RC25536 (BRA1505-003) May 6, 2015
Computer Science

IBM Research Report

Introduction to CircuitML: Modeling Local Processing Units in the *Drosophila* Brain

Daniel Salles Chevitarese
IBM Research
Avenide Pasteur, 138 – Urca
Rio de Janeiro – RJ, 22290-240
Brazil
and
Pnotifical Catholic University

Dilza Szwarcman
Computer Science Department
UEZO
Rio de Janeiro, Brazil

Marley Vellasco
Electrical Engineering Department
Pontifical Catholic University
Rio de Janeiro, Brazil

Research Division
Almaden – Austin – Beijing – Brazil – Cambridge – Dublin – Haifa – India – Kenya – Melbourne – T.J. Watson – Tokyo – Zurich

Introduction to CircuitML: Modeling Local Processing Units in the *Drosophila* Brain

Daniel Salles Chevitarese
Electrical Engineering Department
Pontifical Catholic University
Rio de Janeiro, Brazil
Email: daniel@chevitarese.com.br

Dilza Szwarcman
Computer Science Department
UEZO
Rio de Janeiro, Brazil
Email: dilzamattos@uezo.rj.gov.br

Marley Vellasco - IEEE Senior Member
Electrical Engineering Department
Pontifical Catholic University
Rio de Janeiro, Brazil
Email: marley@ele.puc-rio.br

***Abstract*—The brain of the fruit fly *Drosophila Melanogaster* is an attractive system for studying the logic underlying neural circuits because it implements a rich behavior repertoire with a number of neural components that is five orders of magnitude smaller than that of vertebrates. Analysis of the fly's connectome using a powerful toolkit of well-developed genetic techniques and advanced electrophysiological recording tools enables the fly's neural circuitry to be experimentally mapped into functional units, called Local Processing Units (LPU). Many tools are already available to enable neuroscientists to create an accurate model of the entire fly brain, but none of them provides a method to specify those circuits in a way that both biologists and engineers can work together. Also, the development of plausible LPU models requires the ability to specify and instantiate subcircuits without explicit reference to their constituent neurons and internal connections. To this end, we present a neural circuit specification language called CircuitML for construction of LPUs. CircuitML has been designed as an extension to NeuroML; it provides constructs for defining subcircuits that comprise neural primitives supported by NeuroML. Subcircuits are endowed with interface ports that enable connections to other subcircuits via neural connectivity patterns. We have used CircuitML to specify an LPU-based model of the fly olfactory system.**

Keywords*—*Drosophila melanogaster, simulation models, XML, python language, CUDA.

I. Introduction

The research on the fruit fly transformed this tiny insect into one of the most powerful genetic model organisms [1], which uncovers many developmental principles, genetic regulation and cell signaling. Such principles are conserved across species [1] and may help scientists understand more complex brains and explain some inherited diseases.

In addition to the genetic toolkit, recent advances in experimental methods for precise recordings of the fly's neuronal responses to stimuli [2], [3], [4], [5], [6], as well as in techniques for analyzing the fly's behavioral responses to stimuli [7], [8], [9], [10] have been facilitating the shaping of circuits. Also, progress in the reconstruction of the fly's connectome [11], [12], by using identified neurons - stereotyped neurons that can be located in every fly [13] - has contributed much to circuit modeling.

On the engineering perspective, the reasonable compromise between tractability and richness is one advantage, since flies have an interesting behavioral repertoire and possess, approximately, 150,000 neurons [14], which means five orders of magnitude smaller than that of vertebrates.

Studies on the brain of the *Drosophila Melanogaster* have revealed that it comprises about 40 distinct modular subdivisions, most of which correspond to anatomical regions in the brain associated with specific sensory modalities and locomotion. These modules are referred as Local Processing Units (LPUs), because they possess a characteristic population of local neurons. Given that many LPUs are associated with specific stimulus processing that controls behavior, they can be regarded as the functional building blocks of the fly brain. Also, many LPUs' local synaptic connectivity is organized into distinctive and repeated canonical subcircuits that appear integral to their respective functions. To model these LPUs, it is then highly desirable to be able to specify and connect multiple instances of subcircuit models without having to explicitly refer to their contents.

An important tool that is available for the specification of neural circuitry is NeuroML [15], a meta-language, based on XML (Extensible Markup Language). Although very simple and easy to use, it is very powerful because it allows detailed models and their components to be defined in a standalone form to be used across multiple simulators (NEURON [16], GENESIS [17], MOOSE [18], NEST [19]) and to be archived in a standardized format [15]. NeuroML addresses many compatibility issues between software tools, facilitating the reproduction of published models descriptions and results. It also allows the sharing and reuse of model components and the development of new tools for detailed computational modeling [15].

Although NeuroML addresses the compatibility problem in many ways, it has its limitations regarding the specification of local processing units, since it was intended to address the variety of neurological systems organized in a biological fashion. In order to have specifications of functional units with their canonical subcircuits abstractions, it is necessary to have a tool that offers both a standardized language and support to components that abstract new blocks on the engineering perspective. This is the main objective of the new neural circuit specification language, called CircuitML, presented in this paper.

II. Specifying neural circuits as functional building blocks

CircuitML (CML) is a framework for modeling virtual brains as a set of functional building blocks, instead of representing them as networks of interconnected neurons. Each building block comprises smaller components, allowing one to study, for example, consequences of targeted brain disruption in a "IF-THEN" manner, as proposed by [1]: "if this neuron is removed, what behavior would be affected?" or "if someone changes the motion detection system, then...", or even "if one adds more channels [5] to the fly's olfactory system, then...". It also allows scientists to share new discoveries among research groups, and share those new circuits with the scientific community.

In order to achieve this new level of abstraction, we present the design and implementation of CircuitML, a structured description language, which had its first version published in [20]. CML can describe neuronal circuits at a level above NeuroML (NML) level 3 (NetworkML), inheriting its support to many tools and simulators, and allowing scientists to share and validate their discoveries [15]. In addition, such inheritance gives to CircuitML a great variety of elements, ranging from neuronal morphologies, with MorphML [15], to an entire brain comprised by LPUs.

As a companion tool for CircuitML, we also developed a Python API, called libCircuitML. The main goal of libCircuitML is to provide easy-to-use utilities for the manipulation of CircuitML using pythonic tools familiar to programmers. Such tool can be imported into a Python script allowing users to:

- load and validate CircuitML and NeuroML files;
- parse and edit circuit models, which follow the XML language standard;
- save valid XML files either on NeuroML format or on CircuitML format;
- use additional functionality that make it easier to create large models;
- connect to neurokernel [20], [21], [22] for simulation purposes.

A. CircuitML overview

A CircuitML document consists of XML elements describing the circuit components of the neuronal system. The structure of a valid CircuitML document is defined using XML Schema Definition (XSD) files and, therefore, standard XML handling libraries can be used to check its validity. An error will be generated if, for example, the name attribute is missing from the subcircuit element.

Once an XML file is known to be in accordance to the CircuitML format, the contents of the file can be transformed into other formats in a number of different ways, such as SAX (Simple API for XML) or DOM (Document Object Model). It is also possible to convert the original file onto other text or script formats with Extensible Stylesheet Language (XSL) files, which makes CircuitML accessible from simulators including NEURON, GENESIS and PSICS. This approach has the advantage that applications need not be reimplemented to natively support CircuitML, but can still have access to models in the format.

Before starting to describe CircuitML, it is important to understand the differences between NeuroML and CircuitML. By encapsulating the entire neuronal system into LPUs, CircuitML helps scientists to perceive neural systems as a big circuit with interconnected "chips" (LPUs). Each "chip" has its own and unique functionality and a well-defined interface. Such aspect of CircuitML takes us back to the very beginning of object-oriented theory, when Booch [23] stated that the proposed paradigm would help to manage the complexity of massive software-intensive systems. Also, while NeuroML provides all primitives and functionality declarations, CircuitML provides all data abstraction and information hiding needed to make the entire specification process simpler. The advantages of CircuitML over NeuroML are:

- minimum-to-zero coupling between circuit elements ("chips");
- clearly defined interfaces, allowing the abstraction of data and circuit details inside elements;
- reuse and code clearness.

The current scope of CircuitML covers the definition of functional modules (LPUs) with interfaces, smaller modules, called subcircuits, and the connectivity module that glues all modules together. Figure 1 shows the overall structure of CircuitML.

B. Levels of Abstraction

CircuitML offers new components to abstract Local Processing Units by hiding its local circuitry and exposing projection neurons and input ports. Canonical subcircuits, comprising populations of neurons and their own connectivity, can be instantiated many times, which simplifies the specification code. Figure 2 shows the relationship between the biological scale of information processing in neural systems and CircuiML. The very first level is MorphML followed by ChannelML (level 2) and, then, NetworkML (level 3); all three levels are defined in the NeuroML's abstraction stack. This work introduces a fourth level (CircuitML) over the other ones, which adds the functional modularity through LPUs.

In addition to the abstraction stack, Figure 3 shows one example of functional blocks in the Drosophila olfactory system. Behind the LPU box, there is a representation of part of the fly brain with some neuropils delineated in black, where many of them can be regarded as functional blocks. Inside the box, there is a simple model of the Antenna Lobe LPU (*AL* left hemisphere) with multiple olfactory channels (subcircuits). Each channel comprises olfactory receptor neurons (ORNs) and projection neurons (PNs).

The fourth level of CircuitML defines new components to allow the specification of system modules and the connectivity between them. This level has also the purpose to extend NetworkML providing mecanisms to encapsulate networks and their connectivity into functional modules with standardized interfaces. At this level, brain areas can be specified by Local

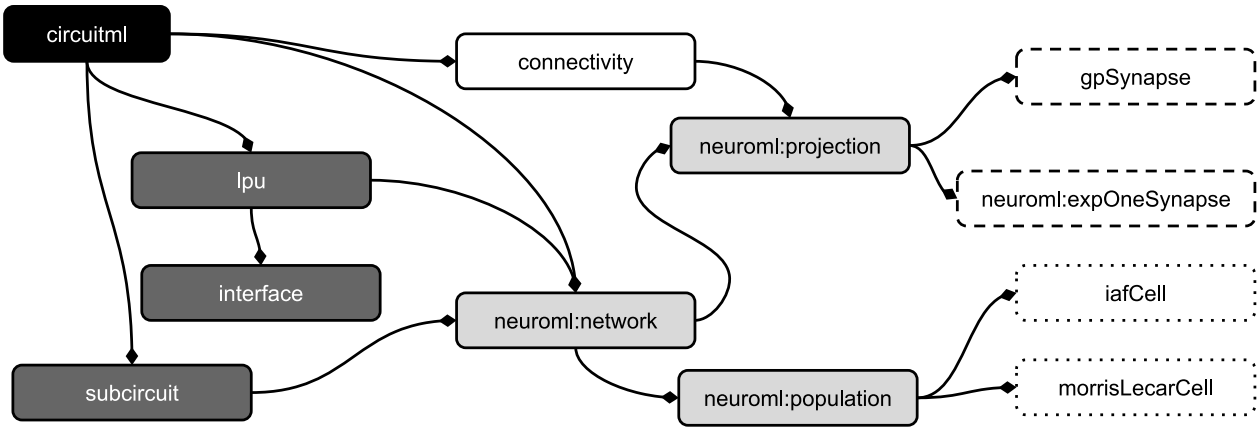

Fig. 1. Overall structure of CircuitML. The top-level element of CircuitML, *circuitml* (black), contains a number of child elements of various types. Dotted elements are cells that are interconnected by synapses (dashed) in a point-to-point fashion. Both cells and synapses elements are encapsulated, respectively, by *populations* and *projections* (light-gray). In turn, populations and projections are encapsulated by *lpu* or *subcircuit* elements (gray), which communicate through *interfaces* (gray) that exposes inner elements. In the white box, the *connectivity* element is presented, which comprises projections of synapses that will be used to connect LPUs.

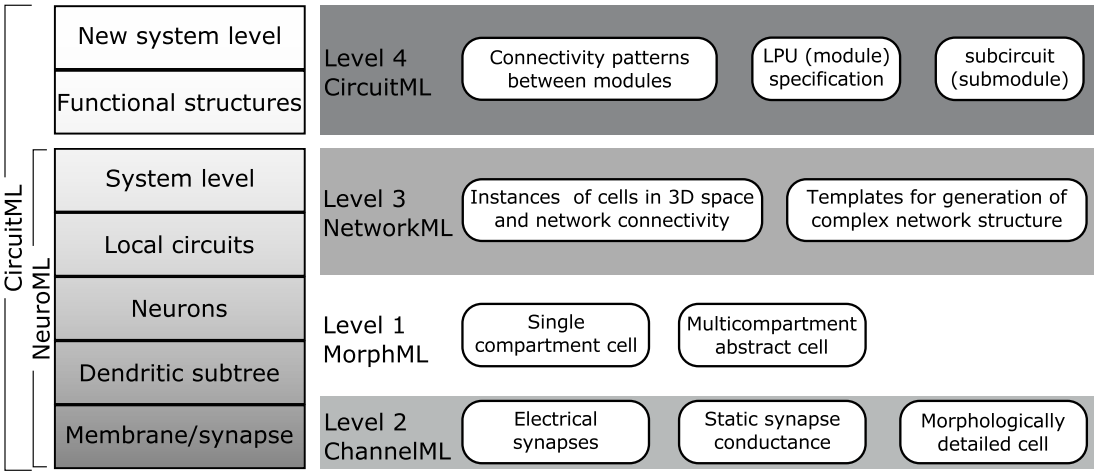

Fig. 2. Abstraction levels in CircuitML and their relationship with the biological scale in neural systems, where MorphML, ChannelML and NetworkML are, respectively, levels 1, 2 and 3 of NeuroML's abstraction stack. On top, the new level is presented comprising the functional structures: LPU and subcircuit; and the connectivity between structures.

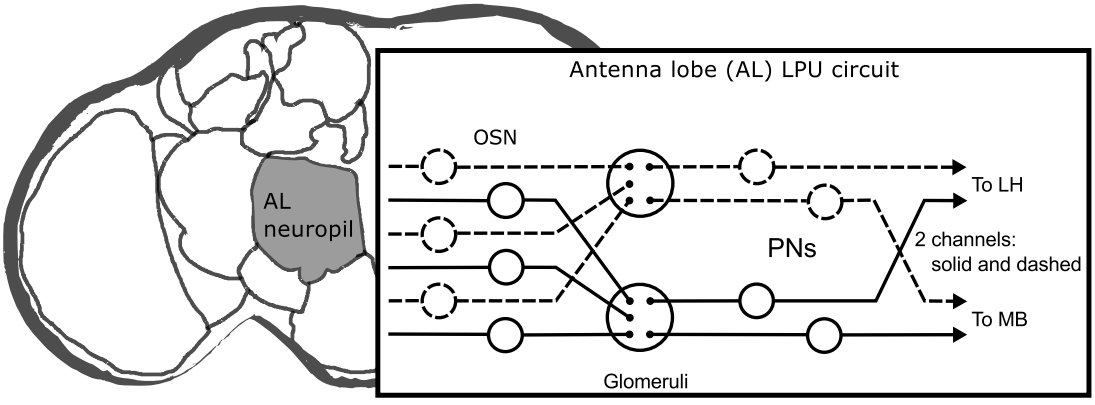

Fig. 3. The fruit-fly brain with many neuropils deliniated on the left hemisphere and the Antenna Lobe neuropil filled in gray. Inside the box, the antenna lobe LPU with two channels, dashed and solid lines, regarding the Antenna Lobe neuropil on the left.

Processing Units (LPU), where each unit stands for a particular functionality.

As mentioned before, local processing units can be compared to chips in a circuit. Each chip has its own internal functionality, which is independent of the external circuit, and has its own standardized interface. Although chips can be very simple, either in its internal circuits or in its functionality, with a small number of elementary kinds of chips combined, it is possible to create complex systems with a great variety of functions.

CircuitML implements four core elements, which are depicted in Figure 1:

1) ***lpu*** (gray box): encapsulates a functional unit with an interface exposing input and output neurons;
2) ***subcircuit*** (gray box): encapsulates smaller circuit parts for reuse inside an LPU. Subcircuits may contain other nested subcircuits, neurons, synapses, and other NeuroML elements;
3) ***interface*** (gray box): generates externally accessible names for neurons comprised by constituent subcircuits;
4) ***connectivity*** (white box): describes synaptic connections between two LPUs.

In Figure 3, some of the CircuitML elements are used to recreate one of the fly's sensory systems, the olfactory: Antenna lobe (AL), which can be specified as an *lpu* element; olfactory channels that can be specified as subcircuits (*subcircuit* element); sets of neurons, where each set can be regarded as a *population* element [15]. On the blue box, lamina (*lpu*) comprises multiple cartridges (*subcircuit*), each of which receives 8 photoreceptors (*population* of photoreceptors).

C. Interface element

The ***interface*** element can be understood as a map between the outside of an LPU and its inner circuits. Listing 1 shows an example of an interface exposing 4 neurons, respectively, 2 input ports and 2 output ports. In line 2, input 0 exposes neuron_0 from population "my_pop", and in line 4, neuron_3 output is exposed by the interface port 2. Notice that, in this example, neuron_2 of the same population is inaccessible from the outside, because it is not listed in "my_interface".

Listing 1. Example of an interface

```
1  <interface id="my_interface">
2   <port id="in_0" in="0", out="my_pop/0"/>
3   <port id="in_1" in="1", out="my_pop/1"/>
4   <port id="out_0" in="my_pop/3" out="2"/>
5   <port id="out_1" in="my_pop/4" out="3"/>
6  </interface>
```

D. Subcircuit element

LPUs may comprise not only networks of neurons, but also smaller functional structures, containing or not neurons, called ***subcircuit***. They are very similar to *network* elements except that they can encapsulate external components, such as filters, pre and post processors, etc. Currently, *subcircuit* supports the same kind of elements that *lpu* does (populations, projections, etc.), but in future releases, it will be possible to add inner elements other then neurons or synapses. This is an important feature for designing a sensory system, for example, that needs pre-processing of analogue inputs or some special spiking encoding.

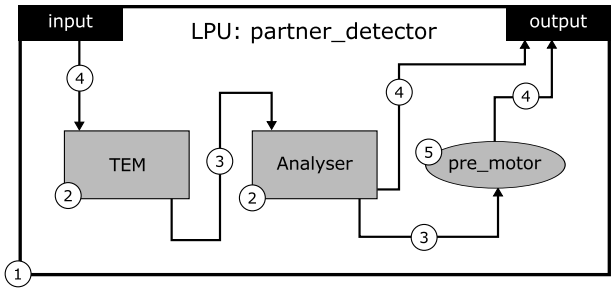

Fig. 4. Example of a fictitious LPU (1) that detects if some input matches the type expected. The boxes numbered with 2 comprise circuits with some function associated and the gray ellipse represents a population of cells that sends some info to the LPU output. Arrows numbered with 4, refers to projections between interface ports and arrows nubered with 3 refers to projections between inner circuits, and between inner circuits, respectively. The ellipse numbered with 5 stands for a population of neurons.

Notice that *subcircuit* elements are not LPUs, either because they have to expose all inner elements (removing the data abstraction), or because they have no complete functionality (disregarding the main point of an LPU).

E. Local Processing Unit element

A simple example of an ***lpu*** is depicted in Figure 4, which shows a basic module called *partner_detector*, containing three main blocks: (1) **TEM** (Time Encoding Machine), which encodes the input analogue signal to spikes, (2) an **analyser** that processes the encoded signal from TEM and (3) a **pre_motor** unit that sends information to the outside of the LPU. In this example, the "partner_detector" LPU classifies inputs into two classes: *valid* and *invalid*. An input is considered *valid* if it was generated by an animal of the same species, and an *invalid* if otherwise. A similar system is found in the auditory system of the fruit fly [24], [25], [26], [27], [28], [29].

The first block (Figure 4 - TEM box) contains a Time Encoding Machine (TEM) [30] that converts analogue signal coming from sensors, for example, to spikes that neurons inside this LPU understand. For simplification purposes, let's consider that someone specified a subcircuit called **tem** that implements a Time Encoding Machine. The output of "my_tem" goes to the "Analyser" block (Figure 4 - Analyser box), which is a network of IAF cells that processes the signal and classifies it as belonging to a valid partner or not. In Listing 2, the Analyser block comprises two populations of IAF cells, respectively, first_step and second_step (lines 6-7). In line 9, a projection connects both populations following a full connected pattern, since no other connectivity pattern was indicated.

Listing 2. Example of the Analyser block

```
1  <circuitml id="ana_block">
2   <refractiaf id="cell_A" threshold="-40mV"
        refractoryPeriod="5ms" capacitance="1nF"
        vleak="-80mV" gleak="100pS" vreset="-70mV"
        v0="-70mV" deltaV="10mV" />
3   <refractiaf id="cell_B" threshold="-35mV"
        refractoryPeriod="6ms" capacitance="2nF"
        vleak="-80mV" gleak="89pS" vreset="-70mV" v0
        ="-70mV" deltaV="10mV" />
4   <expOneSynapse id="syn_1" gbase="0.5nS" erev="0
        mV" tauDecay="3ms" >
5    <network id="analyser">
```

```
    <population id="first_step" cell="cell_A" size
        ="50">
    <population id="second_step" cell="cell_B"
        size="2">
    <!-- Since there is no Connectivity info
        inside this Projection, it will be assumed
         a full connected pattern -->
    <projection id="first_to_second"
        presynapticPopulation="first_step"
        postsynapticPopulation="second_step"
        synapse="syn_1" />
  </network>
</circuitml>
```

The last block (Figure 4 - gray ellipse) comprises a population of neurons that will send their output signals to the interface of the LPU. Listing 3 shows all elements instantiated inside the LPU "partner_detector". In lines 3 and 4 of Listing 3 there is a new element, called **Include**, that substitutes the regular **include**. This new element indicates to libCircuitML parser that the included code must be preprocessed and all internal connectivity must be merged with the code that is importing it. In the end of this process, each LPU will comprise a single internal connectivity matrix, which optimizes both memory usage and access.

Listing 3. Example of the lpu depicted in Figure 4

```
1  <circuitml id="partner_detector">
2   <!-- Including external references -->
3   <Include href="tem_block.xml" />
4   <Include href="ana_block.xml" />
5   <!-- NeuroML elements  -->
6   <iafCell id="pm_cell" reset="-50mV" C="0.03nF"
        thresh="-25mV" leakConductance="1uS"
        leakReversal="-50mV" />
7   <expOneSynapse id="pm_syn" erev="20mV" gbase="65
        nS" tauDecay="3ms" />
8   <expOneSynapse id="ana_syn" erev="15mV" gbase="
        85nS" tauDecay="2ms" />
9   <!-- LPU specification -->
10  <lpu id="partner_detector">
11   <!-- Interface -->
12   <interface id="my_interface">
13    <port id="analogue_input" in="0", out="
          tem_block/tem_block/0"/>
14    <port id="valid_out" in="ana_block/analyser/
          second_step/0" out="1"/>
15    <port id="invalid_out" in="ana_block/analyser/
          second_step/1" out="2"/>
16    <port id="pm_out_0" in="pre_motor/0" out="3"/>
17    ...
18    <port id="pm_out_29" in="pre_motor/29" out="32
          "/>
19   </interface>
20   <!-- Populations -->
21   <population id="tem" structure="tem_block" size
        ="1"/>
22   <population id="analiser" structure ="ana_block
        " size="1" />
23   <population id="pre_motor" component="pm_cell"
        size="30" />
24   <!-- Projections -->
25   <projection id="tem_analyser"
        presynapticPopulation="tem_block/tem_block"
         postsynapticPopulation="ana_block/analyser
        /first_step" synapse="ana_syn" />
26   <projection id="analyser_pm"
        presynapticPopulation="tem_block/tem_block"
         postsynapticPopulation="ana_block/analyser
        /first_step" synapse="ana_syn" />
27   </projection>
28  </lpu>
29 </circuitml>
```

F. *Connectivity element*

The last component presented here is the ***connectivity*** element, which can be understood as the glue between LPUs. Since elements such as *projections* and *connections* are not able to exist outside the LPU, the *connectivity* element acts as a wrapper for *projections* and *connections* that will link not cells or synapses, but interface ports. In order to understand how such element works, let's consider a new system, depicted in Figure 5, where the "partner_detector" (Figure 4 and Listing 3) receives an analogue signal from an external sensor (Figure 5 - top box) and it sends the valid/invalid signals to another LPU that will somehow convert it into a True/False result.

In Listing 4, the *connectivity* on lines 5 to 8 specifies how both LPUs, "partner_detector" and "decode_lpu", will be connected, i.e., the synapse of each connection and which port to link.

Listing 4. Example of two LPUs inter-connected (Figure 5)

```
1  <circuitml id="lpus">
2   <Include href="partner_detector.xml" />
3   <Include href="decode_lpu.xml" />
4   <expOneSynapse id="my_syn" erev="20mV" gbase="65
        nS" tauDecay="3ms" />
5   <connectivity id="pd_to_decoder" lpu1="
        partner_detector" lpu2="decode_lpu">
6    <connection from="partner_detector/valid_out"
         to="decode_lpu/valid" synapse="my_syn"/>
7    <connection from="partner_detector/invalid_out"
         to="decode_lpu/nvalid" synapse="my_syn"/>
8   </connectivity>
9  </circuitml>
```

III. Results

At this point, it is crucial to understand how interconnected LPUs can facilitate the specification of complex systems. In order to be able to model an entire fly brain and then simulate such system, it is imperative to have a good representation scheme to support annotation and modeling [1]. Furthermore, recent progress in neuroinformatics confirmed that comprehensive brain wiring maps are needed to formulate hypotheses about how information flows and is processed inside a brain [14]. These are the two main motifs behind the development of CircuitML: functional modularity and connectivity.

To illustrate how functional structures work, we specified a small demo of the olfactory system of the fruit fly, which is a useful model for studding because of its reduced cell numbers and its similar design with the mammalian olfactory system, which may reflect common functional constraints [31]. The sensory system of the fruit fly has been studied by various groups [32], [33], [34], [31], but the model presented here is based on the Bionet (Columbia University) demo published in [22].

The early olfactory system in *Drosophila* comprises two Antennal Lobes LPUs, one in each side of the brain, as shown in Figure 3 as a gray neuropil. Each of these LPUs has approximately 49 morphologically defined glomeruli that receives information from approximately 30 Olfactory Receptor Neurons (ORN) expressing the same odorant receptor. The axons of each ORN connect to the dendrites of approximately 3 projection neurons (PNs) in the glomeruli that are responsible to transmit olfactory information to the higher regions of the

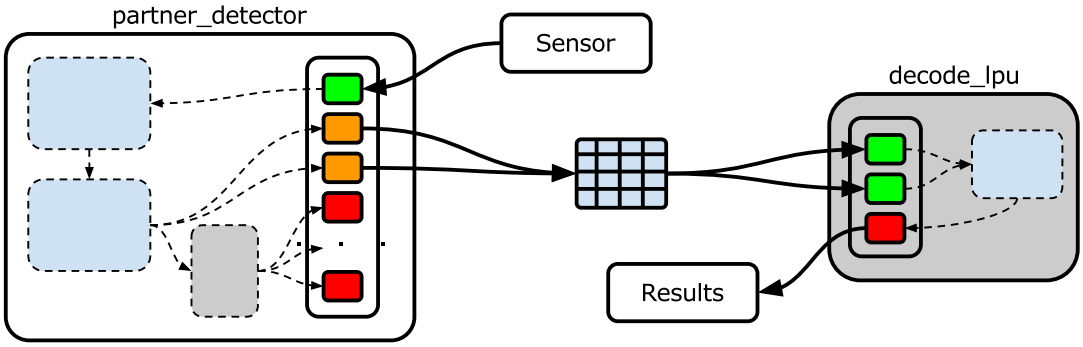

Fig. 5. Two LPUs inter-connected by a connectivity element. Inner circuits are depicted with dashed borders, because they are not visible from the outside of the LPUs.

brain, the mushroom body (MB) and the lateral horn. ORNs also send information to local neurons (LNs), whose connections are restricted to the lobes. The entire early olfactory system in *Drosophila* contains approximately 4000 neurons [31].

The current model, in CircuitML, of *Drosophila's* olfactory system specifies two Antennal Lobes (AL and al) comprising, each, 49 channels - Listing 5. In turn, each channel (Listing 6) encapsulates approximately 30 ORNs transmitting information to 3 projection neurons. PNs can be regarded as the output of the current specification. The entire model comprises 2,800 neurons, or 70% of the fly's entire antenna lobe. All neurons in the system are modeled using the Leaky Integrate-and-Fire (LIF) model [35] and all synaptic currents elicited by spikes are modeled using alpha functions [36]. Parameters for 24 of the glomerular channels are based upon currently available ORN type data [37]; all other parameters are configured with artificial data by Bionet.

In Figure 6, each channel is depicted by lines on different types (solid, dotted, dashed, dash-dot). As discussed before, the antenna lobe contains local neurons (LNs), whose connections are restricted to the lobes that includes synaptic connections between ORNs and PNs, ORNs and LNs, LNs and PNs, and feedback from PNs to LNs (current specification does not include local neurons, yet). By encapsulating some of those neurons into channels, it becomes easier to provide mechanisms by which the activation of different sets of ORNs is transformed into an odor perception in the fly brain that produce a given behavioral output. Eventually, by improving the AL specification, it would be possible to abstract such module and focus on gathering more information about secondary centers' input and how this data is processed.

In Listing 5, some of the lines were removed to reduce space. From line 5 to line 22, *antenna left* is specified with 1470 inputs and 147 outputs, where the connectivity pattern between I/O and the neurons inside the LPU is defined from line 7 to line 16 by using the component *interface*. Since all projection neurons comprised by each glomerulus may emit output visible to other LPUs, their connectivity to the outside is defined inside *interface*, where every port will be exposed by the LPU. Local neurons or ORNs that connect glomeruli do not emit any output to other LPUs and their connectivity is defined inside an LPU component, where they are not exposed by the LPU.

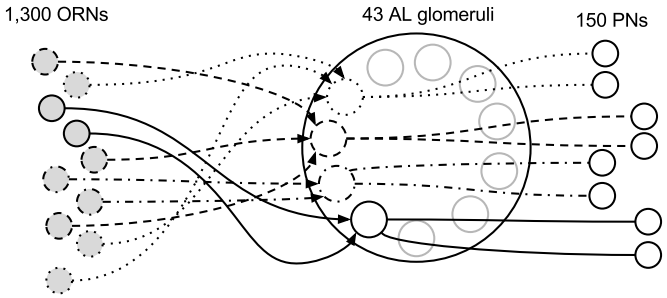

Fig. 6. Circuit of adult olfactory system. The adult olfactory pathway is characterized by converging and diverging connectivity in the AL. Here, each type of line (solid, dotted, dashed, dash-dot) that goes from one ORN to one AL is encapsulated into a structure called channel [31] and represent groups of neurons that express the same odorant receptor.

Listing 5. Simple version of the olfactory system.

```
1  <circuitml id="antenna_demo">
2   <!-- Including external specification -->
3   <Include href="channel_demo.xml" />
4   <!-- First LPU: left antenna -->
5   <lpu id="antenna_left" input="1470" output="147"
        >
6    <!-- LPU I/O -->
7    <interface id="io">
8     <!-- output -->
9     <port id="Or2a_ch0" in="0" out="channel/0/OSN
          /0"/>
10    <!-- ... -->
11    <port id="Or2a_ch49" in="1469" out="channel
          /49/OSN/0"/>
12    <!-- output -->
13    <port id="gl_0_0" in="channel/0/GL/0" out="0"/
          >
14    <!-- ... -->
15    <port id="gl_48_2" in="channel/48/GL/2" out="
          146"/>
16   </interface>
17   <!-- Network specification -->
18   <network id="antenna">
19    <!-- Channels instantiation -->
20    <population id="channels" structure_type="
          channel" size="49"/>
21   </network>
22  </lpu>
23  <!-- Second LPU: right antenna -->
24  <lpu id="antenna_right" input="1470" output="147
        ">
25   <!-- SAME AS ABOVE -->
26  </lpu>
27 </circuitml>
```

The inner network of neurons is specified using channels, which is defined on Listing 6. Synapses and neurons are defined in *synapses.xml* (ln3) and *iafcells.xml* (ln5), respectively. Also, some of the lines were hidden to reduce space. From line 8 to line 21, channel's inner circuit is specified, where all neurons and synapses, once declared on external files, are instantiated by *populations* (for neurons) and *projections* (synapses).

Listing 6. Simple version of the channel subcircuit.

```
1  <subcircuit id="channel_demo">
2   <!-- Synapses types -->
3   <Include href="synapses.xml" />
4   <!-- Neuron cells -->
5   <Include href="iafcells.xml" />
6   <iafCell id="pn" V="-0.07" reset="-0.07" thresh=
        "-0.02" leakConductance="1.01" C="0.07"/>
7   <!-- Channel definition -->
8   <network id="channel">
9    <population id="GL" cell_type="pn" size="3"/>
10   <population id="Osn_default" cell_type="
         osn_default" size="1"/>
11   <population id="Osn_Or43b" cell_type="osn_Or43b
         " size="1"/>
12   <population id="Osn_Or9a" cell_type="osn_Or9a"
         size="1"/>
13   ...
14   <population id="Osn_Or85a" cell_type="osn_Or85a
         " size="1"/>
15   <population id="Osn_Or67a" cell_type="osn_Or67a
         " size="1"/>
16   <!-- Conectivity -->
17   <projection id="proj_osn_glomerulus"
         presynapticPopulation="OSN"
         postsynapticPopulation="GL" synapse="
         syn_osn" />
18   <projection id="Osn_default-DA1"
         presynapticPopulation="Osn_default"
         postsynapticPopulation="GL" synapse="
         osn_default-DA1" />
19   <!-- ... -->
20   <projection id="Osn_Or67a_24-DM6"
         presynapticPopulation="Osn_default"
         postsynapticPopulation="GL" synapse="
         osn_Or67a_24-DM6" />
21  </network>
22 </subcircuit>
```

Before CircuitML, the way many scientists used to specify such system in a common format, would be using graphs, where nodes are neurons and edges are synapses. Using graphs to represent those systems has many advantages on the engineering perspective, but not for the scientist who is specifying the system with thousands, or even, millions of nodes and edges. The specification using *gexf* format (*http://gexf.net*), which is the current standard at Bionet, takes 21,559 lines to specify cells and 49,500 lines to specify synapses.

In addition to the difference between the number of lines necessary to specify the system, CircuitML was envisioned to make the specification much clearer. In Listing 5 and Listing 6 it is easy to understand the big picture of the entire system: the LPUs with input and output, and how circuits are organized and interconnected. Also, using CircuitML, the amount of memory used to save the same specification is reduced. In the case of the *Drosophila's* olfactory system presented in this paper, the graph version uses 3MB not compressed and 210KB compressed, whereas the CircuitML version uses 790KB not compressed and 171KB compressed. Notice that the not compressed version of CircuitML is 3,89 times smaller than the graph version, but the compressed version is only 20% smaller. The reason is that CircuitML specification files tend not to have repetitive text in it, which is usually explored by text compressors.

IV. Conclusion and Future Work

In this paper, we presented the specification language CircuitML that extends NeuroML in order to have neuronal systems designed as functional building blocks. Each of those building blocks, also presented as LPU, may comprise elements ranging from single populations of cells to complex circuitry with many layers of abstractions. Although LPUs can be very complex in the inside, they are endowed with interface ports, which hide such complexity by exposing only necessary parts to be interconnected via neural connectivity patterns, which makes CircuitML a very powerful specification language for neuronal circuitry.

As the *Drosophila Melanogaster* represents a good model for studding LPUs, for the reasons presented here, it may also be an excellent study case for CircuitML. The specification of the entire sensory system of the fruit fly with its individual interconnections and its singularities, may prove that CircuitML is able to address the entire system and also will point to possible issues, such as missing connectivity patterns. In addition, since many parts of those systems are still missing, CircuitML may help scientists to reconstruct the fly brain by providing support to functional building blocks that can be tested and validated independently. By sharing their discoveries, research groups will be able to focus on single LPUs, disregarding other parts that are already encapsulated into other LPUs.

Finally, the simple example presented here, describing the early olfactory system, already shows that the specification of neuronal systems as circuits with interconnected chips that have their own functionality, makes the resulting code cleaner and more systematically delineated. As the visual system is already being studied by many groups, the next steps on the development of CircuitML, also aiming additional innovations, include the specification of the visual system and its complete integration to the olfactory system.

References

[1] J. D. Armstrong, J. I. van Hemert, J. D. Armstrong, and J. I. van Hemert, "Towards a virtual fly brain," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1896, pp. 2387–2397, 2009.

[2] A. J. Kim, A. A. Lazar, and B. S. Yevgeniy, "2d encoding of concentration and concentration gradient in drosophila orns," in *Computational and Systems Neuroscience Meeting, Salt Lake City, Utah*, 2010.

[3] A. J. Kim, A. A. Lazar, and Y. Slutskiy, "System identification of dm4 glomerulus in the drosophila antennal lobe using stationary and non-stationary odor stimuli," *BMC Neuroscience*, vol. 11, no. Suppl 1, p. P174, 2010.

[4] ——, "Drosophila projection neurons encode the acceleration of time-varying odor waveforms," in *Computational and Systems Neuroscience Meeting*, 2011, p. 2.

[5] A. J. Kim, A. A. Lazar, and Y. B. Slutskiy, "System identification of drosophila olfactory sensory neurons," *Journal of computational neuroscience*, vol. 30, no. 1, pp. 143–161, 2011.

[6] R. I. Wilson, "Understanding the functional consequences of synaptic specialization: insight from the¡ i¿ drosophila¡/i¿ antennal lobe," *Current opinion in neurobiology*, vol. 21, no. 2, pp. 254–260, 2011.

[7] S. A. Budick and M. H. Dickinson, "Free-flight responses of drosophila melanogaster to attractive odors," *Journal of experimental biology*, vol. 209, no. 15, pp. 3001–3017, 2006.

[8] A. Y. Katsov and T. R. Clandinin, "Motion processing streams in¡ i¿ drosophila¡/i¿ are behaviorally specialized," *Neuron*, vol. 59, no. 2, pp. 322–335, 2008.

[9] G. Maimon, A. D. Straw, and M. H. Dickinson, "A simple vision-based algorithm for decision making in flying¡ i¿ drosophila¡/i¿," *Current Biology*, vol. 18, no. 6, pp. 464–470, 2008.

[10] M. E. Chiappe, J. D. Seelig, M. B. Reiser, and V. Jayaraman, "Walking modulates speed sensitivity in¡ i¿ drosophila¡/i¿ motion vision," *Current Biology*, vol. 20, no. 16, pp. 1470–1475, 2010.

[11] D. B. Chklovskii, S. Vitaladevuni, and L. K. Scheffer, "Semi-automated reconstruction of neural circuits using electron microscopy," *Current opinion in neurobiology*, vol. 20, no. 5, pp. 667–675, 2010.

[12] S.-y. Takemura, A. Bharioke, Z. Lu, A. Nern, S. Vitaladevuni, P. K. Rivlin, W. T. Katz, D. J. Olbris, S. M. Plaza, P. Winston *et al.*, "A visual motion detection circuit suggested by drosophila connectomics," *Nature*, vol. 500, no. 7461, pp. 175–181, 2013.

[13] S. R. Olsen and R. I. Wilson, "Cracking neural circuits in a tiny brain: new approaches for understanding the neural circuitry of¡ i¿ Drosophila," *Trends in neurosciences*, vol. 31, no. 10, pp. 512–520, 2008.

[14] A.-S. Chiang, C.-Y. Lin, C.-C. Chuang, H.-M. Chang, C.-H. Hsieh, C.-W. Yeh, C.-T. Shih, J.-J. Wu, G.-T. Wang, Y.-C. Chen *et al.*, "Three-Dimensional Reconstruction of Brain-wide Wiring Networks in Drosophila at Single-Cell Resolution," *Current Biology*, vol. 21, no. 1, pp. 1–11, 2011.

[15] P. Gleeson, S. Crook, R. C. Cannon, M. L. Hines, G. O. Billings, M. Farinella, T. M. Morse, A. P. Davison, S. Ray, U. S. Bhalla *et al.*, "NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail," *PLoS computational biology*, vol. 6, no. 6, p. e1000815, 2010.

[16] N. T. Carnevale and M. L. Hines, *The NEURON book*. Cambridge University Press, 2006.

[17] D. Beeman, "GENESIS Modeling Tutorial," *Brains, Minds, and Media*, vol. 1, 2005.

[18] R. M. Cubert and P. A. Fishwick, "MOOSE: an object-oriented multimodeling and simulation application framework," *Simulation*, vol. 6, 1997.

[19] M.-O. Gewaltig and M. Diesmann, "Nest (neural simulation tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[20] D. S. Chevitarese, L. E. Givon, A. A. Lazar, and M. Vellasco, "CircuitML: a Modular Language for Modeling Local Processing Units in the Drosophila Brain," in *Frontiers Neuroinformatics*. Frontiers Research Foundation, Jul. 2013, pp. 80–81.

[21] L. E. Givon and A. A. Lazar, "An open architecture for the massively parallel emulation of the Drosophila brain on multiple GPUs," *BMC Neuroscience*, vol. 13, pp. 1–2, 2012.

[22] ——, "Neurokernel: An open scalable software framework for emulation and validation of drosophila brain models on multiple gpus," *submitted for publication*, 2013.

[23] G. Booch, "Object-oriented development," *Software Engineering, IEEE Transactions on*, no. 2, pp. 211–221, 1986.

[24] A. Kamikouchi, "Auditory neuroscience in fruit flies," *Neuroscience research*, vol. 76, no. 3, pp. 113–118, 2013.

[25] H. T. Spieth, "Courtship behavior in drosophila," *Annual review of entomology*, vol. 19, no. 1, pp. 385–405, 1974.

[26] A. Bretman, J. D. Westmancoat, and T. Chapman, "Male control of mating duration following exposure to rivals in fruitflies," *Journal of insect physiology*, vol. 59, no. 8, pp. 824–827, 2013.

[27] M. J. Kernan, "Mechanotransduction and auditory transduction in drosophila," *Pflügers Archiv-European Journal of Physiology*, vol. 454, no. 5, pp. 703–720, 2007.

[28] M. C. Göpfert and D. Robert, "The mechanical basis of drosophila audition," *Journal of Experimental Biology*, vol. 205, no. 9, pp. 1199–1208, 2002.

[29] F. von Schilcher, "The role of auditory stimuli in the courtship of¡ i¿ drosophila melanogaster¡/i¿," *Animal Behaviour*, vol. 24, no. 1, pp. 18–26, 1976.

[30] A. A. Lazar, E. A. Pnevmatikakis, and Y. Zhou, "Encoding natural scenes with neural circuits with random thresholds," *Vision research*, vol. 50, no. 22, pp. 2200–2212, 2010.

[31] L. B. Vosshall and R. F. Stocker, "Molecular architecture of smell and taste in Drosophila," *Annu. Rev. Neurosci.*, vol. 30, pp. 505–533, 2007.

[32] K.-F. Fischbach and A. Dittrich, "The optic lobe of drosophila melanogaster. i. a golgi analysis of wild-type structure," *Cell and tissue research*, vol. 258, no. 3, pp. 441–475, 1989.

[33] Y. Zhu, A. Nern, S. L. Zipursky, and M. A. Frye, "Peripheral visual circuits functionally segregate motion and phototaxis behaviors in the fly," *Current Biology*, vol. 19, no. 7, pp. 613–619, 2009.

[34] S. J. Caron, V. Ruta, L. Abbott, and R. Axel, "Random convergence of olfactory inputs in the Drosophila mushroom body," *Nature*, 2013.

[35] C. Koch and I. Segev, *Methods in neuronal modeling: from ions to networks*. MIT press, 1998.

[36] R. S. Zucker and W. G. Regehr, "Short-term synaptic plasticity," *Annual review of physiology*, vol. 64, no. 1, pp. 355–405, 2002.

[37] E. A. Hallem and J. R. Carlson, "Coding of odors by a receptor repertoire," *Cell*, vol. 125, no. 1, pp. 143–160, 2006.