Re: LLIR Task-Modell?

Von: Arno Luppold <arno.luppold@tuhh.de> (Hamburg University of Technology)
An: Heiko Falk <Heiko.Falk@tuhh.de>
Datum: 05.01.2016 12:36

Hallo,

Eigentlich ergänzen sich im jetzigen Zustand die beiden Modelle Entrypoints
und TaskEntry eher, als dass sie sich widersprechen.
Soweit ich das sehe, entspricht ein TaskEntry einem Binary, das auf einem
Core ausgeführt werden kann. Es hat sein eigenes Configuration*-Objekt, was
dann sogar heterogene Systeme ermöglicht.
D.h. pro Core gibt es bei uns derzeit einen TaskEntry, auf dem ein Binary
ausgeführt wird.
Dieses Binary wiederum enthält einen (main()) oder mehrere Entrypoints.
Was da eigentlich (finde ich) hauptsächlich verwirrt, ist die verwirrende
Bezeichnung "TaskEntry".

Funktional benötigen wir, denke ich, weiterhin auf jeden Fall einen Weg,
einzelne Funktionen innerhalb einer LLIR als Entrypoint/Task/... zu markieren
und mit Scheduling-Informationen zu annotieren:
* Das "klassische" Mehrprozess-System (wie ich es derzeit verwende)
  bildet mehrere Tasks in einer LLIR ab. Auch die Echtzeitbetriebssysteme,
  die ich kenne, machen das so (bspw. ERIKA oder µC/OS-II).
* Auch einige Mehrkern-Prozessoren (bspw. der Aurix) unterstützen dieses
  Modell. Man kann hier den identischen Code auf allen Cores ausführen,
  über ein Register den aktuellen core abfragen und dann entsprechend
  Funktionen aufrufen.

Wenn wir jetzt auf Multicore + Multiprozess gehen, gibt es auf jedem Core
eine "Box", die einen oder mehrere solche Entrypoints enthält.

Von daher finde ich die von dir da bislang vorgeschlagene Hierarchie da nicht
schlecht. Die LLIR_Task habe ich da auch bislang als Entsprechung der
TaskEntry gesehen.
Innerhalb dieser LLIR_Task kann man dann ja die einzelnen Entrypoints
verwalten, d.h. deren jeweiliges Anregungsmodell, Deadline,... und eben einen
(smart)Pointer o.Ä. auf die LLIR_Function, die Einsprungspunkt ist.

Zur Vereinfachung meiner WCETILP-Basisklasse (um mit Ereignisströmen
arbeiten zu können) habe ich so etwas in der Art (d.h. ausgehend von den
TaskEntry-Objekten mit den entrypoints arbeiten) sogar schon mal im WCC
implementiert:
LLIROPTIMIZATIONS/lliroptimizations/wcetilpcalculation/event_stream.h
und als konkrete Implementierung für periodische Tasks mit Jitter:
LLIROPTIMIZATIONS/lliroptimizations/wcetilpcalculation/event_pj.h

Für die WCRT-Analysen, die der WCC seit Kurzem ja kann (für den
orakel-Solver-Ansatz...), habe ich das ganze Task-Zeugs übrigens schon mal
nochmal WCC-Unabhängig implementiert (aber in
lliroptimizations/wcetilpcalculation noch nicht vollständig glatt gezogen):

LIBMULTIOPT/libmultiopt/rta*.cc bzw rta*.h

Die Anwendung dieser Klasse siehst du in
LLIROPTIMIZATIONS/llirTCanalyses/scheduling/ (bzw. noch unimplementiert in
llirARManalyses/scheduling/ für ARM).

Viele Grüße
Arno

On 2016-01-05 Heiko Falk <Heiko.Falk@tuhh.de> wrote:

> Hoi Arno,
>
>
> als wir uns neulich über Grundzüge einer neuen LLIR unterhalten hatten,
> hatte ich ja den angehängten Vorschlag für eine Klassen-Hierarchie
> rumgeschickt. Dabei hatte ich außer Acht gelassen, dass der WCC ja
> momentan einen Wust unterschiedlicher Task-Modelle enthält: Es gibt WCC
> Tasks und Tasksets, die irgendwie mit der Configuration-Klasse verheiratet
> sind, und es gibt die entrypoint-Annotationen. Beide Varianten sind
> vollkommen unabhängig voneinander und haben eine komplett unterschiedliche
> Semantik.
>
> In meinem Bildchen von neulich hatte ich die Existenz der entrypoints
> verdängt und einfach nur Tasks sehr weit oben in der Klassen-Hierarchie
> vorgesehen, um Scheduling-Sachen zu modellieren. Du arbeitest ja ganz
> praktisch mit diesen WCC-Infrastrukturen auf Task-Ebene. Was denkst Du als
> Praktiker also, wie in einer zukünftigen LLIR die Task-Ebene "schön"
> modelliert sein sollte?
>
>   Heiko
>

--
Dipl.-Ing. Arno Luppold
Institute of Embedded Systems
Hamburg University of Technology
Am Schwarzenberg-Campus 3(E)
21073 Hamburg, Germany

Room: 3.003
Tel: +49 (0)40 42878-3477
Fax: +49 (0)40 42878-2798
http://www.tu-harburg.de/es/