Retrascope - Task #5756

EFSM pre-initial state + initialization action

03/25/2015 11:54 AM - Sergey Smolov

Status:	Closed	**Start date:**	03/25/2015
Priority:	Normal	**Due date:**	
Assignee:	Sergey Smolov	**% Done:**	100%
Category:	Engine (Transformer)	**Estimated time:**	0.00 hour
Target version:	0.1		
Detected in build:	svn	**Published in build:**	20150701

Description

1. Try to construct initialization action that contains assignments to default values (if they're exist in HDL). If failed, print warning message.
2. Construct false pre-inital state & add it to the EFSM model.
3. Link pre-initial state with appropriate target (initial state) by initialization transition with true guard and initialization action.
4. Try to find RESET signal by already implemented heuristic.
5. Store the pre-initial state data.

History

#1 - 03/25/2015 11:54 AM - Sergey Smolov

- Category changed from 81 to Engine (Transformer)

#2 - 03/25/2015 04:11 PM - Sergey Smolov

- Subject changed from EFSM pre-inital state + initialization action to EFSM pre-initial state + initialization action

#3 - 03/26/2015 09:29 AM - Sergey Smolov

- Status changed from New to Open

- % Done changed from 0 to 30

2,3,4 - done.

Work on 1 is in progress (I've sent an email to Zamia team with a question about variables initialization). So today initialization action is empty.

#4 - 04/02/2015 06:54 PM - Sergey Smolov

As for the initial values of variables, the answer from Zamia team is:

Начальное значение хранится в IGObject getInitialValue. Но проблема в том что это может быть выражение. То есть его ещё вычислить надо и оно имеет тип IGOperation. То есть его видимо можно просканировать на предмет IGObjectов. А чтобы вычислить, надо симулятор запускать.

Проблема Замии как мне кажется в том что тут не хранятся обратные ссылки. IG модули знают о своих дочерних объектах, а вот дочки, в отличае от AST дерева, на родителя не факт что ссылаются. На декларацию они ссылаются, а декларации на контейнер нет. Поэтому мы последний раз строили базу данных: пробегали вначале всю структуру (это был netlist: только сигналы и IGInstances -- никаких процессов) и запоминали какие порты гейтов связаны с какими сигналами. Потом уже быстро получалось отображать сигнал на модуль. Тут по-видимому тоже надо нечто подобное делать: пробежать IG-иерархию, типа что IGStructureVisitor делает и создать базу данных -- простое отображение IGObject на родительский IGScope (IGProcess, IGModule и IGStructure реализуют Scope интерфейс). Но это труд и очень большая похоже. Логичнее было бы всё это прямо в Замии реализовать, а может быть и легче на порядок. Но я не решился. Побоялся что дополнительные данные в DB несколько снизят её производительность и главное: создание новых связей требуется связываться со страшной ZamiaDB, а у меня против неё предубеждения.

#5 - 04/19/2015 11:01 AM - Sergey Smolov

More info about default values in VHDL:

```
Сори, вот так это делается. Если вязть исходник

entity LEFT is end entity;

architecture Arch of LEFT is
    signal b1: boolean;
```

```
    constant b2: bit := bit'left;
    constant i1: integer := 1 + 1;
begin
end architecture;
```

В нём задекларировано три объекта (сигналы и константы). Причём у первого не задано начальное значение. Выяснить это можно нехитрым кодом (питон в данном случае)

```
from org.zamia.instgraph.interpreter import IGInterpreterRuntimeEnv, IGInterpreterCode
from org.zamia.vhdl.ast.VHDLNode.ASTErrorMode import EXCEPTION

tl = Toplevel(DMUID.parse("WORK.LEFT"), None)
module = project.getIGM().findModule(tl)

#signal = module.getContainer().getInterfaceList()
#b1 = module.getContainer().resolveObject("B1")
#iv = b1.getInitialValue()
#printf("%s: %s := %s ", b1.getId(), b1.getType(), iv)

location = a.computeSourceLocation()

ic = IGInterpreterCode("Getting initial value", location)
#printf("ic = %s", ic)
env = IGInterpreterRuntimeEnv(ic, project)
#printf("env = %s", env)

for obj in module.getContainer().localItems():
    iv = obj.getInitialValue() # value is expression actually to be evaluated
    if iv != None: # initial value is defined

        #1: evaluating the iv expression
        iv = " expression %s\n evaluated to %s" % (iv, iv.computeStaticValue(env, EXCEPTION, None))

    else: # init value is undefined -- getting default (left) value

        t = obj.getType().computeStaticType(env, EXCEPTION, None )

        #The core of getting left value. It was misnomered 'generateZ'
        left = org.zamia.instgraph.IGStaticValue.generateZ(t.computeStaticType(env, EXCEPTION, None), location
);
        iv = "unspecified, taking %s'left = %s" % (t, left) # must be false

    printf("%s: %s := %s ", obj.getId(), obj.getType(), iv)
```

Питон берёт контейнер модуля, который содержит все декларации переменных. Если начальное выражение obj.getInitialValue задано, то iv.computeStaticValue(env, EXCEPTION, None)) просто вычисляет его. Иначе, нужно получить статический тип объекта и функция generateZ возвратит левое значение по умолчанию. Она по-ошибке названа generateZ, хотя предназначена не для z-значений, а как раз для левых значений по умолчанию в VHDL.

#6 - 05/07/2015 05:11 PM - Sergey Smolov

- Status changed from Open to Resolved

- % Done changed from 30 to 100

The scheme was modified and resolved.

r2021 (trunk).

#7 - 05/07/2015 05:11 PM - Sergey Smolov

- Status changed from Resolved to Verified

#8 - 07/01/2015 11:52 AM - Sergey Smolov

- Status changed from Verified to Closed

- Published in build set to 20150701