Observation, Analysis, Modeling, and Classification of USB Host Controller Operation Under Electro-Magnetic Interference

Natasha Jarus, *Member, IEEE,* Joel Schott, Laika Klingbeil, Sahra Sedigh Sarvestani, *Senior Member, IEEE*
Department of Electrical and Computer Engineering
Missouri University of Science and Technology, Rolla, MO 65409, USA
Email: {jarus, jnskbx, ackgg3, sedighs}@mst.edu

***Abstract*—One source of malfunctions in electronics is electromagnetic interference (EMI), which can cause software glitches, system crashes, and even permanent hardware damage. Our goal is to study the software effects of EMI with the intent of detecting EMI events and potentially mitigating the software errors they cause. We developed a software instrumentation approach which allows us to monitor the operation of system peripherals. We collect a set of baseline sequences where the system is not exposed to interference and then a set of sequences where the system is exposed to EMI.**

To determine whether our instrumentation detected a difference in operation correlated with EMI, we modeled each sequence as a categorical time series and performed statistical analysis. We aggregated measures of variance and autocorrelation for the baseline and EMI-exposed sequences, then compared the statistics of the two sets. These tests revealed an increase in variation in EMI-exposed sequences as well as changes in the autocorrelation structure between the baseline and EMI-exposed datasets. Based on these results, we concluded that our instrumentation is precise enough to capture differences in system operation due to EMI.

To use those differences to predict whether the system is being interfered with, we applied several classification algorithms to the datasets. A key observation from these attempts is that the temporal relationship of the states is critical to classifier performance. We found a gradient boosted random forest classifier to have the best performance identifying sub-sequences of EMI-exposed states.

***Index Terms*—EMC, computers**

I. Introduction

Modern electronics are becoming increasingly susceptible to electromagnetic interference (EMI). As components become smaller and clock speeds and data transfer rates rise, electronics become susceptible to interference from smaller electromagnetic fields [1]. At the same time, high voltage power supplies and other equipment required to build EMI-emitting devices have become less expensive and more easily obtained. Thus, a critical design goal for new electronics is to be able to withstand or recover from such interference.

EMI manifests in electronics in a variety of ways. Program operation may be interrupted, data in memory or being transmitted across a link may be corrupted, or sensors may report bizarre values or stop responding. In the worst case, system resets and permanent hardware damage can occur. For consumer hardware, this might be merely an annoyance, but in safety-critical hardware, such as that in industrial control systems or modern computer-laden vehicles, EMI can have significantly more severe effects. Furthermore, control electronics are typically more vulnerable than consumer electronics due to the wiring which connects a control processor to its numerous sensors and actuators [2].

In order to design EMI-robust and -resilient systems, it is imperative to understand where EMI enters the system, how it travels through the hardware, and the resulting effects it has on hardware and software operation. Our goal is to investigate low-level EMI effects on commercial hardware using software instrumentation. To accomplish this, we have created a novel technique for instrumenting peripheral devices, such as USB controllers, network communication hardware, and sensors. This instrumentation is precise — it captures exactly the operation of the peripheral as seen by the system — without interfering with or burdening the system. Thus it can be used in tandem with the control software a system would be running in the field, making test conditions more realistic.

We desire to use the data collected from our instrumentation to several ends. First, we seek to statistically characterize the operation of a given peripheral both under "typical" operation and under EMI exposure. This will validate that our instrumentation is capable of capturing the effects of EMI on the chosen peripheral. Second, we apply classification algorithms to this data, labeling peripheral operation as either "not experiencing interference" or "experiencing interference".

With these classifiers in hand, we can proceed to more detailed analysis and mitigation. Analysis of operation that is classified as "experiencing interference" will give a low-level understanding of how that particular peripheral is affected by EMI. This may lead to further understanding of points where the hardware is weak to EMI. It may also lead to better EMI mitigation strategies, allowing the system to recover from EMI without requiring a complete reset. Furthermore, we may be able to enable the peripheral to continue operating during an EMI event, albeit at a reduced performance level.

The remainder of this document outlines our work on these goals. We situate our work in the context of related work in Section II. Section III discusses the instrumentation approach, experimental design and procedure, and data processing. Statistical analysis techniques used to achieve our second goal, along with results, are discussed in Section IV. Classification approaches are covered in Section V. Section VI summarizes our results so far and enumerates open questions about our work.

II. Background and Related Literature

The effects of EMI can be categorized according to their mechanism, broadly *no effect*, *interference*, or *destruction*; by their duration, ranging from *observed only for the duration of an EMI event* to *permanent hardware damage*; and by their criticality: [3]

- U **Unknown**: Unobserved or indeterminate due to other failures
- N **No effect**: System fulfils its mission without disturbance
- I **Interference**: Disturbance does not influence the system's function
- II **Degradation**: Disturbance impairs system capability or efficiency
- III **Loss of main function**: Disturbance prevents system from functioning.

Assessing the systemic effect of EMI on a sub-system incorporates these three perspectives plus knowledge of the sub-system's criticality in the function of the system.

Low-level investigation of EMI incorporates both modeling [4] and experimentation [5, 6]. These studies provide a detailed understanding of EMI-related component faults and suggest physical mitigation techniques [7]. However, their precision and experimental rigor (e.g., requirements for PCB design and instrumentation) make them infeasible for analyzing large-scale systems or consumer hardware.

System-level EMI analysis for electromagnetic compatibility purposes primarily focuses on EMI events with type II or III criticality. The effect of interference is determined based on reported alarms, errors recorded in system logs, loss of network traffic, user-visible "glitches" in displays or audio, or system crashes or resets [8, 9]. Such studies can characterize whole systems or focus on the vulnerability of specific sub-systems [10]. A variety of confounding factors must be considered when assessing system vulnerability to EMI, including EMI frequency and waveform, simultaneous exposure to multiple waveforms, waveform reflections, mechanical vibrations, and intermodulation effects [11, 12].

While shielding is an effective method for mitigating EMI effects [2], it is often not feasible or cost-effective, especially for systems that are frequently upgraded or used in environments where EMI exposure has not been characterized. EMI *resilience* takes a complementary approach wherein systems are continually monitored for interference and improved as required [13, 14].

As part of an EMI resilience program, existing system peripherals have been investigated for use as EMI sensors. Errors from USB and PS/2 devices and serial data communication rates have been found to be correlated with EMI exposure [15, 16]. Furthermore, analog sensors including temperature sensors for hard drives and processors and wifi and cellular received power indicators also show anomalies under EMI exposure [17]. In addition to these studies, software instrumentation for electrostatic discharge detection has been investigated in [18–20].

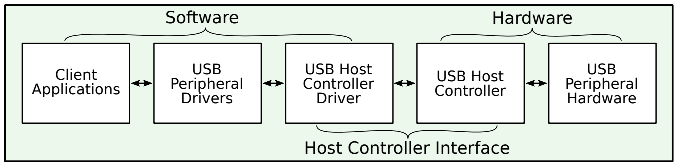


Fig. 1: USB communication architecture

Another key part of EMI resilience is detecting anomalies in sensor data. While there is little published research on detecting EMI-caused anomalies, a number of studies have focused on efficient detection of anomalies in digital and analog sensor data. False data injection, where sensor values are spoofed to hide a physical fault, can be detected through clustering or statistical analysis [21, 22]. Detection of faults or attacks from instrumented software or processor performance counters is also feasible via a combination of classification and statistical correlation [23, 24].

The distinction of our work is twofold. Our instrumentation captures much finer-grained events, offering insight into EMI events with U or I criticality as well as II or III criticality. Furthermore, we pioneer the use of categorical time series statistics and classification algorithms in characterizing and identifying EMI-related operation anomalies. We use these tools to validate our instrumentation and to identify anomalous sequences of events as a foundation for building real-time EMI detection software or for root-cause analysis of EMI-caused failures.

III. Observation

This section details the approach we took to instrument our chosen peripheral, a USB host controller. It also describes our experimental design and the data collected from our experiments.

A. Device & Instrumentation

We have chosen to instrument a USB host controller for our experiments, since USB is widely used and peripherals are readily available. The role of the USB host controller is to handle the physical communication between the host computer and the various USB devices connected to it. All control and data communications to or from attached USB devices go through the host controller. The host controller connects directly to the USB power and data lines, making it an excellent point to detect EMI entering the system through those points. This architecture is depicted in Figure 1.

The host computer communicates with the host controller via a collection of memory locations called *registers*. The host controller updates these registers as it performs various operations. It can also alert the host computer that an event has happened via *interrupts*. Our instrumentation must be based on these registers and interrupts, as they are all the visibility the host computer has into the internal operation of the host controller.

For our experiments, we selected the well-documented and affordable Rock Pro 64 system which uses a Rockchip RK3399 System-on-Chip that has built-in USB 3.0 and 2.0

host controllers. We focus on the USB 2.0 host controller, which conforms to the Enhanced Host Controller Interface (EHCI) specification [25]. This specification details both the physical power and communication requirements for the host controller and the contents of the host controller's registers. We installed a Linux operating system on this device running version `4.4.202-1237-rockchip-ayufan` of the Linux kernel. The kernel contains a driver — a software program — that configures and manages communication with the host controller.

Our instrumentation is performed by making slight modifications to the EHCI host controller driver. This driver has a number of procedures which are run when the host controller causes an interrupt. We modify these procedures to record a snapshot of all the host controller register values before the driver modifies them. In this fashion, we can precisely capture the system-visible operation of the host controller without additional overhead from redundant checks.[1] An example snapshot is shown in Figure 2.

B. Experimental Setup

When performing experiments, we need to generate traffic on the USB bus to stimulate host controller operation and, consequently, the creation of register snapshots by our instrumentation. To keep test conditions consistent, we connect only one USB flash drive to the system and copy the same file to it in each test. This produces constant traffic to the USB host controller for the duration of one EMI injection. To speed up the rate at which experiments were performed, we developed a script which automated this task as well as the task of saving the log of snapshots after an experiment.

Two types of experiments were performed, termed *baseline* and *EMI-exposed*. For both tests, our instrumented driver was used and USB traffic was generated as described above. The baseline tests allow us to capture the typical operation of the host controller and are performed with the device, consisting of the computer and flash drive, sitting on a bench. Our assumption is that any interference arising from these conditions is typical of what the device should withstand under normal operation and thus not meaningful to distinguish from truly interference-free operation.

EMI-exposed tests are conducted with the device placed between the plates of the EMI generator described in [26]. This generator repeatedly produces a 30 MHz damped sinusoidal electric field wave which interferes with the device operation. Tests were carried out to characterize the limits of what the device can withstand without resetting the host computer or causing permanent harm to the device — conditions under which any software instrumentation will struggle to accurately capture system operation. Tests are carried out with the device placed vertically or horizontally in the generator, as orientation can affect how the EMI enters the device.

C. Collected Data

EMI-exposed experiments were carried out on two separate dates. Along with each log of register snapshots, we saved the transferred file as written to the USB drive. None of these files exhibit corruption, so further analysis of that data is not merited. In addition, we recorded the associated field strength and device orientation.

[1]8.6% wall clock overhead; 37.9% CPU cycle overhead.

Register snapshot:
```
Jul 29 21:38:38 rockpro64 kernel: [ 2789.449136]
[EHCI DEBUG DUMP ehci_handshake] 0x10025, 0x8009,
0x37, 0x1abb, 0x0, 0xdfb5d000, 0xdfb5b000, 0x0, 0x1,
0x10025, 0x10025
```

Corresponding state:

Driver Function	`ehci_handshake`
Command	`0x10025`
Status	`0x8009`
Interrupt Enable	`0x37`
TX Fill Tuning	`0x0`
"Configured" Flag	`0x1`
USB Mode	`0x10025`
USB Mode Extended	`0x10025`

Fig. 2: EHCI host controller register snapshot and corresponding state

We gathered a total of 24 baseline sequences and a total of 47 EMI-exposed sequences from our experiments. Note that sequences vary in length due to changes in system operation such as crashes, USB device disconnects, and data frame retransmissions.

We interpret these logs of register snapshots as sequences of host controller operation states; Figure 2 shows an example snapshot and state. States and snapshots correspond one to one; each state is defined by the register values in its corresponding snapshots. We treat two states as equal when their register values are all equal, excluding the values of some registers which are set only by the host controller driver and thus do not reflect host controller operation.[2]

We define the state space to be $\mathcal{S} = \{s_1, \cdots, s_{19}\}$ where each s_i corresponds to a specific unique state observed in our data. See Appendix A for the complete listing of the state space.

IV. Analysis

Our experimental process produces sequences of states; we analyze these sequences to determine whether the sequences from baseline experiments differ from those produced by EMI-exposed experiments. We can model these data as categorical discrete time series: sequences of observations of states from $\mathcal{S}$ over time. These states are discrete, rather than continuous, and unordered, meaning one cannot be ranked less than another. The events where we take our observations are the interrupts from the host controller; the times between events are stochastically distributed. While the space of possible register values is quite large, most of the possible states do not arise in practice: our data contain 19 distinct states. Further analysis is required to determine whether all registers are critical to determining the presence of EMI or if it is possible to reduce the state space further.

Formally, the j^{th} time series contains n_j observations $(x_1, \cdots, x_{n_j})$ from a stochastic process $\{X_t\} \in \mathcal{S}$. The

[2]The registers we exclude are `frame_index`, `segment`, `frame_list`, and `async_next`.

number of observations n_j can be modeled as an observation from a random variable N determining the sampling time of each sequence. We have two datasets: baseline data $\mathcal{D}_B = \{(x_t, 1 \leq t \leq n_j) \mid 1 \leq j \leq 24\}$ and EMI-exposed data $\mathcal{D}_E = \{(x_t, 1 \leq t \leq n_j) \mid 25 \leq j \leq 72\}$. Our assumption is that the sequences in $\mathcal{D}_B$ are generated from one stochastic process and the sequences in $\mathcal{D}_E$ are generated from another. We make no further assumptions about independence or identicality of distribution.

In summary, this section demonstrates that our instrumentation is capable of detecting changes in device operation when exposed to EMI. To do so, we characterize the time series in $\mathcal{D}_B$ and $\mathcal{D}_E$ using categorical time series equivalents of variance and autocorrelation. Significant differences in these measures indicate that our instrumentation is effective.

A. Variance & Dispersion

One question we might ask of a dataset is, "how much variation does it have?" For real-valued data, the variation in a dataset is quantified by *variance*; however, this measure is derived from the mean, which is not defined for categorical data. Instead, we measure *dispersion*: a widely disperse distribution is close to a *uniform distribution*, whereas a minimally disperse distribution is close to a *one-point distribution*. The more disperse a distribution is, the more uncertain we are about the outcome of observing the random variable.

Several dispersion measures have been proposed; we choose two measures which are suitable for our particular datasets. One measure is the *Gini index* [27, §6.2.1], defined as

$$\nu_G = \frac{d+1}{d}\left(1 - \sum_{j \in \mathcal{S}} \pi_j^2\right), \tag{1}$$

where $d+1$ is the number of categories (in our case, 19), $\mathcal{S}$ is the set of categories (in our case, the set of host controller states as defined in §III-C), and π_j is the probability of observing the jth category. The value of the Gini index falls in the range $[0; 1]$, with 0 indicating a one-point, minimally disperse distribution and 1 indicating a uniform, maximally disperse distribution.

The other measure we have chosen is the *extropy* [28] of a distribution, defined as

$$\nu_{Ex} = \frac{-1}{d \ln\left(\frac{d+1}{d}\right)} \sum_{j \in \mathcal{S}} (1 - \pi_j) \ln(1 - \pi_j). \tag{2}$$

Extropy shares the same range and behavior as the Gini index: higher extropy means a more disperse distribution. Extropy differs from information theory's *entropy* in one aspect: entropy is undefined if any $\pi_j = 0$, whereas extropy is undefined if any $\pi_j = 1$. As our datasets do not contain any one-point distributions, we choose extropy as a measure of variance.

Dispersion measures for each time series are aggregated using box plots in Figures 3a and 3b. Both measures qualitatively agree: the baseline time series are less disperse than all but a few of the EMI-exposed time series. In the November dataset, where most interference was near the limit of what the system could handle, all time series are more disperse than baseline time series.

B. Autocorrelation & Serial Dependence

In addition to variance, we can characterise time series data by how repetitive it is. Conceptually, this is determined by how similar the time series is to a copy of itself shifted back by one or more time steps (called *lag*). For real-valued time series, this measure is called *autocorrelation*; however, it depends on expected values, which are not defined for categorical data.

The corresponding categorical time series property is known as *signed serial dependence*. These measure various properties of the *conditional lagged bivariate probabilities* $p_{i|j}(k) = \frac{p_{ij}(k)}{\pi_j} = P(X_t = i \mid X_{t-k} = j)$. $p_{i|j}(k)$ is the probability of observing i at time t given that j was observed at time $t-k$, or, in other words, at *lag* k.

- *Serial Independence* occurs when observing j at $t-k$ gives no insight into what might be observed at t. Equivalently, $p_{i|j} = \pi_i$ for any $i, j \in \mathcal{S}$.
- *Positive Serial Dependence* occurs when observing i at $t-k$ means we observe i again at t. Equivalently, for any i, $p_{i|i} = 1$.
- *Negative Serial Dependence* occurs when observing i at $t-k$ means we will not observe i at t. Equivalently, for any i, $p_{i|i} = 0$.

We select two measures of signed serial dependence that relate to our chosen dispersion measures. The first measure, Cohen's κ [27, §6.3] is defined by

$$\kappa(k) = \frac{\sum_{j \in \mathcal{S}} \left(p_{jj}(k) - \pi_j^2\right)}{1 - \sum_{j \in \mathcal{S}} \pi_j^2}. \tag{3}$$

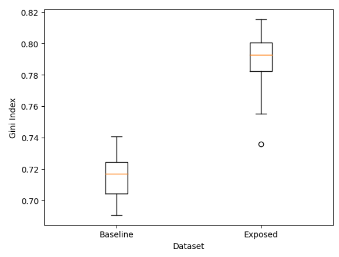
Values of $\kappa(k)$ fall in the range $\left[-\frac{\sum_{j \in \mathcal{S}} \pi_j^2}{1 - \sum_{j \in \mathcal{S}} \pi_j^2}; 1\right]$, with 0 corresponding to serial independence, positive values corresponding to positive serial dependence, and negative values corresponding to negative serial dependence. Note that the denominator of the lower bound for $\kappa(k)$ scales negatively with the Gini index of the marginal distribution $\boldsymbol{\pi}$ of the dataset. Thus, $\kappa(k)$ is undefined when $\boldsymbol{\pi}$ is a one-point distribution. In the independent and identically distributed (i.i.d.) case, that is, when a sequence is serially independent, Cohen's κ follows a normal distribution.

The second measure of serial dependence we selected is a modified version of $\kappa(k)$. This measure, $\kappa^\star(k)$ [29] is defined by

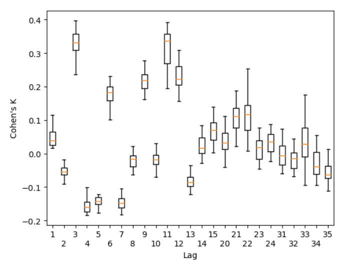
$$\kappa^\star(k) = \sum_{j \in \mathcal{S}} \frac{p_{jj}(k) - \pi_j^2}{1 - \pi_j}. \tag{4}$$

Values of $\kappa^\star(k)$ fall in the range $\left[-\sum_{j \in \mathcal{S}} \frac{\pi_j^2}{1 - \pi_j}; 1\right]$, with positive values corresponding to positive serial dependence and vice versa. As with extropy and $\kappa(k)$, $\kappa^\star(k)$ is undefined in the case that $\boldsymbol{\pi}$ is a one-point distribution. More disperse datasets also have a wider range of $\kappa^\star(k)$ values. In the i.i.d. case, as with Cohen's κ, $\kappa^\star$ follows a normal distribution.

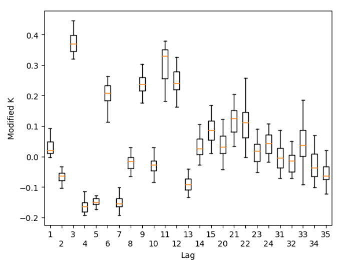
We compute Cohen's κ and $\kappa^\star$ for lags 1–15 and additionally 20–24 and 31–35 to confirm recurring serial dependence



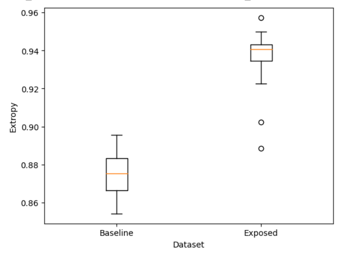
(a) Gini dispersion measure for each dataset as well as for all EMI-exposed time series (“exposed”)



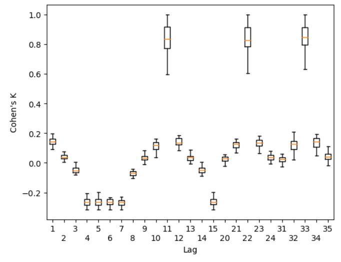
(c) Cohen’s κ for baseline data



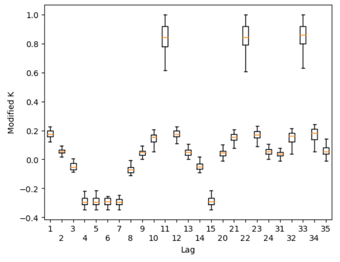
(e) Modified $\kappa^\star$ for baseline data



(b) Extropy dispersion measure for each dataset as well as for all EMI-exposed time series (“exposed”)



(d) Cohen’s κ for EMI-exposed data



(f) Modified $\kappa^\star$ for EMI-exposed data

Fig. 3: (a), (b): Dispersion measures. (c)–(f): Signed serial dependence measures for lags 1–15, 20–24, and 31–35.

	Cohen’s κ				Modified $\kappa^\star$			
	Baseline		EMI-exposed		Baseline		EMI-exposed	
Lag	% +ve	% −ve	% +ve	% −ve	% +ve	% −ve	% +ve	% −ve
1	33.3	0.0	95.6*	0.0	0.0	0.0	42.2	0.0
2	0.0	41.7	2.2	0.0	0.0	0.0	0.0	0.0
3	100.0+	0.0	13.3	4.4	100.0+	0.0	6.7	2.2
4	0.0	100.0+	0.0	100.0+	0.0	83.3+	0.0	97.8+
5	0.0	100.0+	0.0	100.0+	0.0	79.2+	0.0	93.3+
6	100.0+	0.0	4.4	91.1?	91.7+	0.0	2.2	82.2?
7	0.0	100.0+	0.0	100.0+	0.0	66.7+	0.0	97.8+
8	0.0	12.5	0.0	8.9	0.0	0.0	0.0	2.2
9	100.0+	0.0	13.3	0.0	95.8+	0.0	6.7	0.0
10	0.0	4.2	73.3*	0.0	0.0	0.0	33.3	0.0
11	100.0+	0.0	100.0+	0.0	100.0+	0.0	100.0+	0.0
12	100.0+	0.0	97.8*	0.0	100.0+	0.0	48.9	0.0
13	0.0	87.5*	2.2	0.0	0.0	0.0	0.0	0.0
14	20.8	0.0	6.7	2.2	0.0	0.0	0.0	2.2
15	62.5*	0.0	0.0	93.3+	16.7	0.0	0.0	82.2+
20	37.5	0.0	6.7	0.0	0.0	0.0	0.0	0.0
21	83.3+	0.0	82.2*	0.0	50.0+	0.0	35.6	0.0
22	83.3*	0.0	100.0+	0.0	37.5	0.0	100.0+	0.0
23	20.8	0.0	95.6*	0.0	0.0	0.0	46.7	0.0
24	33.3	0.0	4.4	0.0	0.0	0.0	0.0	0.0
31	12.5	4.2	2.2	2.2	4.2	0.0	0.0	0.0
32	0.0	16.7	77.8*	2.2	0.0	0.0	37.8	0.0
33	41.7	8.3	100.0+	0.0	20.8	0.0	97.8+	0.0
34	8.3	33.3	91.1+	0.0	0.0	0.0	51.1+	0.0
35	0.0	54.2	13.3	0.0	0.0	0.0	0.0	0.0

TABLE I: Percent of sequences where serial dependence is significant ($\alpha = 0.05$) at a given lag. Entries marked ‘*’, ‘+’, or ‘?’ have at least 50% of sequences showing significant dependence; entries marked ‘+’ exhibit the same result for both measures; entries marked ‘?’ have at least one sequences showing significant dependence of the opposite sign as well.

at lags that are multiples of 11. Box plots and critical values for the baseline and EMI-exposed datasets are shown in Figures 3c, 3d, 3e, and 3f. Our discussion focuses on dependence exhibited by both measures.

The baseline data measures (figures 3c and 3e) reveal some significant positive serial dependence at small lags, notably lags 3, 6, 9, 11, and 12. Small, but still significant, negative serial dependence is observed in the Cohen’s κ values at lags 4, 5, and 7; however, many of the corresponding values for $\kappa^\star$ are not significant. Above lag 12, serial dependence rapidly vanishes in the baseline dataset aside from mild positive dependence lag 21. In the EMI-exposed data, the only significant positive serial dependence is at lags 11, 22, and 33; in addition, some sequences have positive serial dependence at lag 34. Both Cohen’s κ and $\kappa^\star$ show significant negative serial dependence at lags 4, 5, and 7. In addition, significant negative serial dependence is observed at lag 15.

These results are summarized in Table I. The entries marked ‘?’ deserve additional commentary: at lag 6, most EMI-exposed sequences have negative serial dependence, but one or two have positive serial dependence. This may indicate that the EMI-exposed data is multimodal, with some sequences exhibiting one mode of operation and others another, or it may indicate that certain sequences do not exhibit interference. From the entries marked ‘*’, we can see that the chosen serial dependence measures agree more frequently on the baseline dataset than on the EMI-exposed dataset. The modified $\kappa^\star$ measure universally reports serial dependence less or equally frequently compared to Cohen’s κ.

At a minimum, we can say that there are measurable differences between the baseline and EMI-exposed datasets. Since the magnitudes of the serial dependences do not decrease monotonically, we can conclude the underlying stochastic processes do not have the Markov property. More analysis of the underlying data is necessary before a complete explanation of these differences can be offered. We hypothesize that during "normal" operation, the host controller's operation is characterized by 3 state and 11 state loops. When exposed to EMI, however, the 11-step processes become dominant; perhaps this is due to built-in fault-recovery methods or due to some unforeseen interaction between the experimental setup and the host controller's behavior. The negative serial dependences are less dramatic and are likely a side effect of the 3 and 11 state loops.

V. Classification

Having determined that our instrumentation observes differences in system operation depending on whether it is being interfered with, we study whether these differences can be used as indicators of EMI. The ultimate goals of this investigation are threefold:

1) Identifying sequences of states for further root-cause analysis
2) Detecting anomalies in real time on devices in the field
3) Recovering from interference with minimal interruptions to system operation

This section discusses work towards these goals, focusing on applying classification techniques to our experimental data. Our approach should meet several criteria:

1) Sufficient granularity to identify anomalous subsequences
2) Able to produce results from a stream of data, not just finite-length sequences

We study classifiers which produce a stream of results as well as sliding-window classifiers, both of which meet the second criteria above. Input sequences are synthesized from the experimental data as described in Section III-C to produce data which contains both known uninterfered- and interfered-with subsequences.

A. Classification Events

When developing classifiers for observed system operation, we can choose to classify sequences of register snapshots, or we can classify sequences of *events* derived from these snapshots. One example of such a sequence of derived events is pairs of snapshots $(x_{i-1}, x_i), 1 < i \leq n$ produced from a window of n snapshots. Such a derived event allows classifiers which otherwise ignore the order of states in a window to capture some temporal dependences in the data. We refer to such a pair of states as "lag 1" events; "lag 0" events refer to the sequence of snapshots x_i. This approach can be extended with further "lookback" — a "lag 2" event is the pair (x_{i-2}, x_i). In addition, we can classify over multiple lags: "lag 0 and 1" data contains an event for every observed state s_i as well as an event for every observed state pair (x_{i-1}, x_i).

Since the space of possible events grows exponentially with the number of snapshots used to derive an event, we do not consider triplets (e.g., (x_{i-2}, x_{i-1}, x_i)) of snapshots in this work.

B. Classification Techniques

The first set of techniques we evaluated produce a stream of results: a Hidden Markov Model (HMMs) and a Recurrent Neural Network (RNN) Long Short Term Memory (LSTM) classifier. In our HMM classifier, the hidden states correspond to our classification labels: "Baseline" and "EMI-exposed". When trained, the HMM learns the expected marginal distribution of the classification events in each category. Classifying a sequence of events produces the most likely sequence of classification labels — hidden states — given the observed events. The LSTM approach incorporates a memory of previous events which is considered when classifying a new event. While it seems a promising fit for our problem, we lack sufficient training data to achieve passable classification performance.

The second set of techniques, the sliding window classifiers, all classify on the distribution of events from a window. This choice keeps the input vectors to the classifier small — "lag 0" events require a 19 element vector — and independent of window size, as well as eliminates any overfitting from classifiers associating labels with a particular event's absolute position in the window.

We evaluated four sliding window classification techniques: an Artificial Neural Network (ANN), a Support Vector Machine (SVM), a Random Forest Classifier (RFC), and a Gradient Boosted Classifier (GBC). The ANN classifier features two neuron layers: an input layer the size of the event space and a single neuron sigmoidal output layer. This learns a direct mapping from the event marginal distribution to a value between 0 and 1, which is then compared against a cutoff to determine the classification label applied. The SVM classifier partitions the space of possible event marginal distributions and assigns categories to regions of this space. Random forest classifiers consist of a "forest" of decision trees; the majority vote of the trees determines the classification label applied to an input vector. Internal nodes in these trees pick a particular element of the input vector and compare it to a cutoff to determine which branch of the tree to continue the decision process along; leaf nodes are classification labels. Gradient boosted classifiers are a variation on RFCs wherein trees are not trained independently; instead, each tree is selected to minimize the error of the previous tree.

C. Training and Evaluation Approach

Training and evaluating a classifier intended to detect when a system is experiencing interference requires a dataset of sequences with known-baseline and known-interfered-with subsequences. Recall that our dataset contains either wholly-baseline or wholly-interfered-with sequences, but none that exhibit both modes of operation in one sequence. We devised an approach to synthesize sequences that are suitable for training and evaluating the classifiers from this dataset. Sequences from the baseline dataset and from the EMI-exposed dataset were alternated in either `BEBE` or `BEEB` patterns, with `B` referring to a baseline sequence and `E` an EMI-exposed sequence. These synthesized sequences were then sliced into windows, with the desired label (baseline or EMI-exposed) for the window

Lag	Events
0	x_i
0,1	$x_i, (x_{i-1}, x_i)$
1	(x_{i-1}, x_i)
2	(x_{i-2}, x_i)
3	(x_{i-3}, x_i)
4	(x_{i-4}, x_i)

TABLE II: Events generated from dataset at each lag

determined by the label of the majority of its states. Synthesized sequences could have unbalanced class representation or have balanced class representation by oversampling short baseline or EMI-exposed sequences until all subsequences used to synthesize a sequence contained the same number of snapshots. It is critical that balanced sequences be used only for training and not for classifier evaluation. For each synthetic dataset, 150 sequences were selected at random and 150 windows selected from each sequence for training and evaluation.

In addition to classifier-specific hyperparameters, we have several dataset-level hyperparameters:

- Synthesis pattern: `BEBE` or `BEEB`
- Balanced or unbalanced class representation
- Classification events, or "lags"
- Window length, for sliding-window classifiers

Two scores were selected to evaluate classifier performance: the standard F1 score and Matthews Correlation Coefficient (MCC). The MCC score is balanced: it takes into account true and false positives and negatives, and produces informative scores even when class representations are imbalanced, as they are in our dataset. All classification results are averaged across 5-fold cross-validation.

D. Results

Classifier-specific hyperparameter selection is performed using 5-fold cross-validation; the ideal hyperparameter settings for each classifier are as follows:

ANN	Trained for 100 epochs, learning rate of 0.001, Adam optimization with binary cross entropy loss.
SVM	Linear kernel, cost of 10,000.
RFC	100 trees, max tree depth of 12.
GBC	100 trees, max tree depth of 12, learning rate of 0.1.

We examine the variation in classifier performance due to our dataset-specific hyperparameters. Classifiers were trained for Lag 0; Lags 0,1; Lag 1; Lag 2; Lag 3; and Lag 4. Table II reviews the events used by the classifier at these lags.

The effect of window size is shown in Figure 4a. Note that for Lag 0,1, the window size used is double what is shown on the axis label. Empirically, doubling events requires double the window size to achieve equivalent performance to single-lag classifiers; this is an inherent disadvantage to multiple-lag classifiers. The reduced performance of the Lag 0 classifiers compared to all others indicates that the sequential relationship of state observations plays some role in accurately identifying IEMI events. Optimal window size for each classifier is determined by a one-sided Welch's t-test with $\alpha = 0.05$.

	Metrics				Hyperparameters	
Classifier	F1	MCC	Accuracy	Recall	Window Size	Lag
ANN	0.77	0.68	0.87	0.83	80	4
SVM	0.78	0.69	0.87	0.85	80	4
RFC	0.86	0.81*	0.92*	0.87	60	3
GBC	0.87*	0.81*	0.92*	0.92*	60	3
HMM	0.73	0.59	0.82	0.73	N/A	1

TABLE III: Best classifier configurations and performance. Best values in each column are marked with *.

For the ANN and SVM classifiers, the best window size is 80 states ($p_{\text{ANN}} < 0.01, p_{\text{SVM}} = 0.01$); the RFC and GBC classifiers achieve best results with a window size of 60 states ($p_{\text{RFC}} < 0.01, p_{\text{GBC}} < 0.01$). While peaks in the graph may appear somewhat later, the increase in performance is not statistically significant. Selecting for a smaller window size reduces classifier memory requirements and produces results more amenable to manual analysis.

Performance across different data synthesis approaches is shown in Figure 4b. Again, the performance increase from including sequence information (Lags > 0) can be seen, except for in the HMM. HMM performance, on the other hand, seems to be driven mostly by the number of observable states; fewer observable states leads to better results.

Balancing the class representation in the training dataset does not lead to an improvement in performance as determined by a two-tailed Welch's t-test for inequality, $p > 0.25$. However, we do observe a difference between `BEBE` and `BEEB` data ($p < 0.01$) for all but the HMM classifier. From this, we can conclude that the classifiers struggle the most to classify windows containing states from both `B` and `E` datasets. The increase in performance on the `BEEB` datasets is due to those datasets having only ⅔ of the mixed windows of the `BEBE` datasets. The HMM is an exception to this trend, with equivalent ($p = 0.99$) performance on either dataset.

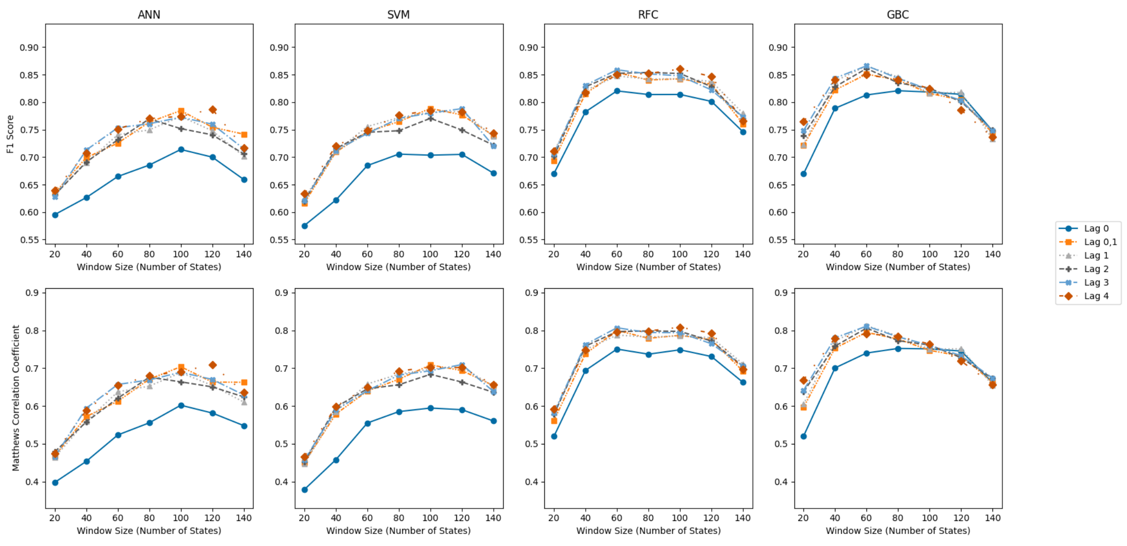
The MCC and F1 scores agree across all results, indicating good classifier performance on identifying both baseline and EMI-exposed states.

Best classification results on the unbalanced `BEBE` dataset are shown in Table III. Of these, the gradient boosted classifier performed the best on this dataset.

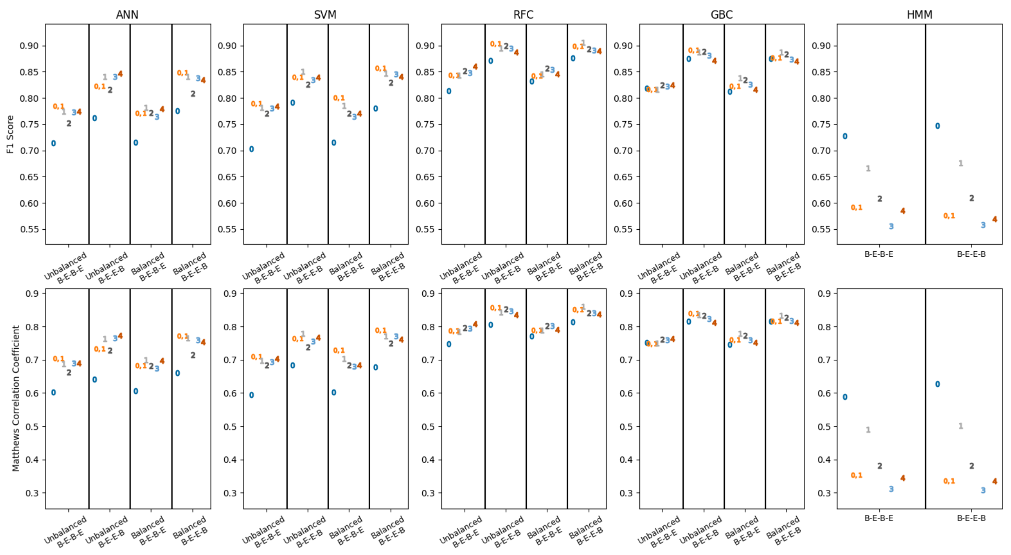
VI. Conclusion

The goal of this project is to detect the effects of electromagnetic interference in system peripherals. We focus on a USB host controller, which is the device responsible for all USB communication to and from a system. We capture sequences of snapshots of a host controller's control registers, both when the system is operating as intended and when it is being exposed to EMI. These sequences are processed into sequences of categorical time-series data for analysis. We divide the data into two datasets: baseline and EMI-exposed.

First, we analyzed various statistical properties of these datasets. The variance, as measured by the Gini index and extropy, differed between the two datasets. Furthermore, the autocorrelation of the two datasets differed. At a minimum,



(a) Classifier performance by window size on the unbalanced `BEBE` synthetic dataset



(b) Classifier performance by synthesized dataset with a window size of 100 (200 for Lag 0,1)

Fig. 4: Comparison of classifier performance

these differences allow us to conclude that our instrumentation is capable of detecting some effects of EMI.

Second, we investigated the ability of classifiers to identify whether system operation was indicative of EMI exposure. We devised a method for synthesizing a dataset for training classifiers from our experimental data. Several classification approaches were evaluated: hidden Markov models, neural networks, support vector machines, random forest classifiers, and gradient boosted classifiers. The gradient boosted random forest classifier performed best on our sliding window classification task, reaching 92% accuracy, 92% recall, and 0.87 F1 score.

Future work will take several directions. Additional data collection, correlated with external measures of EMI such as EMI generator trigger events and device under test power draw, will provide an expanded dataset with finer-grained ground truth for analysis. Stochastic process modeling and expanded statistical analysis will be investigated as an alternative approach to identifying operational anomalies. Anomalous sequences will be identified by classification and modeling techniques for further analysis of system operation, potentially enabling root-cause analysis. Finally, this approach will be expanded to additional system peripherals, leading to a whole-system EMI instrumentation and detection method.

References

[1] D. Pissoort, A. Degraeve, and K. Armstrong, "EMI risk management: A necessity for safe and reliable electronic systems!" in *2015 IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, Sep. 2015, pp. 208–210.

[2] W. A. Radasky, "The role of electromagnetic shielding in dealing with the threat of Intentional Electromagnetic Interference (IEMI)," in *2015 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, Sep. 2015, pp. 1145–1148.

[3] F. Sabath, "Classification of electromagnetic effects at system level," in *Ultra-Wideband, Short Pulse Electromagnetics 9*, F. Sabath, D. Giri, F. Rachidi, and A. Kaelin, Eds. New York, NY: Springer, 2010, pp. 325–333.

[4] T. Liang, G. Spadacini, F. Grassi, and S. A. Pignari, "Worst-case scenarios of radiated-susceptibility effects in a multiport system subject to intentional electromagnetic interference," *IEEE Access*, vol. 7, pp. 76 500–76 512, 2019.

[5] D. S. Guillette, T. J. Clarke, and C. Christodoulou, "Intentional electromagnetic irradiation of a microcontroller," in *2019 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, Sep. 2019, pp. 1214–1218.

[6] F. Burghardt and H. Garbe, "Effects of conducted interference on a microcontroller based on IEC 62132-4 and IEC 62215-3," in *2020 International Symposium on Electromagnetic Compatibility — EMC EUROPE*, Sep. 2020, pp. 1–5, iSSN: 2325-0364.

[7] J. Bai, Y. Shi, and G. Zhao, "Research on electromagnetic interference of vehicle GPS navigation equipment," in *2017 IEEE 5th International Symposium on Electromagnetic Compatibility (EMC-Beijing)*, Oct. 2017, pp. 1–5.

[8] M. Camp, J. Schmitz, and M. Jung, "Vulnerability and coupling behaviour of a TETRA communication system to electromagnetic fields," in *2015 IEEE International Symposium on Electromagnetic Compatibility (EMC)*, Aug. 2015, pp. 344–349.

[9] M. Lanzrath, C. Adami, B. Joerres, G. Lubkowski, M. Joester, M. Suhrke, and T. Pusch, "HPEM vulnerability of smart grid substations coupling paths into typical SCADA devices," in *2017 International Symposium on Electromagnetic Compatibility — EMC EUROPE*, Sep. 2017, pp. 1–6.

[10] C. Mao, F. G. Canavero, Z. Cui, and D. Sun, "System-level vulnerability assessment for EME: From fault tree analysis to bayesian networks—part II: Illustration to microcontroller system," *IEEE Transactions on Electromagnetic Compatibility*, vol. 58, no. 1, pp. 188–196, Feb. 2016.

[11] Y. V. Parfenov, B. A. Titov, L. N. Zdoukhov, and W. A. Radasky, "About the assessment of electronic device immunity to high power electromagnetic pulses," in *2015 7th Asia-Pacific Conference on Environmental Electromagnetics (CEEM)*, Nov. 2015, pp. 428–431.

[12] K. Armstrong and W. A. Radasky, "Extending the normal immunity tests to help prove functional safety," in *2018 IEEE International Symposium on Electromagnetic Compatibility and 2018 IEEE Asia-Pacific Symposium on Electromagnetic Compatibility (EMC/APEMC)*, May 2018, pp. 221–226.

[13] D. V. Giri, R. Hoad, and F. Sabath, "Implications of high-power electromagnetic (HPEM) environments on electronics," *IEEE Electromagnetic Compatibility Magazine*, vol. 9, no. 2, pp. 37–44, 2020.

[14] ——, *High-power electromagnetic effects on electronic systems*. Boston London: Artech House, 2020.

[15] C. Kasmi, J. Lopes-Esteves, N. Picard, M. Renard, B. Beillard, E. Martinod, J. Andrieu, and M. Lalande, "Event logs generated by an operating system running on a COTS computer during IEMI exposure," *IEEE Transactions on Electromagnetic Compatibility*, vol. 56, no. 6, pp. 1723–1726, Dec. 2014.

[16] J. Lopes-Esteves, E. Cottais, and C. Kasmi, "Software instrumentation of an unmanned aerial vehicle for HPEM effects detection," in *2018 2nd URSI Atlantic Radio Science Meeting (AT-RASC)*, May 2018, pp. 1–4.

[17] C. Kasmi, J. Lopes-Esteves, and M. Renard, "Autonomous electromagnetic attacks detection considering a COTS computer as a multi-sensor system," in *2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS)*, Aug. 2014, pp. 1–4.

[18] X. Liu, G. Maghlakelidze, J. Zhou, O. H. Izadi, L. Shen, M. Pommerenke, S. S. Ge, and D. Pommerenke, "Detection of ESD-induced soft failures by analyzing Linux kernel function calls," *IEEE Transactions on Device and Materials Reliability*, vol. 20, no. 1, pp. 128–135, Mar. 2020.

[19] N. Jarus, A. Sabatini, P. Maheshwari, and S. S. Sarvestani, "Software-based monitoring and analysis of a USB host controller subject to electrostatic discharge," in *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, Jun. 2020, pp. 1–7.

[20] N. Jarus, A. Sabatini, P. Maheshwari, and S. Sedigh Sarvestani, "Detection, analysis, and prediction of the effects of electrostatic discharge on a USB host controller," *IEEE Transactions on Electromagnetic Compatibility*, under review.

[21] Y. Kitagawa, T. Ishigooka, and T. Azumi, "Anomaly prediction based on machine learning for memory-constrained devices," *IEICE Transactions on Information and Systems*, vol. E102.D, no. 9, pp. 1797–1807, Sep. 2019.

[22] Y. Li, W. Xue, T. Wu, H. Wang, B. Zhou, S. Aziz, and Y. He, "Intrusion detection of cyber physical energy system based on multivariate ensemble classification," *Energy*, p. 119505, Dec. 2020.

[23] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "MADneSs: a Multi-layer Anomaly Detection Framework for Complex Dynamic Systems," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2019.

[24] M. Alam, S. Bhattacharya, and D. Mukhopadhyay, "Victims can be saviors: A machine learning-based detection for micro-architectural side-channel attacks," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 17, no. 2, pp. 14:1–14:31, Jan. 2021.

[25] Intel Corporation, "Enhanced host controller interface specification for universal serial bus," 2001.

[26] O. H. Izadi, R. K. Frazier, N. Altunyurt, S. Sedigh Sarvestani, D. Pommerenke, and C. Hwang, "A new tunable damped sine-like waveform generator for IEMI applications," in *2020 IEEE International Symposium on Electromagnetic Compatibility Signal/Power Integrity (EMCSI)*, Jul. 2020, pp. 282–286.

[27] C. H. Weiß, *An introduction to discrete-valued time series*. Hoboken, NJ: John Wiley & Sons, 2017.

[28] F. Lad, G. Sanfilippo, and G. Agrò, "Extropy: Complementary dual of Entropy," *Statistical Science*, vol. 30, no. 1, pp. 40–58, Feb. 2015.

[29] C. H. Weiß, "Measures of dispersion and serial dependence in categorical time series," *Econometrics*, vol. 7, no. 2, p. 17, Jun. 2019.

Appendix
Observed State Space

	Registers						
State	Command	Config'd Flag	Interrupt Enable	Status	TX Fill Tuning	Mode	Mode Extended
s_1	0x10005	0x1	0x37	0x0000	0x0	0x10005	0x10005
s_2	0x10005	0x1	0x37	0x0008	0x0	0x10005	0x10005
s_3	0x10025	0x1	0x37	0x8000	0x0	0x10025	0x10025
s_4	0x10025	0x1	0x37	0x8001	0x0	0x10025	0x10025
s_5	0x10025	0x1	0x37	0x8008	0x0	0x10025	0x10025
s_6	0x10025	0x1	0x37	0x8009	0x0	0x10025	0x10025
s_7	0x10025	0x1	0x37	0x8020	0x0	0x10025	0x10025
s_8	0x10025	0x1	0x37	0x8021	0x0	0x10025	0x10025
s_9	0x10025	0x1	0x37	0x8028	0x0	0x10025	0x10025
s_{10}	0x10025	0x1	0x37	0xA000	0x0	0x10025	0x10025
s_{11}	0x10025	0x1	0x37	0xA001	0x0	0x10025	0x10025
s_{12}	0x10025	0x1	0x37	0xA008	0x0	0x10025	0x10025
s_{13}	0x10025	0x1	0x37	0xA009	0x0	0x10025	0x10025
s_{14}	0x10025	0x1	0x37	0xA020	0x0	0x10025	0x10025
s_{15}	0x10025	0x1	0x37	0xA021	0x0	0x10025	0x10025
s_{16}	0x10065	0x1	0x37	0x8000	0x0	0x10025	0x10025
s_{17}	0x10065	0x1	0x37	0x8000	0x0	0x10065	0x10065
s_{18}	0x10065	0x1	0x37	0xA000	0x0	0x10025	0x10025
s_{19}	0x10065	0x1	0x37	0xA020	0x0	0x10025	0x10025

TABLE IV: State space and register values from corresponding snapshot