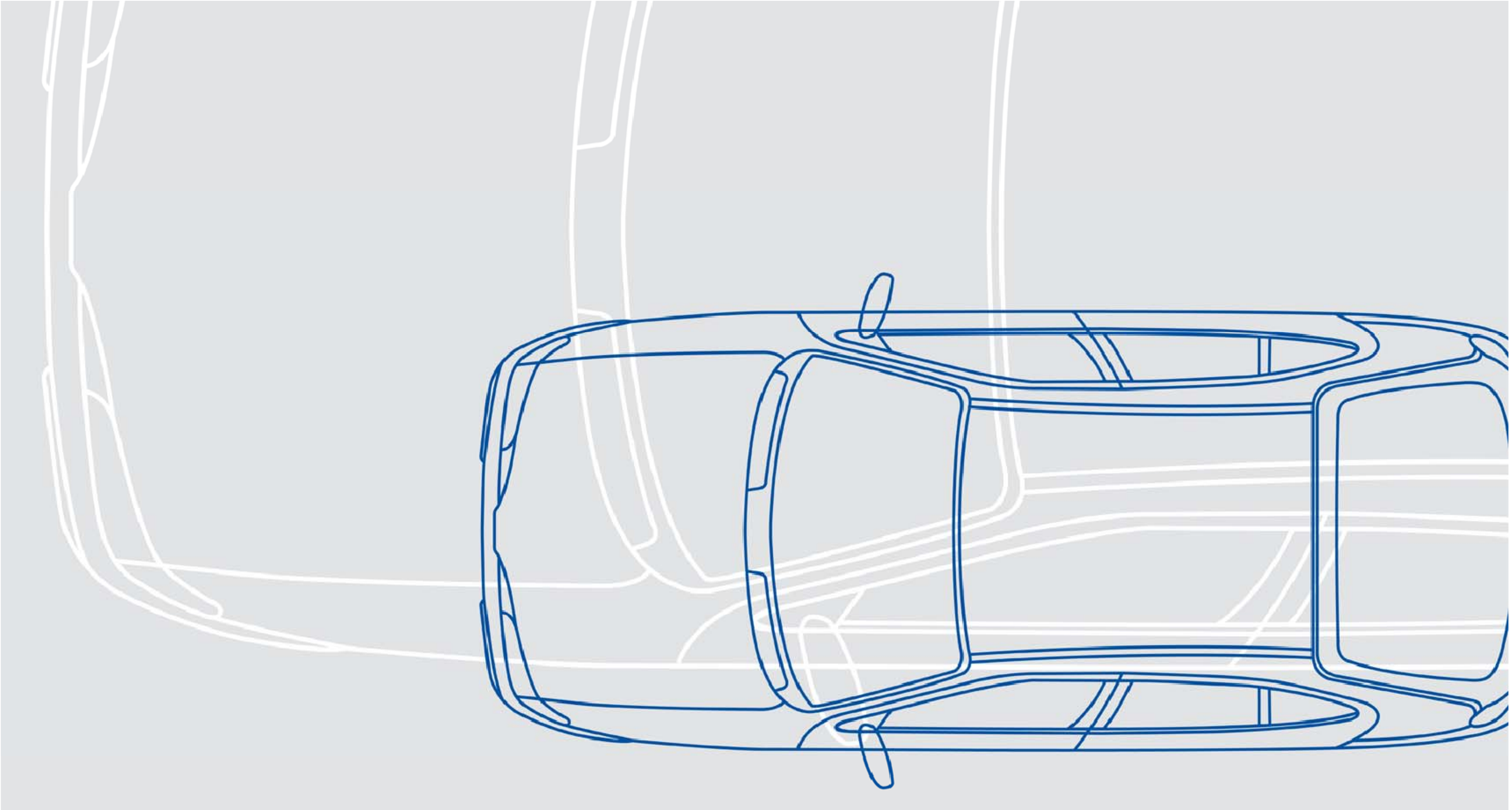
Safety Critical Systems with ASCET



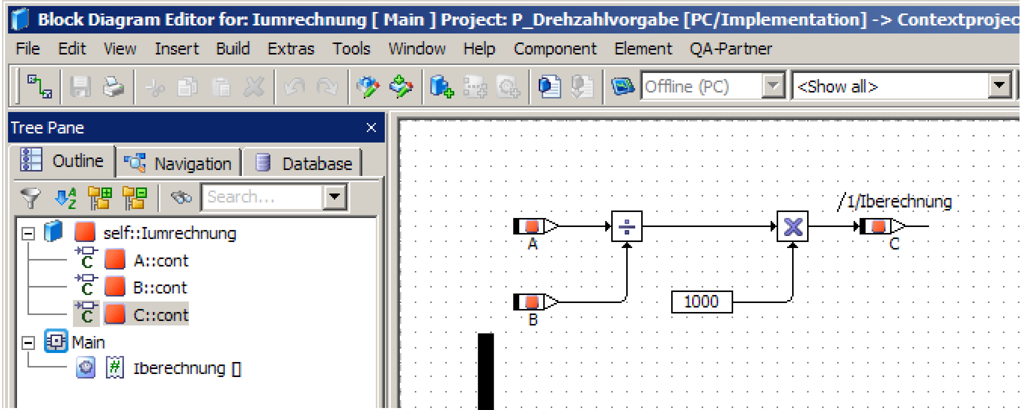
Safety Critical Systems with ASCET

Agenda

- Introduction to ASCET
  - Different types of Models
  - Fixed point arithmetic
- ASCET models vs. UML
- ASCET models vs. Ada
  - Fixed point arithmetic
  - Exceptions
- Conclusion

Safety Critical Systems with ASCET
Introduction to ASCET

Specify the model: domain specific language for the automotive industry

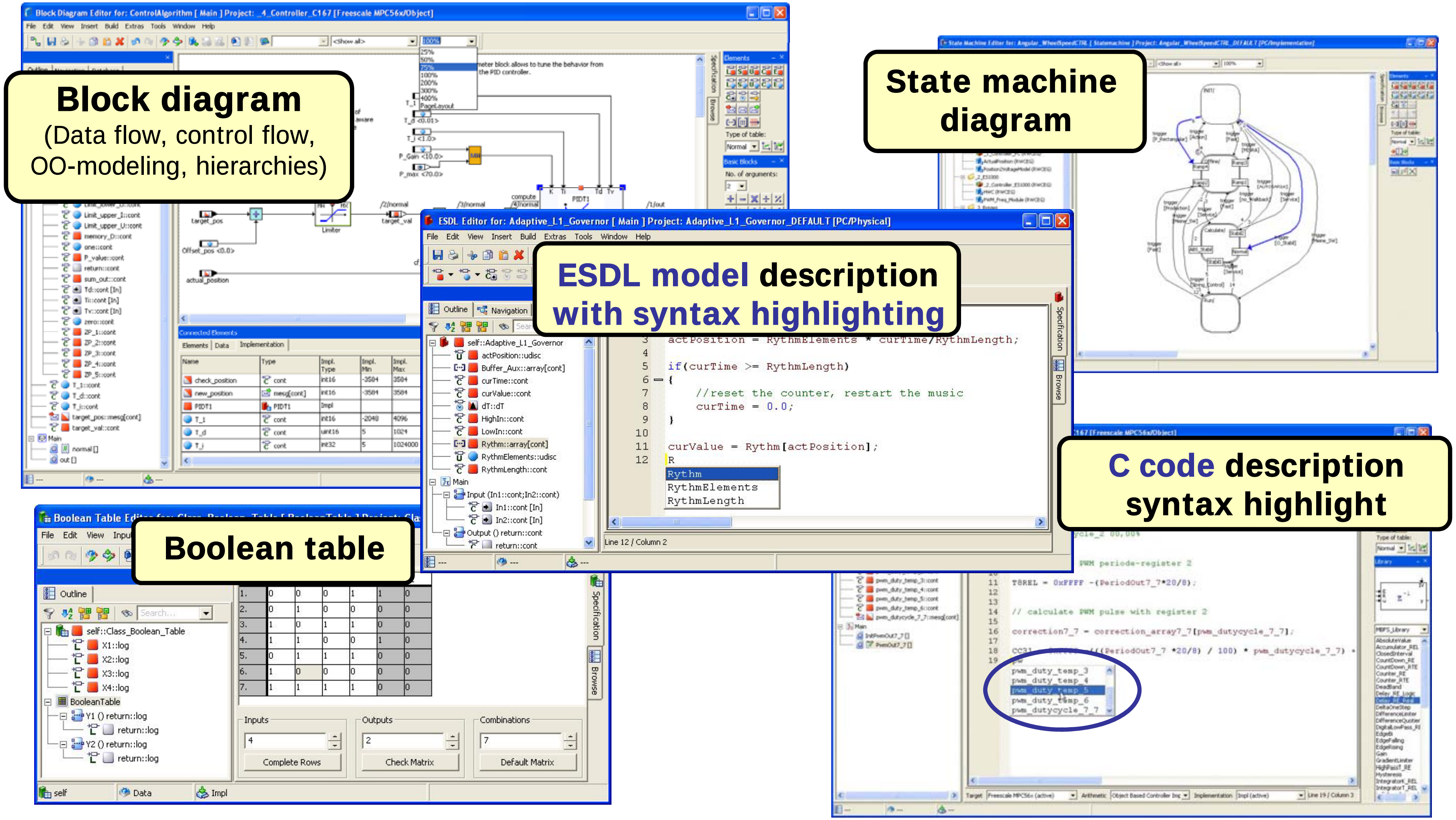


Generate code

```
/* public Iberechnung [] */
void IUMRECHNUNG_IMPL_Iberechnung(void)
{
  sint32 _t1sint32;
  /* Iberechnung: sequence call #1 */
  _t1sint32 = ((uint32)(IUMRECHNUNG_IMPLinstance->A->val * 125) >> 7) /
              IUMRECHNUNG_IMPLinstance->B->val;
  /* assignment to C: min=0, max=65535, hex=4phys+0, limit=(maxBitLength:
true, assign: true), zero incl.=true */
  IUMRECHNUNG_IMPLinstance->C->val =
              (_t1sint32 <= 15) ? ((uint32)_t1sint32 << 12) : 65535;
}
```

Run code in simulation on a PC or deploy on embedded controller

Safety Critical Systems with ASCET
Different types of Models



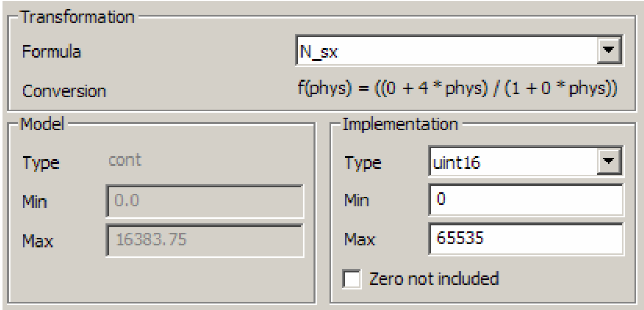
Safety Critical Systems with ASCET
Fixed point arithmetic

Embedded control units are resource constrained:

Floating point arithmetic is expensive => Fixed point arithmetic is used

The model contains the specification of the variables:

Value range, Precision and Data type



Calculation is specified on the physical model:

c = a + b;

The code generator take care of the implementation details

Safety Critical Systems with ASCET
ASCET vs. UML

UML is primarily a design notation:

- Many different diagrams on various levels of abstraction
- Language independent
- Does not contain executable behaviour

ASCET models:

- Fewer diagrammatic styles, and no higher-level abstractions like package or deployment diagrams
- Have both intrinsic value and added value in combination with code generation
- Are executable on multiple different platforms
  - From a 64-bit PC to a 16-bit microcontroller
- Natively supports the C programming language

ETAS

Safety Critical Systems with ASCET
ASCET vs. Ada: Fixed point types

Ada has a strong type system

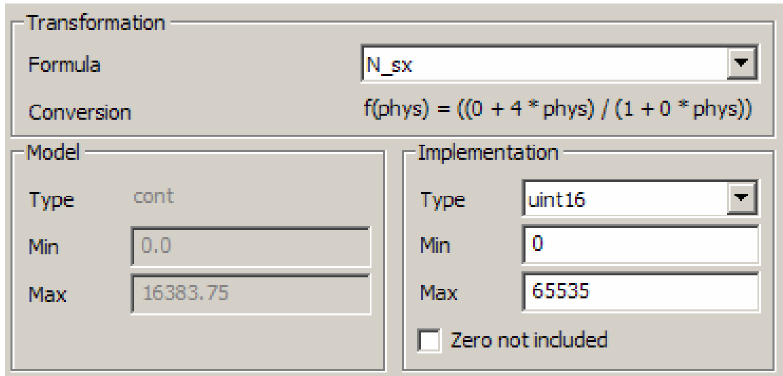
This extends to fixed-point, requiring different types for values of different precision.

Precision/intervals are specified, data type is chosen by the compiler.

```
type VOLT is delta 0.125 range 0.0 .. 255.0;
```

ASCET contains one generic model type “continuous”

- Equivalent to “real” in Ada
- Precision/intervals and data types are specified.
- Model type is realized as “fixed”
- or “float” in the implementation



Safety Critical Systems with ASCET
ASCET vs. Ada: Fixed point arithmetic

Ada:

- Arithmetic between fixed point types is only allowed if they have the same precision, except for multiplication and division, where the target precision must be specified.
- Semantics are specified by the LRM, but complex (compiler is only required to provide *at least* the specified precision - the “small” of the type)

ASCET:

- Arithmetic between all “continuous” expressions is allowed. The code generator takes care of the details: overflow protection, re-scaling, selection of precision in complex expressions.
- Semantics are defined by the code generator
  - And the code generator is proven by use

Safety Critical Systems with ASCET
ASCET vs. Ada: Exceptions

Ada throws runtime exceptions in dangerous situations:

- Array index violations
- Division by Zero
- Integer overflow
- Assignment interval mismatch

The ASCET code generator implements domain-specific default behavior:

- Division by Zero is protected, returning the max value
- Integer overflow is avoided
- Saturated arithmetic on specific microcontrollers is supported
- Assignments are limited to the specified range where necessary
  - Array indices are not limited – cannot assume a default behavior.
  - Array index violations are not expected to occur in practice due to checks in the model or extensive testing.

$$\left\lceil \frac{a}{2} + \frac{b}{2} \right\rceil_{0}^{2^{32-1}} \times 2$$

Safety Critical Systems with ASCET
ASCET vs. Ada: Conclusion

Ada

- is a general purpose language
- Provides basic support for safe fixed point arithmetic
  - If the program compiles, it probably does the right thing
- Requires the programmer to care about possible overflows, re-scaling etc.

ASCET

- is a domain specific language for control algorithms in the automotive industry
- Enables early validation of algorithms using floating point arithmetic
- Provides convenient fixed point arithmetic
  - Code generator makes sensible decisions for the usual problems of overflows, re-scaling etc.
  - Generation of fixed point arithmetic is done consistently
  - Models need to be tested to see if the achieved precision is sufficient

Questions ?