Computing (with) Architecture: Pedagogical Explorations at the Intersection of Design and Mechanical Computation

DIMITRIS PAPANIKOLAOU
University of North Carolina at Charlotte

This essay discusses the development of a pedagogical approach that uses principles of mechanical computation as a means to critically inquiry how information and computation can be manifested and performed tectonically through architecture. Drawing examples from the development of two courses, I pose the question: is computing designable? how might we design architecture that computes? Selected student projects are presented and discussed.

> In the most general sense, computation is the process of storing, transmitting, and transforming information from one form to another.
>
> —Santa Fe Institute

COMPUTATIONAL DESIGN VS. DESIGN COMPUTATION

In his book *Soft Architecture Machines*, Nicholas Negroponte considers three potentials for the computer in relation to Architecture: the computer as a designer, the computer as a partner, and the computer as a physical environment.[1] The first potential, coming from design studies, inquires design as a process, asking the question "is designing computable?" The second and third, coming from cybernetics systems theory, and human-machine interaction inquire design as an artifact and its relationship with a user, asking the question "is inhabiting computable?"

While the relationship between computing and designing has retained strong connections with design theories, the relationship between computing and inhabiting has not. Designing architectural typologies from shape grammar rules [2] has been paralleled to the process of composing syntactically valid sentences from language grammar rules,[3] evolving into a procedural theory of designing. In contrast, designing intelligent environments has done little more than appending computing technology to architecture and assigning anthropomorphic behaviors to it as an afterthought, often without critical reasoning on how computing and architecture may affect each other. Today, 40 years after the MIT Architecture Machine Group created the Media Room,[4] we still point and talk to buildings, walls, and furniture as if they have mind, soul, and composure.

The awkward relationship of computing to architecture is illustrated in the own words of some of its most important thinkers In a seminal text for the field of ubiquitous computing, Mark Weiser, acclaimed that the "most profound technologies are those that disappear."[5] In a discussion with architecture students at the Harvard Graduate School of Design, Negroponte, compared the future of information acquisition to the act of "swallowing a pill."[6] Likewise, Mike Kuniavsky, head of design at Xerox Palo Alto Research Center (PARC), described the coming age of ubiquitous computing as "magic."[7] The difficulty in comprehending formalistically ubiquitous computing is also reflected in the increasing role of metaphors in human-computer interaction (HCI) courses as means to understand unfamiliar concepts in terms of familiar ones.[8] Intelligent buildings, today, are metaphorically seen as "friends," "companions," or "assistants" with a meticulous effort to conceal any visible evidence of technology from their clean surfaces. This attitude is in contrast with the still prevailing tendency in many architecture design studios to formalistically express function, materials, and tectonic clarity.

There are two reasons, I argue, for this. The first is that the medium with which information is manifested (electric voltage) and the speed with which it is processed by a computer are invisible and imperceptibly fast. As a consequence, the actual workings of computation are experientially unnoticeable and difficult to express architecturally. The second reason is that, when architecture students learn Interactive Computing, they are trained in high-level programming languages that further mask the machine code that drives the mechanics of computation. The invisibility of information and computation and the de-contextualization of software from hardware, miss the opportunity to use computation as a medium to drive design and architectural discourse, and mask the potential contribution that architecture can make in the field of computing.

I juxtapose the question "is designing computable?" with the question "is computing designable?" If computation is the process of "storing, transmitting, and transforming information from one form to another" then designing computation must be the process of designing systems capable of using energy to transform their states, from input configurations and rule-driven constraints. Nothing other than inventiveness prevents design and creation of computing machines from such things as building components, human movement, and environmental forces.

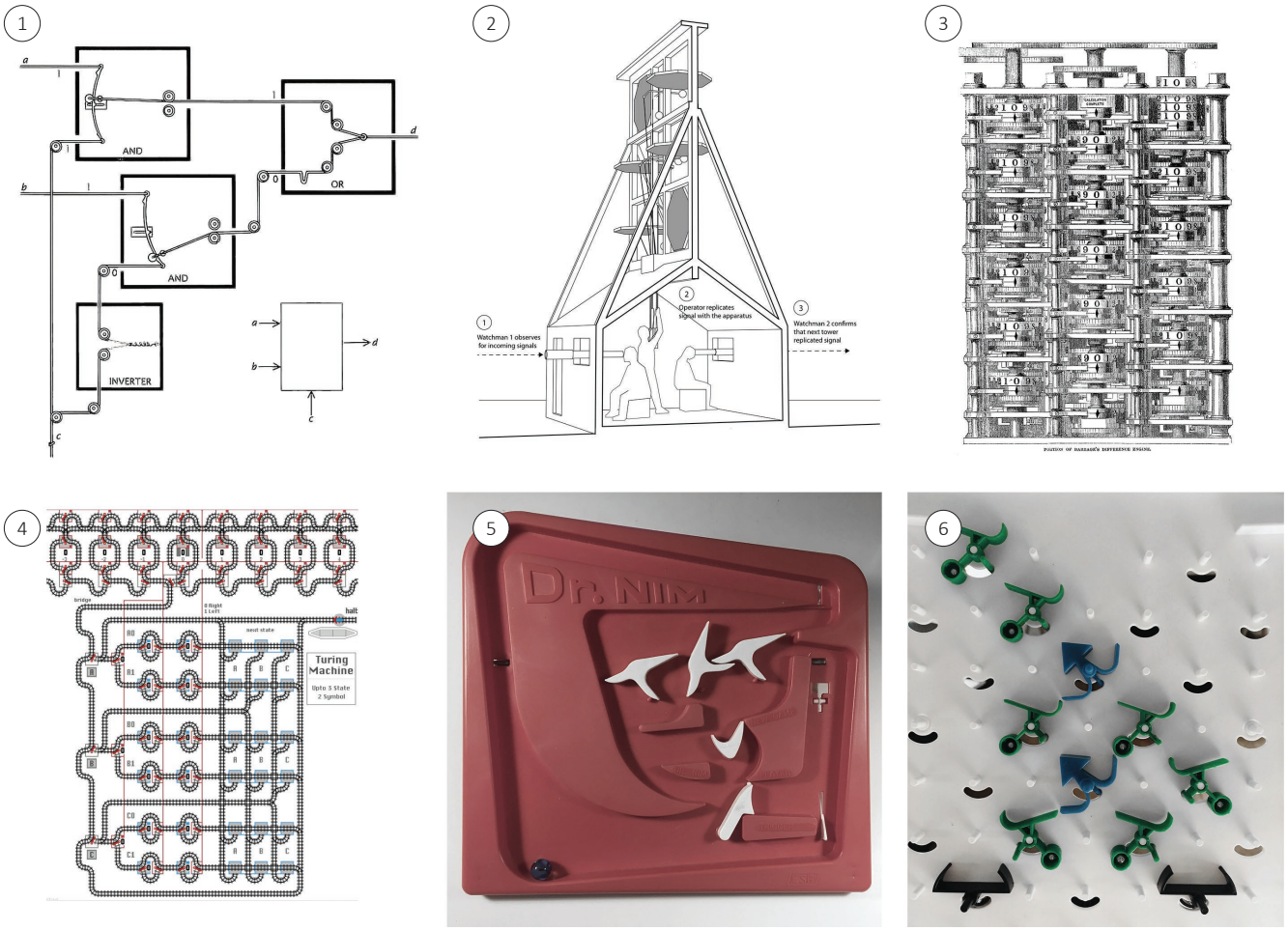

Figure 1. Examples of mechanical computing devices. *1*, An Apraphulian multiplexer (Dewdney 1988). *2*, Optical telegraph building that changes the configuration of its panels to encode 6-bit characters. *3*, The Babbage differential engine. *4*, Turing Trains, implementation of the Digi-Comp II ("Computational Train Track Layouts" n.d.). *5*, Dr. Nim marble-powered mechanical computing toy by J. T. Godfrey, 1966. *6*, Turing Tumble marble-powered mechanical computer toy by Paul Boswell 2018.

This essay describes the development of a pedagogical approach in a project-based seminar and in a graduate-level design studio at the University of North Carolina at Charlotte (UNCC) that aims to engage students in principles of mechanical computation and to develop critical thinking on how information and computation can be manifested and performed tectonically, through architecture. The two courses pose the question: What if the built environment could compute? Can we, and if so, how should we, design environments that compute? Selected student projects from both courses are presented and discussed.

FROM MECHANICAL COMPUTATION TO ARCHITECTURAL COMPUTATION

Traces of the history of computing and informatics can extend as far as the first bonfire relay telecommunication systems in the 12th century BC in Greece that used a binary logic to encode and transmit simple messages. The first working digital computer ever constructed is thought to be an ingenious arrangement of ropes and pulleys in the jungle of Apraphul in New Guinea dating back to 850 AD.[9] Apraphulians used a binary system by using positions of ropes instead of electric voltages. Other early examples include the 18th'century's internet of optical telegraphy.[10] Optical telegraphs were mechanical devices of architectural scale that could store, transmit and process information through physical form for hundreds of miles distance within a matter of few minutes.[11] Charles Babbage invented the difference engine, the first automatic mechanical calculator. The Tinker Toy Computer, a 1975 mechanical computer that plays tic-tac-toe made by MIT students.[12] In 1994, Chalcraft and Greene, published a paper describing how train sets can be Turing complete[13] and designed railway layouts that simulate the operation of a computing machine.[14] Basically, a railway layout represents a program. Turing trains consist of a set of loops and joints. As the train locomotion moves forward, it opens and closed and switches gates, changing the registries. By configuring railway layouts, any program and any problem can be computed. Digi-Comp II, a marble computer designed by John T. Godfrey in 1965, was an automatic binary digital mechanical

computer capable of performing basic arithmetic operations such as addition, subtraction, multiplication and division. By rolling marbles through gravity through latches and gates. The configuration of the gates creates registries and the consecutive passes of marble spheres changes the states of the logic gates. One year later, Godfrey designed Dr. Nim, a plastic digital mechanical computer toy that was published by EST Inc. to be played by one player that was claimed to be unbeatable. Like the Digi Comp II, Dr. Nim consisted of flip-flops (mechanical latches that story memory by their state) that, when combined in cascades, determine the path of a marble. As the marble moves it changes the state of the flip-flops and concurrently the state of the flip-flops determines its path. In 2018, a recent mechanical computer is Turing Tumble which teaches basics of computing. Other examples include fluidic logic: logic circuits with air bubbles.[15] Mechanical computers can be created by linkages. An MIT student developed a half binary adder made from folding linkages[16] Most recently, mechanical computing automata can be designed to create 3D-printed intelligent digital mechanical meta-materials.[17] By 3D printing a cell-structure consisting of logic gates, materials can be designed so that they compute and control their shapes based on their deformation due to applied loads. Despite the physicality of computing, mechanical computation is absent from any discourse in architectural education.

TWO PEDAGOGICAL EXPERIMENTS

The intellectual challenge for students was straightforward. If we can create computation by trains, marbles, linkages, ropes and pulleys, fluids and bubbles, and mechanical latches, then why can't we create computation with architecture, using as energy sources things such as flow of people, flow of tidal forces, wind, traffic of vehicles, and in general any other natural or anthropogenic source of energy? To further explore this issue, I developed two interrelated design assignments. The first, part of an ongoing project-based seminar currently taught in Fall 2018 at UNCC, explored this question practically at an object-scale. The second, a 2nd year M.Arch. graduate diploma design studio also taught at UNCC, explored this question speculatively at a landscape-scale. In what follows, I describe the general pedagogy in each approach and how these approaches were interpreted in selected representative student projects. The description of the two courses is not comparative (many of the pedagogical concepts being explored are still under development) but rather it aims to complement thoughts, ideas, and experiences that add to each other. Since the experiment is ongoing I claim no thorough conclusions but rather objectives, observations and personal reflections.

EXPERIMENT 1: COMPUTING OBJECTS

The first pedagogical exploration is an assignment, part of a design seminar on connective environments taught the School of Architecture at UNCC during fall 2018. Participating students were graduates and came from Architecture, Computer Science, and Software Information Systems backgrounds. The seminar challenged students to imagine the built environment as a medium to connect humans experientially, and to design a project that supported their thesis. Students were presented with the following premise:

> What if the built environment could intelligently sense and mediate our interactions? As we move from the internet of bits to the internet of things, the boundaries between the physical and the digital blur. Yet, designing the built environment as an enabling medium is more than simply retrofitting it with sensors. It requires conceptualizing novel affordances and engineering designs that can successfully implement them. This course introduces design, prototyping, programming, fabrication, and evaluation of interactive objects, systems, and interfaces that enable interactions with and through the built environment. Through a project-based approach, students explore how information technology, human behavior, and material or physical constraints, enable novel affordances, and how these affordances drive engineering design decisions for closing the loop between information and action. Project topics include: connected interactive furniture, human-building interfaces, interactive structures, programmable assemblies, augmented materials, mechanically or digitally computing artifacts.

As students came from various backgrounds, the seminar surveyed foundational skills in designing, prototyping, and programming of interactive physical computing systems without covering them in depth. Students learned how to conceptualize, present, and critique designs in a studio format; how to design, prototype, assemble, program, and assess interactive physical or mechanical systems; how to review state of the art literature in HCI/HBI and how to write a scholarly paper that contextualizes their work in relation to relevant literature. The seminar put an emphasis on critical thinking and on tradeoffs between technical complexity, design decisions, functionality, and quality of craft, when humans are part of the loop. Since the pedagogy challenged students to perceive the built environment not only as a host of computing hardware but also as a potential medium of computation, before introducing digital electronics, the seminar first introduced principles of mechanical computation through an assignment. The objective of the assignment was to develop a critical perspective to digital electronics. The description of the assignment follows.

MECHANICALLY COMPUTING ARTIFACTS

Working in teams of 2-3 persons, students were asked to design and create a mechanically computing device that can interact with a user using a physical medium of their choice. In designing their devices, students were asked to specify what the device does, how it interacts with a user, and how it uses/stores energy to perform computation. In doing so, students had to first understand the concept of a *switch*, a *relay*, and a *gate*. Students were presented with the following premise:

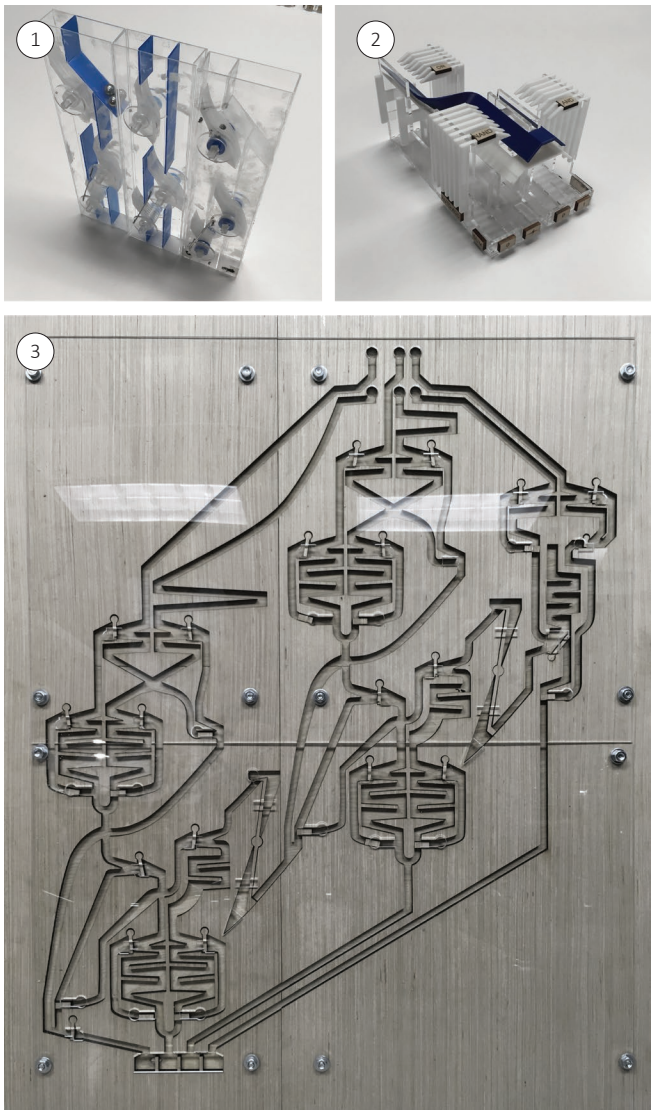

Figure 2. Student projects for fully functional mechanical binary adding machines. The machines can add two 3-bit binary numbers and consist of a series of logic gates. The logic gates consist of flip-flops that change their binary state as marbles pass through them and at the same time control the paths of the marbles. Each project invented a different way to create logic gates, use energy, and represent inputs and outputs.

Is it possible to design a physical mechanism such that it tells us "yes" for some input configurations while "no" for others? Suppose input configurations to be combinations of simple questions that can have a "yes" or "no" state. If such mechanism existed and its inputs were configured by some properties of the environment while its outputs used energy from the environment to configure some other properties of it, then the system would be able to make decisions based on environmental changes, and interact logically with the environment, utilizing natural energy.

Consider each input as a switch with two states. On one hand, the state of a switch can control flow of energy; on the other hand, flow of energy can control the state of a switch. By using the flow of energy that is controlled by one switch to control another switch, cascades of switches can be created that work as relays. Depending on the configuration of the states of two linked switches, a relay can be forward, meaning that a change in the state of the input switch causes a similar change in the state of the output switch, or reverse, meaning that a change in the state of the input switch causes an opposite change in the state of the output switch. By combining forward and backward relays, logic gates can be created, which are the fundamental blocks for building logic circuits. Logic gates take two inputs and produce an output.

APPROACH

The total duration of the assignment was three weeks. Teams began by studying principles of Boolean logic, logic circuits, and by researching historical and contemporary examples of mechanical computation. Next, they developed design ideas for a switch, a relay, and the three basic logic gates (AND, OR, and NOT). Next, each team developed functional prototypes of the three basic logic gates using their relays and a material and fabrication method of their choice. Finally, they combined the logic gates to design and create the more complex mechanically computing machine. In order to narrow down the design exploration given the limited time scope of the assignment, teams were asked to design and prototype a binary adding machine using gravity and marbles as a source of energy (a binary adding machine can be constructed by combining XOR and AND gates).

In designing their relays, teams were asked to consider the following questions: How does your relay harvest, store, and release energy in order to change another relay? Does the output of one gate have the same format as the input? How can multiple relays connect in a cascade such that when one relay changes its state, it triggers its connecting relay to change its state as well? Examples of considered energy forms included mechanical movement, fluidic movement, wind flow, vehicle traffic, electrical current, sound, and pressure.

Three teams developed three different designs shown in figure 2. One team developed a half-adder unit (the basic module that comprises a binary adding machine) using a design strategy for flip-flops similar to the Digi-Comp II. One interesting aspect of this design is that the type of the inputs is different than the type of the outputs. While inputs are configured by the positions of the flip-flops of each unit, the outputs are determined by the positions of the exiting marbles (figs. 2:1 and 3:1). Another team developed a machine consisting of cubic modules. Instead of configuring flip-flops, this machine is programmed by inserting a 3D

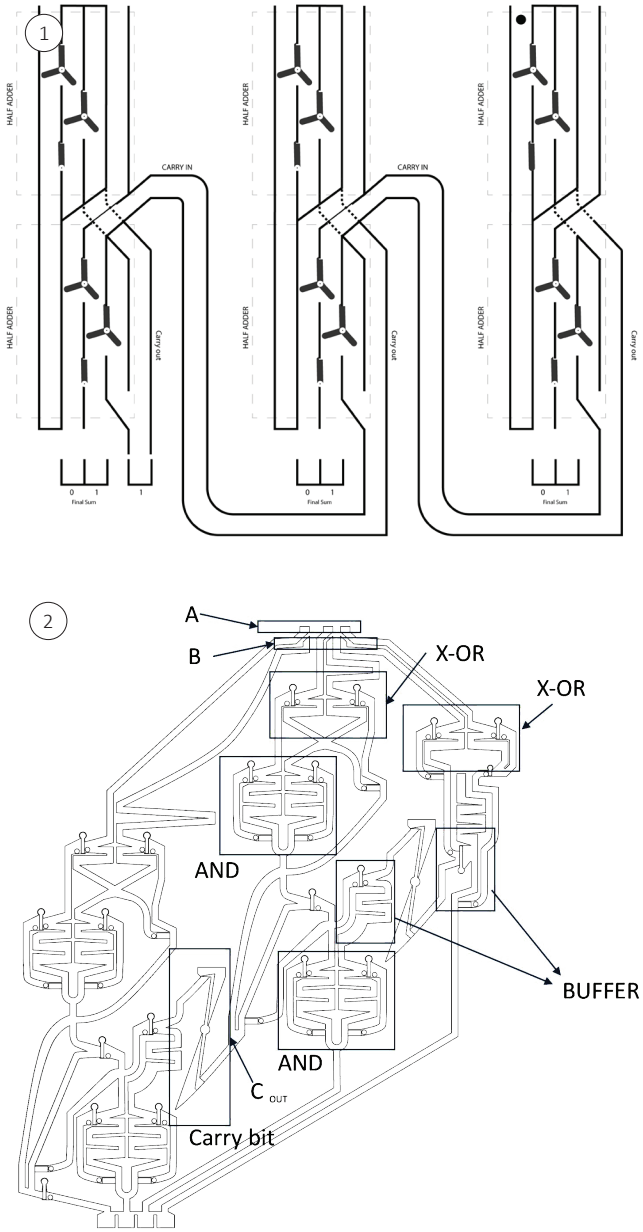

Figure 3: Schematic diagram showing the connection of the half-adder modules in two student projects. Each half-adder module consists of a series of logic gates. *1*, flow of computation goes from right to left: the carry-out marble from a module becomes the carry-in marble to the next module. *2*, flow of computation goes from top to bottom.

printed piece, the form of which determines the logic circuit in a similar manner that punch cards were used to program early-generation computers (figure 2.2). The third team developed a machine that resembled a vertical labyrinth carved in a planar sheet of plywood (figures 2.3 and 3.2). This machine could not be reprogrammed. To guarantee proper sequence of computational sub-procedures, the students designed zig-zag paths that slowed down the trajectories of some of the marbles (delays).

EXPERIMENT 2: COMPUTING LANDSCAPES

The second pedagogical exploration was a graduate diploma design studio, taught at the School of Architecture at UNCC in spring 2018. The studio explored the concepts of architectural/mechanical computation in large scale, asking: is it possible to design urban-, landscape-, or territorial-scale architecture that computes? While the seminar's assignment focused on object-scales allowing students to study mechanical computation in depth, and test their ideas with functional prototypes, the studio explored the topic speculatively, using simulation and representation methods as a means to explore the cyclic behaviors of projects in time. The studio's objective was to illustrate possibilities and gauge critical thinking about the relationship between mechanical computation, building typology, and architectural program. Using the San Francisco Bay area as a context, students were asked to critically position their work to the following premise:

> As we move towards the future, building and urban environments become more distributed, more interconnected, and more shared […] The ontological distinction between what is a building and what is a computer becomes a major architectural statement: just as Modernism gave form to machine functionality, so today, a Second Modernism must give form to computational logic and social synergy.

The studio's objectives included: implement systems thinking in design and integrate physical or digital simulation methods into design; develop a strong thesis statement and position it critically within the architectural computing discourse; become comfortable working within unknown fields of knowledge; challenge pre-established notions of intelligent environments. While the architectural or landscape machines that students designed had to demonstrate a capacity to compute, aesthetics and interpretation of computation into form, as well as the overall thesis on architectural computation were a significant part of critique.

APPROACH

The total duration of the studio was seventeen weeks. Students were asked to "design something that does something in response to something else" by considering (1) form and organization, (2) feedback loop that connects information to action, (3) context that the system responds to, and (4) behavior and transition of the system in time. The studio adopted an engineering systems perspective, defining a system as a "collection of interrelated components that collectively achieve an objective that cannot be achieved by any subcomponent alone." In developing their projects students were asked to clearly articulate objectives, inputs and outputs for their systems, and develop a simulation method to model their behavior (e.g. what state the system takes given a configuration of its inputs). Since students had to demonstrate with simulation the functionality of their concept, simplification of project to its least necessary components was essential.

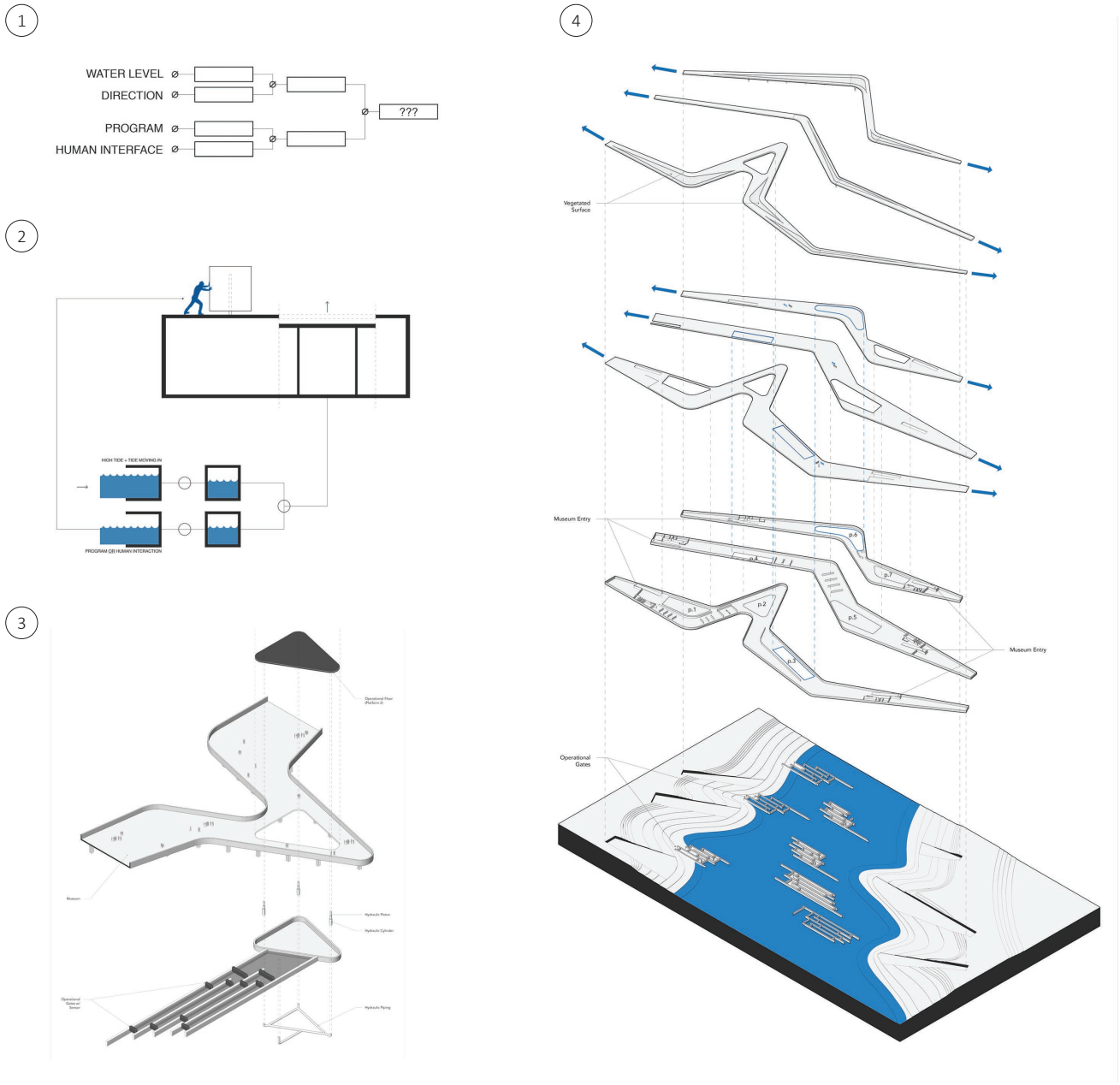

Figure 4. Student design studio project for a hypothetical landscape-scale computer. The landscape computer consists of a series of concatenated logic gates that use tidal energy to change their states in response to anthropogenic and environmental inputs. *1*, Shematic diagram of logic gates and different inputs. *2*, Diagram showing how human input can be used to change the state of a switch that controls the input for a gate. *3 and 4*, Axonometric diagrams showing the different system layers of the project.

The studio approach was organized into three phases, components, organizations, and behaviors, each of which was explored by a combination of digital and physical simulation methods (mechanical models, hydraulic models, system dynamics, multi-agent systems, etc.). These phases are described as follows

Phase 1 focused on components, logic gates, and principles of mechanical computation (weeks 1-5). Students developed typologies of logic gates, logic circuits, and basic interface interactions with inputs and outputs. The objectives were: 1) Articulate design thesis, and clarify scope of project, site, and program. 2) Articulate project as a system, determine goal, component requirements, interactions, input/output variables, users, time scope and anticipated behavior. 3) Familiarize with fundamentals of system design, control feedback loops, causal loop analysis, and dynamics, and how these apply to a project. 5) Understand logic gates, and translate them in physical, sculptural, urban,

Figure 5. Student design studio project for a hypothetical landscape-scale computer. Physical model of the project.

landscape, mechanical, or architectural terms. 6) Identify the main control loop of a project (an if-else logical statement with feedback) and translate it into simple tasks that can be addressed by a design strategy. Critique was based on: Clarity of thesis statement; Concept for project; System analysis of a goal for a project, and key components; Feedback loop analysis; Physical models of logic gates and architectural automata; Site analysis & model; Conceptual preliminary design of project.

Phase 2 focused on system organization and site implementation (weeks 6-10). Students investigated how their components could arrange into larger organization patterns. The objective of the second phase was to implement the systems concept and components developed in the first part to a site and project. Moreover, to explore a simulation method for studying the behavior of a system in time (physical, mechanical, natural, or computational) that can replicate the behavior of a system in time as its input parameters change. Critique was based on: Schematic design of project and an analysis of its organization into parts, interactions, and behavior; Physical/digital simulation model; Physical model of its form/structure; Control loop design.

Phase 3 focused on system simulation, scenario analysis, and project development (weeks 11-16). Students simulated the behavior of their systems in time and mapped the key states that their system takes. The objective of the third phase was to analyze the system's cycle in time and map its states as well as the transition from one state to the other.

Student projects ranged widely. Some projects developed landscapes that used the Bay area's tidal forces as a source of energy. Other projects designed shared mobility systems using flows of cars to and from parking stations as a medium to compute. Other projects designed public markets that used interaction between buyers and sellers as the medium to compute. Figures 4 and 5 show a projected that designed a landscape park that used tidal forces and water gates to change its program based on flow of people and environmental changes. The student described their project as follows.

> Shifting Shorelines: Changing Contours within the Landscape: The San Francisco Bay is facing many regional challenges regarding climate change and natural hazards. This project consists of animating landscape systems with tidal change and human activity to manifest dynamic transformations of coastal shorelines. The project seeks

to highlight the issues the Bay is facing by magnifying the changes occurring within the natural landscape and its effect on the surrounding context. This operational museum will further educate visitors on these changes through direct engagement with building elements and the landscape system.

DISCUSSION: LEARNING COMPUTING WITH HANDS

In discussing teaching of computer sciences to architects, Negroponte emphasized the importance for architecture students to learn computing tactilely with their hands.[18] The combination of the Boolean logic truth tables and the hands-on assignments, aimed to give students a different perspective on what computation is. In the words of the students from the assignment on Computing Objects:

> Survey Question (author): In the first part of the class we focus on computation and information theory not digitally but mechanically. In what ways does this approach affect or challenge your understanding or existing preconceptions of computation, specifically in relation to architecture?
>
> Student A: At first, I didn't know computation from this deep perspective. Now I really understand the vastness of computation. The first 2 projects were quite different and many aspects and quite similar many. Both of them were enjoyable. I really liked the relation of computation and architecture in the last project.
>
> Student B: Computation is always thought to be this mystic and difficult field to get into. The Mechanical computation, certainly helped in demystifying it and helped in understanding that static structure can do complex computations. The concept of space, time, form, composition, functionality, etc which are central to any architectural project were applicable to the design and understanding of mechanical systems and for making a mechanical adder.
>
> Student C: It is definitely a side I have yet to experience in architecture. Most of my experience is purely digital and theoretical, so applying it to real life is an interesting challenge.
>
> Student D: Specifically, being a CS student, the concept of computing using physical media has not only been a complete 180 in terms of approach but also has made me broaden my understanding of computation capabilities overall.

The purpose of this pedagogy is, of course, not to suggest that computing in architecture should be mechanical as opposed to digital but rather to engage students in thinking that computation is not an abstract notion but rather something that can be designed not only symbolically but also physically. As such, its connection to architecture may extend beyond that of a host of computing to that of a medium of computing. As this pedagogical investigation is still in progress, it is early to provide comprehensive conclusions. This discourse, I believe, is valuable within the context of architectural computing education. As computing and informatics become inseparable aspects of social life, their academic disciplines become increasingly more influential on the built environment. In contrast, a reluctance of Architecture to embrace these changes critically, may slowly push the discipline out of its own domain. Independently of the degree of elaboration of each project, all student investigations provided a tactile perspective on the mechanics of computation that helped students not only to deepen their understanding on the topic but most importantly to challenge their preconceptions about what intelligent architecture is and about what (if any) the boundary between architecture and computation might be.

ACKNOWLEDGEMENTS

The work of the following students is shown (listed in order of appearance): Seyedehsan Aboutorabi, Gavin Reeb and Saquib Sarwar (collaborative); Manoj Deshpande, Rachel Lutes and Carlos Martinez (collaborative); Nicklaus Williams and Sri Yeswanth Tadimalla (collaborative). Nicole Peterson (individual).

ENDNOTES

1 Nicholas Negroponte, *Soft Architecture Machines*. (Cambridge, MA: The MIT Press, 1975).

2 George Stiny and James Gips, "Shape Grammars and the Generative Specification of Painting and Sculpture," *Information Processing* 71 (1972): 1460–65.

3 William J. Mitchell, *The Logic of Architecture: Design, Computation, and Cognition* (Cambridge, MA: The MIT Press, 1989).

4 Noam Chomsky, *Syntactic Structures Janua Linguarum NR 4* (The Hague: Mouton, 1964).

5 Richard A. Bolt, "'Put-That-There': Voice and Gesture at the Graphics Interface," *ACM SIGGRAPH Computer Graphics* 14, no. 3 (1980): 262–270, doi:10.1145/965105.807503.

6 Mark Weiser, "The Computer for the 21st Century," *Scientific American* 265, no. 3 (1991).

7 Dimitris Papanikolaou, "The Future of Information Acquisition is Like Swallowing a Pill," (2013).

8 Mike Kuniavsky, "The Coming Age of Magic - Orange Cone," accessed September 29, 2018. http://www.orangecone.com/archives/2006/10/the_coming_age.html.

9 George Lakoff, "The Contemporary Theory of Metaphor," UC Berekley, 1993. http://www.escholarship.org/uc/item/54g7j6zh.

10 George Lakoff, *Metaphors We Live By* (Chicago: University of Chicago Press, 1980).

11 A. K. Dewdney, "Computer Recreations," *Scientific American* 258, no. 4 (1988): 118–21.

12 Dimitris Papanikolaou, "Choreographies of Information: The Architectural Internet of the Eighteenth Century's Optical Telegraphy," in *New Geographies* 7 (Cambridge, MA: Harvard Graduate School of Design Press, n.d.), 45–55.

13 Gerard J. Holzmann and Björn Pehrson, "The First Data Networks," *Scientific American* 270, no. 1 (1994): 124. doi:10.1038/scientificamerican0194-124.

14 A. K. Dewdney, "A Tinkertoy Computer That Plays Tic-Tac-Toe. (Computer Recreations) (Column)," *Scientific American* 261, no. 4 (1989): 120–23.

15 Adam Chalcraft and Michael Greene, "Train Sets," *Eureka* 53 (1994): 5–12.

16 Brian Hayes, "Trains of Thought," *American Scientist; Research Triangle Park* 95, no. 2 (April 2007): 108.

17 Manu Prakash and Neil Gershenfeld, "Microfluidic Bubble Logic," *Science* 315, no. 5813 (2007): 832. doi:10.1126/science.1136907.

18 Xiaoji Chen, "The Linkage Computer," Xiaoji Chen's Design Weblog," December 13, 2010. http://www.xiaoji-chen.com/2010/the-linkage-computer.

19 Alexandra Ion et al., "Digital Mechanical Metamaterials," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI '17* (New York: ACM, 2017), 977–988. doi:10.1145/3025453.3025624.

20 Negroponte, *Soft Architecture Machines*.