31st Annual International Conference of the IEEE EMBS
Minneapolis, Minnesota, USA, September 2-6, 2009

# Digitally-Bypassed Transducers: Interfacing Digital Mockups to Real-Time Medical Equipment

Scott Sirowy, Tony Givargis, *Member, IEEE*, and Frank Vahid, *Senior Member, IEEE*

***Abstract*—Medical device software is sometimes initially developed by using a PC simulation environment that executes models of both the device and a physiological system, and then later by connecting the actual medical device to a physical mockup of the physiological system. An alternative is to connect the medical device to a *digital mockup* of the physiological system, such that the device believes it is interacting with a physiological system, but in fact all interaction is entirely digital. Developing medical device software by interfacing with a digital mockup enables development without costly or dangerous physical mockups, and enables execution that is faster or slower than real time. We introduce digitally-bypassed transducers, which involve a small amount of hardware and software additions, and which enable interfacing with digital mockups.**

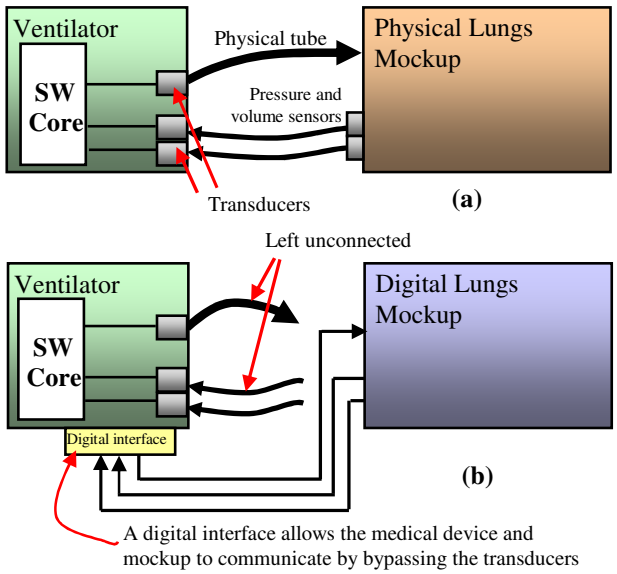
## I. Introduction

Medical device software is developed using several approaches. One approach uses modeling software on a PC. The designer develops models for both the medical device, such as a pacemaker or ventilator, and the physiological system with which the device interacts, such as a heart or lung. Such a modeling software approach supports rapid software changes, supports simulations that execute faster (or slower) than real-time, and avoids potential safety issues that would arise when interacting with a physical system. Device software may be migrated to an actual medical device platform for further development. Later in development, medical device software may be run on the actual medical device and connected to a physical mockup of the physiological system, shown in Figure 1(a). Physical mockups range from simple structures, such as a balloon representing a lung, to computerized mechanical parts that dynamically react [6], that can be set to mimic a range of conditions, and whose internal sensors can be interfaced to a computer for analysis and debugging. One disadvantage of interfacing to off-the-shelf physical mockups is the inability to adapt to new features, especially features not easily mimicked via mechanical means. For example, a future ventilator may sense human-generated nitric oxide concentrations (recently discovered to be significant in respiratory issues [10]) and adapt the output gas mix in response. However, no existing computerized mechanical test lung generates nitric oxide, nor is it clear how to create one.

An alternative is to connect the actual medical device to a digital mockup of the physiological system. A *digital mockup* is a behavioral model that emulates the physical system. In such a case, the medical device software believes it is interacting with a physiological system, but in fact all interaction is through a digital interface, as in Figure 1(b). Digital mockups combine the flexibility and faster-than-real-time execution benefits of PC simulation models with the advantages of developing software on an actual medical device. Digital mockups are also potentially less costly than physical mockups, which can cost tens of thousands of dollars.

We describe *digitally-bypassed transducers,* which enable the interfacing of medical equipment with digital mockups. Digitally-bypassed transducers require only a few minor hardware and software additions, but enable medical software development on actual medical equipment early in the design process, or enable avoidance of costly or dangerous physical mockups. Digitally-bypassed transducers also enable development modes where the medical device and digital mockup can run faster (or slower) than real time. Running faster than real-time might be beneficial when

**Figure 1:** Physical mockup vs. digital mockups: (a) a physical mockup directly connects to a medical device's transducers, (b) a digital mockup uses a digital interface that can bypass transducers.




Manuscript received April 6, 2009. This work was supported in part by the National Science Foundation under Grant CNS-0614957 and by Office of Naval Research under the Contract Number N00014-07-C-0311

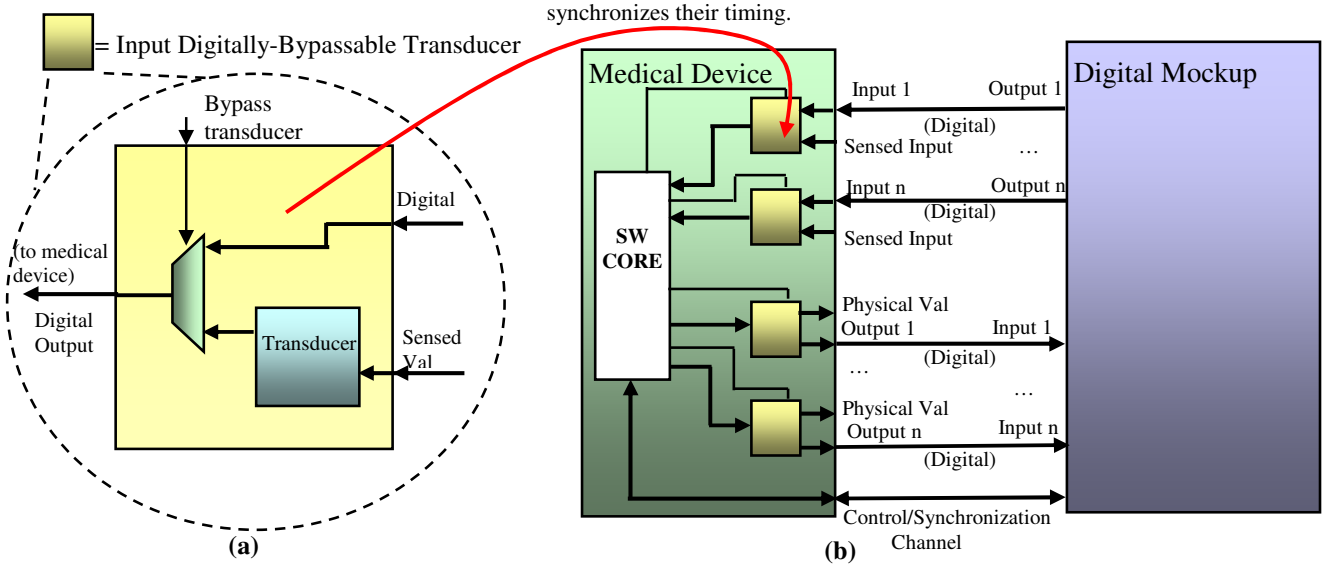
Scott Sirowy is with the Department of Computer Science, University of California, Riverside, CA 92507 USA (e-mail: ssirowy@cs.ucr.edu).

Tony Givargis is with the Department of Computer Science, University of California, Irvine, CA 92697, USA (e-mail: givargis@uci.edu).

Frank Vahid is with the Department of Computer Science, University of California, Riverside, CA, 92507 USA, (e-mail: vahid@cs.ucr.edu).

**Figure 2:** Digitally bypassing the transducers: **(a)** An input digitally-bypassable transducer. **(b)** The connections required to enable connectivity with a digital mockup. The synchronization and control channel enables digital connections between the device and mockup, as well as synchronizes their timing.




testing multiple versions of an algorithm, or when a designer needs to simulate a night's worth of breathing in minutes. Digitally-bypassed transducers also enable software development when transducers are not yet present, or when mechanized values (like nitric oxide concentrations) are difficult to create. Digitally-bypassed transducers are already used in some domains, e.g., space satellite software development, but a standard methodology has yet to evolve.

## II. Related Work

There has been some work developing digital mockups that interface to medical devices. Pimentel and Tirat-Gefen [8][9] developed real-time digital mockups that interfaced to medical devices by connecting symmetric D/A (digital-to-analog) and A/D (analog-to-digital) cards to each side. Our work focuses on modest modifications to the medical device hardware and software such that a digital mockup could be connected directly. Our approach still allows the addition of D/A and A/D attachments, but with the added advantage of allowing a designer to completely stay in the digital domain, and to accommodate situations where D/A or A/D conversions are complex (e.g., in the case of gas generation or sensing). Other researchers have developed real-time physiological models [1], with a focus on describing the necessary architectures to achieve real-time.

Several research efforts have emphasized creating and cataloging detailed physiological models [2][7][11]. Those models are targeted for PC-based simulation, yet could be used as a basis for digital mockups. Further, many physiological models are highly complex, often requiring hours or days to simulate a few seconds [5]. Our initial focus is on real-time digital mockups.

There has been much work in the domain of synchronization mechanisms for distributed systems. Lamport [4] describes methods to order events in a distributed system. Kopetz [3] also specifies clock synchronization methods, but describes techniques used for a more general network topology. In contrast, our system consists of only two directly connected components, and thus is a simpler synchronization problem because uncertainties in a general network need not be considered.

## III. Digital Mockup/Medical Device Interfacing

### A. *Digital Mockup Interface*

There are several options for interfacing digital mockups with medical devices. One option is to add A/D and D/A interface cards and connect the two devices, as in [8][9]. This approach has the advantage that the medical device hardware and software do not have to be changed. However, the approach is limited to certain types of analog signals. Another option is to design two versions of the medical device, with one version interacting solely with a digital model. Designing two devices could be expensive, and doesn't offer the flexibility of using the same device on real physiological systems.

Another option is to augment the medical device with digitally-bypassed transducers. A *digitally-bypassed transducer* replaces the traditional transducer in a medical device and allows for bypassing the transducer input/output when connected to a digital mockup. Figure 2(a) shows the internals of a digitally-bypassed transducer that would be connected to an input on the medical device. The digitally-bypassed transducer has two inputs, a sensed value from either a real physiological system or a physical mockup, and a digital value coming from a digital mockup. The sensed value is converted to a digitized value through a standard transducer and the digital mockup input passes through, bypassing the transducer. The transducer output and the digital input feed into a multiplexor and the output is selected based on the connectivity of a digital mockup. A similar digitally-bypassed transducer is also required to be

connected internally to the output of a medical device. That digitally-bypassed transducer accepts one digital input from the medical device software core, and output the passed through digital value, and the output from a traditional transducer. In this case, no multiplexor is needed. Figure 2(b) shows a medical device designed with multiple digitally-bypassed transducers.

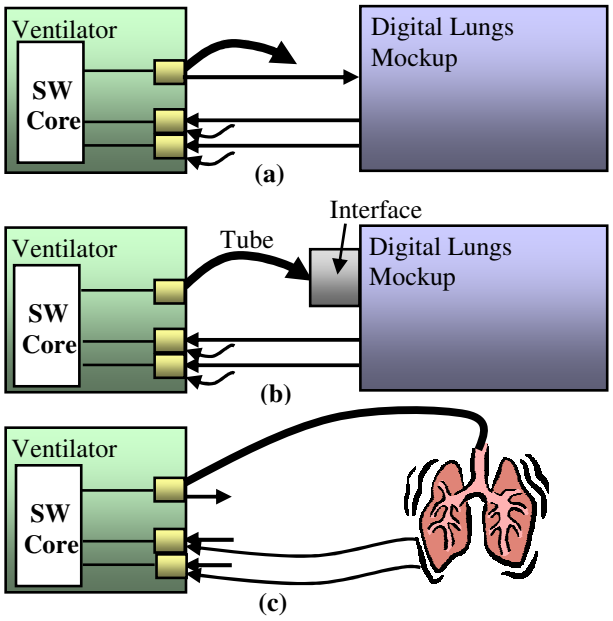
Digitally-bypassed transducers still allow the designer to connect interface D/A and A/D cards to both devices, but with the added flexibility of being able to operate digitally. For many medical devices, digitally-bypassed transducers would require only tens of dollars of extra cost to add serial digital connectivity. Furthermore, when debugging medical device software in the field, the digitally-bypassed transducers would provide a debug portal to allow a designer to connect a digital mockup as needed.

Digitally-bypassed transducers potentially enable many modes of medical device software development. For instance, in the early stages of medical device development, a designer could connect to a digital mockup via the digitally-bypassed transducers, as shown in Figure 3(a). Further development would lead to scenarios where some real connections were made through medical device outputs and interface cards, like Figure 3(b). Finally, the same device could be connected fully to a real physiological system, as in Figure 3(c).

### B. *Digital Mockup Synchronization*

Real-time medical devices also require synchronization mechanisms to correctly communicate with digital mockups since there is a possibility the digital mockup executes faster or slower than real time. We address this issue by adding a dedicated digital channel between the medical device and digital mockup. The dedicated channel is used to exchange connectivity information, sampling rates, and mockup execution speeds in a protocol called *rate synchronization.* On initial connection, the medical device and digital mockup perform a simple handshake to assert the digital connectivity. The assertion in turn signals the digitally-bypassed transducers to pass through the correct digital values. The medical device also sends its own execution speed. After connection, the medical device sends its required sampling rate. Sending the medical device sampling rate is necessary because the digital mockup may be outputting samples at a lesser rate in order to maintain real-time execution. For example, a medical device might sample at 1000 samples a second, but a digital mockup is initially configured to simulate at a granularity of a hundredth of a second, in which case the digital mockup will generate one hundred samples per second. There are several options for the two devices in such a case. One option is to decrease the medical device's sampling rate. Another option is to have the digital mockup simulate at a finer granularity to meet the medical device's sampling requirements. The latter option has the advantage that the medical device software doesn't have to give up accuracy by decreasing its sample rate. Finally, the medical

**Figure 3:** Digitally-bypassed transducers enable many modes of execution of the medical device software: (a) all digitally-bypassed transducers option, (b) combined digital and transducers, (c) full interaction with physical system.




device optionally instructs the digital mockup how fast to execute. Real time execution is initially assumed, but communicating the execution speed gives the two devices the option to simulate faster or slower than real time. Of course, the digital mockup has a maximum number of samples it can generate, so the digital mockup is responsible for informing the medical device whether or not it can meet the execution speed and sampling rate the medical device requests. In the event that the digital mockup cannot maintain the sampling rate, the digital mockup informs the medical device of the maximum execution speed it *can* maintain while still generating samples at the desired medical device sampling rate. The devices agree on either the requested execution speed or the digital mockup maximum speed, and execution begins.

We considered other synchronization options, but did not find the others to work as well as *rate synchronization*. One option was to have the digital mockup and medical device work in lock step, each computing and communicating one sample before progressing to the next time step. Such an approach enables fine-grained levels of granularity, and ensures the two devices are always synchronized. But, lock-stepped control would require a more complicated communication protocol, and fine-grained debug capability could be attained by synchronizing the two devices to run slower than real time. Another option was to have the devices send time-stamped values to each other. The digitally-bypassed transducers would be responsible for monitoring the time stamp and feeding the data to the medical device software at the right time, possibly interpolating data when the time stamps weren't at the rate

the medical device software required. However, the time-stamped approach added unnecessary complexity to the digitally-bypassed transducers and the synchronization communication layer, and did not seem necessary due to the devices having small communication time.

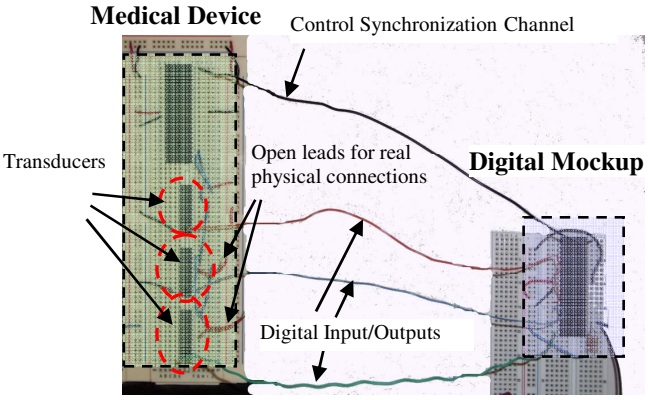
## IV. Experiments

We conducted several experiments to test the digitally-bypassable transducers connectivity and synchronization mechanisms. We also tested scenarios where transducers weren't available or operating properly. For our experiments, we designed two medical device and digital mockup prototype combinations. One model was a pacemaker and a beating heart. The other prototype modeled a ventilator and lung and was based off a model obtained from the NSR Physiome Project [7]. To closely model existing pacemakers and ventilators, each medical device was initially run in real-time and to sample at 1 KHz frequency. Figure 4 illustrates one of our prototype setups for a pacemaker and digital heart mockup. We implemented the medical device software and the digital mockup on off-the-shelf microcontrollers. We created the digitally-bypassable transducers by using the A/D functionality and bypass software on smaller off-the-shelf microcontrollers. Figure 4(a) shows the internals of the pacemaker. The larger processor is implementing the medical device software, and the three smaller processors modeling the digitally-bypassed transducers. In the figure, the lower two digitally-bypassed transducers have two inputs, a connected digital input, and an open analog lead. The digital inputs are connected to the digital heart mockup in Figure 4(b) by means of one-way serial connections.

We tested the device's connection mechanism by first running the pacemaker and the digital heart mockup separately. The digital heart mockup was initially configured to simulate a normal beating heart, and the pacemaker executed assuming its inputs were coming from the analog leads. The digital heart mockup and the pacemaker were both running in real time. Since the pacemaker and heart were not digitally connected, the pacemaker noticed no activity on the analog leads, and thus output periodic paces to the unconnected heart. When connected, the medical device successfully connected to the digital heart mockup, and stopped pacing the heart since the medical device was now receiving a regular heart beat.

We tested the medical device's synchronization mechanisms by configuring the granularity of the digital mockup's simulated time step. We first adjusted the granularity of the digital mockup time step to generate samples at twice the rate the medical device sampled. In this case, the medical device maintained its own 1 KHz sampling frequency, and sampled every other sample correctly. Next, we adjusted the digital mockups simulated time step so it would only generate one hundred samples a second. On connection, the medical device correctly instructed the digital mockup to increase its sample generation frequency.

**Figure 4:** Experimental setup. The mockups have several one-way serial connections, including a control/synchronization channel.




To test simulations that could run faster or slower than real time, we configured the medical device to run faster than real time by adjusting internal timer settings. For the pacemaker/digital heart example, we were able to speed up the execution by over 10X faster than real-time, and still achieve correct simulation results. We further tested the time synchronization mechanisms by slowing down the medical device to correctly operate on a heart that beats once a minute.

There also may be situations in early software development where a transducer isn't available or isn't operating correctly. We tested such a scenario by modeling a digitally-bypassed transducer to only pass through the digital data. The pacemaker software still worked properly as long as the digital heart mockup was properly connected.

## References


[1] Botros, N., Akaaboune, M., Alghazo, J., and Alhreish, M. 2000. Hardware Realization of Biological Mechanisms Using VHDL and FPGAs.

[2] IUPS Physiome Project. http://www.physiome.org.nz/

[3] Kopetz, H. and Ochsenreiter, W. 1987. Clock synchronization in distributed real-time systems. *IEEE Trans. Comput.* 36, 8 (Aug. 1987), 933-940.

[4] Lamport, L. 1978. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM* 21, 7 (Jul. 1978), 558-56

[5] MedGadget Internet Journal. 2008. Supercomputer Creates Most Advanced Heart Model. http://medgadget.com/archives/2008/01/worlds_biggest_heart_model_simulated_1.html.

[6] Michigan Instruments. Training and Test Lung (TTL) and PneuView software, http://www.michiganinstruments.com/resp-ttl.htm, 2009.

[7] NSR Physiome Project. http://nsr.bioeng.washington.edu

[8] Pimentel, J. and Tirat-Gefen, Y. 2006. Hardware Acceleration for Real Time Simulation of Physiological Systems. EMBS. pp 218-223.

[9] Pimentel, J. and Tirat-Gefen, Y. 2006. Real-Time Simulation of Physiological Systems. Proceedings of the IEEE 32nd Annual Northeast Bioengineering Conference. pg. 159-160.

[10] Shin, H. and George, S. Impact of Axial Diffusion on Nitric Oxide Exchange in the Lungs. Journal of Applied Physiology. 2002.

[11] Tawhai, M., and Ben-Tal, A. 2004. Multiscale Modeling for the Lung Physiome. Cardiovascular Engineering: An International Journal, Vol. 4, No. 1., March 2004. pp 19-26.