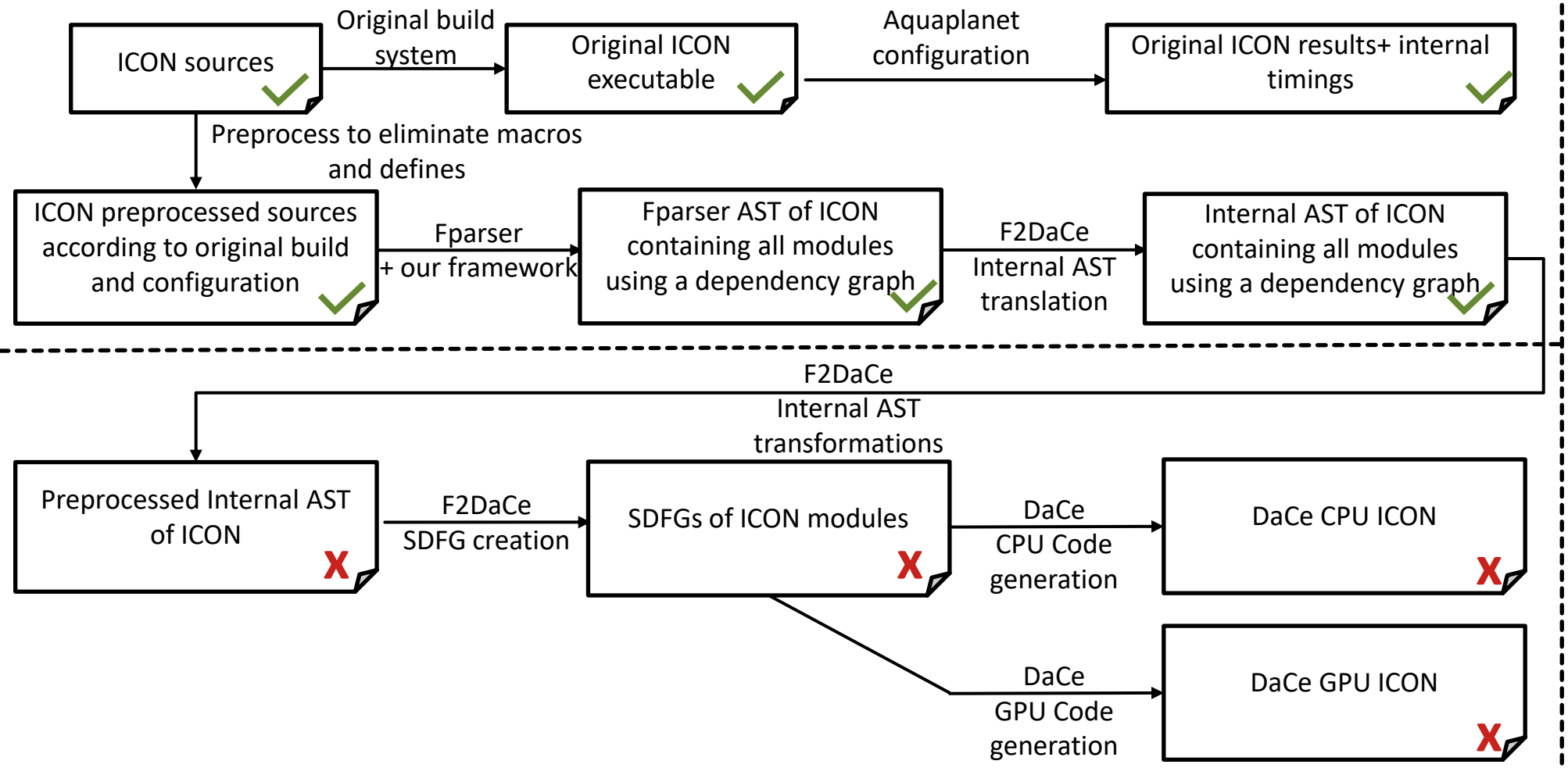
F2DaCe + ICON overall system and overall progress



Additional aspects:

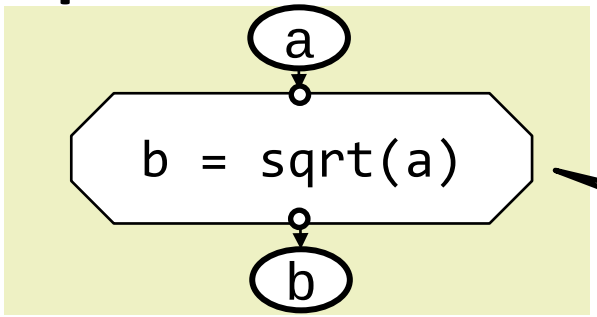
Framework for Value/Timing instrumentation for original ICON

Framework for Value checking/Timing for individual module SDFGs

Together, we need an automatic testing framework to quickly discover bugs

Intrinsic Functions – types by example

1. **SQRT, SIN, COSH**



Python tasklet, therefore analyzable!

2. **SUM, ANY, ALL**

   Eliminated via AST transformation.

Some interesting tradeoff questions: when to codegen using a break?

3. **SELECTED_INT_KIND –** direct evaluation during AST processing
4. **PRESENT, OPTIONAL –** (hopefully) direct evaluation during AST processing
5. **MATMUL –** rewrite/lift as einsums and rely on DaCe optimization

Namespaces…

from module X import a=>a_1

integer a

Rename of imported object- can be function, type, symbol or data container.

- **Proposed solution:**
  - Global rename: if renaming(s) exist for a variable **var_name** to **new_var_name$_{1..N}$** rename both **var_name** and all **new_var_name$_{1..N}$** to **__dace_<module_of_var_name>_var_name** everywhere
  - This should eliminate shadowing issues caused by renamings while still ensuring global uniqueness at the granularity of modules
  - Not a full solution -> DaCe might still have issues with shadowing when inlining SDFGs or lose optimization opportunities because of it. (Need to investigate)

Functions from Object Oriented Programming

FUNCTION test(a)

CLASS(*) a

- **a can be anything!**
- This can even be theoretically decided at runtime
- I hope we might be able to narrow this down through AST analysis for ICON and generate multiple functions, one for each datatype actually used in the call tree: