Chapter 4

Ultra-dense crossbar molecular memory circuits

4.1 Introduction

The past four decades have witnessed extraordinary advances in computation that have revolutionized the way people communicate and process information. Sustained progress in this field has largely been driven by the consistent reduction of silicon-based microelectronic-device dimensions with accompanying increases in device density. This guiding principle of advancing computational technology through regular increases in device integration is widely referred to as 'scaled CMOS' after the ubiquitous complementary metal-oxide-semiconductor integrated circuit, and has driven the now famous exponential increase in computational performance as measured by any number of metrics (*e.g.*, speed, size, cost, power consumption, etc.). However, there are strong indications that continued scaling of conventional CMOS technology may falter in the near future due to physical and (perhaps more importantly) financial considerations[1]. This has led to a growing consensus within the semiconductor industry that continued improvements in information processing technology will likely occur through the development of alternative materials, patterning methods, and architectures[2, 3] that can be

integrated into the well-established silicon CMOS infrastructure in the near term[1, 4] while being scalable in the long term. Ideally, any new technology should be compatible with conventional CMOS to bridge computational requirements during its assimilation period while having the intrinsic potential to continue the exponential pace of computational performance once traditionally-scaled CMOS comes to an end.

These considerations have brought a great deal of attention to the possibility of engineering molecules for use as the active electronic components in otherwise solid-state computational systems[5]. While the idea of using molecules to mimic traditional computational functions is not new[6], it is only within the past decade that molecules have been integrated into hybrid solid-state/molecular devices to perform the traditional computational functions of rectification[7], storage[8], and logic[9]. Although a complete picture of electronic transport through molecular junctions continues to elude theorists[10, 11], molecules have nonetheless empirically demonstrated their potential for computation. Additionally, a number of methods have been reported for assembling small numbers of nanowire[12, 13] or carbon-nanotube[14, 15] devices. However, while these studies demonstrate individual device scalability, they seldom address issues such as device pitch or density, which are equally important from a technology standpoint.

To that end, the Heath group began a research program focused on the concept of developing an ultra-dense molecular electronic computer architecture where the various computational elements would be tiled together through high-density arrays of nanowires (NWs) (Figure 4-1)[5]. Along with my co-workers, my research in the Heath group has focused on a number of the components shown in Figure 4-1 for realizing this multifunctional computational architecture. These included the development of

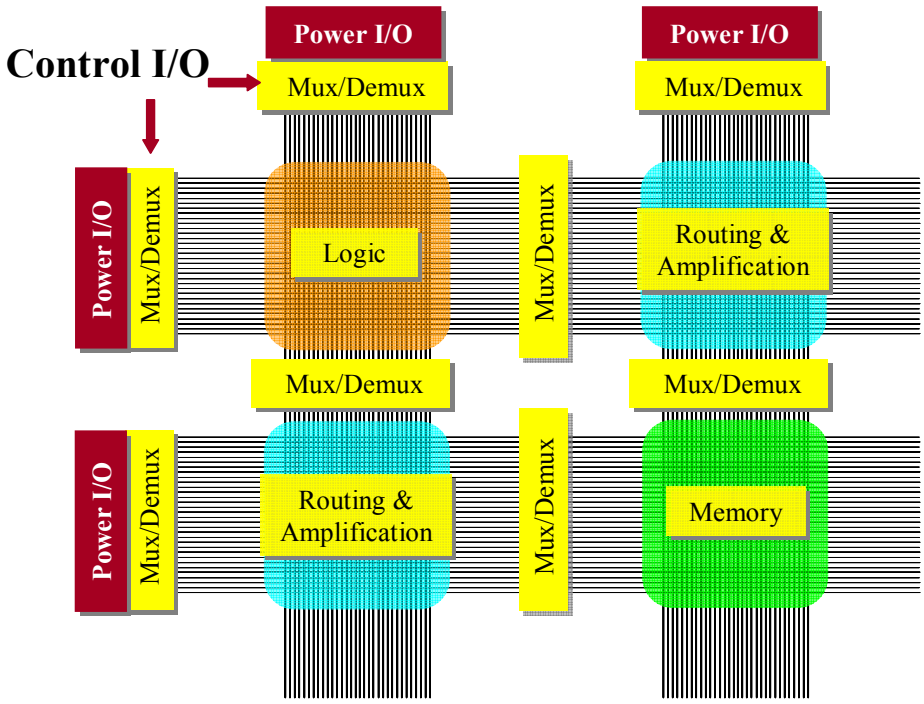


Figure 4-1. Schematic diagram of a nanoelectronic crossbar circuit architecture. The various computational building-blocks are shown tiled together through ultra-dense nanowire arrays. Multiplexers (Mux) and/or demultiplexers (Demux) control signals within the circuit and to outside electronics (power I/O). This structure is both defect-tolerant and amendable to non-lithographic fabrication techniques.

techniques for patterning ultra-high-density arrays of silicon NWs with precisely controlled electronic properties (Chapter 2), and the demonstration of a field-effect transistor (FET)-based demultiplexer capable of bridging from the sub-micrometer length scales of conventional CMOS technology to the nanometer length scales of molecular electronics (Chapter 3). In this chapter, I will discuss the integration of sub-lithographic patterning techniques and [2]rotaxane-based molecular materials for the fabrication of an ultra-dense, error-tolerant, 160,000-bit molecular electronic crossbar memory patterned at a density of 100 gigabits per square centimeter (1×10^{11} bits cm^{-2}). Before describing in detail the fabrication and testing of this memory, I will give a brief introduction to the

rich science underlying the switching mechanism of bistable [2]rotaxane molecules and their integration into high-density crossbar architectures.

4.2 The [2]rotaxane switching cycle

Figure 4-2.A shows the molecular structure of a bistable [2]rotaxane $HRT5^{4+}$ used in the crossbar molecular memory circuits discussed herein. This molecule was synthesized by Dr. Hsian-Rong Tseng of the J. Fraser Stoddart group at UCLA using the techniques of supramolecular[16] template-directed[17, 18] chemical synthesis. [2]Rotaxanes consist of two mechanically-interlocked components: an amphiphilic dumbbell-shaped component and a π-electron-accepting cyclobis(paraquat-*p*-phenylene) ($CBPQT^{4+}$) ring (blue). The dumbbell component features two bulky stoppers (light blue and grey) on either end to prevent the ring from slipping off the [2]rotaxane shaft and to facilitate orientational incorporation into solid-state devices via Langmuir-Blodgett techniques. The $CBPQT^{4+}$ ring can translate along the shaft of the dumbbell-shaped component to sit at one of two π-electron-donating recognition sites: the tetrathiafulvalene (TTF) unit (green) or the 1,5-dioxynapthalene unit (DNP) (red). [2]Rotaxanes have been extensively studied experimentally[19-29] and theoretically[30-33] in a variety of environments to elucidate the physical mechanism underlying their switching behavior, and to quantify the switching kinetics and equilibrium thermodynamics. Under ambient conditions, the $CBPQT^{4+}$ ring preferentially encircles the TTF unit over the DNP unit. For the [2]rotaxane $RTTF^{4+}$ (which is very similar to $HRT5^{4+}$) in acetonitrile solution, this equilibrium is greater than

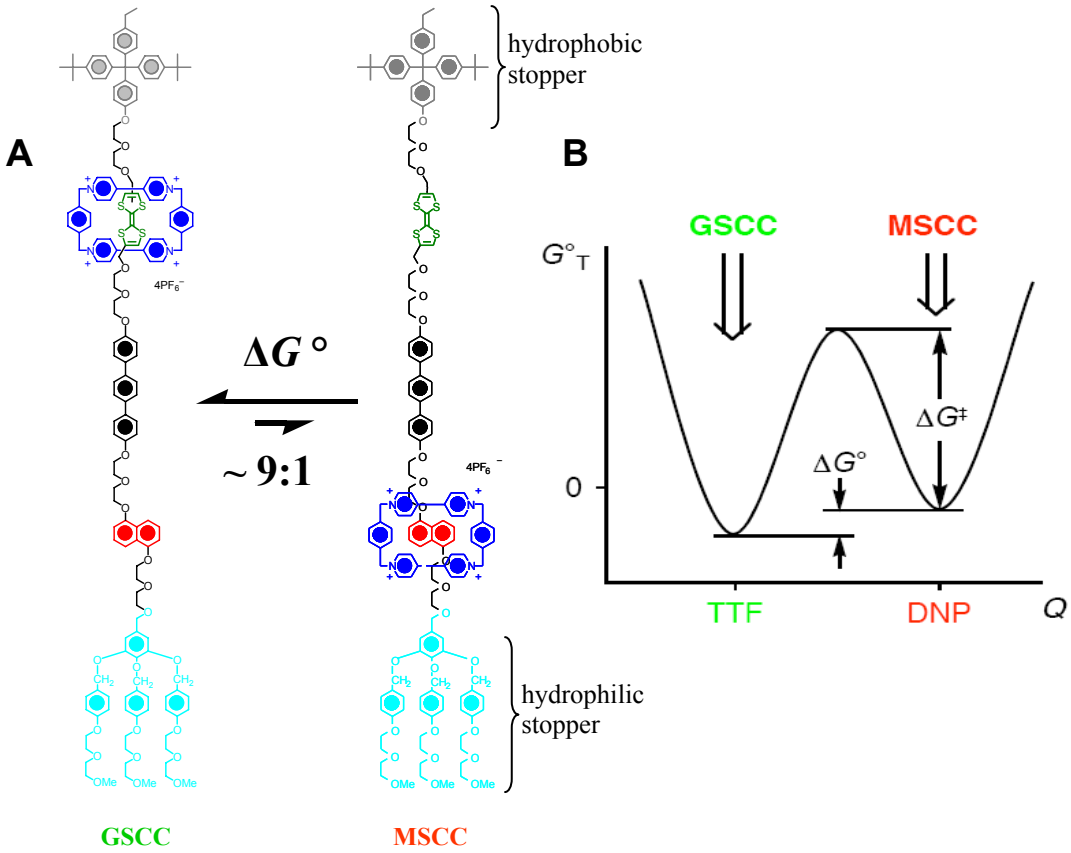


Figure 4-2. Molecular structure and energy diagram of the bistable [2]rotaxane HRT5^{4+}. A. At equilibrium, the ground-state co-conformation (GSCC) is energetically favored over the metastable-state co-conformation (MSCC) by a free energy of $\Delta G°$. This corresponds to a GSCC-to-MSCC distribution of about 9:1. Within a Si/mol/Ti MSTJ (see text), the molecule is oriented with the (light blue) hydrophilic stopper in contact with the Si electrode and the grey hydrophobic stopper in contact with the Ti electrode. **B.** The potential energy landscape revealing the basis of bistability in [2]rotaxane molecular switches is plotted against the reaction coordinate, *Q,* representing translation of the ring from the TTF unit to the DNP unit. The rate of relaxation from the MSCC ('1') state to the GSCC ('0') state depends on the energy barrier, $\Delta G^{‡}$, which increases with the viscosity of the physical environment (*e.g.*, $\Delta G^{‡}_{\text{solid-state}} > \Delta G^{‡}_{\text{solution}}$).

90 percent, and is described by a free energy change of $\Delta G° = +1.6$ kcal/mol (Figure 4-2.B) when the CBPQT^{4+} ring moves from the TTF to the DNP unit[20]. Thus, the co-conformation with the CBPQT^{4+} ring encircling the TTF unit is referred to as the ground-state co-conformation of the molecule. Recent experiments[20] have shown that the ground state equilibrium distribution of these molecules is dominated by molecular structure,

with the physical environment (*e.g.*, solution or solid-state) of the molecular-switch playing only a minor role. Within a silicon/molecule(s)/titanium (Si/mol/Ti) solid-state molecular-switch tunnel junction (MSTJ), the ground-state co-conformation corresponds to the low conductivity or binary '0' state of the molecule.

The universal[19] molecular-mechanical switching mechanism of bistable [2]rotaxanes is shown in Figure 4-3. Starting at the ground-state co-conformation ($CBPQT^{4+}$ ring encircling the TTF unit), the first two oxidation states of the molecule result from sequential oxidations of the TTF unit corresponding to the reaction $TTF^0 \xrightarrow{-e^-} TTF^{\bullet+} \xrightarrow{-e^-} TTF^{2+}$. Upon forming the $TTF^{\bullet+}$ radical cation, coulombic repulsion between the $CBPQT^{4+}$ ring and the $TTF^{\bullet+}$ unit results in translation of the ring from the TTF unit to the DNP unit. This process occurs on a millisecond time scale[27-29], and is believed to convert all of the molecules from their ground-state co-conformation to

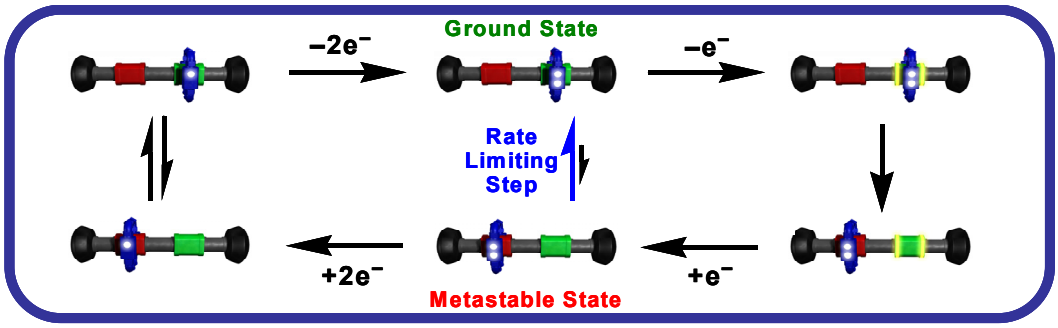


Figure 4-3. Bistable [2]rotaxane switching cycle. Starting from the ground state with the $CBPQT^{4+}$ ring (blue, each white dot corresponding to a +2 charge) encircling the TTF (green) unit and moving clockwise: The TTF unit is oxidized (highlighted now) resulting in translation of the ring from the TTF to DNP (red) unit and formation of the metastable state after the TTF unit regains neutrality. The molecule can relax back to the ground state through the rate-limiting kinetic step, or through the clockwise loop on the left in which the ring is reduced resulting in recovery of the ground state at least one thousand times faster.

a translational isomer with the ring encircling the DNP unit, referred to as the metastable-state co-conformation of the molecule. Within a Si/mol/Ti MSTJ, this co-conformation corresponds to the high conductivity or binary '1' state of the molecule, and is obtained by applying a positive voltage of about 1.5 V across the molecule (with respect to the hydrophilic stopper, or equivalently, silicon electrode; see Figure 4-2.A).

Neutrality is quickly restored to the TTF unit in the absence of an oxidizing potential; however, the $CBPQT^{4+}$ ring continues to encircle the DNP unit for a period of time due the energy barrier of $\Delta G^{\ddagger}$ = 22.5 kcal/mol ($HRT5^{4+}$) shown in Figure 4-2.B. Recovery of the ground- to metastable-state equilibrium distribution (~9:1) is a thermally activated process, and temperature-dependent relaxation-time measurements have been used to understand the kinetics of this relaxation[20, 34]. From a device perspective, relaxation from the metastable-state co-conformation ('1' state) to the ground-state co-conformation ('0' state) corresponds to the volatility or bit-retention time of a Si/mol/Ti MSTJ. For the [2]rotaxane $RTTF^{4+}$ in a 50-μm^2 MSTJ, this relaxation time was measured to be about 58 minutes at room temperature[20].

The ground-state equilibrium distribution with the $CBPQT^{4+}$ ring encircling the TTF unit (> 90%) can be recovered at least 1000 times more quickly (this is a lower limit; the actual value was not obtained experimentally)[25, 35, 36] by electrochemically reducing the two bipyridinium units in the $CBPQT^{4+}$ ring to their radical cations corresponding to the reaction $CBPQT^{4+} \xrightarrow{+2e^-} CBPQT^{\bullet\bullet 2+}$. According to previous investigations[37, 38], the doubly-reduced $CBPQT^{\bullet\bullet 2+}$ ring then loses its affinity for the π-electron-donating DNP recognition site, and the molecule relaxes back to its ground-state co-conformation with the ring encircling the TTF unit before neutrality is restored. In

terms of the solid-state switching mechanism, this corresponds to switching the molecule from its high conductivity '1' state to its low conductivity '0' state, and is accomplished by applying a negative voltage of about 1.5 V to the molecule at the hydrophilic stopper, or equivalently, to the silicon electrode of a Si/mol/Ti MSTJ.

4.3 The crossbar architecture

The crossbar (Figure 4-1) is an attractive architecture for nanoelectronic circuitry for a number of reasons[5, 9, 39]. First, nanoelectronic circuits based on the crossbar structure are tolerant of manufacturing defects[40, 41]. Each device in the crossbar structure can be uniquely addressed by two crossed wires that define the junction. If initial testing reveals that a device is defective; its address can be stored and routed around during future computations. This characteristic becomes increasingly important as electronic devices approach macromolecular dimensions and non-traditional (and imperfect) fabrication methods (*e.g.*, self-assembly) are employed. A proof-of-concept demonstration that robust computation can be obtained from a configurable circuit with defective components was given by Hewlett Packard's defect-tolerant, custom-configurable computing machine, Teramac[42]. The Teramac computer had nearly a quarter million hardware defects, but through the use of testing and configuration algorithms, it could be transformed into a robust computing machine.

Second, the crossbar architecture can be fabricated without using lithographic techniques. This is important because it is doubtful that conventional lithographic

techniques will ever be able to achieve the resolution necessary to cost-effectively fabricate a truly nanoelectronic architecture[1, 43]. Self-assembly and other non-traditional patterning methods typically generate highly regular structures, so they aren't practical for fabricating the arbitrarily complex architectures characteristic of traditional CMOS microelectronics. A crossbar structure, however, consists of only two sets of straight, aligned wires and can be readily fabricated using a variety of non-lithographic techniques. Indeed, several groups have demonstrated methods for assembling nanowires (NWs) into crossbar structures using fluidic alignment[12, 44], Langmuir-Blodgett alignment[45], and imprinting[46-48], and various architectural concepts have been introduced that can take advantage of such circuits[49-52]. However, only the superlattice nanowire pattern transfer (SNAP) method (described in Chapter 2) has been successful in producing NW arrays aligned over the length scales required for large-scale circuitry.

Third, the highly ordered nature of NW arrays has enabled the development of demultiplexing architectures capable of addressing 2^n NWs using order (n) number of control wire pairs (see Chapter 3)[53-55]. These architectures allow the selection of an individual NW from within an array that has been patterned at sub-lithographic density using relatively large wires patterned using traditional lithographic processing. This demonstrates that crossbar architectures can exhibit excellent scaling from the microscale to the nanoscale, in addition to being compatible with standard CMOS microelectronic technology.

Finally, the crossbar architecture is the highest-density two-dimensional circuit for which every device can be independently addressed[56]. Wiring overhead in a crossbar circuit is minimized because the NWs defining a junction are used to both configure and

read the device. This is in contrast to conventional CMOS-based configurable devices that require one set of wires (address lines) to configure the device and another set of wires (data lines) to read it. The ability to independently address every component in the circuit is useful for memory applications, but also enables the circuit to be fully tested for manufacturing defects so these can be routed around during configuration. However, taking advantage of the inherent density afforded by the crossbar architecture requires the development of electrically active thin-film materials that function within the two-terminal junctions of the circuit.

4.4 [2]Rotaxane molecular electronic crossbar circuits

Two-terminal molecular switches, such as [2]rotaxanes, have a number of important advantages (and some disadvantages) over more-developed two-terminal electronic materials such as ferroelectrics[57]. As discussed above, comprehensive experimental and theoretical investigations have verified the distinctly molecular basis of [2]rotaxane electrical switching. Thus, devices based on these switches should scale to macromolecular dimensions without a significant change in the switching characteristics. Solid-state-based switching materials are unlikely to exhibit similar scaling since they arise from inherently bulk properties. Two-terminal devices based on these materials are switched by applying a field across the junction to polarize crystallographic domains. The hysteresis of this polarization disappears, and the device no longer switches, once the

junctions are made smaller than the domain size of the material. For ferromagnetic materials, this is referred to as the superparamagnetic limit[58].

Another advantage of [2]rotaxane-based devices is that the voltage at which the molecular switches are opened or closed is very stable. This is because the switching mechanism is based on an electrochemical process in which current has to flow in or out of molecular orbitals before the molecule isomerizes to its high- or low-resistance co-conformations. In contrast, domain polarization is driven by nucleation events, and so is intrinsically statistical. A consequence of this nucleation-driven switching mechanism is that the field required to switch solid-state-based devices can fluctuate randomly from one device to the next within a crossbar circuit, or even from one switching cycle to the next within a single device.

In a crossbar structure, a given junction is switched by applying a voltage, V, across the wires defining the junction. To avoid switching every junction sharing one of the two address lines, V is split into $-\frac{1}{2}V$ and $+\frac{1}{2}V$ components and applied symmetrically across the two wires of the junction. Thus, junctions in the given row and column only receive half of the required switching voltage and should not switch. Nevertheless, because the required field for domain polarization is subject to statistical fluctuation, $\pm\frac{1}{2}V$ occasionally generates a sufficient field to inadvertently switch junctions that received only half the switching voltage. This is generally referred to as the 'half-select' problem and is a generic problem for crossbar circuits utilizing domain-switched electronic materials. To the contrary, the half-select problem has not been observed in the [2]rotaxane-based crossbar circuits discuss herein (discussed in Section 4.6).

Despite the significant advantages of [2]rotaxane switches in terms of scalability and operability in crossbar structures, these switches do have some drawbacks. For one, the relaxation of the switch from its low-resistance binary '1' state to its high-resistance binary '0' state is thermally activated. Thus, [2]rotaxane-based devices will show temperature-dependent variations in their operation. Another drawback is that [2]rotaxane molecular-switch tunnel junctions (MSTJs) are observed to stop functioning after a relatively low number of write cycles. In large MSTJs (~50 μm^2), this number ranges from 100 to 1000 cycles[9, 39], and is significantly less in nanometer-scale junctions. A possible explanation is that molecules along the perimeter of a junction are more susceptible to environmental damage. Reducing the area of the junction increases the fraction of molecules found along the perimeter, thus resulting in a lower average number of write cycles. Finally, because the switching mechanism is due to large-amplitude molecular mechanical motion, it is relatively slow. The solid-state kinetic processes responsible for molecular mechanical switching have been quantified for a variety of bistable [2]rotaxanes, revealing that switching occurs on a millisecond time scale[27, 29]. While quite slow compared to conventional CMOS-based switches, this is not a significant limitation since in highly parallel architectures, such as the crossbar, computational speed may be generated by switching many devices at once rather than quickly switching one device at a time.

4.5 A 160,000 bit memory circuit patterned at 1×10^{11} bits/cm^2

In 2002, the Heath and Stoddart groups reported on the use of bistable [2]rotaxane molecules as the active elements within a 64-bit molecular electronic random access memory (RAM) circuit that utilized micrometer-scale wiring[9]. While this work successfully demonstrated that [2]rotaxane molecules could be used as the active elements within in a solid-state crossbar memory circuit to store, read out, and erase small data strings, it did not take advantage of the unique scalability offered by molecular components. This would have required methods for patterning circuits with macromolecular feature sizes and pitches. The Superlattice Nanowire Pattern Transfer (SNAP) method, which can pattern ultra-dense arrays of NWs aligned over millimeter length scales, provides this capability.

A major focus of my research in the Heath group was to integrate SNAP-fabricated NW arrays with [2]rotaxane molecular materials to demonstrate an ultra-dense crossbar molecular electronic memory circuit patterned at macromolecular dimensions. In addition to demonstrating device density, we wanted to demonstrate large-scale device integration. To that end, the SNAP method was extended from previous reports[59, 60] to generate arrays of 400 NWs that were used to construct and test a 400-by-400 crossbar memory circuit at extreme dimensions. As Figure 4-4 shows, the entire 160,000-bit crossbar circuit is approximately the size of a white blood cell (~13×13 μm^2).

The fabrication of this molecular memory circuit proved to be a significant challenge on many fronts. First and foremost was the inherent difficulty in making

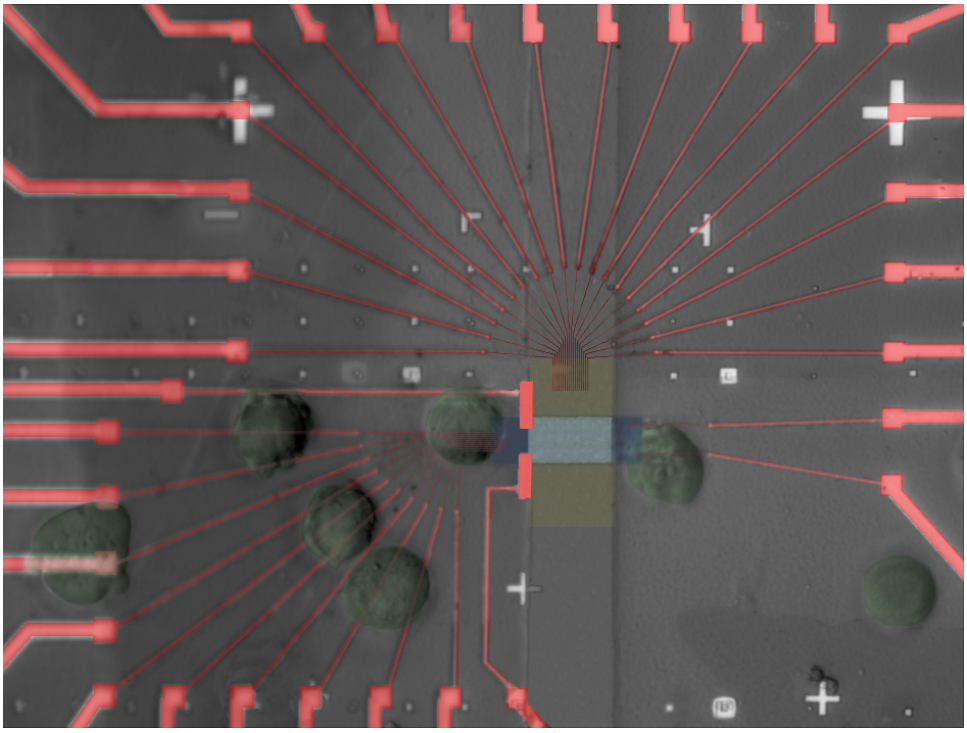


Figure 4-4. A false-colored optical micrograph of a memory circuit with white blood cells for scale. White blood cells (shown in green) approximately 15 micrometers in diameter were sprinkled over this functional memory circuit to provide a biological metric for the level of device integration accomplished in this work. All 160,000 bits are contained within the intersection of the yellow and blue rectangles.

devices at the density described herein, and of integrating those devices into large-scale functional circuits. This is emphasized by contrasting the level of device integration in our molecular memory with its analog in conventional microelectronic technology, the dynamic random access memory (DRAM) circuit. The 2005 International Technology Roadmap for Semiconductors (ITRS) consortium reports[1] that current DRAM circuits are patterned with a memory cell size of 0.04 μm^2 and a density of 1.5×10^9 bits/cm^2. For comparison, the molecular memory described here is about two orders of magnitude more dense with a memory cell size of 0.001 μm^2 and device density of 1×10^{11} bits/cm^2. In fact, this level of integration is on par with 'ultimately scaled' CMOS-based

microelectronic technology, which ITRS projects may reach a cell area of 0.001 μm^2 and density of 5×10^{10} bits/cm^2 by the year 2020.

An additional fabrication challenge was developing a process flow compatible with the delicate [2]rotaxane molecular monolayers. This was accomplished by adopting a fabrication scheme in which the memory was built up sequentially, with the molecular monolayer incorporated as close to the final step as possible, and then protecting that monolayer during subsequent processing steps. It also required establishing electronic-measurement protocols that could be employed to follow the conductivity status of the NWs during the entire nanofabrication procedure. Details of this process flow, along with the various electronic testing protocols, are discussed in Section 4.6. However, a list of the major steps in memory fabrication is as follows:

1. Use SNAP to fabricate the bottom array of Si NW electrodes.
2. Pattern all necessary electrical contacts using electron beam lithography.
3. Planarize the chip using a spin-on glass.
4. Deposit the [2]rotaxane monolayer and evaporate a thin Ti layer on top.
5. Deposit a Pt NW array over the molecule/Ti layer perpendicular to the Si NWs.
6. Using dry etching, transfer the Pt NW pattern to the underlying Ti layer.

Note that, in the last step, the Pt NW pattern only serves as a mask to define an array of Ti NWs from a continuous Ti film. This fabrication protocol not only minimizes the number of processing steps after deposition of the molecular monolayer, but also uses the Ti layer to both protect the molecules and serve as the top electrode in the molecular-

switch tunnel junctions. This technique has been shown, via infrared spectroscopy in conjunction with electronic transport measurements, to protect the functional sites of the [2]rotaxane molecules by reacting with the hydrophobic end groups while leaving the functional regions of the molecule unscathed (Figure 4-2.A)[22].

The structure of our crossbar molecular memory circuit is shown in Figure 4-5.A, and consists of a bottom electrode set of 400 Si NWs (16-nm wide, 33-nm pitch; highly phosphorous doped, as discussed in Chapter 2, $n = 5\times10^{19}$ cm^{-3}) crossed by a top electrode set of 400 Ti NWs (16-nm wide, 33-nm pitch) sandwiching a monolayer of bistable [2]rotaxanes. Each bit corresponds to an individual molecular-switch tunnel junction defined by a Si bottom NW and Ti top NW, and contains approximately 350 [2]rotaxane molecules. The solid-state switching signature of the bistable [2]rotaxanes that were used in this study has been shown to originate from electrochemically addressable, molecular mechanical switching for C/mol/metal or Si/mol/metal junctions[61], but not for metal/mol/metal wire junctions[62]. The desire to utilize molecular mechanical bistable switches as the storage elements is what dictated the choice of the Si NW/mol/Ti NW (Si/mol/Ti) crossbar structure.

Electrical contacts were established to several bottom and top NWs to allow for testing of up to 180 ‘effective’ bits (ebits) from the central region of the crossbar, but only 128 were actually tested due to measurement constraints. The ‘effective’ prefix is used because SNAP-fabricated NWs are patterned beyond the resolution of lithographic methods[63], so each contact bridges 2–4 NWs (Figure 4-5.B). As a result, most of the tested ebits contained an average of 4–9 junctions. We recently reported on a demultiplexer[54] (see Chapter 3) that would allow for this memory circuit to be fully

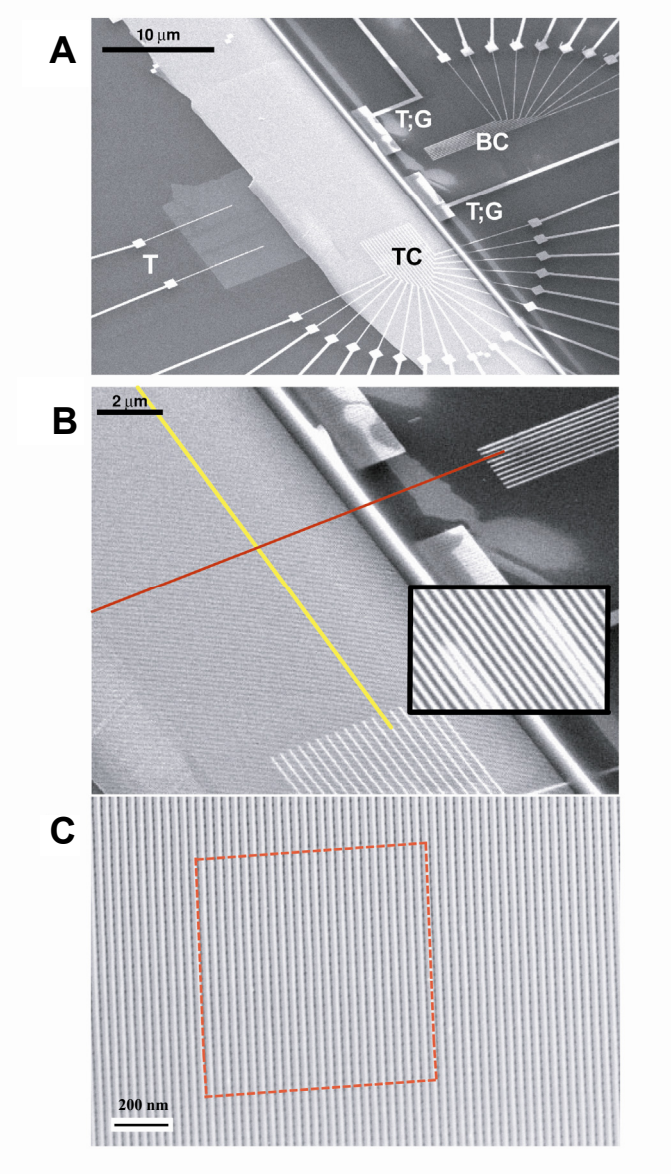


Figure 4-5. Scanning electron micrographs (SEMs) of the NW crossbar memory. A. Image of the entire circuit. The array of 400 bottom Si NWs is seen as the light grey rectangular patch extending diagonally up from bottom left. The top array of 400 Ti NWs is covered by the SNAP template of 400 Pt NWs, and extends diagonally down from top left. Testing contacts (T) are for monitoring the electrical properties of the Si NWs during the fabrication steps. Two of those contacts are also grounding contacts (G), and are used for grounding most of the Si NWs during the memory evaluation, writing, and reading steps. Electron-beam-lithography patterned 18 top (TC) and 10 bottom (BC) contacts are also visible. The scale bar is 10 micrometers. **B.** An SEM image showing the cross-point of top and bottom NW electrodes. Each cross-point corresponds to an 'effective bit' in memory testing because (inset) the electron-beam-lithography defined contacts bridged 2–4 nanowires. The scale bar is 2 micrometers. **C.** High-resolution SEM of approximately 2500 junctions out of a 160,000-junction nanowire crossbar circuit. The red square highlights an area of the memory that is equivalent to the number of bits that were tested. The scale bar is 200 nanometers.

tested, including the ability to address each junction independently. However, implementation of that demultiplexer would have added significant complexity to an already demanding nanofabrication procedure, and wasn't necessary to demonstrate the viability of this circuit. (This limitation simply adds some level of uncertainty to our estimates of device yield.) Assuming 4–9 junctions per ebit, the 128 ebits tested represents between 0.5–0.7 percent of the full 160,000-bit crossbar circuit distributed across 6 percent of the device area (Figure 4-5.C). We believe that this relatively small portion of the crossbar is representative of the overall circuit. This belief is based upon the fact that we have fabricated approximately 50 full 160,000-junction crossbar memory circuits, four of which have been fully tested as memories. Each of those tested memory circuits yielded similar results.

By scanning-electron-microscopy inspection, the 160,000-junction crossbar appeared to be structurally defect-free, with no evidence of broken, wandering, or electrically shorted NWs (Figure 4-5.B). Nevertheless, there were a large number of electrical defects. Comprehensive electrical characterization was used to determine the address locations of both working and defective ebits, as well as to provide insight into the nature of the defective ebits. This was done by first applying +1.5 V relative to the Si NW electrodes to set all ebits to '1', or alternatively to switch the [2]rotaxane molecules to their metastable-state co-conformation. Each ebit was then read sequentially using a non-perturbing +0.2 V bias. Application of –1.5 V to the Si electrode was then used to set all ebits to '0'; this effectively returned the active molecular monolayer to its ground-state co-conformation. The status of each of the 128 ebits was then read again. The 1/0 current ratios are presented in Figure 4-6.A. Approximately 50 percent of the tested ebits

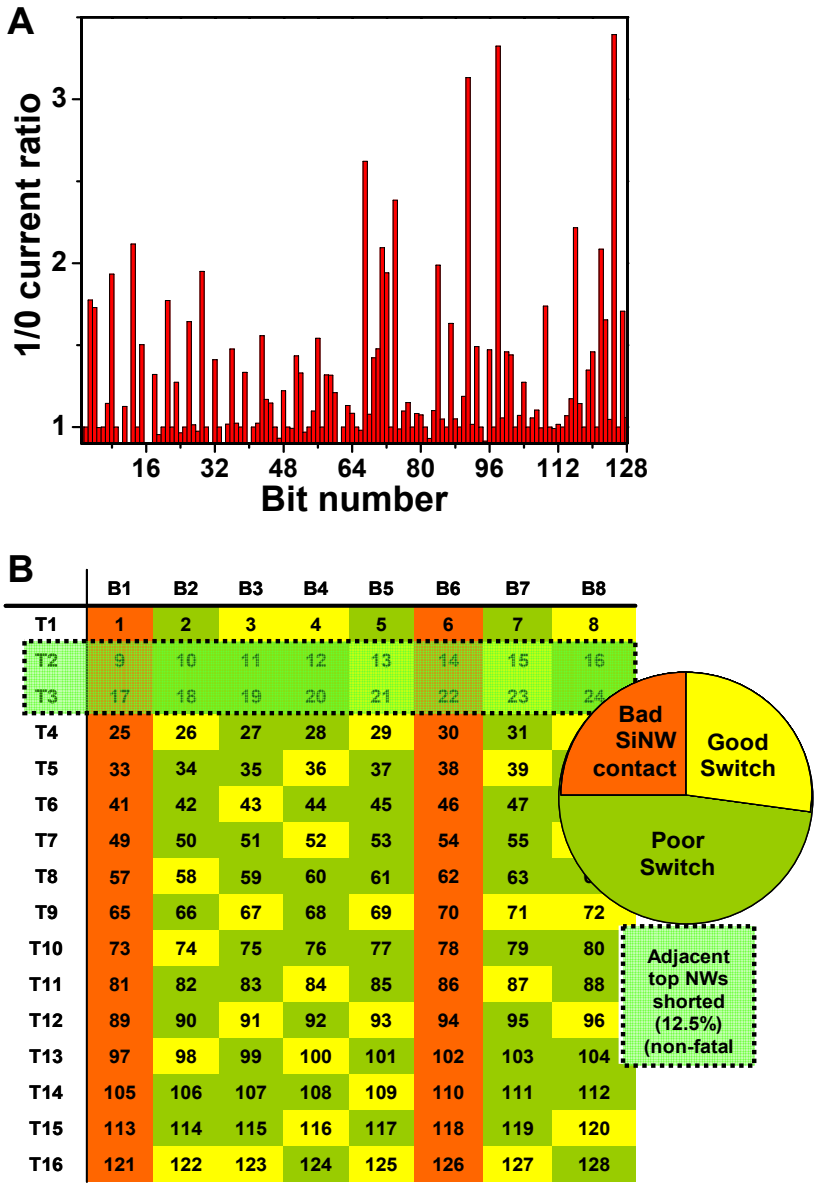


Figure 4-6. Data from evaluating the performance of 128 ebits within the crossbar memory circuit. **A.** The current ratio of the '1' state divided by the '0' state of the tested ebits. Note that many of the ebits exhibit little to no switching response. Those ebits are defective. **B.** A map of the defective and usable ebits, along with a pie-chart giving the testing statistics. Note that, except for the bad Si NW contacts on bottom electrodes B1 and B6, and the shorted top electrodes, T2 and T3, the defective and good bits are randomly distributed. Poor switches can be divided into two types: Type I defects (26% of the 128 tested) are ebits that exhibited an open-circuit conductance and a low- or zero-amplitude switching response when tested. Type II defects (22%) are non-switchable ebits that exhibited a conductance similar to that of a closed switch. In both cases, the 1/0 ratio is near unity.

yielded some sort of switching response; however, some of those ebits may have been exhibiting behavior originating from assorted parasitic current pathways through the crossbar array.

Multiple current pathways between an input and output electrode are an inherent drawback of crossbar architectures wherein each junction is electrically connected to every other junction. Thus, when many devices are switched from the '0' to the '1' state, the current through the non-switching devices can also change due to a modification of the effective resistance of these parasitic loops. A standard remedy is to incorporate diodes at each crosspoint[64] that will suppress parasitic loops by acting as one-way current valves. Although the molecule/Ti interface yields some built-in rectification, we have additionally fabricated micrometer-scale molecular electronic memory circuits with a vertical p-n doping gradient through each junction[19]. This resulted in improved memory performance that should, in principle, extend to the nanometer-scale memory described here. For this prototype circuit, however, we found it sufficient to simply ground all NW electrodes not being used during a read cycle in conjunction with the establishment of a threshold for a 'good' ebit based upon a minimum 1/0 current ratio of ~1.5. About 25 percent of the ebits passed this threshold. While this yield may be low for a mature technology, we are very encouraged by this result in an unpackaged first-generation circuit.

Defective bits impacted memory performance with varying levels of severity (Figure 4-6.B). Bits with a 1/0 ratio of unity were classified as 'poor switches' and resulted from switches stuck in either the '1' or '0' state. Poorly switching bits only lead to a proportional loss in memory performance. Bad contacts to the NWs, however,

removed an entire row of bits from memory operation. In a similar vein, two Ti electrodes that are shorted together effectively turn two rows of bits into one row of usable bits—also removing a row of bits from operation and doubling the number of junctions in the ebits for that row. We believe the majority of these defects resulted from sub-nanometer variations in the reactive-ion etching process that was employed to define the top Ti NWs. As will be explained more fully in the next section, these Ti NWs originate as a uniform thin film (~20 nm) that is deposited on top of the [2]rotaxane Langmuir monolayer. The SNAP process is used to deposit 400 Pt NWs on top of this film, and those Pt NWs serve as an etch mask for defining the 400 top Ti electrodes. The capability of etching tools to define nanostructures at the narrow pitches required here is largely unexplored and, in fact, this etching step was one of the most challenging nanofabrication steps for constructing the memory.

Isolated devices, or crossbar memories patterned at substantially lower densities and with larger wires, can typically be prepared with a nearly 100 percent yield. The capability of etching tools to define nanostructures at the narrow pitches required here is largely unexplored and, in fact, was one of the most challenging nanofabrication steps in constructing the memory.

An important result from the defect map shown in Figure 4-6.B is that the good and bad ebits are randomly dispersed throughout the matrix, implying that the good junctions are not correlated to one another. However, the ultimate test of any memory circuit is whether it can store information. Based upon the defect map shown in Figure 4-6.B, and taking advantage of the inherent defect tolerance of the crossbar architecture[42], we were able to identify the addresses of good ebits, and from those addresses configure

an operational memory. This is demonstrated by the data of Figure 4-7 in which we have utilized 24 out of 30 operational ebits to write a string of '1's and '0's that represent the ASCII characters for 'CIT,' short for 'California Institute of Technology.'

Our principle motivation for utilizing bistable [2]rotaxane molecules as the active elements within this memory is that even though we are measuring of order 100 molecules in each junction, the change in conductivity correlated with the two conformational states is a single-molecule property[20, 27, 34]. The implication is that the switching signature should be effectively size-invariant (neglecting statistical effects), meaning that it should scale down to the macromolecular dimensions that characterize these crossbar junctions. In fact, the success of these molecules at this scale implies that next-generation devices using only tens of molecules may be possible. While it may be unlikely that these digital circuits will scale to a density that is only limited by the size of the molecular switches, it should be possible to significantly increase the bit density over what is described here (Section 4.6).

Previous work (see Section 4.2) has quantified the thermodynamic and kinetic parameters that describe both the bistability and the switching mechanism of the [2]rotaxane (Figure 4-2.A) and related molecules in a variety of environments. Those measurements required robust switching devices that could be cycled many times and at various temperatures. The junctions measured here were much more delicate: While all good ebits could be cycled multiple times (as evidenced by the testing and writing steps), most ebits failed after a half-dozen cycles or so. While the exact failure mode is still under investigation, it is worth noting that these junctions have a very large perimeter-to-area ratio, and that molecules along the perimeter of a junction are likely to be more

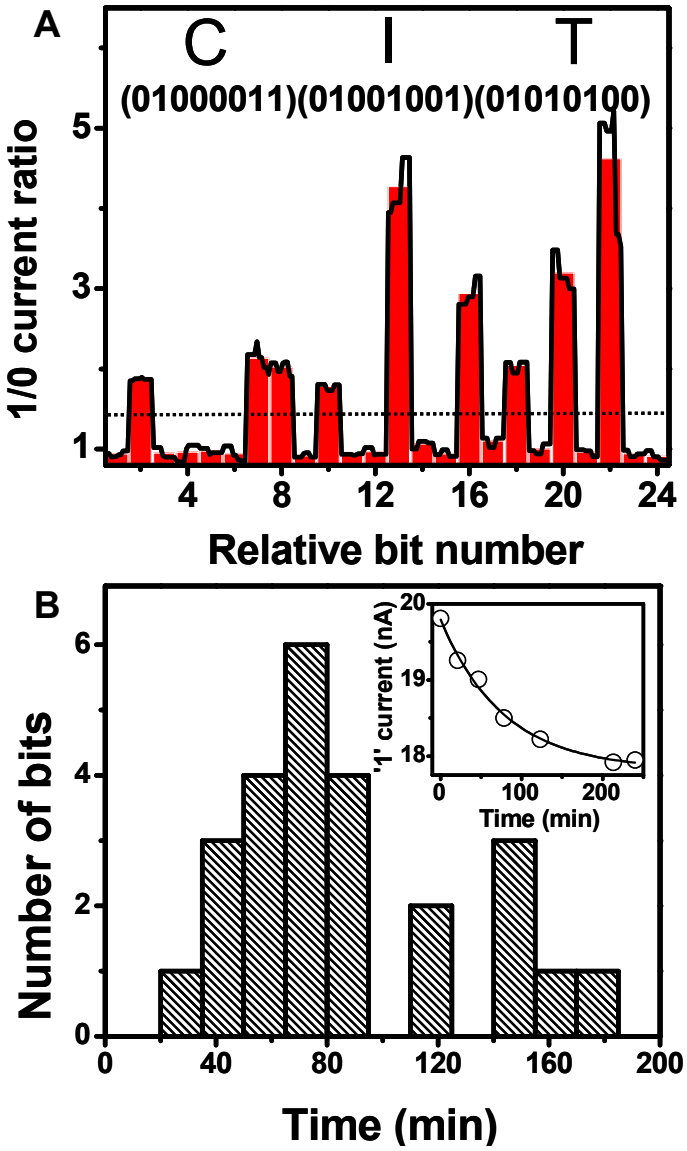


Figure 4-7. Demonstration of memory storage and retention characteristics from the molecular electronic crossbar memory. A. A demonstration of point-addressability within the crossbar. Good ebits were selected from the defect mapping of the tested portion of the crossbar. A string of '0's and '1's corresponding to ASCII characters for 'CIT' (abbreviation for California Institute of Technology) were stored and read out sequentially. The dotted line indicates the separation between a '0' and '1' state of the individual ebits. The black trace is raw data showing ten sequential readings of each bit while the red bars represent the average of those ten readings. Note that deviations of individual readings from their average are well separated from the threshold 1/0 line. **B.** A histogram representing the 1/*e* decay time of the '1' state to the '0' state. The 25 ebits represented in the data each were 'large' ebits, comprised of approximately 100 junctions, to increase the measurement signal to noise. Raw data from a single large ebit is shown in the inset. The line is a single exponential fit used to extract the decay time.

susceptible to processing damage or contamination since the circuit is measured under ambient conditions (*i.e.*, the circuit is unpackaged). Despite this difficulty, we were able to measure the rate of relaxation from the 1→0 state for many of the ebits (Figure 4-7.B). From a device perspective, this time represents the volatility, or memory retention time, of the ebits. With respect to the bistable [2]rotaxane switching cycle, this time represents a room-temperature measurement of the rate-limiting kinetic step within the switching cycle wherein the metastable co-conformation relaxes to the ground state. Our measured rate (90 ± 40 minutes; median decay = 75 minutes) was statistically equivalent to the rate reported for much larger devices (50 μm^2 junction area) containing the same [2]rotaxane switches (58 ± 5 minutes) and measured using a more comprehensive thermodynamic analysis[20]. Thus, our results are consistent with previous reports of a molecular mechanism for the memory operation[9].

4.6 Crossbar molecular memory circuit fabrication and testing

In this section, I will discuss the details of memory fabrication and testing. In an effort to keep this section more or less self-contained, some of the material mentioned above is repeated (albeit with greater detail). A bottom-up approach was critical to the successful fabrication of this memory. This approach both minimized the number of processing steps following deposition of the delicate molecular monolayer, as well as protected the molecules from remaining processing steps. The following in-depth description of how

this memory was fabricated will proceed with an analogous structure, that is, from the bottom up.

The 160,000-junction crossbar memory described above consists of 400 Si nanowire (NW) bottom electrodes of 16-nm width and 33-nm pitch, crossed with 400 Ti NW top electrodes of the same dimensions, and with a monolayer of bistable [2]rotaxane molecules sandwiched in between. We have previously reported on using the superlattice nanowire pattern transfer (SNAP) technique to fabricate highly ordered arrays of metal and Si NWs of up to 128 NWs. For this work, the SNAP technique was extended to create 400-element NW arrays of both the bottom and top electrode materials, and so was the primary patterning method for achieving the 1×10^{11}-cm^{-2} bit density of the crossbar. The SNAP NW fabrication procedure is described in detail in Chapter 2. Briefly, SNAP is a 'top-down,' non-photolithographic technique that uses molecular-beam epitaxy (MBE) to create a physical template for NW patterning. This template is used to deposit an array of Pt NWs onto an epoxy-coated thin-film material. The Pt array then serves as an etch mask to transfer the NW pattern into the underlying thin-film. This technique enables the fabrication of ultra-dense arrays of high-aspect-ratio (length to width routinely > 10^6) Si and metal NWs that are aligned over millimeter length scales, without the need for a secondary alignment step after NW fabrication.

4.6.1 Fabrication and contact to bottom Si nanowire electrodes

An overview of the process flow used to fabricate the memory is shown in Figure 4-8. The Si NW array was fabricated as described in Chapter 2. The starting wafer for the Si

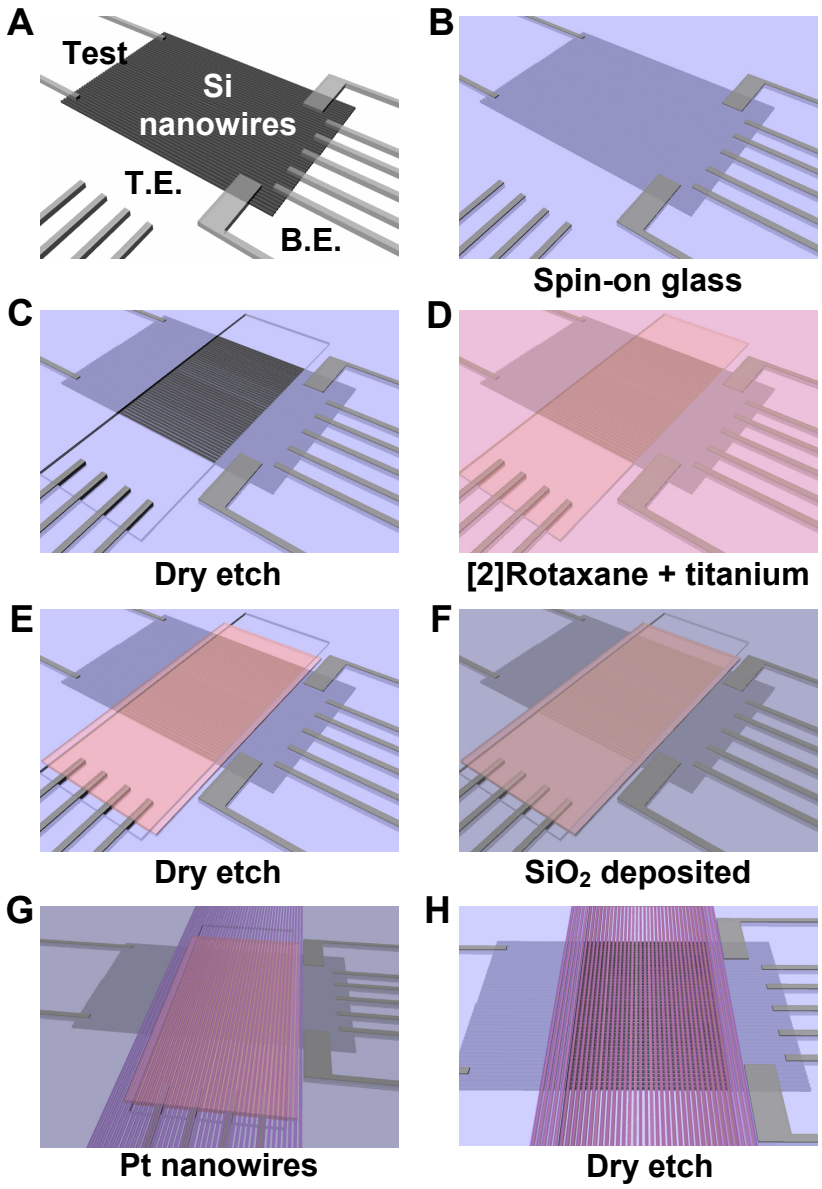


Figure 4-8. Process flow for fabricating the 160,000 bit molecular electronic memory circuit at 10^{11} bits/cm^2. A. A section of SNAP-patterned SiNW bottom electrodes are electrically contacted to electron-beam lithography patterned metal electrodes. **B.** The entire circuit is coated with SiO_2 using an optimized spin-on-glass procedure. **C.** The active memory region is exposed using lithographic patterning followed by CF_4 dry etching. **D.** The bistable [2]rotaxane Langmuir monolayer is deposited on top of the Si NWs and then protected by the deposition of a Ti layer. **E.** The molecule/Ti layer is etched everywhere except for the active memory region. **F.** An evaporated SiO_2 insulating layer is deposited over the entire chip. **G.** An array of Pt NWs is deposited on top of the Ti/SiO_2 layer at a right angle to the bottom Si NWs using the SNAP method. **H.** The Pt NW pattern is transferred, using BCl_3 dry etching, to the underlying Ti layer to form an array of top Ti NW electrodes, and the crossbar structure is complete.

NWs was a 33-nm-thick silicon-on-insulator (SOI) substrate with a 250-nm-thick buried oxide (Simgui, Shanghai, China). This wafer was highly diffusion doped (phosphorous; $n=5\times10^{19}$ cm^{-3}) to ensure that NWs fabricated from it would maintain robust conductivity throughout the various nanofabrication procedures, in addition to forming ohmic contacts with Ti-Pt leads. This proved to be important in later stages where the Si NW surface is unavoidably etched. To fabricate the Si NW array, an array of Pt NWs was deposited onto the doped SOI substrate using the SNAP method, and high-frequency (40 MHz) fluorine-based (CF_4 to He 20:30, 5 mTorr, 40 W) reactive-ion etching was used to transfer the Pt NW pattern into the underlying Si epilayer to form an approximately 2-millimeter-long array of Si NWs. The Pt NW array was then dissolved in hot aqua regia (1:4 conc. HCl to conc. HNO_3, 120° C, 10 min) and the Si NW array was sectioned into a 30-μm-long region using a lithographically-patterned Al mask and three sequential reactive-ion etch (RIE) steps. The first was a high-power O_2 RIE (20 mTorr, 100 W, 2 min) to remove any residual epoxy (from the SNAP procedure), then a brief SF_6 RIE (5 mTorr, 30 W, 30 sec) to remove any unmasked Si, and finally a low-power O_2 RIE (20 mTorr, 10 W, 2 min) to oxidize any pinholes through the insulating oxide that may have been bored out from the first two RIE steps. We had occasionally observed leakage current through the insulating oxide when this last step was omitted.

Ten electrical contacts to these bottom Si NWs, as well as 18 contacts that are intended for the top Ti NWs, were defined at this point using standard electron-beam lithography (EBL) patterning and electron-beam evaporation to produce wires consisting of a 15-nm Ti adhesion layer followed by 50 nm of Pt (Appendix 4.1). Immediately prior to metal evaporation, the Si NWs were cleaned using a gentle O_2 plasma (20 mTorr, 10

W, 30 seconds) followed by a 5-second dip in buffered oxide etch (BOE) (6:1; NH_4F to HF) solution to remove the Si NW native oxide. After metal lift-off, the chip was annealed at 450 °C in N_2 for 5 minutes. In addition to promoting the formation of ohmic contacts, this anneal helped to prevent peeling of the smallest lithographically-defined wires during the spin-on glass step described below.

Figure 4-9.A shows an SEM image of the memory circuit at the stage in which the Si NWs and all of the external electrical contacts have been created. Note that there are four sets of EBL-defined contacts. The 18 narrow contacts at the bottom left of the image (nominal width of 70 nm at 300-nm pitch) will eventually connect to the top Ti NW electrodes and are used for testing of the final memory circuit. The ten narrow contacts to the Si NWs at the bottom right (nominal width of 60 nm at 300-nm pitch) of the image are also used for testing of the memory circuit. Finally there are two narrow test electrodes at the top left and two wide electrodes at the bottom right. The wide electrodes contact about two-thirds of all the Si NWs and serve a dual function. First, they ground unused Si NWs during memory testing to minimize parasitic current loops through the crossbar. (This procedure approximates how a fully multiplexed crossbar circuit would be utilized.) Second, when used in conjunction with the two narrow test-electrodes on the opposite side of Si-NW array, they enable testing of the Si-NW conductivity at various stages throughout the memory-fabrication processes. This testing procedure provided invaluable feedback for finely tuning and tracking many of the fabrication processes, most notably the etching procedures described below. Once these various contacts were established, robust Si-NW conductivity was confirmed via current-voltage (*I-V*) measurements. If the Si NWs were measured to be poor conductors (a very infrequent

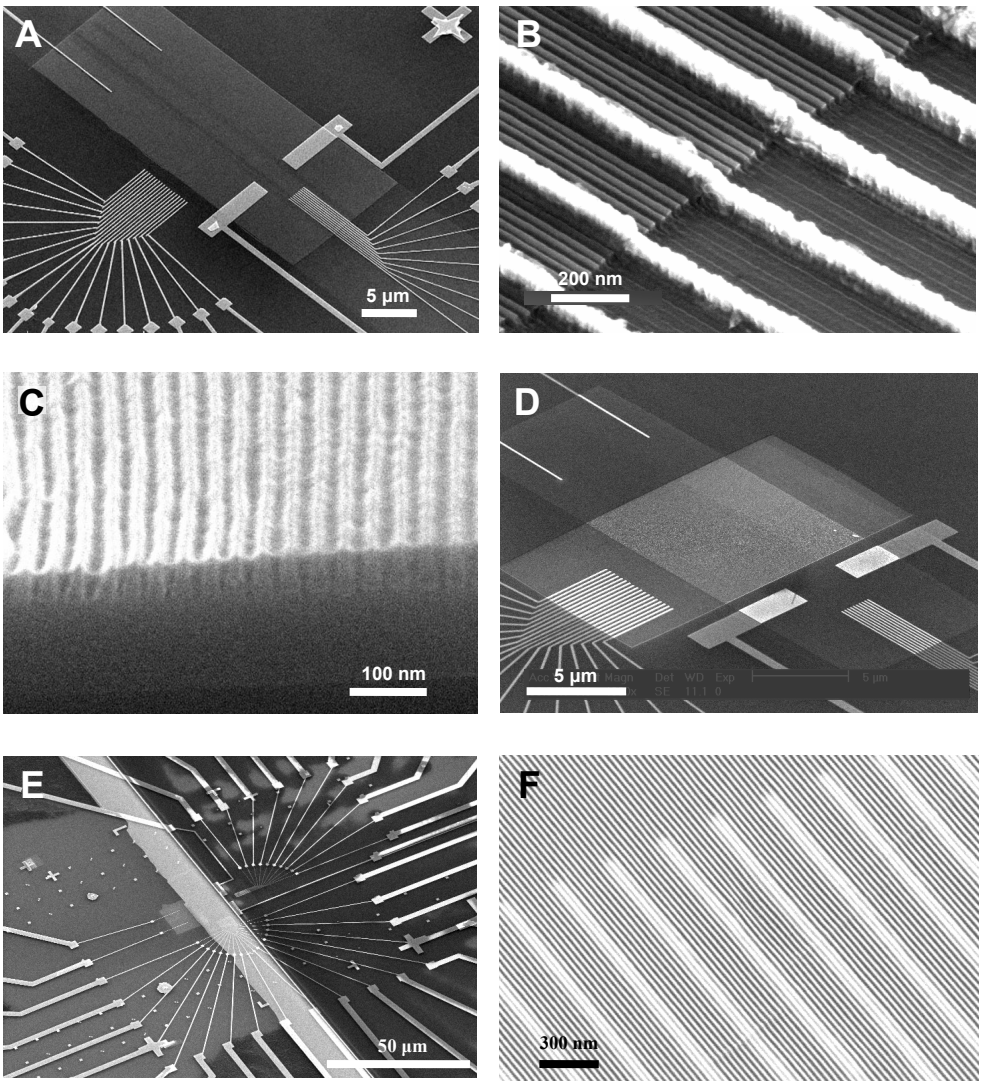


Figure 4-9. Representative scanning electron micrographs illustrating the crossbar memory fabrication process. A. A 30-micrometer-long section of 400 Si NWs with electron-beam lithography (EBL) defined contacts to the Si NWs (bottom right) and pre-patterned contacts to the Ti-NW array (not deposited at this stage) (bottom left). **B.** Representative contacts to Si NWs showing each EBL-defined metal lead is about 70-nm wide and contacts 2–4 NWs. **C.** Micrograph verifying that the spin-on-glass layer fills the narrow trenches between the Si NWs. The chip was cleaved after the planarization process to allow for the view shown here. **D.** Lithographically patterned window in the SOG film defining the memory active region. **E.** Deposition of 400 Pt NWs over the memory active region. Note the Pt NWs extend for about a millimeter in either direction. **F.** Micrograph of the Ti NWs contacting pre-patterned EBL-defined leads after transferring the Pt NW pattern to the underlying [2]rotaxane/Ti layer.

occurrence) or if there was any measurable leakage current through the insulating oxide, the chip was discarded. Additionally, the chip was discarded if *I-V* measurements showed rectification at the contacts (unless ohmic behavior could be established through further efforts).

The device was then planarized using an optimized spin-on-glass (SOG) procedure (Accuglass 214, Honeywell Electronic Materials, Sunnyvale CA) (Figure 4-8.B). This planarization process is critical because the SOG not only protects Si NWs and EBL defined wiring outside of the active memory region from damage that can arise during subsequent processing steps, but it also prevents evaporated Ti from entering the gaps between Si NWs where it would be extremely difficult to remove. For the SOG to fill the narrow gaps between adjacent Si NWs (Figure 4-9.C), it had to be applied to the surface of the chip while under vacuum (< 1 mTorr). This was accomplished by placing the chip into a clean Erlenmeyer flask* sealed with an air-tight rubber septum and piercing the septum with a syringe needle attached via tubing to a diffusion pump. After a couple of minutes to ensure evacuation of the flask, a scrupulously-clean glass syringe and a 5-inch metal needle were used to generously apply the SOG liquid to the chip surface (while maintaining vacuum with the other needle). The chip was then immediately taken out of the flask and spun at 5000 RPM to ensure a uniform film.

At this point, the chip was inspected using a light microscope to look for any particulates on the surface. If any were found (which almost inevitably there were), the SOG film was stripped from the chip with extremely delicate swabbing while immersed in methanol, followed by repeatedly rinsing in methanol and isopropyl alcohol and drying

* The flask was modified with a cylindrical glass pedestal for mounting the chip off the flask bottom.

under a stream of N_2. The repeated solvent rinses followed by blowing with N_2 seemed to be particularly effective at removing particles from the surface. Sonication was avoided because it would occasionally damage the finest EBL-defined wires. The SOG was then re-applied to the chip under ambient conditions, spun at 5000 RPM, and re-checked for particulates. Note that vacuum application was found to be unnecessary for subsequent SOG applications, probably because the SOG applied under vacuum continued to wet the NW gaps through subsequent methanol cleanings. This entire procedure was repeated until no particulates were seen on the chip surface. It was very important to ensure that the chip was rigorously clean before proceeding, since particles on the surface would frequently result in an unsuccessful transfer of the Pt NW array (which is required in a later step to define the top Ti NW electrodes for the memory).

After globally thinning the SOG layer to 50 nm using a CF_4 plasma (10 mTorr, 40 W), an opening in polymethyl-methacrylate (PMMA) was lithographically defined over the Si NWs, and the tips of the 18 EBL-defined contacts (Figure 4-8.C). The SOG was then further etched until the tops of the underlying Si NWs were exposed (Figure 4-9.C). This step was monitored by periodically measuring the Si NW conductivity using the test electrodes. The majority of dopant atoms in the Si NWs reside within 10 nm of the surface (Chapter 2), so the NW conductivity is very sensitive to any etching of the surface. This unique feature of SNAP-fabricated Si NWs makes it very straightforward to etch back the SOG until just the tops of the Si NWs are exposed, since the etch end-point can be precisely determined by a small drop in the Si NW conductivity. At this stage, the entire memory circuit is under SOG (and thus electrically isolated from any further top

processing) except for the lithographically-defined opening over the Si NWs and the 18 EBL-defined contacts. This opening defines the active memory region (Figure 4-9.D).

4.6.2 Deposition of molecules and top electrode materials

A monolayer of bistable [2]rotaxane switches was prepared by Langmuir-Blodgett techniques and transferred onto the device, as reported previously[9, 22]. A thin film of Ti (20 nm) was then evaporated over the entire chip (Figure 4-8.D). This Ti layer serves to protect the molecules from further top processing and will later be patterned into the crossbar top electrodes. As briefly mentioned above, this Ti layer additionally adds (favorable) current rectification to each Si/mol/Ti MSTJ to reduce the impact of parasitic current pathways within the crossbar circuit. The amount of rectification is dependent upon the amount of Ti oxidation that occurs at the molecule/Ti interface, which, in turn, depends upon the vacuum level of the metal deposition system[65-67]. For this work, the Ti was deposited at a pressure of approximately 5×10^{-7} Torr. For micrometer-scale Si/mol/Ti MSTJs, this typically produced a rectification of about 10:1 at 1 V.

Using photolithographic techniques and BCl_3 RIE (5 mTorr, 30 W), the molecule/Ti layer was then removed from everywhere except for the memory active region where electrical contact to the underlying Si NWs is made (Figure 4-8.E). Patterning the Ti film is important for two reasons: First, it prevents the deposited Ti film from bridging (shorting) EBL-defined wiring that protrudes from the SOG film (explained below). Second, it removes the requirement of precise NW registry over the entire length (> 1 millimeter) of the Pt NW array deposited in a later step (Figure 4-9.E).

This is important because the dry etch used to define the Ti NW pattern from the Pt NW stencil can cause adjacent Pt NWs to wander into each other (thus shorting the corresponding Ti NWs beneath) in regions with non-uniform epoxy (used to adhere the Pt NW array to the surface; explained further below). Over the length of a typical SNAP-fabricated NW array, the likelihood of this occurrence is expected to be almost certain.

Next, a thin (~15 nm) SiO_2 layer was deposited over the entire substrate to isolate the EBL-defined wires from the Pt NWs to be deposited in the next step. (Recall that the EBL-defined wires are 65-nm tall and the SOG was globally thinned to 50 nm; Figure 4-8.F.) It may seem that the SiO_2 deposition can be avoided by etching the SOG film to a thickness greater than the EBL-defined wire height. However, this results in a larger recess in the SOG opening that defines the memory active region. Spin-coated epoxy used for Pt NW deposition in the next step fills this opening, which, as explained below, can be problematic during subsequent etching.

A thin layer of epoxy (~10 nm) is then spin-coated onto the chip and the SNAP technique is used to deposit an array of 400 Pt NWs over the Ti/SiO_2/epoxy layer at a right angle to the underlying Si NWs (Figure 4-8.G and Figure 4-9.E). Finally, careful BCl_3 RIE (5 mTorr, 30 W) was used to transfer the Pt NW pattern to the underlying Ti/SiO_2/epoxy film, thus forming Ti NW top electrodes (Figure 4-8.H). Although the Pt NW array is in excess of 1 mm long (Figure 4-9.F), the top Ti electrodes of the crossbar circuit only extend from the tips of the 18 EBL-defined leads to a couple of micrometers past the underlying Si NW array. The etch endpoint was determined by monitoring the

cross-conductance of the top Ti NWs* (Figure 4-10.A). Complete transfer of the Pt NW pattern to the underlying Ti film is indicated by a fall in the cross-conductance to about ten nanoSiemens (nS). Note that the cross-conductance does not go to zero since the Ti electrodes, while physically separated, are still electrically coupled through the crossbar

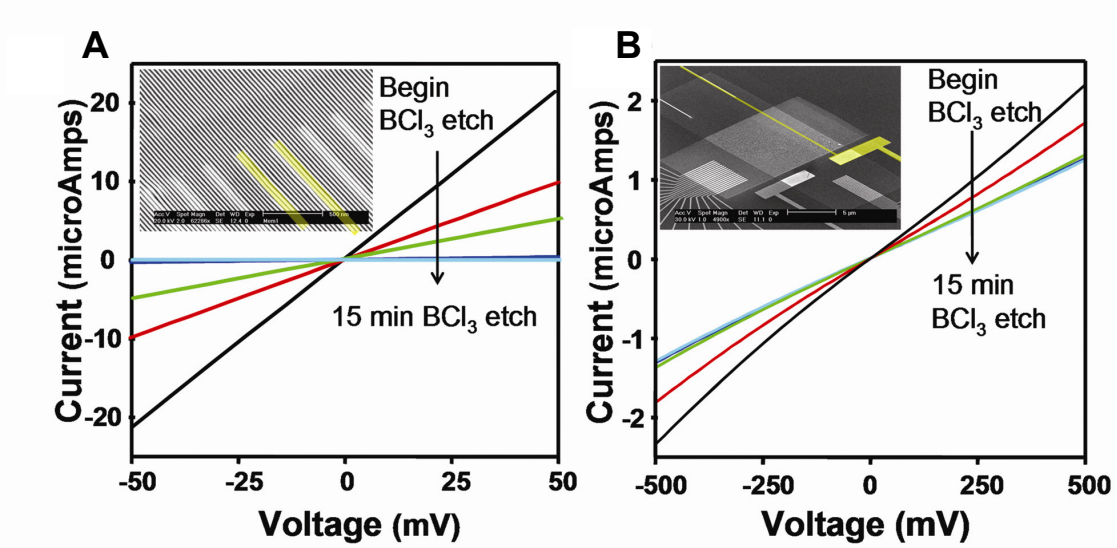


Figure 4-10. Conductance monitoring during the Ti layer etching. A. Cross-conductance measurements between electrical contacts to the top nanowire array were performed to monitor the Ti layer etching. When the current drops to sub-10 nanoAmps, the top Ti electrodes are separated. The inset scanning electron microscope (SEM) image shows two representative contacts to the top Ti electrodes as highlighted in yellow. It is the cross-conductance between such contacts that was used for this measurement. **B.** The Si NW conductance was measured throughout the Ti layer etching to ensure that Si NWs were not damaged. The SEM image (inset) shows the current pathway that was measured.

junctions and the underlying Si NWs. The health of the underlying Si NWs throughout the Ti-etching process was also monitored, as shown in Figure 4-10.B.

The use of Ti as the top electrode material here was necessary since its high reactivity prevents metal from spiking across the [2]rotaxane monolayer during evaporation[22]. However, Ti is a difficult metal to etch because it forms a tough TiO_2 layer

* Interferometric end-point detection cannot be used here since the etch rate for Ti within a 16-nm trench is likely to be quite different from that of a regular Ti surface.

at its surface[68]. Highly-directional BCl_3 reactive ion etching was used because it provides the needed momentum to erode the TiO_2 layer while additionally providing reactive Cl ions to chemically remove Ti. Figure 4-11 shows a fully functional memory circuit in which the final Pt NW deposition (using SNAP) was substituted by EBL-patterned Pt microwires (about 210 nm in width) of variable spacing down to 90 nm (less than 3 times the SNAP Pt NW pitch). The [2]rotaxane/Ti film was etched using an iterative procedure so the Ti electrode cross-conductance could be periodically checked. Also, SEM analysis was used to gauge how effectively the BCl_3 etch removed Ti from between the Pt microwires after each etch iteration. Note that SEM analysis cannot be used to track the etch progress using SNAP fabricated Pt NWs because 1) the SNAP NW spacing is too narrow, and 2) the electron beam heats up the underlying epoxy causing the Pt NWs to collapse into each other. SEM analysis confirmed complete separation of the underlying Ti electrodes after the Ti electrode cross-conductance fell to below 50 nS (Figure 4-11.A). Taking into account that the (210-nm-wide) EBL-defined Ti microwires are about five times wider than the combined average number Ti NWs defining a row of ebits (3×16 nm = 48 nm), we reasoned that Ti NWs patterned from SNAP-fabricated Pt NWs would be separated when their cross-conductance fell to below 10 nS. We have found this metric to be accurate with many memory chips. Note that while the Ti electrode cross-conductance at separation scales linearly with the Pt wire width, the total etch time does not scale predictably with the Pt wire spacing. In fact, the total etch time to achieve Ti NW separation can vary considerably from one memory chip to the next with nominally identical SNAP Pt NW arrays.

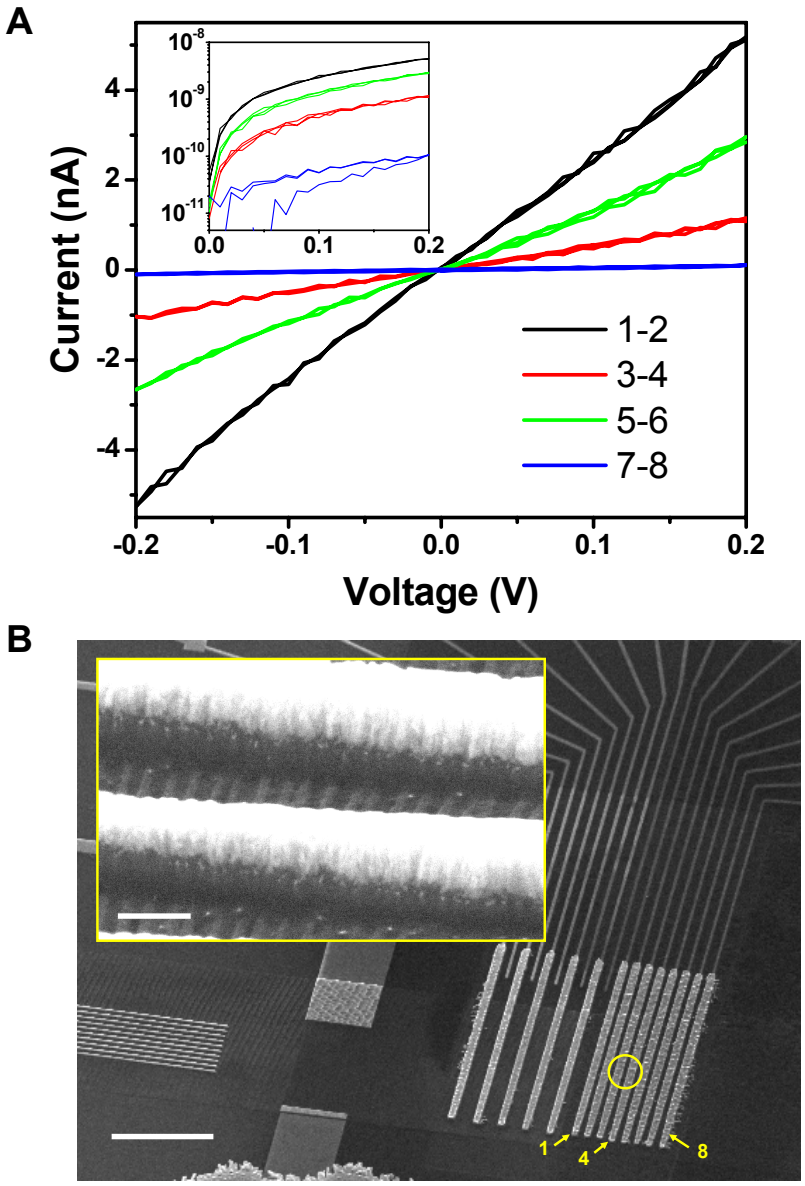


Figure 4-11. Diagnostic Ti etching between electron-beam lithography (EBL)-defined Pt microwires. A. Ti electrode cross-conductance after separation. The numbers in the legend correspond to cross-conductance measurements between the 90-nm-spaced EBL-defined microwires numbered (sequentially from 1 to 8) in panel B. The cross-conductance values after separation are 0.5 nS (blue trace, corresponding to microwires 7–8), 5 nS (red trace, microwires 3–4), 15 nS (green trace, microwires 5–6), and 25 nS (black trace, microwires 1–2). **B.** SEM images of the diagnostic memory circuit. The background shows the EBL-defined Pt microwires patterned over the memory active region to form a Si NW/Ti microwire crossbar. Scale bar is 2.5 μm. The inset is a high-resolution SEM image of the region between two Pt microwires indicated by the yellow circle. Note the Ti has been cleanly removed, revealing the underlying Si NWs. The scale bar is 100 nm.

To form Ti NWs as the top electrodes from SNAP deposited Pt NWs, the chip was etched (via BCl_3) in intervals of time ranging from five minutes (at the beginning) to 30 seconds (near the end) so that periodic conductance measurements could be made to monitor the etch progress. This significantly increased the total BCl_3 etch time because the chip had to be periodically removed from the vacuum environment of our reactive ion etcher resulting in re-growth of surface TiO_2. The total BCl_3 etch time ranged from 15 to 20 minutes, although a large fraction of that time was undoubtedly spent etching re-grown TiO_2 at the beginning of each BCl_3 etch iteration. As will be discussed below, this unavoidably* long etch time can lead to fidelity problems in transferring the Pt NW pattern to the underlying Ti film.

The Ti etching step described above proved to be one of the most challenging aspects of memory fabrication and required the simultaneous optimization of a number of correlated factors. This included the BCl_3 etch recipe described above, the depth of the SOG recess defining the memory active region, and the epoxy used to bond the Pt NW array to the Ti/SiO_2 film. This epoxy fills in the SOG recess, thus separating the Ti/SiO_2 film from the Pt NWs by a relatively thick organic spacer. If the epoxy is too thick significant undercutting can occur. This leads to blurring of the Ti NW pattern and wandering of individual Pt NWs on top of a sea of shifting epoxy. Frequently the Pt NWs would wander so much they would short into each other (Figure 4-12), resulting in Ti NW top electrodes that could not be physically separated (more on this below). This problem was exacerbated with the relatively long BCl_3 etch times required to define the Ti NWs.

* In principle, the etch time could be reduced considerably by monitoring the conductance in-situ with the BCl_3 etch so the time-consuming TiO_2-removal step would only need to be done once. Obviously, our RIE system did not have this feature.

The epoxy thickness over the Ti film was reduced by decreasing the recess of the SOG window defining the memory active region. This was accomplished by globally thinning the SOG film beforehand. Trial and error in conjunction with atomic force microscopy (AFM) measurements of the SOG recess and surrounding region after curing the spin-coated epoxy (without the deposited Pt NWs) determined the optimal SOG

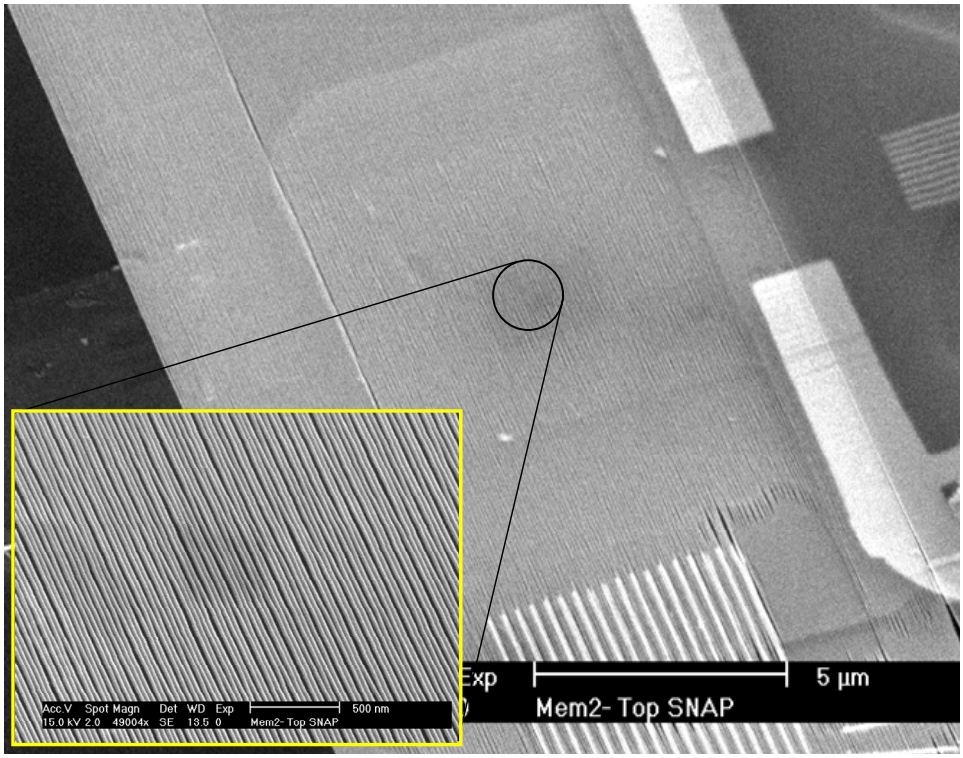


Figure 4-12. SEM image of a crossbar memory circuit before optimization of the Ti NW fabrication parameters. This image shows wandering of individual Pt NWs due to shifting epoxy from the BCl_3 etch used to define the Ti NW array. The inset shows a zoomed-in view from the center of the memory crossbar region. The scale bar in the inset is 500 nm.

thickness to be about 50 nm. Thinning the SOG further required reducing the thickness (65 nm) of the EBL-defined wires so they did not protrude from the SOG + SiO_2 film (keeping the thickness of SiO_2 deposited over the Ti film to be constant at 15 nm[*]). However, this led to problems in making reliable contact to the Si NWs, since the

[*] Increasing the SiO_2 thickness beyond 15 nm adversely affected the etch fidelity.

deposited metal would be more prone to becoming discontinuous at the ends of the Si NWs where there is a step of 33 nm (the starting SOI thickness) (Figure 4-9.B). It should be noted, however, that more advanced metal deposition systems capable of depositing metal conformally would eliminate this constraint.

While reducing the epoxy thickness over the Ti film was necessary to achieve high-fidelity pattern transfer of the Pt NW mask, it was not sufficient. This required improving the epoxy recipe to make it more resistant to undercutting without making it overly difficult to etch using O_2 or BCl_3. (A O_2 etch of 5 mTorr at 40 W always preceded the initial BCl_3 etch to remove epoxy from between the SNAP-fabricated Pt NWs.) After some trial and error, the optimal epoxy recipe was determined to be a modified version of Allied High Tech (Rancho Dominguez, CA) Epoxy Bond 110. (5 drops part A, 1 drop part B, 2 drops of dibutyl phthalate, and diluted with 10 ml of anhydrous tetrahydrofuran. The dibutyl phthalate is a plasticizer that makes the epoxy easier to etch from between Pt NWs.)

4.6.3 Memory testing

The memory circuit was tested using a Probe 2000 (San Jose, CA) custom-built probe card (Appendix 4.2) and a Keithley 707A switching matrix in conjunction with a Keithley 7174A low-current matrix card for off-chip demultiplexing. Individual ebits (containing 4–16 crossbar junctions, but most often containing nine crossbar junctions) were electrically addressed within the 2-D cross-point array by the intersection of 2–4 sequential Si NW bottom electrodes and 2–4 sequential Ti NW top electrodes. Individual

molecular junctions were set to their low resistance, or ‘1’ state, through the application of a positive 1.5–2.3 V pulse (voltages are referenced to the bottom Si NW electrode) of 0.2-second duration. A junction was set to its ‘0’ or high resistance state through application of a −1.5 V pulse, also of 0.2-second duration. To avoid switching an entire column or row of bits, the switching voltage was split between the two electrodes defining the ebit. Thus, to write a ‘1’ with +2 V, a single Si NW electrode is charged to +1 V, while a single Ti NW electrode is set to −1 V, and only where they cross does the junction receive the full +2 V switching voltage. Half-selected bits, that is, bits receiving only half the switching voltage, were never observed to switch. Individual ebits were read by applying a small, non-perturbing +0.2 V bias to the bottom Si NW electrode and grounding the top Ti NW electrode through a Stanford Research Systems SR-570 current pre-amplifier. Bits not being read were held at ground to reduce parasitic current pathways through the crossbar array. Note that all the electrical writing and reading operations described in this work were done sequentially.

Configuring the memory circuit for information storage proceeded as follows. Initially, all ebits were read with +0.2 V to document their baseline current. The value of this baseline current varied from being greater than the current through the junction when set to its low-resistance or ‘1’ state to being less than the current through the junction when set to its high-resistance or ‘0’ state. However, after a (good) bit had been switched though the application of ± 1.5 V, it performed reliably (*i.e.*, on current > off current) until it no longer exhibited switching behavior. After the baseline current was read, all ebits were switched to their ‘1’ state, read out, then set to their ‘0’ state and again read out (Figure 4-13 shows raw data). Good ebits were identified as those with 1/0 current

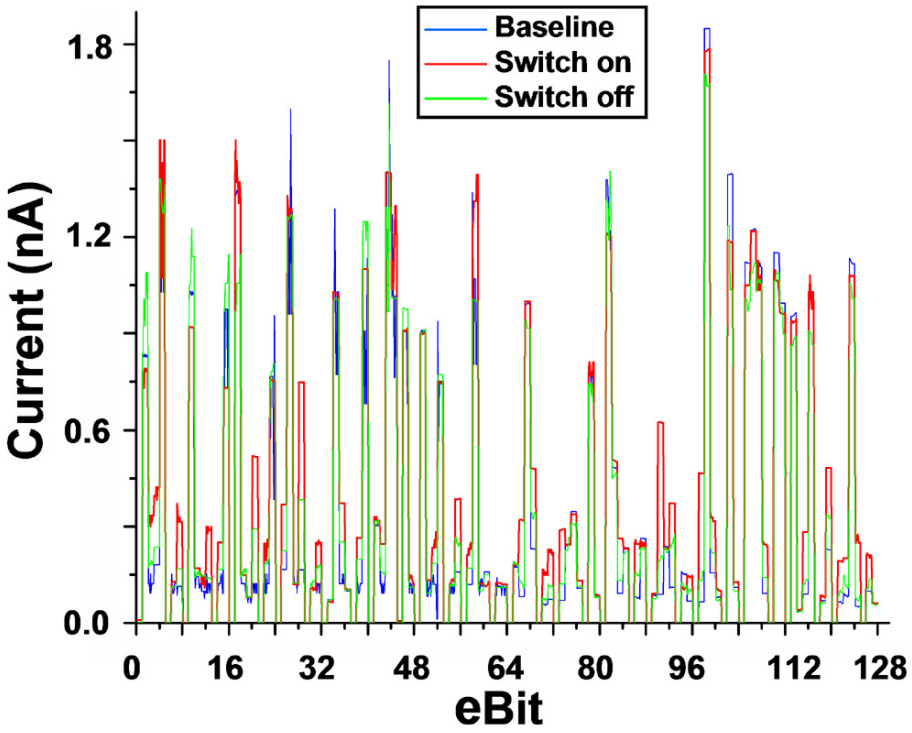


Figure 4-13. Raw switching data from a molecular electronic crossbar memory circuit. The raw data in this figure was used to generate the 1/0 plot and defect matrix shown in Figure 4-6.

ratios roughly greater than or equal to 1.5. Bad ebits fell into a few classes, with the two most common groups being ebits that were either poor switches with little or no switching response or open circuits. In both cases, the 1/0 ratio was unity. Adjacent Ti top electrodes that were shorted together were identified when the ebits addressed by those electrodes were not independently addressable. This is evidenced by an 8-bit periodicity in the response of bits sharing a single Si NW bottom electrode and the shorted Ti top electrodes. This can be seen from the bit matrix in Figure 4-6.B where the shorting of top electrodes T2 and T3 results in nearly equivalent responses from bits 13 & 21 and bits 15 & 23. A more-severe case of top Ti NW shorting is shown in Figure 4-14, which corresponds to the memory circuit shown in Figure 4-12. Even though this type of defect is not completely fatal (*i.e.*, two rows of ebits could still be utilized as a single row

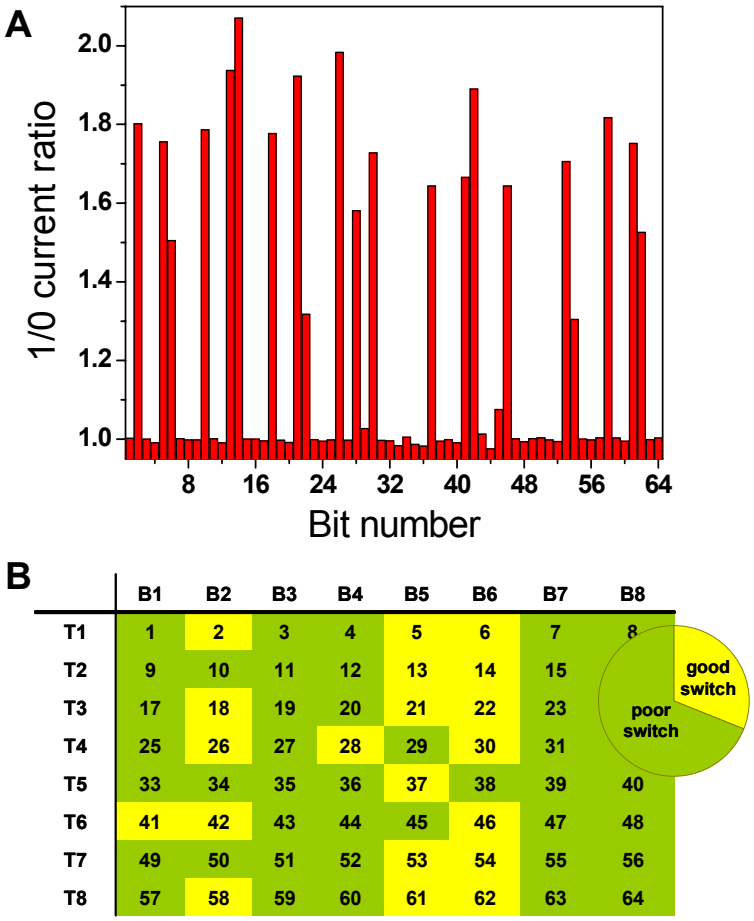


Figure 4-14. Data from a memory circuit with extensive Ti NW top electrode shorting. **A.** 1/0 current ratio measurements from 64 bits of the memory circuit shown in Figure 4-12. **B.** Defect map of the good and defective ebits with a pie chart showing the testing statistics. Good ebits were defined as ebits with 1/0 ratios greater than 1.2 (31% of the tested bits). Note that most of the good ebits (yellow) are clustered together within columns (Si NW bottom electrodes), indicating severe Ti NW (rows) shorting.

of twice-as-large ebits), we did not use ebits associated with shorted top electrode defects.

Once the good ebits were identified, they were used to store and read out small strings of information written in standard ASCII code. The maximum number of ebits that could be tested was 180, but our electronics were configured to test 128 ebits (less than 1 percent of the actual crossbar). Based on results from similarly fabricated memory

circuits, we believe this small subset of measured bits is representative and sufficient to demonstrate the key concepts of this memory.

To increase the measured current, volatility measurements were carried out with approximately 100 (~30 ebits) junctions in parallel. The junctions were switched to their '1' or low resistance state, as described above, and the current was periodically measured through the parallel combination of all 100 junctions at discrete time points. Note that defective switches stuck in a low conductivity state contribute little signal to the parallel combination while defective switches stuck in a high conductivity state only add a constant offset to the decaying current. There could be an unknown number of defective switches (and hence junctions) stuck in their '1' state; however, the current through a parallel combination of functional and defective junctions will decay with the same time constant as that of the functional junctions as long as the number of defective junctions is not too large. We occasionally observed parallel combinations with a large fraction of defective junctions. These were identified by an approximately constant measured current as a function of time. Data from these parallel combinations were not used.

If a defective junction is shorted (*i.e.*, the Ti top electrode is shorted to the bottom Si electrode through direct metal contact), the current through the junction would be orders of magnitude higher than expected and readily identified. We did not observe such junctions in the memory circuits described above.

4.7 Limitations of the SNAP process for crossbar circuits

The nanofabrication methods described above for creating the 160,000-bit crossbar memory circuit can be significantly extended in terms of both memory size and bit density. For [2]rotaxane-based molecular electronic memory circuits, proper choice of electrode materials within the crossbar has proven to be very important for successful memory operation[61]; that is, having Si bottom electrodes and metallic top electrodes with a thin Ti layer to protect the underlying [2]rotaxane monolayer was key. Arrays of SNAP-fabricated Pt NWs only serve as stencils for forming the crossbar electrodes. To be used in a crossbar memory, the SNAP NW pattern must be transferred to Si or Ti NWs for the bottom and top electrodes, respectively. Thus, it is not just the SNAP process, but the ability to translate the initially deposited SNAP NWs to form other NWs that ultimately limits the size and density of the circuitry that can be fabricated.

Figure 4-15 (left micrograph) shows an array of 7-nm-wide, 15-nm-tall single crystal Si NWs patterned at 13-nm pitch. This array could be used to produce to a crossbar molecular memory circuit at about six times the density of this work (~ 6×10^{11} bits cm^{-2}). While this array may not represent the density limit of what could be achieved, densities in excess of 1×10^{12} cm^{-2} may difficult to obtain using these patterning methods and conventional nanofabrication tools. Similarly, the 160,000-bit crossbar described herein can be extended in terms of the total number of bits by using larger-element SNAP NW arrays. Figure 4-15 (right two micrographs) shows SEM images of an array of 1400 Si NWs formed using the SNAP method. An array this size makes possible

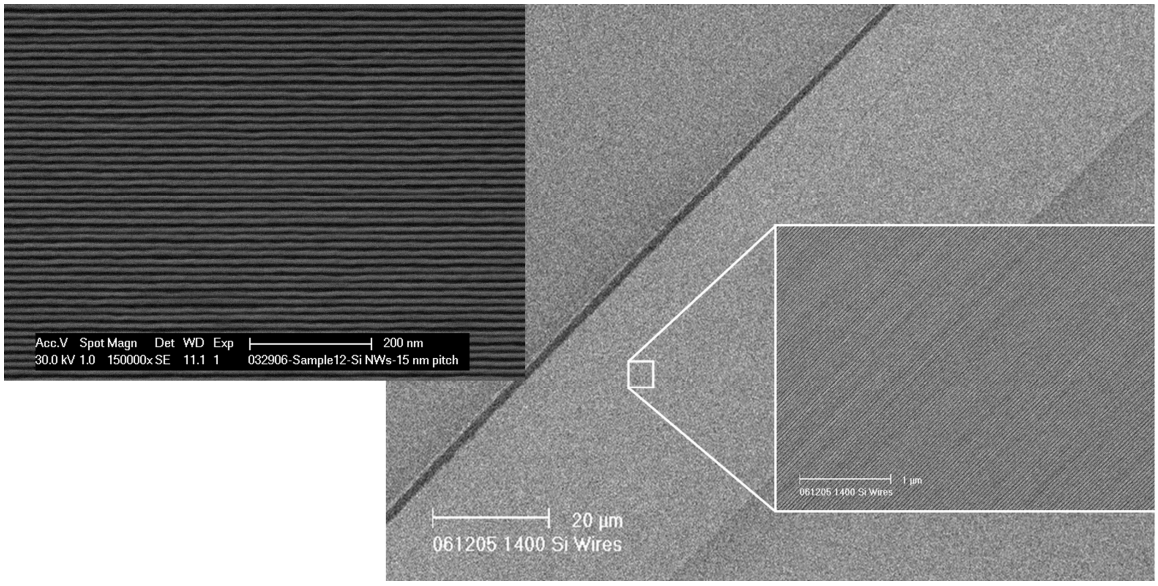


Figure 4-15. Next-generation crossbar molecular memory circuits using SNAP patterning. (left) An array of 7-nm-wide Si NWs patterned at 13-nm pitch could produce a crossbar molecular memory circuit with six times the bit density of this work. **(right)** An array of 1400 Si NWs patterned at 33-nm pitch could provide enough nanowires to produce an approximately two-million-bit crossbar molecular memory circuit. The inset shows an expanded view of the array, which is virtually free of defective nanowires.

the construction of an approximately two-million-bit crossbar molecular memory circuit and it is certainly possible to further expand this concept to substantially larger structures.

From a manufacturing perspective, a significant limitation of the SNAP process is that each NW array must be fabricated serially using a labor-intensive process (despite SNAP being a parallel patterning method in that all NWs within an array are created simultaneously). For instance, in a single day a worker can usually fabricate not more than 10–20 arrays of Si NWs. However, a recent collaboration with Stan Williams' group at Hewlett Packard labs (Palo Alto, CA) has demonstrated that nanoimprinting can be used to replicate SNAP NWs and to form crossbar structures[47]. This indicates that high-

throughput parallel fabrication methods can be developed, even at the near molecular-densities described in this work.

4.8 Concluding remarks

Many challenges remain to be addressed before the type of crossbar molecular memory described here can be practically implemented. For example, areas of future interest include finding faster and more-robust molecular switches, addressing nanofabrication challenges associated with improving the fidelity of these tools and procedures, and meeting engineering challenges such as those involved with combining demultiplexing architectures, such as those described in Chapter 3, with crossbar circuits[69]. Nevertheless, this circuit stands as a new benchmark for nanoelectronic device integration and provides evidence that at least some of the most challenging scientific issues associated with integrating nanowires, molecular materials, and defect-tolerant circuit architectures at extreme dimensions are solvable. The circuits described in this work represent significant advances in sub-lithographic patterning, large-scale assembly of nanoscale electronic devices, and the integration of molecular and solid state materials. Furthermore, recently published nanoimprinting results imply that methods for the high-throughput manufacturing of these types of circuits are possible[46-48].

4.9 References

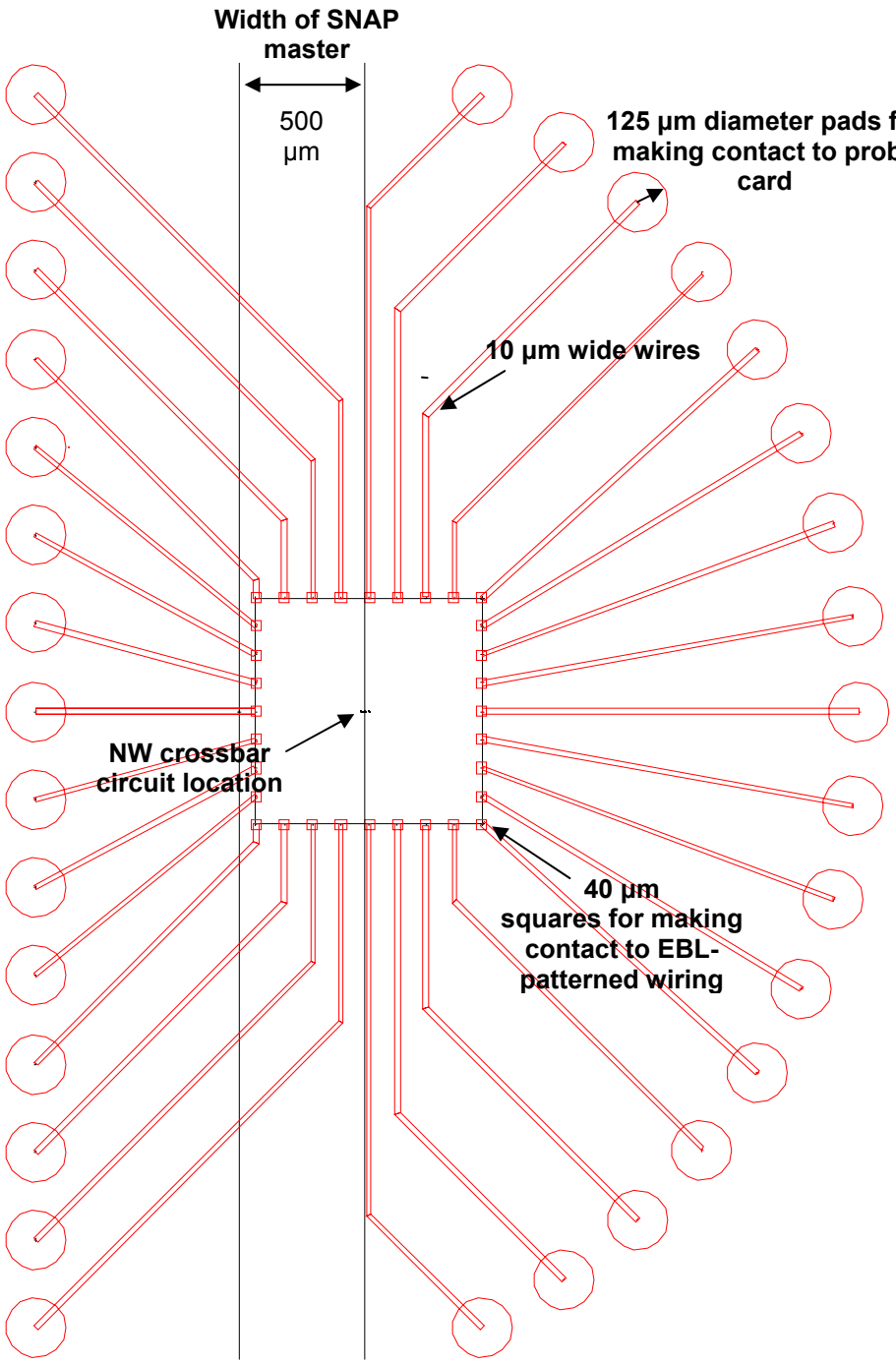
1. The International Technology Roadmap for Semiconductors (ITRS), 2005 Edn. San Jose, CA, Semiconductor Industry Association.
2. Cavin, R. K. et al. A long-term view of research targets in nanoelectronics. Journal of Nanoparticle Research 7, 573–586 (2005).
3. Galatsis, K. et al. Emerging memory devices – Nontraditional possibilities based on nanomaterials and nanostructures. Ieee Circuits & Devices 22, 12–21 (2006).
4. Cahen, D. & Hodes, G. Molecules and electronic materials. Advanced Materials 14, 789–798 (2002).
5. Heath, J. R. & Ratner, M. A. Molecular electronics. Physics Today 56, 43–49 (2003).
6. Aviram, A. & Ratner, M. A. Molecular Rectifiers. Chemical Physics Letters 29, 277–283 (1974).
7. Kornilovitch, P. E., Bratkovsky, A. M. & Williams, R. S. Current rectification by molecules with asymmetric tunneling barriers. Physical Review B, 66, (2002).
8. Blum, A. S. et al. Molecularly inherent voltage-controlled conductance switching. Nature Materials 4, 167–172 (2005).
9. Luo, Y. et al. Two-dimensional molecular electronics circuits. ChemPhysChem 3, 519–525 (2002).
10. McCreery, R. L. Molecular Electronic Junctions. Chem. Mater. 16, 4477–4496 (2004).
11. Joachim, C. & Ratner, M. A. Molecular Electronics Special Feature: Molecular electronics: Some views on transport junctions and beyond. PNAS 102, 8801–8808 (2005).
12. Huang, Y. et al. Logic gates and computation from assembled nanowire building blocks. Science 294, 1313–1317 (2001).
13. Duan, X. F., Huang, Y. & Lieber, C. M. Nonvolatile memory and programmable logic from molecule-gated nanowires. Nano Letters 2, 487–490 (2002).
14. Bachtold, A., Hadley, P., Nakanishi, T. & Dekker, C. Logic circuits with carbon nanotube transistors. Science 294, 1317–1320 (2001).
15. Chen, Z. H. et al. An integrated logic circuit assembled on a single carbon nanotube. Science 311, 1735–1735 (2006).
16. Lehn, J. M. Toward self-organization and complex matter. Science 295, 2400–2403 (2002).
17. Tseng, H. R., Vignon, S. A. & Stoddart, J. F. Toward chemically controlled nanoscale molecular machinery. Angewandte Chemie-International Edition 42, 1491–1495 (2003).
18. Stoddart, J. F. & Tseng, H. R. Chemical synthesis gets a fillip from molecular recognition and self-assembly processes. Proceedings of the National Academy of Sciences of the United States of America 99, 4797–4800 (2002).
19. Beckman, R. et al. Spiers Memorial Lecture – Molecular mechanics and molecular electronics. Faraday Discussions 131, 9–22 (2006).

20. Choi, J. W. et al. Ground-state equilibrium thermodynamics and switching kinetics of bistable [2]rotaxanes switched in solution, polymer gels, and molecular electronic devices. Chemistry-a European Journal 12, 261–279 (2005).
21. David W. Steuerman, H.-R. T., Andrea J. Peters, Amar H. Flood, Jan O. Jeppesen, Kent A. Nielsen, J. Fraser Stoddart, James R. Heath. Molecular-Mechanical Switch-Based Solid-State Electrochromic Devices. Angewandte Chemie International Edition 43, 6486–6491 (2004).
22. DeIonno, E., Tseng, H. R., Harvey, D. D., Stoddart, J. F. & Heath, J. R. Infrared Spectroscopic Characterization of [2]Rotaxane Molecular Switch Tunnel Junction Devices. J. Phys. Chem. B 110, 7609–7612 (2006).
23. Flood, A. H. et al. The role of physical environment on molecular electromechanical switching. Chemistry-a European Journal 10, 6558–6564 (2004).
24. Pease, A. R. et al. Switching devices based on interlocked molecules. Accounts of Chemical Research 34, 433–444 (2001).
25. Tseng, H. R., Wu, D. M., Fang, N. X. L., Zhang, X. & Stoddart, J. F. The metastability of an electrochemically controlled nanoscale machine on gold surfaces. Chemphyschem 5, 111–116 (2004).
26. Asakawa, M. et al. Current/voltage characteristics of monolayers of redox-switchable [2]catenanes on gold. Advanced Materials 12, 1099–1102 (2000).
27. Katz, E., Lioubashevsky, O. & Willner, I. Electromechanics of a redox-active rotaxane in a monolayer assembly on an electrode. J. Am. Chem. Soc. 126, 15520–32 (2004).
28. Katz, E., Baron, R., Willner, I., Richke, N. & Levine, R. D. Temperature-dependent and friction-controlled electrochemically induced shuttling along molecular strings associated with electrodes. Chemphyschem 6, 2179–2189 (2005).
29. Katz, E., Sheeney-Haj-Ichia, L. & Willner, I. Electrical contacting of glucose oxidase in a redox-active rotaxane configuration. Angewandte Chemie-International Edition 43, 3292–3300 (2004).
30. Jang, S. S. et al. Structures and properties of self-assembled monolayers of bistable [2]rotaxanes on Au(111) surfaces from molecular dynamics simulations validated with experiment. Journal of the American Chemical Society 127, 1563–1575 (2005).
31. Jang, S. S. et al. Molecular dynamics simulation of amphiphilic bistable [2]rotaxane Langmuir monolayers at the air/water interface. Journal of the American Chemical Society 127, 14804–14816 (2005).
32. Jang, Y. H. & Goddard, W. A. Mechanism of oxidative shuttling for [2]rotaxane in a Stoddart-Heath molecular switch: Density functional theory study with continuum-solvation model. Journal of Physical Chemistry B 110, 7660–7665 (2006).
33. Deng, W. Q., Muller, R. P. & Goddard, W. A. Mechanism of the Stoddart-Heath bistable rotaxane molecular switch. Journal of the American Chemical Society 126, 13562–13563 (2004).
34. Steuerman, D. W. et al. Molecular-mechanical switch-based solid-state electrochromic devices. Angew. Chem. Int. Ed. 43, 6486–91 (2004).

35. Collier, C. P. et al. Molecular-based electronically switchable tunnel junction devices. Journal of the American Chemical Society 123, 12632–12641 (2001).
36. Collier, C. P. et al. A [2]catenane-based solid state electronically reconfigurable switch. Science 289, 1172–1175 (2000).
37. Jeppesen, J. O. et al. Amphiphilic bistable rotaxanes. Chemistry-a European Journal 9, 2982–3007 (2003).
38. Nielsen, M. B. et al. Binding studies between tetrathiafulvalene derivatives and cyclobis(paraquat-p-phenylene). Journal of Organic Chemistry 66, 3559 (2001).
39. Collier, C. P. et al. Electronically configurable molecular-based logic gates. Science 285, 391–394 (1999).
40. Lee, M. H., Kim, Y. K. & Choi, Y. H. A defect-tolerant memory architecture for molecular electronics. Ieee Transactions on Nanotechnology 3, 152–157 (2004).
41. DeHon, A. & Naeimi, H. Seven strategies for tolerating highly defective fabrication. Ieee Design & Test of Computers 22, 306–315 (2005).
42. Heath, J. R., Kuekes, P. J., Snider, G. S. & Williams, R. S. A defect-tolerant computer architecture: Opportunities for nanotechnology. Science 280, 1716–1721 (1998).
43. Ito, T. & Okazaki, S. Pushing the limits of lithography. Nature 406, 1027–1031 (2000).
44. Huang, Y., Duan, X. F., Wei, Q. Q. & Lieber, C. M. Directed assembly of one-dimensional nanostructures into functional networks. Science 291, 630–633 (2001).
45. Whang, D., Jin, S., Wu, Y. & Lieber, C. M. Large-scale hierarchical organization of nanowire arrays for integrated nanosystems. Nano Letters 3, 1255–1259 (2003).
46. Chen, Y. et al. Nanoscale molecular-switch devices fabricated by imprint lithography. Applied Physics Letters 82, 1610–1612 (2003).
47. Jung, G. Y. et al. Circuit fabrication at 17 nm half-pitch by nanoimprint lithography. Nano Letters 6, 351–354 (2006).
48. Wu, W. et al. One-kilobit cross-bar molecular memory circuits at 30-nm half-pitch fabricated by nanoimprint lithography. Applied Physics a-Materials Science & Processing 80, 1173–1178 (2005).
49. Ziegler, M. M. et al. in Molecular Electronics Iii 312–330 (New York Acad Sciences, New York, 2003).
50. Stan, M. R., Franzon, P. D., Goldstein, S. C., Lach, J. C. & Ziegler, M. M. Molecular electronics: From devices and interconnect to circuits and architecture. Proceedings of the Ieee 91, 1940–1957 (2003).
51. Kuekes, P. J., Stewart, D. R. & Williams, R. S. The crossbar latch: Logic value storage, restoration, and inversion in crossbar circuits. Journal of Applied Physics 97 (2005).
52. Snider, G., Kuekes, P., Hogg, T. & Williams, R. S. Nanoelectronic architectures. Applied Physics a-Materials Science & Processing 80, 1183–1195 (2005).
53. Chen, Y. et al. Nanoscale molecular-switch crossbar circuits. Nanotechnology 14, 462–468 (2003).

54. Beckman, R., Johnston-Halperin, E., Luo, Y., Green, J. E. & Heath, J. R. Bridging dimensions: Demultiplexing ultrahigh-density nanowire circuits. Science 310, 465–468 (2005).
55. Zhong, Z. H., Wang, D. L., Cui, Y., Bockrath, M. W. & Lieber, C. M. Nanowire crossbar arrays as address decoders for integrated nanosystems. Science 302, 1377–1379 (2003).
56. DeHon, A., Goldstein, S. C., Kuekes, P. J. & Lincoln, P. Nonphotolithographic nanoscale memory density prospects. Ieee Transactions on Nanotechnology 4, 215–228 (2005).
57. Waser, R. & Rudiger, A. Ferroelectrics – Pushing towards the digital storage limit. Nature Materials 3, 81–82 (2004).
58. Allwood, D. A. et al. Magnetic domain-wall logic. Science 309, 1688–1692 (2005).
59. Beckman, R. A. et al. Fabrication of conducting Si nanowire arrays. Journal of Applied Physics 96, 5921–5923 (2004).
60. Melosh, N. A. et al. Ultrahigh-density nanowire lattices and circuits. Science 300, 112–115 (2003).
61. Diehl, M. R. et al. Single-walled carbon nanotube based molecular switch tunnel junctions. Chemphyschem 4, 1335–1339 (2003).
62. Stewart, D. R. et al. Molecule-independent electrical switching in Pt/organic monolayer/Ti devices. Nano Letters 4, 133–136 (2004).
63. Vieu, C. et al. Electron beam lithography: resolution limits and applications. Applied Surface Science 164, 111–117 (2000).
64. Parkin, S. S. P. et al. Exchange-biased magnetic tunnel junctions and application to nonvolatile magnetic random access memory (invited). Journal of Applied Physics 85, 5828–5833 (1999).
65. McCreery, R. et al. Molecular rectification and conductance switching in carbon-based molecular junctions by structural rearrangement accompanying electron injection. J. Am. Chem. Soc. 125, 10748–58 (2003).
66. Nowak, A. M. & McCreery, R. L. In situ Raman spectroscopy of bias-induced structural changes in nitroazobenzene molecular electronic junctions. Journal of the American Chemical Society 126, 16621–16631 (2004).
67. Nowak, A. M. & McCreery, R. L. Characterization of carbon/nitroazobenzene/titanium molecular electronic junctions with photoelectron and Raman spectroscopy. Analytical Chemistry 76, 1089–1097 (2004).
68. Handbook of Semiconductor Manufacturing Technology (eds. Nishi, Y. & Doering, R.) (Marcel Dekker, Inc., New York, 2000).
69. Ziegler, M. M. & Stan, M. R. CMOS/nano co-design for crossbar-based molecular electronic systems. Ieee Transactions on Nanotechnology 2, 217–230 (2003).

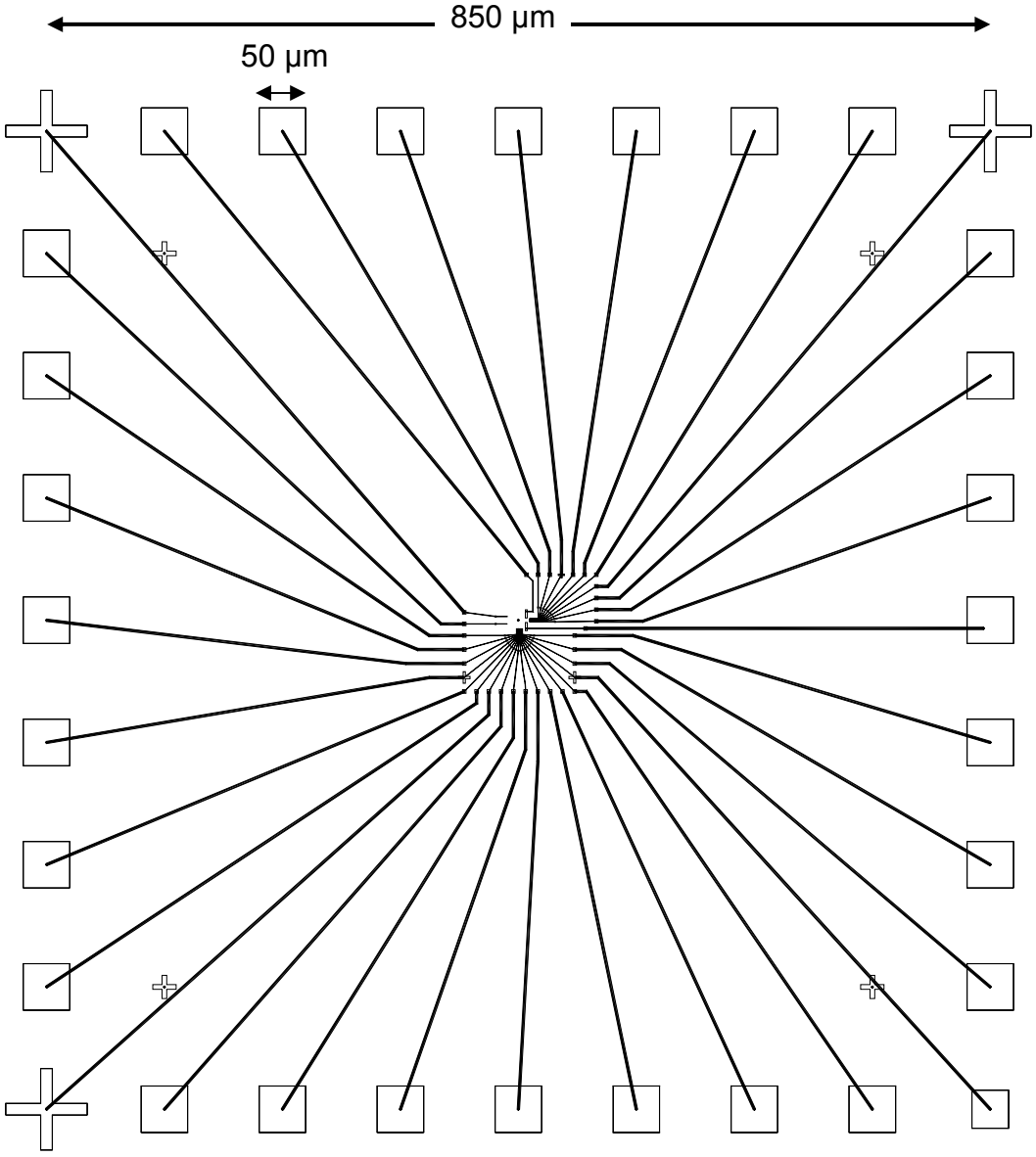
Appendix 4.1 Details of lithographically-patterned structures



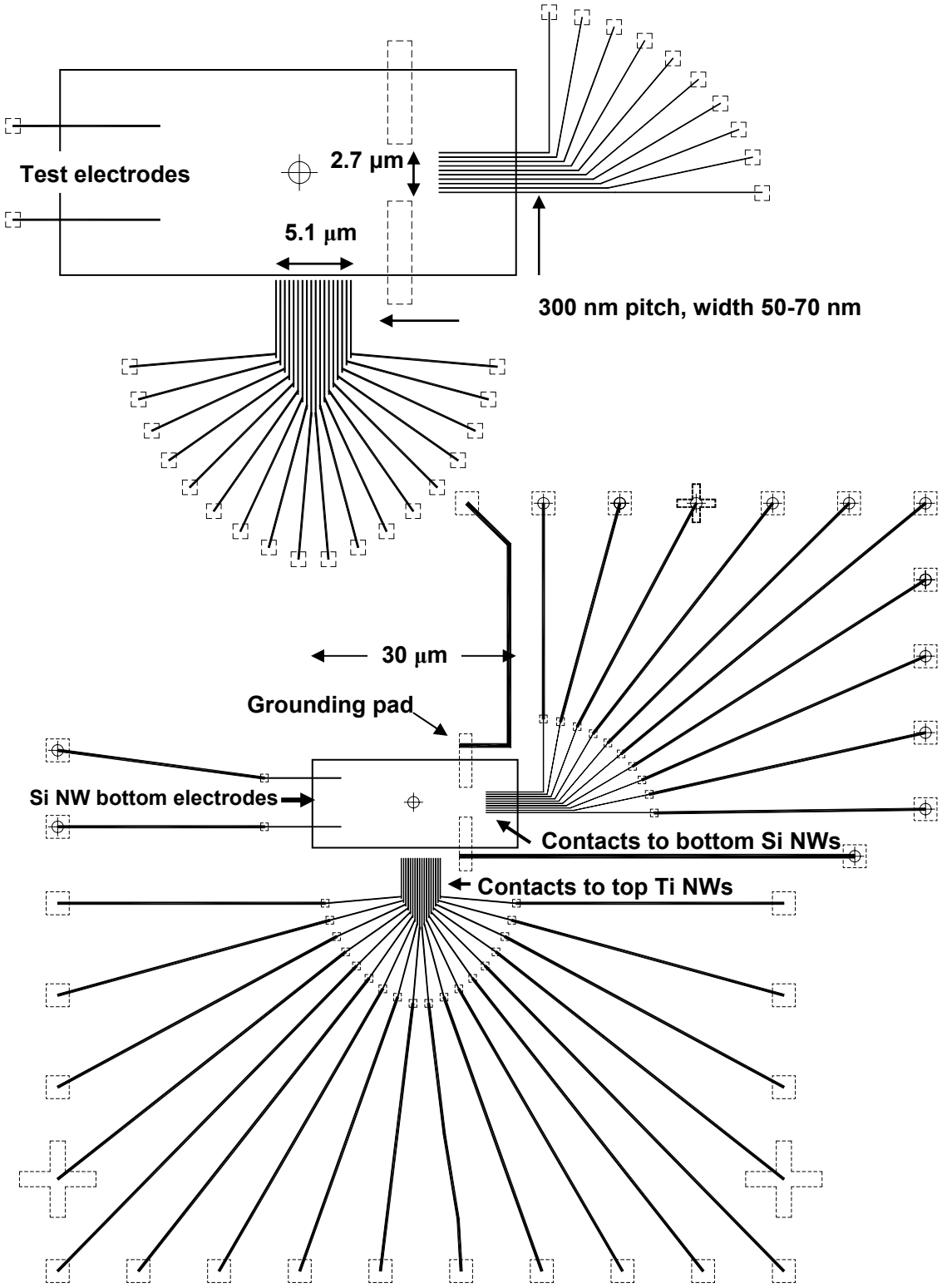
Optical lithography patterned structure. The coordinates for the pads are on the next page. A printed photomask (Output City, Eoway CA) was used to expose the pattern in AZ 5214 (Clariant). AZ 400K was used for development and metal lift-off was in acetone.

Coordinates for the optical mask pattern on previous page (1 unit = 1 micrometer) with the origin at the lower left pad.

PAD #	*X COORDINATE*	*Y COORDINATE*
1	0.0000	0.0000
2	0.0000	350.0000
3	0.0000	700.0000
4	0.0000	1050.0000
5	0.0000	1400.0000
6	0.0000	1750.0000
7	0.0000	2100.0000
8	0.0000	2450.0000
9	0.0000	2800.0000
10	0.0000	3150.0000
11	0.0000	3500.0000
12	0.0000	3850.0000
13	0.0000	4200.0000
14	0.0000	4550.0000
15	0.0000	4900.0000
16	1778.0000	4900.0000
17	2101.6561	4705.6744
18	2395.6589	4468.8663
19	2654.4711	4194.0355
20	2873.2185	3886.3582
21	3047.7811	3551.6291
22	3174.8715	3196.1523
23	3252.0959	2826.6227
24	3278.0000	2450.0000
25	3252.0959	2073.3773
26	3174.8715	1703.8477
27	3047.7811	1348.3709
28	2873.2185	1013.6418
29	2654.4711	705.9645
30	2395.6589	431.1337
31	2101.6561	194.3256
32	1778.0000	0.0000

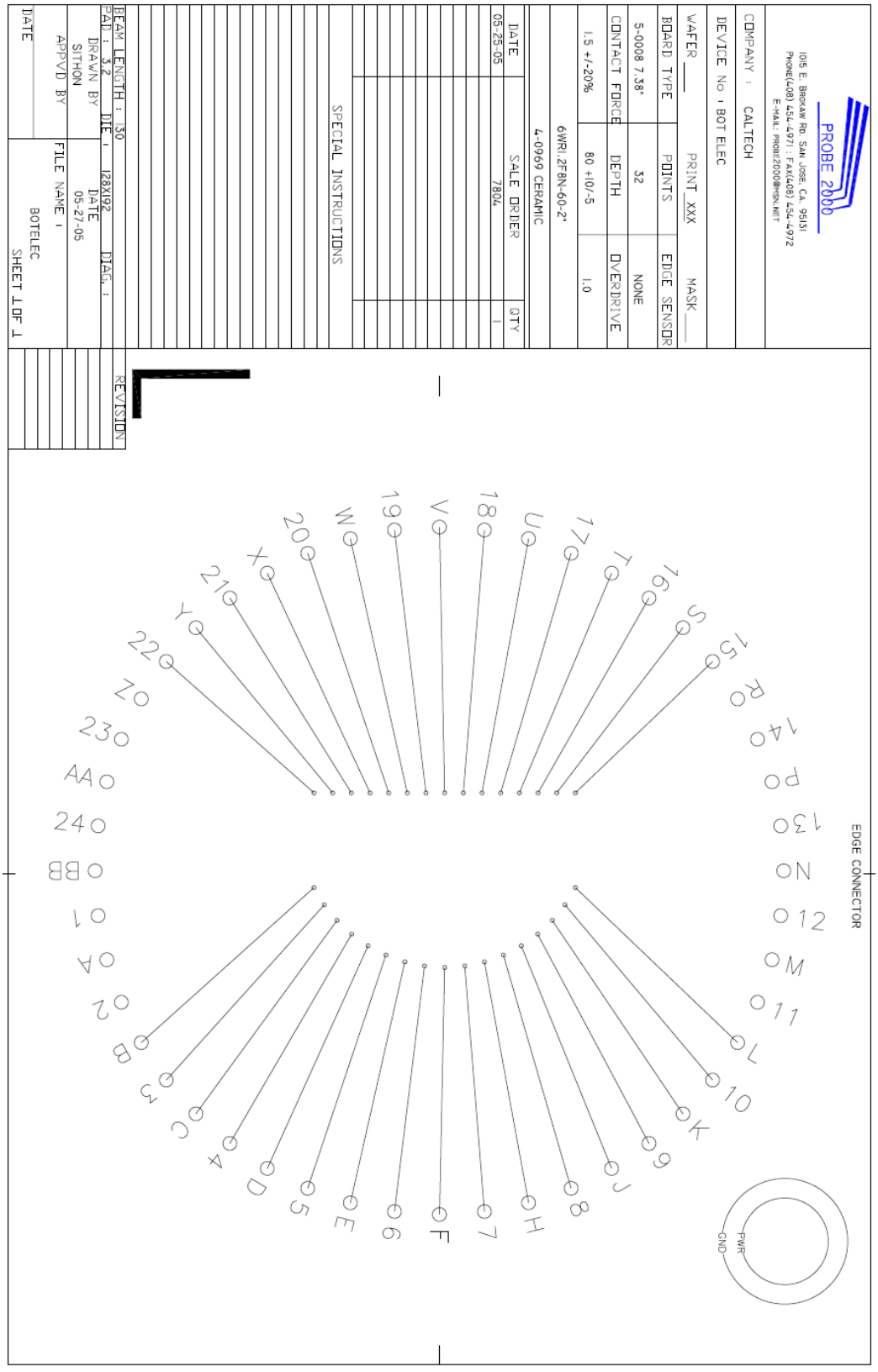


Outermost electron-beam lithography written structures. The large 50-μm pads and crosses at the periphery make contact to optical-lithography-defined pads that in turn fan-out to large circular pads (125 μm diameter) for making contact to a custom-built probe card. All electron-beam lithography was done using an FEI XL-30 SEM with the Nanometer Pattern Generation System (NPGS) version 6.0 (J.C. Nabity Systems) to expose regions of 3% polymethyl-methacrylate (PMMA) over 2.25% MMA. Pattern development was done with 1:3 methyl isobutyl ketone to isopropyl alcohol. Metal depositions were done using an electron-beam evaporator (Semicore Corp, CHA-Mark 40; Freemont, CA), and lift-off was in acetone.



Intermediate-to-smallest EBL-patterned structures.

Appendix 4.2 Memory probe card specifications



Appendix 4.3 NI LabWindows /CVI code used for memory reading/writing operations

---

Note that much of the code below has been commented out (/*…*/). I nonetheless left those portions intact in its original location within the code. This code was written primarily by Dr. Yi Luo for use with a Keithley 707A switching matrix, National Instruments (NI) DAQ PC card (v. 4.8), and a Stanford Research Systems SR-570 current pre-amplifier.

```
#include <gpib.h>
//#include <windows.h>
#include <utility.h>
//#include "decl-32.h"
#include <stdio.h>
#include <string.h>
#include <userint.h>
#include <dataacq.h>
#include <ansi_c.h>
#include "MUX_AC.h"

static int daq, daq1;
FILE *fp_out;
int Device1;
int cross_point[9][9],set_bit[9][9];
int num_read, all_switch,all_control=-1, ramp, ramp_num=20;
double time_write, time_read, volt_write_on,volt_write_off, volt_read, volt_hold, threshold_high,threshold_low;
double adch0,adch1, volt_ramp0, volt_ramp1, ramp_rate;
const char tmp_file[10]="tmp.dat";

void main(){
        int i;
        Device1=ibdev(0,18,0,10,1,0);                                   /* initiate 707A  */
        ibwrt(Device1,"REMOTE",6);                                              /* enable remote mode   */
        ibwrt(Device1,"E0X",3);                                                 /* Point to present relays  */

        daq = LoadPanel (0, "MUX_AC.uir", MUX);
        DisplayPanel (daq);
        i=AI_Clear (1);
        RunUserInterface ();
}
int select_ind (int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)
        {
        daq1 = LoadPanel (1, "MUX_AC.uir", MUX1);
        DisplayPanel (daq1);
        return 1;
        }
int close_selection(int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)
        {
        int i,m;
        i=HidePanel(daq1);
        return 0;
        }
int switch_control(int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)
        {
                        int m;
                        if(all_control==-1){
                                m=SetCtrlAttribute(daq,MUX_ALL_SWITCHES, ATTR_DIMMED, 0);
                        }
                        else{
                                m=SetCtrlAttribute(daq,MUX_ALL_SWITCHES, ATTR_DIMMED, 1);
                        }
                        all_control=all_control*(-1);
                        return 1;
        }
int configure_ind (int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)
        {
        int i,j,k,m,i_ramp;
        char c[5],d[6];
        if (all_control!=1){
        m = GetCtrlVal (daq, MUX_Switch1_1, &cross_point[1][1]);
        m = GetCtrlVal (daq, MUX_Switch1_2, &cross_point[1][2]);
        m = GetCtrlVal (daq, MUX_Switch1_3, &cross_point[1][3]);
```

```
m = GetCtrlVal (daq, MUX_Switch1_4, &cross_point[1][4]);
m = GetCtrlVal (daq, MUX_Switch1_5, &cross_point[1][5]);
m = GetCtrlVal (daq, MUX_Switch1_6, &cross_point[1][6]);
m = GetCtrlVal (daq, MUX_Switch1_7, &cross_point[1][7]);
m = GetCtrlVal (daq, MUX_Switch1_8, &cross_point[1][8]);
m = GetCtrlVal (daq, MUX_Switch2_1, &cross_point[2][1]);
m = GetCtrlVal (daq, MUX_Switch2_2, &cross_point[2][2]);
m = GetCtrlVal (daq, MUX_Switch2_3, &cross_point[2][3]);
m = GetCtrlVal (daq, MUX_Switch2_4, &cross_point[2][4]);
m = GetCtrlVal (daq, MUX_Switch2_5, &cross_point[2][5]);
m = GetCtrlVal (daq, MUX_Switch2_6, &cross_point[2][6]);
m = GetCtrlVal (daq, MUX_Switch2_7, &cross_point[2][7]);
m = GetCtrlVal (daq, MUX_Switch2_8, &cross_point[2][8]);
m = GetCtrlVal (daq, MUX_Switch3_1, &cross_point[3][1]);
m = GetCtrlVal (daq, MUX_Switch3_2, &cross_point[3][2]);
m = GetCtrlVal (daq, MUX_Switch3_3, &cross_point[3][3]);
m = GetCtrlVal (daq, MUX_Switch3_4, &cross_point[3][4]);
m = GetCtrlVal (daq, MUX_Switch3_5, &cross_point[3][5]);
m = GetCtrlVal (daq, MUX_Switch3_6, &cross_point[3][6]);
m = GetCtrlVal (daq, MUX_Switch3_7, &cross_point[3][7]);
m = GetCtrlVal (daq, MUX_Switch3_8, &cross_point[3][8]);
m = GetCtrlVal (daq, MUX_Switch4_1, &cross_point[4][1]);
m = GetCtrlVal (daq, MUX_Switch4_2, &cross_point[4][2]);
m = GetCtrlVal (daq, MUX_Switch4_3, &cross_point[4][3]);
m = GetCtrlVal (daq, MUX_Switch4_4, &cross_point[4][4]);
m = GetCtrlVal (daq, MUX_Switch4_5, &cross_point[4][5]);
m = GetCtrlVal (daq, MUX_Switch4_6, &cross_point[4][6]);
m = GetCtrlVal (daq, MUX_Switch4_7, &cross_point[4][7]);
m = GetCtrlVal (daq, MUX_Switch4_8, &cross_point[4][8]);
m = GetCtrlVal (daq, MUX_Switch5_1, &cross_point[5][1]);
m = GetCtrlVal (daq, MUX_Switch5_2, &cross_point[5][2]);
m = GetCtrlVal (daq, MUX_Switch5_3, &cross_point[5][3]);
m = GetCtrlVal (daq, MUX_Switch5_4, &cross_point[5][4]);
m = GetCtrlVal (daq, MUX_Switch5_5, &cross_point[5][5]);
m = GetCtrlVal (daq, MUX_Switch5_6, &cross_point[5][6]);
m = GetCtrlVal (daq, MUX_Switch5_7, &cross_point[5][7]);
m = GetCtrlVal (daq, MUX_Switch5_8, &cross_point[5][8]);
m = GetCtrlVal (daq, MUX_Switch6_1, &cross_point[6][1]);
m = GetCtrlVal (daq, MUX_Switch6_2, &cross_point[6][2]);
m = GetCtrlVal (daq, MUX_Switch6_3, &cross_point[6][3]);
m = GetCtrlVal (daq, MUX_Switch6_4, &cross_point[6][4]);
m = GetCtrlVal (daq, MUX_Switch6_5, &cross_point[6][5]);
m = GetCtrlVal (daq, MUX_Switch6_6, &cross_point[6][6]);
m = GetCtrlVal (daq, MUX_Switch6_7, &cross_point[6][7]);
m = GetCtrlVal (daq, MUX_Switch6_8, &cross_point[6][8]);
m = GetCtrlVal (daq, MUX_Switch7_1, &cross_point[7][1]);
m = GetCtrlVal (daq, MUX_Switch7_2, &cross_point[7][2]);
m = GetCtrlVal (daq, MUX_Switch7_3, &cross_point[7][3]);
m = GetCtrlVal (daq, MUX_Switch7_4, &cross_point[7][4]);
m = GetCtrlVal (daq, MUX_Switch7_5, &cross_point[7][5]);
m = GetCtrlVal (daq, MUX_Switch7_6, &cross_point[7][6]);
m = GetCtrlVal (daq, MUX_Switch7_7, &cross_point[7][7]);
m = GetCtrlVal (daq, MUX_Switch7_8, &cross_point[7][8]);
m = GetCtrlVal (daq, MUX_Switch8_1, &cross_point[8][1]);
m = GetCtrlVal (daq, MUX_Switch8_2, &cross_point[8][2]);
m = GetCtrlVal (daq, MUX_Switch8_3, &cross_point[8][3]);
m = GetCtrlVal (daq, MUX_Switch8_4, &cross_point[8][4]);
m = GetCtrlVal (daq, MUX_Switch8_5, &cross_point[8][5]);
m = GetCtrlVal (daq, MUX_Switch8_6, &cross_point[8][6]);
m = GetCtrlVal (daq, MUX_Switch8_7, &cross_point[8][7]);
m = GetCtrlVal (daq, MUX_Switch8_8, &cross_point[8][8]);
}
else{
        m = GetCtrlVal (daq, MUX_ALL_SWITCHES, &all_switch);
        for(i=1;i<=8;i++){
                for(j=1;j<=8;j++){
                cross_point[i][j]=all_switch;
                }
        }
}
m = GetCtrlVal (daq1, MUX1_Switch1_1, &set_bit[1][1]);
m = GetCtrlVal (daq1, MUX1_Switch1_2, &set_bit[1][2]);
m = GetCtrlVal (daq1, MUX1_Switch1_3, &set_bit[1][3]);
m = GetCtrlVal (daq1, MUX1_Switch1_4, &set_bit[1][4]);
m = GetCtrlVal (daq1, MUX1_Switch1_5, &set_bit[1][5]);
m = GetCtrlVal (daq1, MUX1_Switch1_6, &set_bit[1][6]);
m = GetCtrlVal (daq1, MUX1_Switch1_7, &set_bit[1][7]);
m = GetCtrlVal (daq1, MUX1_Switch1_8, &set_bit[1][8]);
m = GetCtrlVal (daq1, MUX1_Switch2_1, &set_bit[2][1]);
m = GetCtrlVal (daq1, MUX1_Switch2_2, &set_bit[2][2]);
m = GetCtrlVal (daq1, MUX1_Switch2_3, &set_bit[2][3]);
m = GetCtrlVal (daq1, MUX1_Switch2_4, &set_bit[2][4]);
m = GetCtrlVal (daq1, MUX1_Switch2_5, &set_bit[2][5]);
m = GetCtrlVal (daq1, MUX1_Switch2_6, &set_bit[2][6]);
m = GetCtrlVal (daq1, MUX1_Switch2_7, &set_bit[2][7]);
m = GetCtrlVal (daq1, MUX1_Switch2_8, &set_bit[2][8]);
m = GetCtrlVal (daq1, MUX1_Switch3_1, &set_bit[3][1]);
m = GetCtrlVal (daq1, MUX1_Switch3_2, &set_bit[3][2]);
m = GetCtrlVal (daq1, MUX1_Switch3_3, &set_bit[3][3]);
m = GetCtrlVal (daq1, MUX1_Switch3_4, &set_bit[3][4]);
m = GetCtrlVal (daq1, MUX1_Switch3_5, &set_bit[3][5]);
m = GetCtrlVal (daq1, MUX1_Switch3_6, &set_bit[3][6]);
m = GetCtrlVal (daq1, MUX1_Switch3_7, &set_bit[3][7]);
```

```
m = GetCtrlVal (daq1, MUX1_Switch3_8, &set_bit[3][8]);
m = GetCtrlVal (daq1, MUX1_Switch4_1, &set_bit[4][1]);
m = GetCtrlVal (daq1, MUX1_Switch4_2, &set_bit[4][2]);
m = GetCtrlVal (daq1, MUX1_Switch4_3, &set_bit[4][3]);
m = GetCtrlVal (daq1, MUX1_Switch4_4, &set_bit[4][4]);
m = GetCtrlVal (daq1, MUX1_Switch4_5, &set_bit[4][5]);
m = GetCtrlVal (daq1, MUX1_Switch4_6, &set_bit[4][6]);
m = GetCtrlVal (daq1, MUX1_Switch4_7, &set_bit[4][7]);
m = GetCtrlVal (daq1, MUX1_Switch4_8, &set_bit[4][8]);
m = GetCtrlVal (daq1, MUX1_Switch5_1, &set_bit[5][1]);
m = GetCtrlVal (daq1, MUX1_Switch5_2, &set_bit[5][2]);
m = GetCtrlVal (daq1, MUX1_Switch5_3, &set_bit[5][3]);
m = GetCtrlVal (daq1, MUX1_Switch5_4, &set_bit[5][4]);
m = GetCtrlVal (daq1, MUX1_Switch5_5, &set_bit[5][5]);
m = GetCtrlVal (daq1, MUX1_Switch5_6, &set_bit[5][6]);
m = GetCtrlVal (daq1, MUX1_Switch5_7, &set_bit[5][7]);
m = GetCtrlVal (daq1, MUX1_Switch5_8, &set_bit[5][8]);
m = GetCtrlVal (daq1, MUX1_Switch6_1, &set_bit[6][1]);
m = GetCtrlVal (daq1, MUX1_Switch6_2, &set_bit[6][2]);
m = GetCtrlVal (daq1, MUX1_Switch6_3, &set_bit[6][3]);
m = GetCtrlVal (daq1, MUX1_Switch6_4, &set_bit[6][4]);
m = GetCtrlVal (daq1, MUX1_Switch6_5, &set_bit[6][5]);
m = GetCtrlVal (daq1, MUX1_Switch6_6, &set_bit[6][6]);
m = GetCtrlVal (daq1, MUX1_Switch6_7, &set_bit[6][7]);
m = GetCtrlVal (daq1, MUX1_Switch6_8, &set_bit[6][8]);
m = GetCtrlVal (daq1, MUX1_Switch7_1, &set_bit[7][1]);
m = GetCtrlVal (daq1, MUX1_Switch7_2, &set_bit[7][2]);
m = GetCtrlVal (daq1, MUX1_Switch7_3, &set_bit[7][3]);
m = GetCtrlVal (daq1, MUX1_Switch7_4, &set_bit[7][4]);
m = GetCtrlVal (daq1, MUX1_Switch7_5, &set_bit[7][5]);
m = GetCtrlVal (daq1, MUX1_Switch7_6, &set_bit[7][6]);
m = GetCtrlVal (daq1, MUX1_Switch7_7, &set_bit[7][7]);
m = GetCtrlVal (daq1, MUX1_Switch7_8, &set_bit[7][8]);
m = GetCtrlVal (daq1, MUX1_Switch8_1, &set_bit[8][1]);
m = GetCtrlVal (daq1, MUX1_Switch8_2, &set_bit[8][2]);
m = GetCtrlVal (daq1, MUX1_Switch8_3, &set_bit[8][3]);
m = GetCtrlVal (daq1, MUX1_Switch8_4, &set_bit[8][4]);
m = GetCtrlVal (daq1, MUX1_Switch8_5, &set_bit[8][5]);
m = GetCtrlVal (daq1, MUX1_Switch8_6, &set_bit[8][6]);
m = GetCtrlVal (daq1, MUX1_Switch8_7, &set_bit[8][7]);
m = GetCtrlVal (daq1, MUX1_Switch8_8, &set_bit[8][8]);
m = GetCtrlVal (daq, MUX_TIME_WRITE, &time_write);
m = GetCtrlVal (daq, MUX_VOLT_WRITE_ON, &volt_write_on);
m = GetCtrlVal (daq, MUX_VOLT_WRITE_OFF, &volt_write_off);
m = GetCtrlVal (daq, MUX_VOLT_HOLD, &volt_hold);
m = GetCtrlVal (daq, MUX_Ramp, &ramp);
m = GetCtrlVal (daq, MUX_Ramp_Rate, &ramp_rate);

/********   starting the loop of configuring   *******/

    /********  test ********/
m=SetCtrlVal(daq,MUX_STOP_SCAN,1);
m=SetCtrlVal(daq,MUX_Config_complete,0);
m=SetCtrlVal(daq,MUX_Memory_Check_Done,0);
ibwrt(Device1,"CA72X",5);                                   /* dummy line */
ibwrt(Device1,"NA72X",5);
for(i=1;i<=8;i++){
        for(j=1;j<=8;j++){
        if (set_bit[i][j]==1){                              /* check if the bit is selected   */
                c[0]='C';
                c[1]='B';
                c[2]=(char)(48+i);
                c[3]='X';
                c[4]='\0';
                ibwrt(Device1,c,4);
                c[0]='N';
                c[1]='A';
                c[2]=(char)(48+i);
                c[3]='X';
                c[4]='\0';
                ibwrt(Device1,c,4);
                if (j<2){
                c[0]='C';
                c[1]='C';
                c[2]=(char)(48+j+8);
                c[3]='X';
                c[4]='\0';
                ibwrt(Device1,c,4);
                c[0]='N';
                c[1]='H';
                c[2]=(char)(48+j+8);
                c[3]='X';
                c[4]='\0';
                ibwrt(Device1,c,4);
                }
                else{
                d[0]='C';
                d[1]='C';
                d[2]='1';                                   /* two-digit  */
                d[3]=(char)(48+j-2);
                d[4]='X';
                d[5]='\0';
                ibwrt(Device1,d,5);
```

```
                                        d[0]='N';
                                        d[1]='H';
                                        d[2]='1';                                                       /* two-digit  */
                                        d[3]=(char)(48+j-2);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        }
                                for(k=1;k<=16;k++){
                                        if((k!=i)&&(k!=j+8)){
                                                if (k<10){
                                                        if (k<=8){
                                                        c[0]='C';
                                                        c[1]='A';           /* apply -1.0 volt to rows from Keithley 5-25-01  */
                                                        c[2]=(char)(48+k);
                                                        c[3]='X';
                                                        c[4]='\0';
                                                        ibwrt(Device1,c,4);
                                                        }
                                                        else{
                                                        c[0]='C';
                                                        c[1]='H';
/*  Ground the columns */
                                                        c[2]=(char)(48+k);
                                                        c[3]='X';
                                                        c[4]='\0';
                                                        ibwrt(Device1,c,4);
                                                        }
                                                }
                                                else{
                                                d[0]='C';
                                                d[1]='H';
                                                d[2]='1';
                                                d[3]=(char)(48+k-10);
                                                d[4]='X';
                                                d[5]='\0';
                                                ibwrt(Device1,d,5);
                                                }
                                        }
                                }
                                /*  set write voltage */
                                Delay(0.1);
                                printf("\a");
                                if(ramp==1){
                                        if(cross_point[i][j]==1){
                                                volt_ramp0=volt_hold;
                                                volt_ramp1=0.0;
                                                for(i_ramp=1; i_ramp<=ramp_num; i_ramp++){
/* ramp-up */
                                                        volt_ramp0=volt_ramp0 + (volt_write_on/2-volt_hold)/ramp_num;
                                                        volt_ramp1=volt_ramp1 + (volt_write_on/2)/ramp_num;
                                                        m = AO_VWrite (1, 0, volt_ramp0);
                                                        m = AO_VWrite (1, 1, (-volt_ramp1-0.06225)/0.9938);
                                                        Delay(volt_write_on/ramp_num/ramp_rate);
                                                }
                                                Delay(time_write);                                                                      /* hold */
                                                for(i_ramp=1; i_ramp<=ramp_num; i_ramp++){
/* ramp-down */
                                                        volt_ramp0=volt_ramp0 - (volt_write_on/2-volt_hold)/ramp_num;
                                                        volt_ramp1=volt_ramp1 - (volt_write_on/2)/ramp_num;
                                                        m = AO_VWrite (1, 0, volt_ramp0);
                                                        m = AO_VWrite (1, 1, (-volt_ramp1-0.06225)/0.9938);
                                                        Delay(volt_write_on/ramp_num/ramp_rate);
                                                }
                                                                                                                        }
                                        else{
                                                volt_ramp0=volt_hold;
                                                volt_ramp1=0.0;
                                                for(i_ramp=1; i_ramp<=ramp_num; i_ramp++){
/* ramp-up */
                                                        volt_ramp0=volt_ramp0 + (volt_write_off/2-volt_hold)/ramp_num;
                                                        volt_ramp1=volt_ramp1 + (volt_write_off/2)/ramp_num;
                                                        m = AO_VWrite (1, 0, volt_ramp0);
                                                        m = AO_VWrite (1, 1, (-volt_ramp1-0.06225)/0.9938);
                                                        Delay(volt_write_off/ramp_num/ramp_rate);
                                                }
                                                Delay(time_write);                                                                      /* hold */
                                                for(i_ramp=1; i_ramp<=ramp_num; i_ramp++){
/* ramp-down */
                                                        volt_ramp0=volt_ramp0 - (volt_write_off/2-volt_hold)/ramp_num;
                                                        volt_ramp1=volt_ramp1 - (volt_write_off/2)/ramp_num;
                                                        m = AO_VWrite (1, 0, volt_ramp0);
                                                        m = AO_VWrite (1, 1, (-volt_ramp1-0.06225)/0.9938);
                                                        Delay(volt_write_off/ramp_num/ramp_rate);
                                                }
                                        }
                                }   /* with ramp */
                                else{
                                        if(cross_point[i][j]==1){
                                                m = AO_VWrite (1, 0, (volt_write_on/2));
                                                m = AO_VWrite (1, 1, (-volt_write_on/2-0.06225)/0.9938);
                                        }
                                        else{
```

```
                                                m = AO_VWrite (1, 0, (volt_write_off/2));
                                                m = AO_VWrite (1, 1, (-volt_write_off/2-0.06225)/0.9938);
                                        }
                                        Delay(time_write);
                                        m = AO_VWrite (1, 0, volt_hold);
                                        m = AO_VWrite (1, 1, -0.06225/0.9938);
                                }                               /* no ramp */
                        /*****  set holding voltage to the row, and Ground to the column  *****/
                                        c[0]='C';
                                        c[1]='A';
                                        c[2]=(char)(48+i);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        c[0]='N';
                                        c[1]='B';
                                        c[2]=(char)(48+i);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        if (j<2){
                                        c[0]='C';
                                        c[1]='H';
                                        c[2]=(char)(48+j+8);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        c[0]='N';
                                        c[1]='C';
                                        c[2]=(char)(48+j+8);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        }
                                        else{
                                        d[0]='C';
                                        d[1]='H';
                                        d[2]='1';                                                       /* two-digit  */
                                        d[3]=(char)(48+j-2);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        d[0]='N';
                                        d[1]='C';
                                        d[2]='1';                                                       /* two-digit  */
                                        d[3]=(char)(48+j-2);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        }
                                /*ibwrt(Device1,"P0X",3);                       open all relays  5-21-01 */
                        }   /*   finish setting one selected bit  */
                        }   /* j */
                        }
                /* close i loop  */
                m=SetCtrlVal(daq,MUX_Config_complete,1);
                return 1;
                }
int configure (int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2)
                {
                int i,j,k,m,i_ramp;
                char c[5],d[6];
                if (all_control !=1){

                m = GetCtrlVal (daq, MUX_Switch1_1, &cross_point[1][1]);
                m = GetCtrlVal (daq, MUX_Switch1_2, &cross_point[1][2]);
                m = GetCtrlVal (daq, MUX_Switch1_3, &cross_point[1][3]);
                m = GetCtrlVal (daq, MUX_Switch1_4, &cross_point[1][4]);
                m = GetCtrlVal (daq, MUX_Switch1_5, &cross_point[1][5]);
                m = GetCtrlVal (daq, MUX_Switch1_6, &cross_point[1][6]);
                m = GetCtrlVal (daq, MUX_Switch1_7, &cross_point[1][7]);
                m = GetCtrlVal (daq, MUX_Switch1_8, &cross_point[1][8]);
                m = GetCtrlVal (daq, MUX_Switch2_1, &cross_point[2][1]);
                m = GetCtrlVal (daq, MUX_Switch2_2, &cross_point[2][2]);
                m = GetCtrlVal (daq, MUX_Switch2_3, &cross_point[2][3]);
                m = GetCtrlVal (daq, MUX_Switch2_4, &cross_point[2][4]);
                m = GetCtrlVal (daq, MUX_Switch2_5, &cross_point[2][5]);
                m = GetCtrlVal (daq, MUX_Switch2_6, &cross_point[2][6]);
                m = GetCtrlVal (daq, MUX_Switch2_7, &cross_point[2][7]);
                m = GetCtrlVal (daq, MUX_Switch2_8, &cross_point[2][8]);
                m = GetCtrlVal (daq, MUX_Switch3_1, &cross_point[3][1]);
                m = GetCtrlVal (daq, MUX_Switch3_2, &cross_point[3][2]);
                m = GetCtrlVal (daq, MUX_Switch3_3, &cross_point[3][3]);
                m = GetCtrlVal (daq, MUX_Switch3_4, &cross_point[3][4]);
                m = GetCtrlVal (daq, MUX_Switch3_5, &cross_point[3][5]);
                m = GetCtrlVal (daq, MUX_Switch3_6, &cross_point[3][6]);
                m = GetCtrlVal (daq, MUX_Switch3_7, &cross_point[3][7]);
                m = GetCtrlVal (daq, MUX_Switch3_8, &cross_point[3][8]);
                m = GetCtrlVal (daq, MUX_Switch4_1, &cross_point[4][1]);
                m = GetCtrlVal (daq, MUX_Switch4_2, &cross_point[4][2]);
                m = GetCtrlVal (daq, MUX_Switch4_3, &cross_point[4][3]);
                m = GetCtrlVal (daq, MUX_Switch4_4, &cross_point[4][4]);
```

```
		m = GetCtrlVal (daq, MUX_Switch4_5, &cross_point[4][5]);
		m = GetCtrlVal (daq, MUX_Switch4_6, &cross_point[4][6]);
		m = GetCtrlVal (daq, MUX_Switch4_7, &cross_point[4][7]);
		m = GetCtrlVal (daq, MUX_Switch4_8, &cross_point[4][8]);
		m = GetCtrlVal (daq, MUX_Switch5_1, &cross_point[5][1]);
		m = GetCtrlVal (daq, MUX_Switch5_2, &cross_point[5][2]);
		m = GetCtrlVal (daq, MUX_Switch5_3, &cross_point[5][3]);
		m = GetCtrlVal (daq, MUX_Switch5_4, &cross_point[5][4]);
		m = GetCtrlVal (daq, MUX_Switch5_5, &cross_point[5][5]);
		m = GetCtrlVal (daq, MUX_Switch5_6, &cross_point[5][6]);
		m = GetCtrlVal (daq, MUX_Switch5_7, &cross_point[5][7]);
		m = GetCtrlVal (daq, MUX_Switch5_8, &cross_point[5][8]);
		m = GetCtrlVal (daq, MUX_Switch6_1, &cross_point[6][1]);
		m = GetCtrlVal (daq, MUX_Switch6_2, &cross_point[6][2]);
		m = GetCtrlVal (daq, MUX_Switch6_3, &cross_point[6][3]);
		m = GetCtrlVal (daq, MUX_Switch6_4, &cross_point[6][4]);
		m = GetCtrlVal (daq, MUX_Switch6_5, &cross_point[6][5]);
		m = GetCtrlVal (daq, MUX_Switch6_6, &cross_point[6][6]);
		m = GetCtrlVal (daq, MUX_Switch6_7, &cross_point[6][7]);
		m = GetCtrlVal (daq, MUX_Switch6_8, &cross_point[6][8]);
		m = GetCtrlVal (daq, MUX_Switch7_1, &cross_point[7][1]);
		m = GetCtrlVal (daq, MUX_Switch7_2, &cross_point[7][2]);
		m = GetCtrlVal (daq, MUX_Switch7_3, &cross_point[7][3]);
		m = GetCtrlVal (daq, MUX_Switch7_4, &cross_point[7][4]);
		m = GetCtrlVal (daq, MUX_Switch7_5, &cross_point[7][5]);
		m = GetCtrlVal (daq, MUX_Switch7_6, &cross_point[7][6]);
		m = GetCtrlVal (daq, MUX_Switch7_7, &cross_point[7][7]);
		m = GetCtrlVal (daq, MUX_Switch7_8, &cross_point[7][8]);
		m = GetCtrlVal (daq, MUX_Switch8_1, &cross_point[8][1]);
		m = GetCtrlVal (daq, MUX_Switch8_2, &cross_point[8][2]);
		m = GetCtrlVal (daq, MUX_Switch8_3, &cross_point[8][3]);
		m = GetCtrlVal (daq, MUX_Switch8_4, &cross_point[8][4]);
		m = GetCtrlVal (daq, MUX_Switch8_5, &cross_point[8][5]);
		m = GetCtrlVal (daq, MUX_Switch8_6, &cross_point[8][6]);
		m = GetCtrlVal (daq, MUX_Switch8_7, &cross_point[8][7]);
		m = GetCtrlVal (daq, MUX_Switch8_8, &cross_point[8][8]);
		}
		else{
			m = GetCtrlVal (daq, MUX_ALL_SWITCHES, &all_switch);
			for(i=1;i<=8;i++){
				for(j=1;j<=8;j++){
				cross_point[i][j]=all_switch;
				}
			}
		}
		m = GetCtrlVal (daq, MUX_TIME_WRITE, &time_write);
		m = GetCtrlVal (daq, MUX_VOLT_WRITE_ON, &volt_write_on);
		m = GetCtrlVal (daq, MUX_VOLT_WRITE_OFF, &volt_write_off);
		m = GetCtrlVal (daq, MUX_VOLT_HOLD, &volt_hold);
		m = GetCtrlVal (daq, MUX_Ramp, &ramp);
		m = GetCtrlVal (daq, MUX_Ramp_Rate, &ramp_rate);

/********   starting the loop of configuring   *******/

	/********  test ********/
		m=SetCtrlVal(daq,MUX_STOP_SCAN,1);
		m=SetCtrlVal(daq,MUX_Config_complete,0);
		m=SetCtrlVal(daq,MUX_Memory_Check_Done,0);
		ibwrt(Device1,"CA25X",5);                                        /* dummy line */
		ibwrt(Device1,"NA25X",5);
		for(i=1;i<=8;i++){
			for(j=1;j<=8;j++){
			/*		if(cross_point[i][j]==1){                                  2-17-01      */
					c[0]='C';
					c[1]='B';
					c[2]=(char)(48+i);
					c[3]='X';
					c[4]='\0';
					ibwrt(Device1,c,4);
					c[0]='N';
					c[1]='A';
					c[2]=(char)(48+i);
					c[3]='X';
					c[4]='\0';
					ibwrt(Device1,c,4);
					if (j<2){
					c[0]='C';
					c[1]='C';
					c[2]=(char)(48+j+8);
					c[3]='X';
					c[4]='\0';
					ibwrt(Device1,c,4);
					c[0]='N';
					c[1]='H';
					c[2]=(char)(48+j+8);
					c[3]='X';
					c[4]='\0';
					ibwrt(Device1,c,4);
					}
					else{
					d[0]='C';
					d[1]='C';
					d[2]='1';                                                  /* two-digit  */
```

```
                                        d[3]=(char)(48+j-2);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        d[0]='N';
                                        d[1]='H';
                                        d[2]='1';                                                       /* two-digit  */
                                        d[3]=(char)(48+j-2);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        }
                /*              }
                                else{
                                        c[0]='C';
                                        c[1]='B';
                                        c[2]=(char)(48+i);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        if (j<4){
                                        c[0]='C';
                                        c[1]='A';
                                        c[2]=(char)(48+j+6);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        }
                                        else{
                                        d[0]='C';
                                        d[1]='A';
                                        d[2]='1';
                                        d[3]=(char)(48+j-4);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        }
                                }
                  2-17-01       */
                                for(k=1;k<=16;k++){
                                        if((k!=i)&&(k!=j+8)){
                                                if (k<10){
                                                        if (k<=8){
                                                        c[0]='C';
                                                        c[1]='A';           /* apply -1.0 volt to rows from Keithley 5-25-01  */
                                                        c[2]=(char)(48+k);
                                                        c[3]='X';
                                                        c[4]='\0';
                                                        ibwrt(Device1,c,4);
                                                        }
                                                        else{
                                                        c[0]='C';
                                                        c[1]='H';
/*  Ground the columns */
                                                        c[2]=(char)(48+k);
                                                        c[3]='X';
                                                        c[4]='\0';
                                                        ibwrt(Device1,c,4);
                                                        }
                                                }
                                                else{
                                                d[0]='C';
                                                d[1]='H';
                                                d[2]='1';
                                                d[3]=(char)(48+k-10);
                                                d[4]='X';
                                                d[5]='\0';
                                                ibwrt(Device1,d,5);
                                                }
                                        }
                                }
                                /*  set write voltage */
                                Delay(0.1);
                                printf("\a");
                                if(ramp==1){
                                        if(cross_point[i][j]==1){
                                                volt_ramp0=volt_hold;
                                                volt_ramp1=0.0;
                                                for(i_ramp=1; i_ramp<=ramp_num; i_ramp++){
/* ramp-up */
                                                        volt_ramp0=volt_ramp0 + (volt_write_on/2-volt_hold)/ramp_num;
                                                        volt_ramp1=volt_ramp1 + (volt_write_on/2)/ramp_num;
                                                        m = AO_VWrite (1, 0, volt_ramp0);
                                                        m = AO_VWrite (1, 1, (-volt_ramp1-0.06225)/0.9938);
                                                        Delay(volt_write_on/ramp_num/ramp_rate);
                                                        Delay(-volt_write_on/ramp_num/ramp_rate);
                                                }

                                                Delay(time_write);                                                                      /* hold */
                                                for(i_ramp=1; i_ramp<=ramp_num; i_ramp++){
/* ramp-down */
                                                        volt_ramp0=volt_ramp0 - (volt_write_on/2-volt_hold)/ramp_num;
                                                        volt_ramp1=volt_ramp1 - (volt_write_on/2)/ramp_num;
```

```
                                                m = AO_VWrite (1, 0, volt_ramp0);
                                                m = AO_VWrite (1, 1, (-volt_ramp1-0.06225)/0.9938);
                                                Delay(volt_write_on/ramp_num/ramp_rate);
                                                Delay(-volt_write_on/ramp_num/ramp_rate);
                                        }
                                }
                                else{
                                        volt_ramp0=volt_hold;
                                        volt_ramp1=0.0;
                                        for(i_ramp=1; i_ramp<=ramp_num; i_ramp++){
                        /* ramp-up */
                                                volt_ramp0=volt_ramp0 + (volt_write_off/2-volt_hold)/ramp_num;
                                                volt_ramp1=volt_ramp1 + (volt_write_off/2)/ramp_num;
                                                m = AO_VWrite (1, 0, volt_ramp0);
                                                m = AO_VWrite (1, 1, (-volt_ramp1-0.06225)/0.9938);
                                                Delay(volt_write_off/ramp_num/ramp_rate);
                                                Delay(-volt_write_off/ramp_num/ramp_rate);
                                        }
                                        Delay(time_write);                                                              /* hold */
                                        for(i_ramp=1; i_ramp<=ramp_num; i_ramp++){
                        /* ramp-down */
                                                volt_ramp0=volt_ramp0 - (volt_write_off/2-volt_hold)/ramp_num;
                                                volt_ramp1=volt_ramp1 - (volt_write_off/2)/ramp_num;
                                                m = AO_VWrite (1, 0, volt_ramp0);
                                                m = AO_VWrite (1, 1, (-volt_ramp1-0.06225)/0.9938);
                                                Delay(volt_write_off/ramp_num/ramp_rate);
                                                Delay(-volt_write_off/ramp_num/ramp_rate);
                                        }
                                }
                        }               /* with ramp */
                        else{
                                if(cross_point[i][j]==1){
                                        m = AO_VWrite (1, 0, (volt_write_on/2));
                                        m = AO_VWrite (1, 1, (-volt_write_on/2-0.06225)/0.9938);
                                }
                                else{
                                        m = AO_VWrite (1, 0, (volt_write_off/2));
                                        m = AO_VWrite (1, 1, (-volt_write_off/2-0.06225)/0.9938);
                                }
                                Delay(time_write);
                                m = AO_VWrite (1, 0, volt_hold);
                                m = AO_VWrite (1, 1, -0.06225/0.9938);
                        }                       /* no ramp */
                /*****  set holding voltage to the row, and Ground to the column  *****/
                                c[0]='C';
                                c[1]='A';
                                c[2]=(char)(48+i);
                                c[3]='X';
                                c[4]='\0';
                                ibwrt(Device1,c,4);
                                c[0]='N';
                                c[1]='B';
                                c[2]=(char)(48+i);
                                c[3]='X';
                                c[4]='\0';
                                ibwrt(Device1,c,4);
                                if (j<2){
                                c[0]='C';
                                c[1]='H';
                                c[2]=(char)(48+j+8);
                                c[3]='X';
                                c[4]='\0';
                                ibwrt(Device1,c,4);
                                c[0]='N';
                                c[1]='C';
                                c[2]=(char)(48+j+8);
                                c[3]='X';
                                c[4]='\0';
                                ibwrt(Device1,c,4);
                                }
                                else{
                                d[0]='C';
                                d[1]='H';
                                d[2]='1';                                                               /* two-digit  */
                                d[3]=(char)(48+j-2);
                                d[4]='X';
                                d[5]='\0';
                                ibwrt(Device1,d,5);
                                d[0]='N';
                                d[1]='C';
                                d[2]='1';                                                               /* two-digit  */
                                d[3]=(char)(48+j-2);
                                d[4]='X';
                                d[5]='\0';
                                ibwrt(Device1,d,5);
                                }
                        /*   ibwrt(Device1,"P0X",3);                    /*  open all relays (skipped  5-25-01) */
                }
        }
/* close the loop  */
m=SetCtrlVal(daq,MUX_Config_complete,1);
return 1;
}
```

```
int logic_check(int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2){
        /*SetCtrlVal(daq,MUX_STOP_SCAN,1);      */
        return 1;
}
int memory_check(int panel, int control, int event,
                        void *callbackData, int eventData1, int eventData2){
        /*SetCtrlVal(daq,MUX_STOP_SCAN,1);      */
int i,j,k,ii,m;
double r_dummy;
int fail[9][9];
double AD0[9][9][100],AD1[9][9][100];
char c[5],d[6];
        DeleteGraphPlot (daq, MUX_GRAPH, -1, VAL_IMMEDIATE_DRAW);
        m=SetCtrlVal(daq,MUX_Memory_Check_Done,0);
        m=SetCtrlVal(daq,MUX_Set_phase,0);
/*
        m=SetCtrlAttribute(daq,MUX_switch1_1r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch1_2r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch1_3r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch1_4r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch1_5r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch1_6r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch2_1r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch2_2r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch2_3r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch2_4r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch2_5r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch2_6r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch3_1r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch3_2r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch3_3r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch3_4r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch3_5r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch3_6r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch4_1r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch4_2r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch4_3r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch4_4r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch4_5r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch4_6r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch5_1r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch5_2r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch5_3r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch5_4r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch5_5r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch5_6r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch6_1r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch6_2r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch6_3r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch6_4r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch6_5r, ATTR_DIMMED, TRUE);
        m=SetCtrlAttribute(daq,MUX_switch6_6r, ATTR_DIMMED, TRUE);
12-12-01  LED's removed and kept in an untitled panel    */
        for(i=1;i<=8;i++){
                for(j=1;j<=8;j++){
                PlotLine(daq, MUX_GRAPH, (i-1)*8+(j-1), cross_point[i][j], (i-1)*8+j, cross_point[i][j],VAL_BLUE);
                }
        }
        m = GetCtrlVal (daq, MUX_TIME_READ, &time_read);
        m = GetCtrlVal (daq, MUX_VOLT_READ, &volt_read);
        m = GetCtrlVal (daq, MUX_Threshold_High, &threshold_high);
        m = GetCtrlVal (daq, MUX_Threshold_Low, &threshold_low);
        m = GetCtrlVal (daq, MUX_NUM_READ, &num_read);
        m = GetCtrlVal (daq, MUX_VOLT_HOLD, &volt_hold);
        fp_out=fopen(tmp_file,"w");
/*      Device1=ibdev(0,18,0,10,1,0);                   /* initiate 707A  */
/*      ibwrt(Device1,"E0X",3);                                         /* Point to present relays  */
        for(i=1;i<=8;i++){
                for(j=1;j<=8;j++){
                        c[0]='C';
                        c[1]='D';                       /* use relay row D to read (Vread+AC from function generater)  */
                    2]=(char)(48+i);
                        c[3]='X';
                        c[4]='\0';
                        ibwrt(Device1,c,4);
                        c[0]='N';
                        c[1]='A';
                        c[2]=(char)(48+i);
                        c[3]='X';
                        c[4]='\0';
                        ibwrt(Device1,c,4);
                        if (j<2){
                        c[0]='C';
                        c[1]='G';                                               /* amp-meter  */
                        c[2]=(char)(48+j+8);
                        c[3]='X';
                        c[4]='\0';
                        ibwrt(Device1,c,4);
                        c[0]='N';
                        c[1]='H';                                               /*  GND  */
                        c[2]=(char)(48+j+8);
                        c[3]='X';
```

```
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        }
                                        else{
                                        d[0]='C';
                                        d[1]='G';                                       /* amp-meter  */
                                        d[2]='1';
                                        d[3]=(char)(48+j-2);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        d[0]='N';
                                        d[1]='H';                                       /*  GND  */
                                        d[2]='1';
                                        d[3]=(char)(48+j-2);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        }
                                for(k=1;k<=16;k++){
                                        if((k!=i)&&(k!=j+8)){
                                                if (k<10){
                                                        if (k<=8){
                                                        c[0]='C';
                                                        c[1]='A';           /* apply -1.0 volt to rows from Keithley 5-25-01  */
                                                        c[2]=(char)(48+k);
                                                        c[3]='X';
                                                        c[4]='\0';
                                                        ibwrt(Device1,c,4);
                                                        }
                                                        else{
                                                        c[0]='C';
                                                        c[1]='H';
        /*  Ground the columns */
                                                        c[2]=(char)(48+k);
                                                        c[3]='X';
                                                        c[4]='\0';
                                                        ibwrt(Device1,c,4);
                                                        }
                                                }
                                                else{
                                                d[0]='C';
                                                d[1]='H';
                                                d[2]='1';
                                                d[3]=(char)(48+k-10);
                                                d[4]='X';
                                                d[5]='\0';
                                                ibwrt(Device1,d,5);
                                                }
                                        }
                                }

                                /*  set read voltage and measure the current  */
                                /* Delay (0.1);                                 5-25-01  */
                                printf("\a");
                                m = AO_VWrite (1, 0, volt_read);            /* channel 0's output goes to relay row B directly
                                                                                and goes to row D through function generater  */
                                Delay (0.1);                                /* delay after setting the read voltage */
                                /*manually set phase on the lock-in  5_28_01  */
                                /*
                                m=SetCtrlVal(daq,MUX_Set_phase,1);
                                scanf("%f",r_dummy);
                                m=SetCtrlVal(daq,MUX_Set_phase,0);          taken out for non-volatile devices   6-5-01*/

                                for (ii=0;ii<num_read;ii++){
                                m = AI_VRead (1, 0, 1, &adch0);                     /* output from current amplifier  */
                                m = AI_VRead (1, 1, 1, &adch1);                     /* output from lock-in amplifier   */

                                AD0[i][j][ii]=-adch0;                               /****  Current Amplifier revise the polarity!!  *****/
                                AD1[i][j][ii]=adch1;

                                if(ii>0) m=PlotLine (daq, MUX_GRAPH, 8.0*(i-1)+j-1+(double)(ii-1)/(double)(num_read-1), AD0[i][j][ii-1], 8.0*(i-1)+j-
1+(double)(ii)/(double)(num_read-1), AD0[i][j][ii], VAL_RED);
                                if(ii>0) m=PlotLine (daq, MUX_GRAPH, 8.0*(i-1)+j-1+(double)(ii-1)/(double)(num_read-1), AD1[i][j][ii-1], 8.0*(i-1)+j-
1+(double)(ii)/(double)(num_read-1), AD1[i][j][ii], VAL_GREEN);

                                Delay (time_read/num_read);
                                }
        /*****  set holding voltage to the row, and Ground to the column  *****/
                                        c[0]='C';
                                        c[1]='A';
                                        c[2]=(char)(48+i);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        c[0]='N';
                                        c[1]='D';
                                        c[2]=(char)(48+i);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        if (j<2){
                                        c[0]='C';
```

```
                                        c[1]='H';
                                        c[2]=(char)(48+j+8);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        c[0]='N';
                                        c[1]='G';
                                        c[2]=(char)(48+j+8);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        }
                                        else{
                                        d[0]='C';
                                        d[1]='H';
                                        d[2]='1';                                                       /* two-digit  */
                                        d[3]=(char)(48+j-2);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        d[0]='N';
                                        d[1]='G';
                                        d[2]='1';                                                       /* two-digit  */
                                        d[3]=(char)(48+j-2);
                                        d[4]='X';
                                        d[5]='\0';
                                        ibwrt(Device1,d,5);
                                        }
                                fail[i][j]=0;
                                if(cross_point[i][j]==0){
                                        for (ii=0;ii<num_read;ii++){
                                                if(AD0[i][j][ii]>threshold_low) {
                                                        fail[i][j]=1;
                                                        break;
                                                }
                                        }
                                }
                                if(cross_point[i][j]==1){
                                        for (ii=0;ii<num_read;ii++){
                                                if(AD0[i][j][ii]<threshold_high) {
                                                        fail[i][j]=1;
                                                        break;
                                                }
                                        }
                                }
                                /*  m = AO_VWrite (1, 0, 0.0);
                                ibwrt(Device1,"P0X",3);                                                 5-25-01  */
                        }
                }
/* close the loop  */

/*****  set holding voltage to the rows, and Ground to the columns  *****/
                for(k=1;k<=16;k++){
                        if (k<10){
                                if (k<=8){
                                        c[0]='C';
                                        c[1]='A';                       /* apply -1.0 volt to rows from Keithley 5-25-01  */
                                        c[2]=(char)(48+k);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        }
                                        else{
                                        c[0]='C';
                                        c[1]='H';                                       /*  Ground the columns */
                                        c[2]=(char)(48+k);
                                        c[3]='X';
                                        c[4]='\0';
                                        ibwrt(Device1,c,4);
                                        }
                                }
                                else{
                                d[0]='C';
                                d[1]='H';
                                d[2]='1';
                                d[3]=(char)(48+k-10);
                                d[4]='X';
                                d[5]='\0';
                                ibwrt(Device1,d,5);
                                }
                        }
/*
if(fail[1][1]==0) {
                m=SetCtrlAttribute(daq,MUX_switch1_1r, ATTR_DIMMED, FALSE);
                m=SetCtrlVal(daq,MUX_switch1_1r,cross_point[1][1]);}
if(fail[1][2]==0) {
                m=SetCtrlAttribute(daq,MUX_switch1_2r, ATTR_DIMMED, FALSE);
                m=SetCtrlVal(daq,MUX_switch1_2r,cross_point[1][2]);}
if(fail[1][3]==0) {
                m=SetCtrlAttribute(daq,MUX_switch1_3r, ATTR_DIMMED, FALSE);
                m=SetCtrlVal(daq,MUX_switch1_3r,cross_point[1][3]);}
if(fail[1][4]==0) {
                m=SetCtrlAttribute(daq,MUX_switch1_4r, ATTR_DIMMED, FALSE);
```

```
		m=SetCtrlVal(daq,MUX_switch1_4r,cross_point[1][4]);}
if(fail[1][5]==0) {
		m=SetCtrlAttribute(daq,MUX_switch1_5r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch1_5r,cross_point[1][5]);}
if(fail[1][6]==0) {
		m=SetCtrlAttribute(daq,MUX_switch1_6r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch1_6r,cross_point[1][6]);}
if(fail[2][1]==0) {
		m=SetCtrlAttribute(daq,MUX_switch2_1r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch2_1r,cross_point[2][1]);}
if(fail[2][2]==0) {
		m=SetCtrlAttribute(daq,MUX_switch2_2r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch2_2r,cross_point[2][2]);}
if(fail[2][3]==0) {
		m=SetCtrlAttribute(daq,MUX_switch2_3r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch2_3r,cross_point[2][3]);}
if(fail[2][4]==0) {
		m=SetCtrlAttribute(daq,MUX_switch2_4r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch2_4r,cross_point[2][4]);}
if(fail[2][5]==0) {
		m=SetCtrlAttribute(daq,MUX_switch2_5r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch2_5r,cross_point[2][5]);}
if(fail[2][6]==0) {
		m=SetCtrlAttribute(daq,MUX_switch2_6r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch2_6r,cross_point[2][6]);}
if(fail[3][1]==0) {
		m=SetCtrlAttribute(daq,MUX_switch3_1r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch3_1r,cross_point[3][1]);}
if(fail[3][2]==0) {
		m=SetCtrlAttribute(daq,MUX_switch3_2r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch3_2r,cross_point[3][2]);}
if(fail[3][3]==0) {
		m=SetCtrlAttribute(daq,MUX_switch3_3r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch3_3r,cross_point[3][3]);}
if(fail[3][4]==0) {
		m=SetCtrlAttribute(daq,MUX_switch3_4r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch3_4r,cross_point[3][4]);}
if(fail[3][5]==0) {
		m=SetCtrlAttribute(daq,MUX_switch3_5r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch3_5r,cross_point[3][5]);}
if(fail[3][6]==0) {
		m=SetCtrlAttribute(daq,MUX_switch3_6r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch3_6r,cross_point[3][6]);}
if(fail[4][1]==0) {
		m=SetCtrlAttribute(daq,MUX_switch4_1r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch4_1r,cross_point[4][1]);}
if(fail[4][2]==0) {
		m=SetCtrlAttribute(daq,MUX_switch4_2r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch4_2r,cross_point[4][2]);}
if(fail[4][3]==0) {
		m=SetCtrlAttribute(daq,MUX_switch4_3r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch4_3r,cross_point[4][3]);}
if(fail[4][4]==0) {
		m=SetCtrlAttribute(daq,MUX_switch4_4r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch4_4r,cross_point[4][4]);}
if(fail[4][5]==0) {
		m=SetCtrlAttribute(daq,MUX_switch4_5r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch4_5r,cross_point[4][5]);}
if(fail[4][6]==0) {
		m=SetCtrlAttribute(daq,MUX_switch4_6r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch4_6r,cross_point[4][6]);}

if(fail[5][1]==0) {
		m=SetCtrlAttribute(daq,MUX_switch5_1r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch5_1r,cross_point[5][1]);}
if(fail[5][2]==0) {
		m=SetCtrlAttribute(daq,MUX_switch5_2r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch5_2r,cross_point[5][2]);}
if(fail[5][3]==0) {
		m=SetCtrlAttribute(daq,MUX_switch5_3r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch5_3r,cross_point[5][3]);}
if(fail[5][4]==0) {
		m=SetCtrlAttribute(daq,MUX_switch5_4r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch5_4r,cross_point[5][4]);}
if(fail[5][5]==0) {
		m=SetCtrlAttribute(daq,MUX_switch5_5r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch5_5r,cross_point[5][5]);}
if(fail[5][6]==0) {
		m=SetCtrlAttribute(daq,MUX_switch5_6r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch5_6r,cross_point[5][6]);}
if(fail[6][1]==0) {
		m=SetCtrlAttribute(daq,MUX_switch6_1r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch6_1r,cross_point[6][1]);}
if(fail[6][2]==0) {
		m=SetCtrlAttribute(daq,MUX_switch6_2r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch6_2r,cross_point[6][2]);}
if(fail[6][3]==0) {
		m=SetCtrlAttribute(daq,MUX_switch6_3r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch6_3r,cross_point[6][3]);}
if(fail[6][4]==0) {
		m=SetCtrlAttribute(daq,MUX_switch6_4r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch6_4r,cross_point[6][4]);}
if(fail[6][5]==0) {
```

```
		m=SetCtrlAttribute(daq,MUX_switch6_5r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch6_5r,cross_point[6][5]);}
if(fail[6][6]==0) {
		m=SetCtrlAttribute(daq,MUX_switch6_6r, ATTR_DIMMED, FALSE);
		m=SetCtrlVal(daq,MUX_switch6_6r,cross_point[6][6]);}

  12-12-01	taken out, because the LED's are removed*/

m=SetCtrlVal(daq,MUX_Memory_Check_Done,1);
		for(i=1;i<=8;i++){
			for(j=1;j<=8;j++){
				for (k=0;k<num_read;k++){
					fprintf(fp_out, "%d %d %d %f %f\n", i, j, cross_point[i][j], AD0[i][j][k], AD1[i][j][k]);
					}
				}
			}
		fclose(fp_out);
		return 1;
}
int stop(int panel, int control, int event,
			void *callbackData, int eventData1, int eventData2){
		return 1;
}
int save_file(int panel, int control, int event,
			void *callbackData, int eventData1, int eventData2){
		int i;
		int tmp1[6400],tmp2[6400],tmp3[6400];
		float tmp4[6400], tmp5[6400];
		char line[100];
		char name[30];
		fp_out=fopen (tmp_file,"r");
		for (i = 0; i < num_read*64; ++i)
			{
			fgets(line,sizeof(line),fp_out);
			sscanf(line,"%d %d %d %f %f", &tmp1[i], &tmp2[i], &tmp3[i], &tmp4[i], &tmp5[i]);
			}
		 fclose(fp_out);
		PromptPopup ("SAVE FILE", "Enter the file name (*.txt).", name, 20);
		fp_out=fopen(name,"w");
		for (i =0; i < num_read*64; ++i)
			fprintf(fp_out,"%d %f %f\n",tmp3[i], tmp4[i], tmp5[i]);
		fclose(fp_out);
		return 1;
}
int quit(int panel, int control, int event,
			void *callbackData, int eventData1, int eventData2)
{
		int i;
		switch (event) {
			case EVENT_COMMIT:
				i = AO_VWrite (1, 0, 0.0);
				i = AO_VWrite (1, 1, 0.0);
				ibwrt(Device1,"P0X",3);
				QuitUserInterface (0);
				break;
			case EVENT_RIGHT_CLICK:
				break;
		}
		return 0;
}
/*
int load_individual_panel (int panel, int control, int event, void *callbackData, int eventData1, int eventData2)
{
		daq1 = LoadPanel (0, "MUX.uir",SET_INDIVI);
		DisplayPanel (daq1);
		return 0;
}
*/
int clear (int panel, int control, int event,
			void *callbackData, int eventData1, int eventData2)
{
		int i;
		switch (event) {
			case EVENT_COMMIT:
				DeleteGraphPlot (daq, MUX_GRAPH, -1, VAL_IMMEDIATE_DRAW);
/*				DeleteGraphPlot (daq, DAQ_GRAPH_2, -1, VAL_IMMEDIATE_DRAW);		*/
				break;
		}
		return 0;
}
```