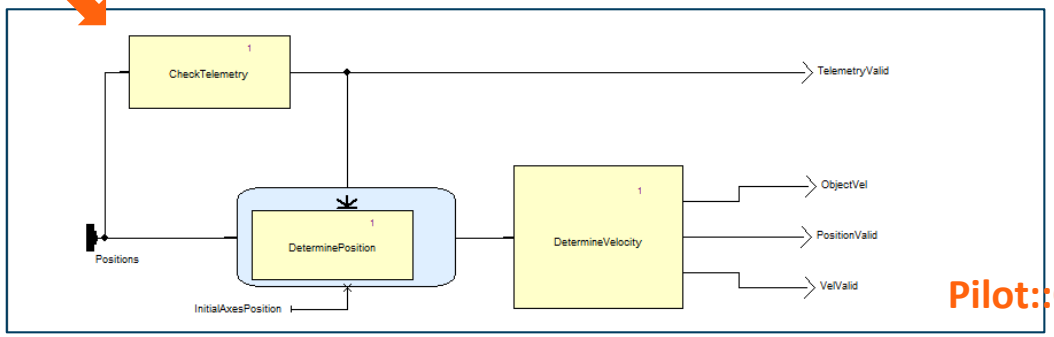
	Name ⇑	Stack	CStack
	sqrt	48	168
	numtest	32	248
	ldexp	48	216
	ispos	16	184
	frexp	32	200
	__errno	16	232

Using TSO: Stack Analyzer Report

- **Other features:**
  - **Sort by column** any “Stack Use” results table (by Stack or CStack fields)
  - **Locate each function** in the SCADE model

Reset functions

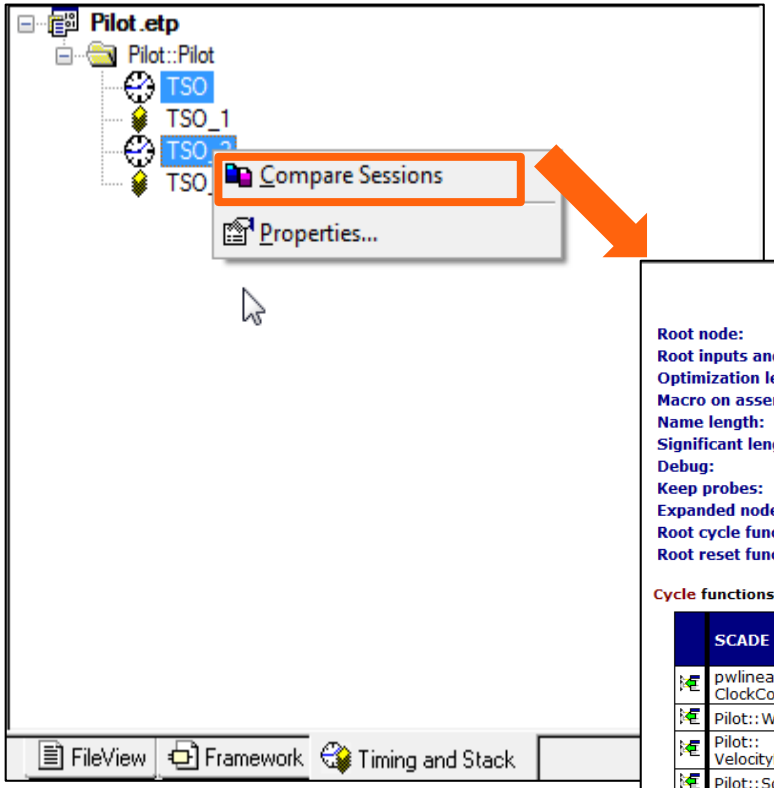
	SCADE Path	Stack ⇑	CStack
	Pilot::VelocityRegulation	24	80
	Pilot::DetermineVelocity	24	96
	Pilot::Pilot	24	48
	Pilot::CheckPosition	24	72
	Pilot::CheckTelemetry	8	80
	Pilot::Command	8	56
	pw_linear::ClockCounter	0	80
	Pilot::CorrectVelocityAxes	0	80
	Pilot::Delegate	0	96



Pilot::CheckPosition

Using TSO: Compare Timing Sessions

- Select two Timing sessions in the "Timing and Stack" tab
- Right-click *Compare sessions*



Sessions Comparison Between TSO(ref) and TSO_2(vs)

Root node:	Pilot::Pilot	/ Pilot::Pilot
Root inputs and outputs as global variables:	false	/ false
Optimization level:	1	/ 1
Macro on assert:	false	/ false
Name length:	200	/ 200
Significant length:	31	/ 31
Debug:	false	/ false
Keep probes:	false	/ false
Expanded nodes:	_	/ Pilot::CheckPosition
Root cycle function CWCET:	16140 cycles	/ 15390 cycles
Root reset function CWCET:	2071 cycles	/ 1747 cycles

Cycle functions

SCADE Path	Calls				WCET (sum)				WCET (max)				WCET (avg)			
	ref	vs	Diff	%	ref	vs	Diff	%	ref	vs	Diff	%	ref	vs	Diff	
pwlinear::ClockCounter	1	1	0	0.00	173	175	2	1.14	173	175	2	1.14	173.00	175.00	2.00	1
Pilot::Waiting_FL	1	1	0	0.00	267	293	26	8.87	267	293	26	8.87	267.00	293.00	26.00	8
Pilot::VelocityRegulation	1	1	0	0.00	945	862	-83	-9.63	945	862	-83	-9.63	945.00	862.00	-83.00	-9.
Pilot::SquareSum	3	3	0	0.00	18	159	141	88.68	10	81	71	87.65	6.00	53.00	47.00	88
Pilot::Rising_FL_imported	0	0	0	0.00	0	0	0	0.00	0	0	0	0.00	-1.#J	-1.#J	1.#R	0
Pilot::Pilot	1	1	0	0.00	1072	1347	275	20.42	1072	1347	275	20.42	1072.00	1347.00	275.00	20

Using TSO: Compare Timing Sessions

Sessions Comparison Between TSO(ref) and TSO_2(vs)

Root node:	Pilot::Pilot	/ Pilot::Pilot
Root inputs and outputs as global variables:	false	/ false
Optimization level:	1	/ 1
Macro on assert:	false	/ false
Name length:	200	/ 200
Significant length:	31	/ 31
Debug:	false	/ false
Keep probes:	false	/ false
Expanded nodes:	_	/ Pilot::CheckPosition
Root cycle function CWCET:	16140 cycles	/ 15390 cycles
Root reset function CWCET:	2071 cycles	/ 1747 cycles

Cycle functions

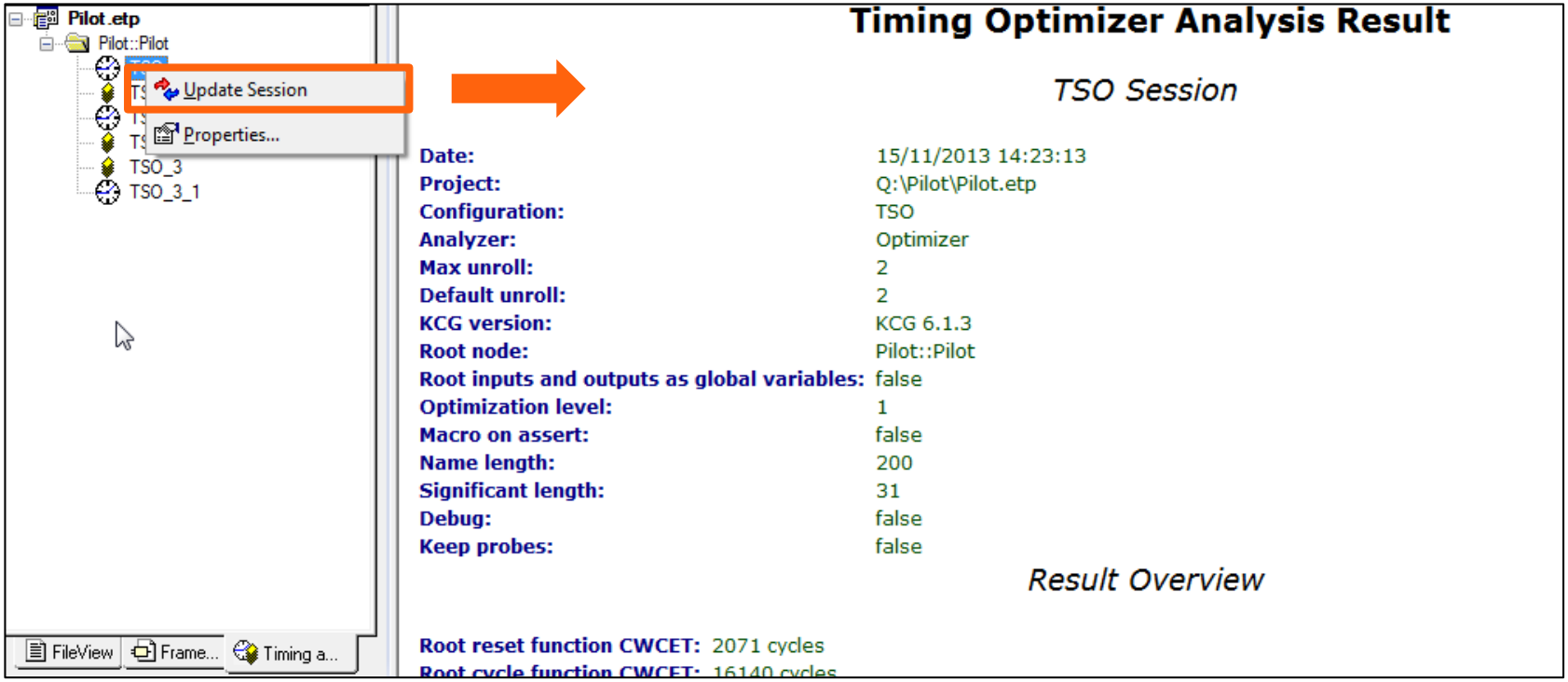
SCADE Path	Calls				WCET (sum)				WCET (max)				WCET (avg)			
	ref	vs	Diff	%	ref	vs	Diff	%	ref	vs	Diff	%	ref	vs	Diff	%
pwlinear::ClockCounter	1	1	0	0.00	173	175	2	1.14	173	175	2	1.14	173.00	175.00	2.00	1
Pilot::Waiting_FL	1	1	0	0.00	267	293	26	8.87	267	293	26	8.87	267.00	293.00	26.00	8
Pilot::VelocityRegulation	1	1	0	0.00	945	862	-83	-9.63	945	862	-83	-9.63	945.00	862.00	-83.00	-9.
Pilot::SquareSum	3	3	0	0.00	18	159	141	88.68	10	81	71	87.65	6.00	53.00	47.00	88
Pilot::Rising_FL_imported	0	0	0	0.00	0	0	0	0.00	0	0	0	0.00	-1.#J	-1.#J	1.#R	0
Pilot::Pilot	1	1	0	0.00	1072	1347	275	20.42	1072	1347	275	20.42	1072.00	1347.00	275.00	20

- **Sessions comparison**
  - Results view
    - Summary of each session configuration
    - Results tables
  - A Diff Report allowing to compare results between
    - The first session (**ref** column) and
    - The second one (**vs** column)

  - A Diff column computes the difference between sessions
    - To easily identify differences, values are represented in two different colors:
      - Values that represent a benefit of time are colored in **green**
      - Values that represent a loss of time are colored in **red**

Using TSO: Update a Timing Session

- **Select a Timing session in the "Timing and Stack" tab**
- **Right-click *Update session***
  - Overwrites the timing results recorded from a previous session using the same configuration



LABS 1

EXERCISE 1

Exercise 1: Goal

- **Learn how to**
  - create a Timing Analysis session in this SCADE model, where an imported and function operators are implemented
  - create a Stack Use Analysis session in this SCADE model, where an imported and function operators are implemented

Prerequisites

- **To avoid problems due to Windows length path limitation**
  - To work on this exercise, use a virtual drive named “Z”
    - Launch a Dos cmd
    - Specifies the virtual drive “Z:” to assign the path of the user Labs folder to obtain the following virtual path “Z:\Prerequisites\Exercise 1”
    - subst Z: <Prerequisites Exercise 1 path>

Tip: Run the batch environment named CmdVirtual_TSO_1.bat located at the root of the TSO Exercises folder

Exercise 1

- **Step 1.1:**
  - Open "Z:\Prerequisites\Exercise 1\impOp.etp"

- **Step 1.2:**
  - Generate "TSO" Timing Optimizer Analysis session
    - Build the code related to "TSO" configuration
    - Launch Timing Analyzer Optimizer for this configuration
  - Show the "TSO" session results
    - Locate the imported and function operators

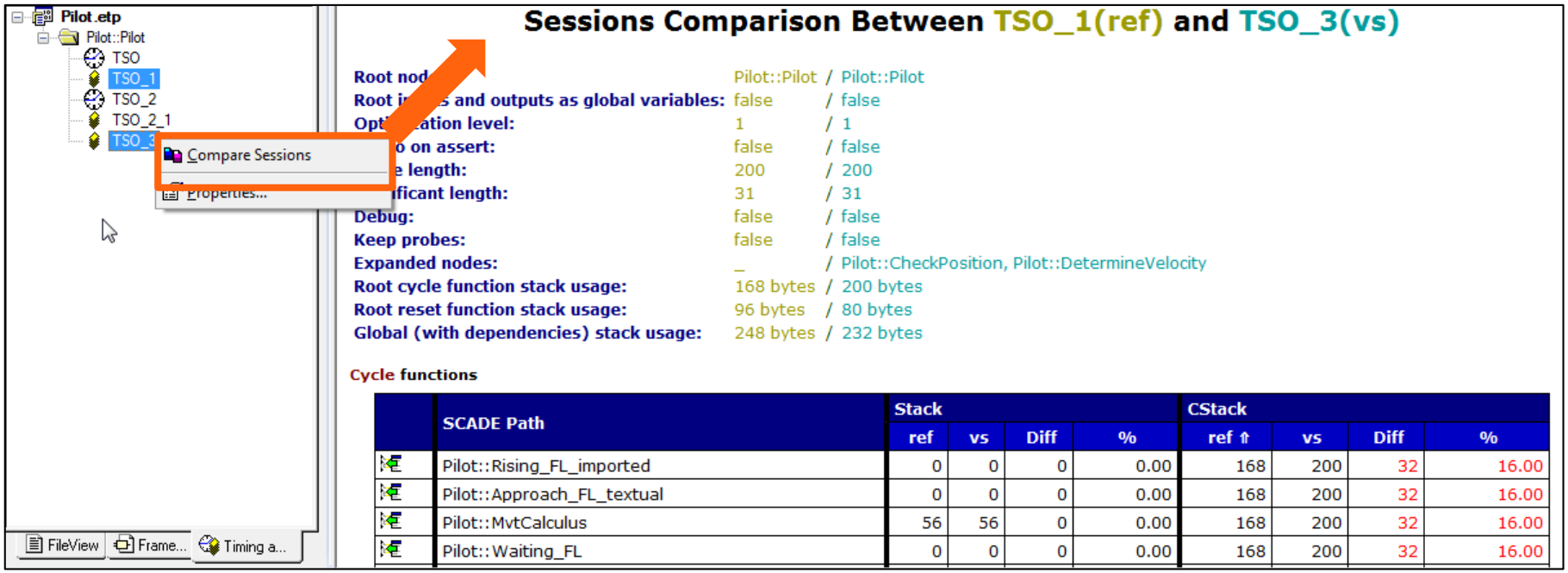
- **Step 1.3:**
  - Generate "TSO_1" Stack Optimizer Analysis session
    - Launch Stack Analyzer Optimizer for the "TSO" configuration
  - Show the "TSO_1" session results
    - Locate the imported and function operators

Exercise 1

- **Step 1.4:**
  - Customize the Timing and Stack Optimizer to avoid to measure `myFunc` in each analysis
    - Create a new configuration named “TSO_AIS” based on TSO one
    - Set “Prerequisites\Exercise 1\myAIS.ais” in the ***User ais file*** field
    - Generate “TSO_AIS” Timing Optimizer Analysis session
      - Build the code related to “TSO_AIS” configuration
      - Launch Timing Analyzer Optimizer for this configuration
      - Show the “TSO_AIS” session results
        - Check that `myFunc` takes 0 cycles
    - Generate “TSO_AIS” Stack Optimizer Analysis session
      - Launch Stack Analyzer Optimizer for the “TSO_AIS” configuration
      - Show the “TSO_AIS” session results
        - Check that `myFunc` takes exactly 16 bytes of stack

Using TSO: Compare Stack Sessions

- Select two Stack sessions in the "Timing and Stack" tab
- Right-click *Compare sessions*



SCADE Path	Stack				CStack			
	ref	vs	Diff	%	ref ⇑	vs	Diff	%
Pilot::Rising_FL_imported	0	0	0	0.00	168	200	32	16.00
Pilot::Approach_FL_textual	0	0	0	0.00	168	200	32	16.00
Pilot::MvtCalculus	56	56	0	0.00	168	200	32	16.00
Pilot::Waiting_FL	0	0	0	0.00	168	200	32	16.00

Using TSO: Compare Stack Sessions

Sessions Comparison Between TSO_1(ref) and TSO_3(vs)

Root node:	Pilot::Pilot	/ Pilot::Pilot
Root inputs and outputs as global variables:	false	/ false
Optimization level:	1	/ 1
Macro on assert:	false	/ false
Name length:	200	/ 200
Significant length:	31	/ 31
Debug:	false	/ false
Keep probes:	false	/ false
Expanded nodes:	_	/ Pilot::CheckPosition, Pilot::DetermineVelocity
Root cycle function stack usage:	168 bytes	/ 200 bytes
Root reset function stack usage:	96 bytes	/ 80 bytes
Global (with dependencies) stack usage:	248 bytes	/ 232 bytes

Cycle functions

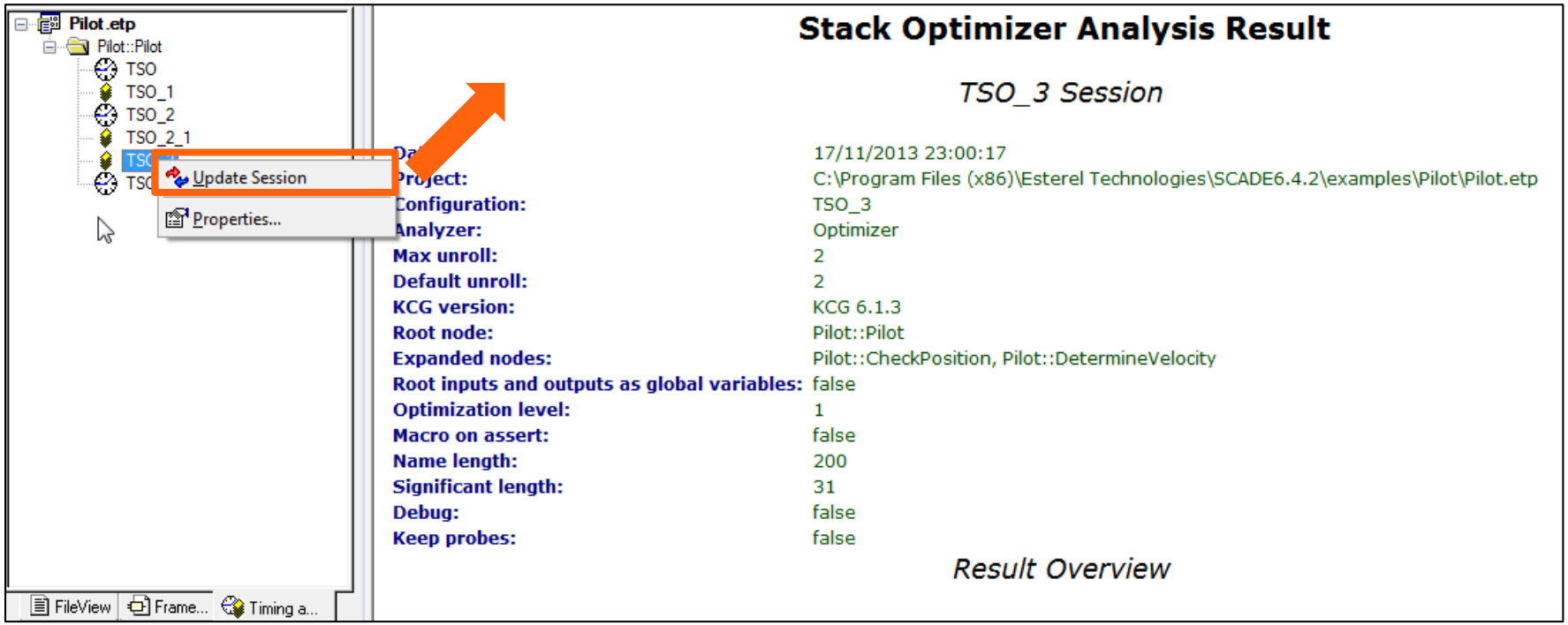
SCADE Path	Stack				CStack			
	ref	vs	Diff	%	ref ⇑	vs	Diff	%
Pilot::Rising_FL_imported	0	0	0	0.00	168	200	32	16.00
Pilot::Approach_FL_textual	0	0	0	0.00	168	200	32	16.00
Pilot::MvtCalculus	56	56	0	0.00	168	200	32	16.00
Pilot::Waiting_FL	0	0	0	0.00	168	200	32	16.00
Pilot::CorrectVelocityAxes	0	0	0	0.00	144	176	32	18.18
Pilot::VelocityRegulation	32	32	0	0.00	144	176	32	18.18
Pilot::SquareSum	0	0	0	0.00	120	104	-16	-15.38
Pilot::Module	24	24	0	0.00	120	104	-16	-15.38
Pilot::Command	64	64	0	0.00	112	144	32	22.22

- **Sessions comparison**
  - Results view
    - Summary of each session configuration
    - Results tables
  - A Diff Report allowing to compare results between
    - The first session (**ref** column) and
    - The second one (**vs** column)

- A Diff column computes the difference between sessions
  - To easily identify differences, values are represented in two different colors:
    - Values that represent a benefit of stack size are colored in **green**
    - Values that represent a loss of stack size are colored in **red**

Using TSO: Update a Stack Session

- **Select a Stack session in the "Timing and Stack" tab**
- **Right-click *Update session***
  - Overwrites the stack use results recorded from a previous session using the same configuration



LABS 2

Using TSO: Model Optimization

- **Use Expansion for in-lining a function**
  - Enables cross-boundaries optimizations
    - Faster execution time expected
    - Less allocated variables
    - But bigger compilation units
      - **Be aware that full expansion especially stresses KCG and C compiler**
  - "Merges" the stack of the callee with the caller one
    - Possibly harm total stack size
  - Disables direct commandability
    - Impact on test strategy: test at higher level becomes mandatory, but not easy in all cases

Using TSO: Model Optimization

- **Use Optimization level to get better performances**
  - `-O <level>` in the KCG command line, <level> = 0 to 3
  - 0: no optimization
    - Level to use for simulation/debug
  - 1: local variables optimization and Data flow simplification
    - Remove unused, redundant, not observable and used once variables
    - Limited visibility for simulation/debug (tunable with the help of probes and "keep" pragma)
    - Level by default to use
  - 2: memories optimization and factorization
    - Included Level 1 optimizations, merge of loops and better Boolean variables assignments
    - Limited visibility for simulation/debug (tunable with the help of probes and "keep" pragma)
  - 3: structure and array copies optimization
    - Included Level 2 optimizations
    - Level 2 and 3 are used to resolve issues on runtime performance
  - A best optimization can improve code generation and compilation

Using TSO: More on Model Optimization

- **To know more on how to optimize SCADE Suite models for code performance…**
- **To do practical exercises…**
- **… Please, request the SCADE Advanced Training:**
  - **Optimize SCADE Suite Models and Code Performance**
    - How the SCADE Suite model architecture influences performances
    - How to manage SCADE Suite main functions and constructs
    - How SCADE Suite KCG code generation options impact runtime performance of the generated code
    - How to use efficient modeling patterns
- **Training courses on TSV and TSO (interactive mode and .ais format file) are arranged by AbsInt on demand. Please don't hesitate to contact support@absint.com (http://www.absint.com/services.htm)**

ANNEX

Annex: Brief Description Of AIS Format

- **TSO can be supplied with several kinds of annotations and specifications that help it**
  - To do its analysis
  - To improve the precision of the results
- **The format of these annotations and specifications is called AIS (saved in .ais files)**

Annotation/Specification	Analyser*
The clock rate of the processor	T
Naming the compiler	A
Specification of contexts	T
Declaring additional start points for timing analysis	T
Declaring that addresses are routine entries	A
End specifications telling exec2crl to stop decoding	A
Control-flow: targets of computed branches, never-returning routines, etc.	A
Addresses of memory accesses	S - T
Specification of register values and definition of user registers	s -T

*** Analyser:**

1. **S** is relevant for stack analysis
2. **s** describes annotation of minor importance for stack analysis
3. **T** is relevant for timing analysis
4. **A** is relevant for all analyses in an uniform way

Annex: Brief Description Of AIS Format

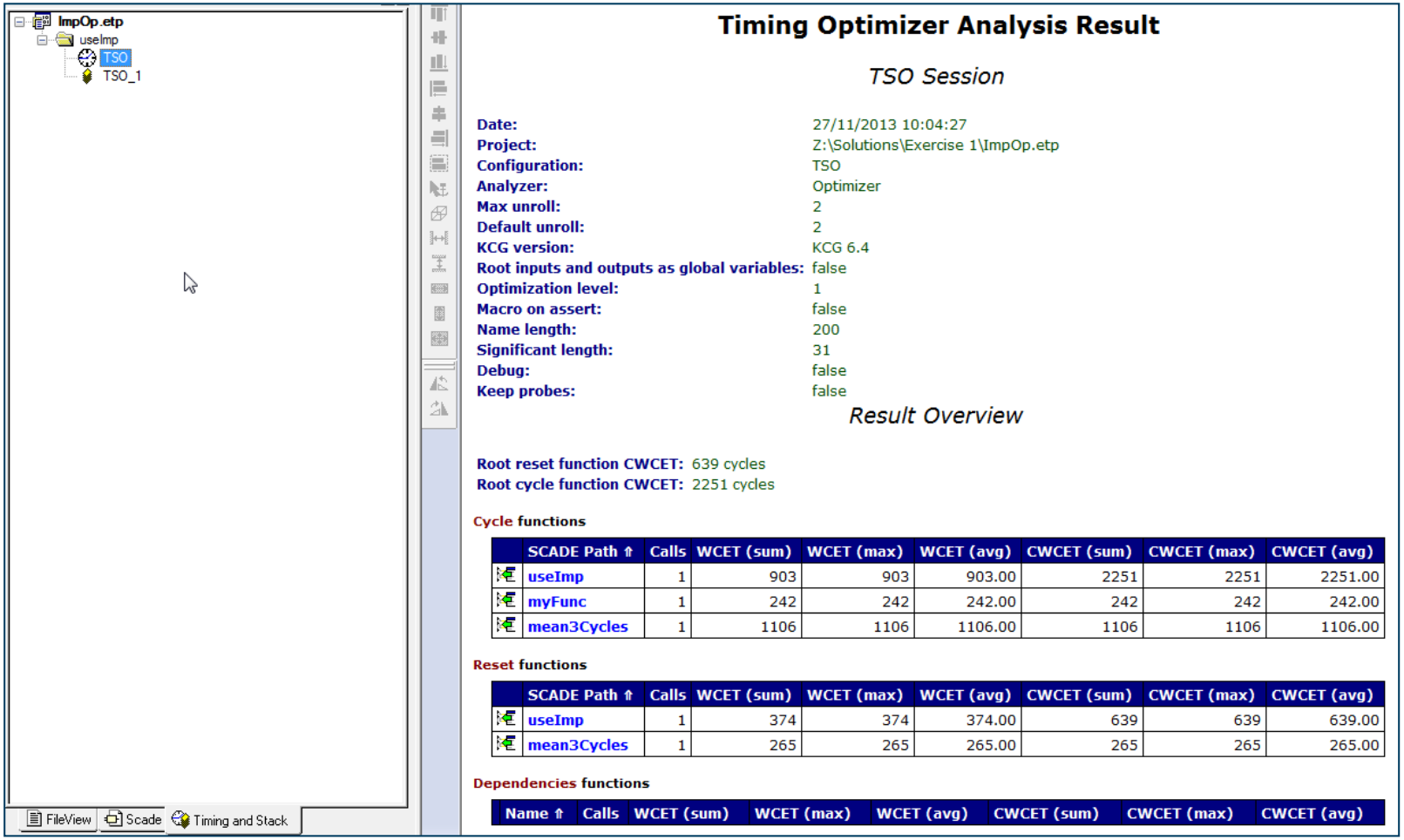
Annotation/Specification	Analyser*
Assertions, e.g., about register values	A
Describing the cache(s)	T
Properties of memory areas including timing	A
Additional execution time	T
Requesting the WCET contribution of non-routine snippets	T
Code snippets that should not be analyzed and their properties	A
Violation of the calling conventions	A
Handling of software interrupts	A
Infeasible code, i.e. code that is never executed	A
Values of conditions	A
Recursion: maximum call depth - number of recursive routines calls	S - T
Iteration counts of ordinary loops	s -T
Timing specification for loops	T
Iteration counts of irreducible loops – and more	T
Commands that cause timing analysis to produce certain output	T
Introduction of symbolic names for program points and areas	A

*** Analyser:**

1. **S** is relevant for stack analysis
2. **s** describes annotation of minor importance for stack analysis
3. **T** is relevant for timing analysis
4. **A** is relevant for all analyses in an uniform way

EXERCISE SOLUTION

Step 1.2: Timing Analysis session



Timing Optimizer Analysis Result

TSO Session

Date:	27/11/2013 10:04:27
Project:	Z:\Solutions\Exercise 1\ImpOp.etp
Configuration:	TSO
Analyzer:	Optimizer
Max unroll:	2
Default unroll:	2
KCG version:	KCG 6.4
Root inputs and outputs as global variables:	false
Optimization level:	1
Macro on assert:	false
Name length:	200
Significant length:	31
Debug:	false
Keep probes:	false

Result Overview

Root reset function CWCET: 639 cycles
Root cycle function CWCET: 2251 cycles

Cycle functions

SCADE Path ⇑	Calls	WCET (sum)	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
useImp	1	903	903	903.00	2251	2251	2251.00
myFunc	1	242	242	242.00	242	242	242.00
mean3Cycles	1	1106	1106	1106.00	1106	1106	1106.00

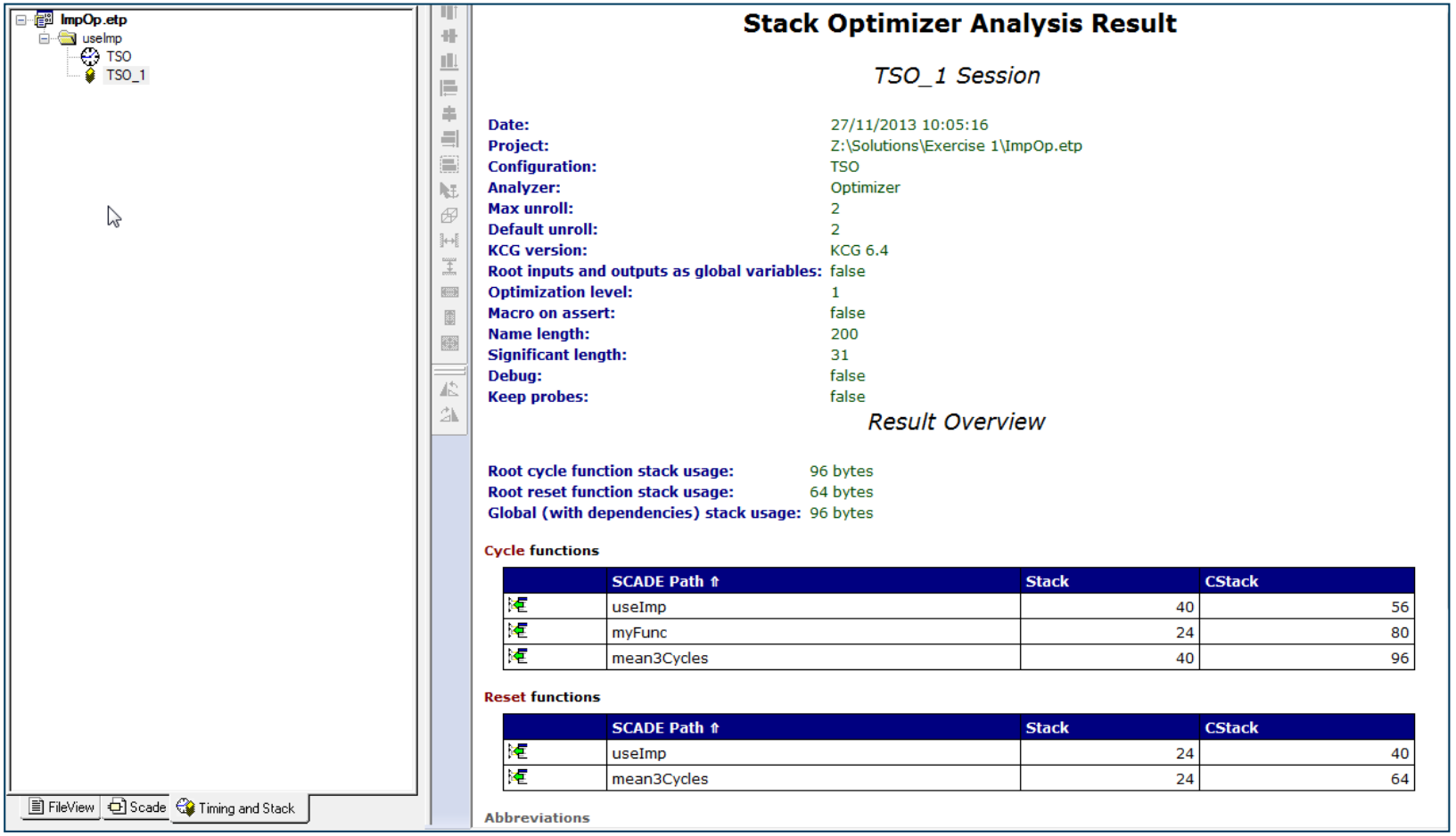
Reset functions

SCADE Path ⇑	Calls	WCET (sum)	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
useImp	1	374	374	374.00	639	639	639.00
mean3Cycles	1	265	265	265.00	265	265	265.00

Dependencies functions

Name ⇑	Calls	WCET (sum)	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)

Step 1.3: Stack Analysis session



Step 1.4: Timing Analysis session

Timing Optimizer Analysis Result

TSO_AIS Session

Date:	20/12/2013 12:24:52
Project:	Z:\Solutions\Exercise 1\ImpOp.etp
Configuration:	TSO_AIS
Analyzer:	Optimizer
User AIS file:	Z:\Solutions\Exercise 1\myAIS.ais
Max unroll:	2
Default unroll:	2
KCG version:	KCG 6.4
Root inputs and outputs as global variables:	false
Optimization level:	1
Macro on assert:	false
Name length:	200
Significant length:	31
Debug:	false
Keep probes:	false

Result Overview

Root reset function CWCET: 574 cycles
Root cycle function CWCET: 978 cycles

Cycle functions

	SCADE Path ⇑	Calls	WCET (sum)	WCET (max)	WCET (avg)	CWCET (sum)	CWCET (max)	CWCET (avg)
	useImp	1	925	925	925.00	978	978	978.00
	myFunc	0	0	0	-1.#J	0	0	-1.#J
	mean3Cycles	1	53	53	53.00	53	53	53.00

Step 1.4: Stack Analysis session

Stack Optimizer Analysis Result

TSO_AIS_1 Session

Date:	03/01/2014 14:53:20
Project:	Z:\Solutions\Exercise 1\ImpOp.etp
Configuration:	TSO_AIS
Analyzer:	Optimizer
User AIS file:	Z:\Solutions\Exercise 1\myAIS.ais
Max unroll:	2
Default unroll:	2
KCG version:	KCG 6.4
Root inputs and outputs as global variables:	false
Optimization level:	1
Macro on assert:	false
Name length:	200
Significant length:	31
Debug:	false
Keep probes:	false

Result Overview

Root cycle function stack usage:	56 bytes
Root reset function stack usage:	64 bytes
Global (with dependencies) stack usage:	64 bytes

Cycle functions

	SCADE Path ⇑	Stack	CStack
	useImp	40	56
	myFunc	0	56
	mean3Cycles	0	0