Levo state trace output

David A. Morano

Northeastern University

1. introduction

This document will describe the Levo machine clock state output trace format. Although the Levo machine being sequenced is done so at the moment using the LevoSim simulator, this is not strictly necessary and future Levo simulators may be used instead. In any event, the output state of the Levo machine may contain similar elements across more than one simulator.

1.1 Levo machine

The output state trace described in this document is the registered state information from a Levo machine. More information on the Levo machine and its design philosophy can be found in the Technical Report by Uht. [1]

1.2 LevoSim

The current vehicle used to sequence a Levo machine is the LevoSim simulator. This simulator implements the actual hardware components of a Levo machine using software. Hardware components are modeled individually and are interconnected to form the whole machine. The whole machine is somewhat hierarchical and this hierarchy is usually expressed within the simulator also with its various software pieces. For most of the whole Levo machine, each hardware component is implemented with a corresponding software component and possibly several subcomponents each of which can have their own subcomponents.

The simulated hardware components process inputs that are usually the outputs from hardware registers and produce new next-state. The processing of combinatorial signals occurs over time through a mechanism in the simulator known as clock phases. These both somewhat mimic the phases of the hardware clock period as well as for the provision of synchronization points in time for the communication of signal between different hardware components. All hardware components are also clocked. Typically, all clocked hardware components transition the previously computed next-state signals to be the current state of the new clock period.

In this way, the whole of the simulator resembles very closely the hardware of an actual machine, transitioning synchronously through states in many state machines. More information on the LevoSim simulator is currently only found in some outline oriented documentation that exists for most of the major components as well as in the source code itself. All information for LevoSim (documentation and source code) is currently at the University of Rhode Island in the normal Levo research file systems.

1.3 XML

The Levo output state trace will make use of XML as the structuring of all machine state data. There will be two files produced by LevoSim for the output state trace. One file will be the XML proper. The other file will be an index of the XML file. The index file is an index of the start of clock blocks in the XML file. The index file contains both the starting offset of clock blocks in the XML file as well as a means for finding an XML clock block that contains a full dump of all machine registers (that are being dumped at all). Most of the rest of this document will describe the details of these two files.

2. XML usage

The decision to use XML as the means to structure the output trace data was largely quite pragmatic. A means was needed to both tag data with a name as well as a way to structure related data hierarchically (thus matching the hierarchy of the hardware components themselves). XML conveniently provides both of these capabilities. XML is also rather simple to write and to understand giving it an advantage over other methods. XML also brings a significant disadvantage than other approaches. The main disadvantage of using XML for our purposes is that since it is all ASCII, it tends to consume a large amount of space for relatively modest information content. Further, closing tags are really redundant but take up space none-the- less. However, it is felt that the advantages of XML outweigh the disadvantages for our immediate goals.

A brief outline of how XML is used in the state output will now be presented. XML consists of tags (opening and closing) and the data associated with those tags. Tags may introduce blocks that contain more tags, et cetera. Some definitions of some XML structures is given next.

leaf tags	these are tags that include only data elements in them and no further tags
block tags	these tags introduce a block that only has other tags in it
generic block tags	these are block tags but may appear in several other tag blocks and will have the same meaning everywhere

For purposes of this document, block tags and generic tags will be described individually. Leaf tags will always be described along with the block tags that enclose them.

2.1 miscellaneous tags

Some tags are not part of the Levo machine hardware but serve to frame other important information. Two tags are in this category so far. These are described below:

2.1.1 the configuration tag This tag appears as the very first tab in the XML file. This block tag given the information on the configuration of the Levo machine that was sequenced. It contains several leaf tags that give the parameters used to configure the Levo machine. These tags are:

nsgrows	a decimal number that gives the number of Sharing Group rows in the machine
nsgcols	a decimal number that this gives the number of Sharing Group columns in the machine
nasrpsg	a decimal number that gives the number of Active Station rows per each Sharing Group; note that the number of Active Station columns in each Sharing Group is currently fixed at two
nrfspan	a decimal number that gives the span (in SGs) of the Register Forwarding buses
nrbspan	a decimal number that gives the span (in SGs) of the Register Backwarding buses
npfspan	a decimal number that gives the span (in SGs) of the Predicate Forwarding buses
nmfspan	a decimal number that gives the span (in SGs) of the Memory Forwarding buses
nmbspan	a decimal number that gives the span (in SGs) of the Memory Backwarding buses
ndeepaths	a decimal number that gives the number of possible DEE paths in the machine
nprpas	a decimal number that gives the number of possible Predicate Registers that can be held in any single Active Station
ifetchwidth	a decimal number that gives the with (in 32- bit units) of the Instruction Response bus
mfbinter	this is a number in hexadecimal that gives the bits that form the bus interleave index for the Memory Forwarding bus between the execution window and the memory subsystem

mbbinter	this is a number in hexadecimal that gives the bits that form the bus interleave index for the Memory Backwarding bus between the execution window and the memory subsystem
mwbinter	this is a number in hexadecimal that gives the bits that form the bus interleave index for the Memory Write bus between the execution window and the memory subsystem
iw:rftype	this is a decimal number that gives the type of register forwarding that is used in the machine
iw:btrbsize	this is a decimal number that gives the size of the Branch Tracking Buffer
iw:sqsize	this is a decimal number that gives the Store Queue FIFO depth
iw:wmfinter	this is a number in hexadecimal that gives the bits that form the bus interleave index for the Memory Forwarding bus within the execution window
iw:wmbinter	this is a number in hexadecimal that gives the bits that form the bus interleave index for the Memory Backwarding bus within the execution window

2.1.2 clock tag This tag is used for all other basic blocks in the trace output. It currently only has two sub- tags. Tags that can appear in this tag block are:

value	this is a leaf tag that gives the clock number of the current clock block
levo	this is a block tag that encloses all Levo machine hardware component tags

2.2 Levo machine component tags

This section describes the Levo machine hardware component tags. These tags are divided into two major types. The block tags used within for Levo machine components fall into two types. These are the generic block tags and the non- generic block tags.

2.2.1 Non-generic tags This section describes most of the block tags used.

2.2.1.1 levo tag This tag encloses all other Levo machine component tags. It's sub- tags are:

lmem	this tag encloses all memory subsystem block tags
iw	this tag encloses all execution window block tags
busgroup	this is a generic block tag that describes a group of related buses
busname	this is a generic block tag that describes a single bus

2.2.1.2 lmem tag This tag represents the top of the entire memory subsystem of the Levo machine. The sub- tags for this tag defined so far so far are:

uid	a hexadecimal number that gives the unique ID of this block

2.2.1.3 the iw tag This block tag introduces the Levo machine execution window. This is a big one! This contains most of the machine components of the machine. The sub- tags of this block are:

lifetch	this block tag introduces the LIFETCH hardware component
llb	this block tag introduces the LLB (load buffer) hardware component
sg	this is a block tag that introduces a Sharing Group logical arrangement of Levo hardware components
lwq	this is a block tag that introduces the Levo Write Queue (LWQ) hardware component

2.2.1.4 the lifetch tag Currently, there are no sub- tags in this block.

2.2.1.5 the llb tag This block tag introduces the state for the Levo Load Buffer (LLB) hardware component. This is basically just a register of decoded instructions. The sub- tag for this block are:

entry	this is a block tag that introduces one entry of many for the LLB component; the number of entries is the height of an AS column

2.2.1.6 the entry tag This tag introduces one entry (row) of the LLB. Its sub- tags are:

row	a leaf tag containing a decimal number giving the row index of this entry
valid	a leaf tag containing a single binary digit indicating whether the *valid* bit is set in this entry or not
used	a leaf tag containing a single binary digit indicating whether the entry contains an instruction of some sort or otherwise unused
ia	if the entry is used (see above), then this leaf tag will appear containing a hexadecimal number giving the instruction address of the instruction in this LLB entry
instr	if the entry is used (see above), then this leaf tag will appear containing a hexadecimal number giving the 32- bit instruction word for the instruction in this LLB entry
instrdis	if the entry is used (see above), then this leaf tag will appear containing an ASCII string of the disassembled instruction that is in this LLB entry

2.2.1.7 the sg tag This block tag introduces a Sharing Group (SG). A Sharing Group is a logical arrangement of several hardware components. There can be many Sharing Groups in the Levo machine execution window. Not all of the components within a Sharing Group are currently programmed to output state but some are. The sub- tags within this block tag are:

sgid	a leaf tag giving in decimal the identification of this Sharing Group (IDs are small positive integers); SGIDs are unique within the whole Levo machine execution window
ascol	this is a block tag that introduces a column of Active Station element (described later) component groups
lpe	this is a block tag that introduces a Levo Processing Element (LPE) hardware component

2.2.1.8 the ascol tag The sub- tags in this block tag are:

id	this leaf tag gives in decimal the ID of this column within the Sharing Group
aselem	this is a block tag that introduces an Active Station element within this Sharing Group

2.2.1.9 the aselem tag This is a block tag that exists with Sharing Groups. This contains a single The sub- tags of this block tag are:

id	this leaf tag contains a decimal number giving the relative ID of this Active Station element within the Active Station column grouping
asrow	this leaf tag contains a decimal number giving the execution window row index of this Active Station element within the context of the entire execution window

asid	this leaf tag contains a decimal number that is the ID of the enclosed Active Station (to be described later) within the context of the entire Levo machine execution window
as	this block tag introduces the state within an Active Station hardware component

2.2.1.10 the as tag This tag introduces the state within a single Active Station hardware component. There are often many of these in any configured Levo machine. Current restrictions on the geometry of the Levo machine as implemented in LevoSim requires that there be at least two Active Stations in each column of the execution window of the machine. The sub- tag are:

uid	this leaf tag contains a hexadecimal number giving the unique ID of this Active Station component; UIDs are useful for tracking hardware components in the software
asid	this leaf tag contains a decimal number giving the ID of this Active Station within the execution window (same as *asid* in other block tags); this ID is useful for human tracking of ASes
used	this leaf tag contains a single binary digit indicating whether this AS has an instruction in it or if it is otherwise unused
path	this leaf tag contains a decimal number that indicates what path the current AS is associated with; paths are small non- zero positive integers and will not exceed in value the number of DEE paths configured for the current machine plus one
tt	this leaf tag contains a decimal number indicating the time- tag associated with the current AS; the number is a small positive integer
ia	if the entry is used (see above), then this leaf tag will appear containing a hexadecimal number giving the instruction address of the instruction in this AS
instr	if the entry is used (see above), then this leaf tag will appear containing a hexadecimal number giving the 32- bit instruction word for the instruction in this AS
instrdis	if the entry is used (see above), then this leaf tag will appear containing an ASCII string of the disassembled instruction that is in this AS
opclass	this leaf tag contains a decimal number indicating the class of the instruction that has been issued to this AS; the number is a small positive integer
opexec	this leaf tag contains a decimal number indicating the decoded opcode of the instruction that has been issued to this AS; the number is a small positive integer
srcreg	this is a block tag that contains a source register set for this AS; there are currently up to four of these in any AS
dstreg	this is a block tag that contains a destination register set for this AS; this destination register set is not the output of the instruction execution in the current AS but rather the input snoop/snarf information on this register; there are currently up to three of these in any AS

2.2.1.11 srcreg and dstreg tags These block tags appear in Active Station blocks and are currently very similar. A source register set contains the register information associated with one source operand to the instruction loaded in an AS. A destination register set contains the register information associated with one destination operand for the instruction loaded in the AS. The term 'destination' may be misleading as this block contains state information for the input of this register to the current AS. The values in a destination register set are those for the operand before any possible computation by the current AS. Therefore both source and destination register sets described here are inputs to as AS and are typically snooped and snarfed identically. Both of these block tags currently have the following sub- tags:

name	a leaf tag containing an ASCII string taken to the name of the register set
addr	a leaf tag containing a hexadecimal number of the address of the register operand
dv	a leaf tag containing a hexadecimal number that is the current data value of the register operand
path	a leaf tag containing a decimal number giving the path that this register operand came from
tt	a leaf tag containing a decimal number giving the time- tag value of the source of this register operand; this is the time- tag of the hardware component where the operand was generated from

2.2.1.12 the lpe tag This is a block tag that introduces the state for the Levo Processing Element hardware component. Currently, there is no clearly defined state for this component so there are not yet any sub- tags.

2.2.1.13 the lwq tag This tag introduces the Levo Write Queue (LWQ) hardware component. There are not yet any sub- tags for this component.

2.2.2 Generic tags This section describes some generic block tags. These tags are used in more than one place. That is, they may be enclosed by more than one other block tag.

2.2.2.1 the busgroup tag This tag introduces a block that describes a group of related buses. Some of the sub- tags of this block tag are optional. Usually only one other block tag will be enclosed by a*busname* tag. The *possible* sub- tags are:

name	this is a leaf tag whose value is a name assigned to the group of buses
rfbus	this is a generic block tag that introducing a RFBUS type hardware component
libus	this is a generic block tag that introducing a LIBUS type hardware component

2.2.2.2 the busname tag This tag introduces a single bus. The enclosed bus is usually described by a generic block sub- tag. Some of the sub- tags of this block tag are optional. Usually only one other block tag will be enclosed by a *busname* tag. The *possible* sub- tags are:

name	this is a leaf tag whose value is a name assigned to the group of buses
rfbus	this is a generic block tag that introducing a RFBUS type hardware component
libus	this is a generic block tag that introducing a LIBUS type hardware component

2.2.2.3 the rfbus tag This block tag introduces the state for the RFBUS hardware component. The sub- tags are:

uid	a leaf tag containing a hexadecimal number giving the unique ID of the component
busy	a leaf tag containing a single binary digit specifying whether the bus is currently busy or not
hold	a leaf tag containing a single binary digit specifying whether the bus is currently being held (stalled) or not
dp	a leaf tag containing a single binary digit specifying whether the bus has any data present in the current clock
lflowgroup	this is a generic block tag that contains a LFLOWGROUP object of related bus data

2.2.2.4 the libus tag This block tag introduces the state for the RFBUS hardware component. The sub- tags are:

uid	a leaf tag containing a hexadecimal number giving the unique ID of the component
busy	a leaf tag containing a single binary digit specifying whether the bus is currently busy or not
hold	a leaf tag containing a single binary digit specifying whether the bus is currently being held (stalled) or not
dp	a leaf tag containing a single binary digit specifying whether the bus has any data present in the current clock
datalane	this is a generic block tag that contains a single lane of data for multi- laned buses; there can be many of these tags in a single *libus* tag

2.2.2.5 the datalane tag This block tag introduces the information for a single lane of data for some outer block tag. The sub- tags are:

id	a leaf tag containing a decimal number giving the relative ID of the component
lflowgroup	a block tag that contains the related data elements of an LFLOWGROUP hardware object

2.2.2.6 the lflowgroup tag This is a block tag that introduces the information for the LFLOWGROUP hardware object. This object generally corresponds with a "transaction" in the Levo machine. Transactions contain both architectural data as well as micro- architectural meta- data describing the context of the architectural data. The sub- tags are:

ftt	a leaf tag containing a decimal number giving the forwarder's time tag value; a forwarder was a component that last forwarded this LFLOWGROUP if any
ott	a leaf tag containing a decimal number giving the originator's time tag value; an originator was a component that first originated this LFLOWGROUP
tt	a leaf tag containing a decimal number giving the time tag value of this present LFLOWGROUP; this is the time tag used for snooping of this LFLOWGROUP
seq	a leaf tag containing a decimal number giving the sequence number value of this present LFLOWGROUP
path	a leaf tag containing a decimal number giving the path for this present LFLOWGROUP
addr	a leaf tag containing a hexadecimal number giving the address of this present LFLOWGROUP
dv	a leaf tag containing a hexadecimal number giving the data value of this present LFLOWGROUP
dp	a leaf tag containing a decimal number giving the data- present bits (the low four bits) for the data in this present LFLOWGROUP
trans	a leaf tag containing a decimal number giving the transaction code for this present LFLOWGROUP

3. Acknowledgments

Many thanks go to Sean Langford for the initial discussions I had with him about using XML for the output state dump and his enthusiasm for the idea.

REFERENCES

1. , A.K. Uht, D.A. Morano, A. Khalafi, M. de Alba, T. Wenisch, M. Ashouei, and D.R. Kaeli, *IPC in the 10's via Resource Flow Computing with Levo*, 2001- 09- 01.

CONTENTS