La inseguridad del hardware actual

Alfonso Pili, Iñaky Vydra, Santiago Barrionuevo

Abstract—Este articulo realizara estudio de la seguridad en la arquitectura del hardware, profundizara en temas como las vulnerabilidades de los actuales procesadores, sus respectivas mitigaciones y por ultimo se detallaran los campos de investigacion en proceso que ayudan a prevenir o preparanos ante estos ataques maliciosos.

Index Terms—Predicciones de saltos, ejecucion especulativa, Meltodown, Spectre y sus variantes,Ataques de canales laterales, Mitigaciones y prevenciones.

1 Introduccion

La seguridad de la informacion es un tema cada vez mas revelante en la era digital. Los procesadores son el corazon de cualquier sistema informatico, y su seguridad es crucial para garantizar la integridad y la confidencialidad de los datos. Sin embargo, la complejidad de la microarquitectura de los procesadores modernos y la constante evolucion de las amenazas informaticas hacen que la seguridad de estos dispositivos sea un desafio cada vez mas complejo.

2 Optimizaciones

El rendimiento de los procesadores modernos ha mejorado gracias a varias técnicas. Como ya no es tan fácil seguir reduciendo el tamaño de los transistores o aumentar la frecuencia del reloj, se han buscado otras soluciones a nivel de arquitectura. La clave está en el paralelismo: optimizar y extender las rutas de instrucciones para ejecutar más operaciones al mismo tiempo dentro de un hilo, además de aumentar la cantidad de núcleos físicos y lógicos en un procesador.

Las optimizaciones claves de los procesadores modernos son: la **ejecución especulativa**, **ejecución fuera de orden**y la **predicción de bifurcaciones**(Branch Prediction).

- **Ejecución Especulativa**: Permite al procesador ejecutar instrucciones "por adelantado" **(aprovechando el paralelismo y la predicción de bifurcaciones)[?]** para evitar tiempos de espera. Si la predicción es incorrecta, los resultados especulativos se descartan, aunque pueden dejar rastros en la caché, lo que ha dado lugar a vulnerabilidades.
- **Ejecución Fuera de Orden**: Aprovecha al máximo los recursos del procesador ejecutando instrucciones en paralelo en distintas unidades de ejecución. Mientras una unidad está ocupada, las demás pueden procesar otras instrucciones de forma simultánea, siempre que no rompa las reglas arquitectónicas dadas por el hardware.
- **Predicción de Bifurcaciones**: Es una técnica utilizada para decidir qué rama de un condicional es más probable que se ejecute. Esto optimiza el flujo de instrucciones al anticiparse al resultado de la condición, evitando interrupciones en el pipeline del procesador y mejorando el rendimiento general.

Los procesadores modernos implementan estas técnicas de optimización para mejorar el rendimiento. Sin embargo, estas mismas técnicas pueden ser explotadas para crear **canales laterales** de los cuales hablaremos mas adelante en la seccion de **Vulnerabilidades**.

3 Niveles de Privilegios

Antes de hablar de las vulnerabilidades por explotación de canales laterales de caché aprovechando las optimizaciones del microprocesador, es fundamental entender cómo se gestionan los privilegios de las instrucciones para interactuar con la CPU y la memoria, ya que esto influye directamente en la seguridad del sistema y en la posibilidad de ataques que exploten estas interacciones.
En la Fig. 1 se pueden apreciar 4 anillos de seguridad en las arquitecturas x86, aunque normalmente los sistemas operativos solo utilizan el nivel 0 y el nivel 3 lo que les permite utilizar unicamente 1 bit para verificar el modo en el cual debe trabajar el procesador.

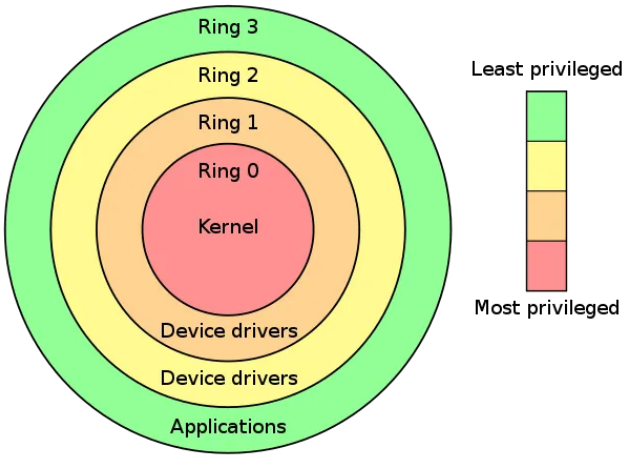

Fig. 1. Distribucion jerarquica de los privilegios.

Cada modo tiene una región de memoria mapeada diferente; ese aislamiento está dado por la **virtualización de la memoria**. Por lo tanto, el modo usuario no puede acceder

a la misma información que el modo kernel, al menos de manera directa.

El **modo usuario** es el responsable del anillo 3, el cual solo ejecuta código a nivel software. Es decir, está aislado tanto del sistema operativo como del kernel; sus recursos son limitados y, por cada intento de modificación de memoria, manejo de archivos o contacto con el hardware, se realiza una **syscall**. Estas llamadas al sistema permiten al programa cambiar al modo kernel, dejando que el sistema operativo termine con las instrucciones necesarias y luego vuelva al modo usuario sin violar la seguridad. De aquí podemos deducir que, en el anillo de nivel 0, se ejecuta únicamente código a nivel kernel. Este nivel es administrado por el **modo kernel** y es el nivel de seguridad más importante. El modo kernel tiene contacto directo con el hardware, es decir, tiene acceso a todo el mapeado de la memoria, incluidas las tablas de páginas, donde se encuentra nuestra información más sensible, como contraseñas, credenciales y demás datos privados. Por lo tanto, el principal problema aquí es que, si algún programa malicioso llegara a ejecutar instrucciones en modo kernel o explotara la ejecución especulativa y fuera de orden para ejecutar instrucciones sin los privilegios necesarios, se pondría en riesgo toda nuestra seguridad, ya que podrían acceder a las páginas del kernel sin siquiera tener los privilegios adecuados.

4 VULNERABILIDADES

La seguridad de los sistemas puede verse comprometida por distintos tipos de vulnerabilidades. En este artículo, haremos hincapié en los **ataques de canales laterales** (del inglés Side Channel Attacks), que explotan información derivada de las condiciones físicas o lógicas durante la ejecución de un sistema para extraer datos sensibles.

¿A qué nos referimos con ”condiciones físicas y lógicas”? A que estos ataques no se basan en vulnerabilidades directas en algoritmos o software, sino en la medición de:

Condiciones físicas: Fenómenos observables en el hardware, como el consumo de energía, el tiempo de ejecución de operaciones o las emisiones electromagnéticas/acústicas.

Condiciones lógicas: Los comportamientos de la implementación del sistema, como el estado de la memoria caché, el mapeo de memoria o los patrones de acceso a recursos compartidos.

Estas fugas de información indirecta son los llamados **canales laterales**, que permiten al atacante obtener datos secretos sin acceder directamente a ellos.
Existen muchismos casos de canales laterales que se pueden explotar pero en este informe solo vamos a dentrar en el canal de temporizacion de cache (cahce timing side-channel), el cual es utilizado por Meltdown y Spectre.

4.1 Spectre

Spectre es una vulnerabilidad que explota la ejecución especulativa para engañar al procesador y hacer que acceda a datos sensibles en la memoria, rompiendo el aislamiento entre procesos.
Afecta casi todos los procesadores modernos (Intel, AMD, ARM).
Fue descubierto por dos equipos de investigadores: Jann Horn (Google Project Zero) y Paul Kocher, en colaboración con Daniel Genkin (University of Pennsylvania, University of Maryland), Mike Hamburg (Rambus), Moritz Lipp (Graz University of Technology) y Yuval Yarom (University of Adelaide, Data61).
Funciona de la siguiente manera: Primero, el atacante manipula la unidad de predicción de ramas del procesador para inducir una predicción incorrecta sobre las instrucciones que se ejecutarán. De esta manera, el procesador se prepara para ejecutar instrucciones especulativamente basadas en esa suposición errónea. A continuación, el procesador, confiando en su predicción equivocada, ejecuta especulativamente instrucciones que no debería ejecutar. Estas instrucciones pueden incluir accesos a áreas de memoria privadas, como datos de otros procesos o del sistema operativo. Aunque el procesador eventualmente detecta el error y descarta estas instrucciones, los efectos de esos accesos permanecen en la memoria caché. Por último, el atacante explota estos efectos secundarios midiendo los tiempos de acceso a la caché con técnicas como Flush+Reload. Si el acceso a una dirección de memoria es más rápido, significa que los datos fueron cargados especulativamente y están en la caché. Si el acceso es más lento, significa que los datos no estaban en caché. A través de estos tiempos de acceso, el atacante puede reconstruir información sensible sin necesidad de acceder directamente a la memoria protegida.
A continuacion, presentaremos 2 variantes de Spectre:

4.1.1 *Spectre-PHT*

En este tipo de ataque, el atacante manipula el predictor de saltos para que el procesador especule incorrectamente sobre una comparación en un programa. En el siguiente codigo de C:

```
if (x < array1_size)
    y = array2[array1[x] * 4096];
```

Fig. 2. Fragmento de cdigo en C ilustrando el concepto de manipulacion de prediccion de saltos.

x es un valor controlado por el atacante. Normalmente, el procesador verifica que x este dentro de los limites de array1, pero si el predictor de saltos ha sido entrenado para asumir que x es siempre valido, el procesador accedera a array1[x] sin verificar los limites. Si x es un valor malicioso, puede hacer que array1[x] apunte a una direccion confidencial. Luego array2[array1[x] * 4096] almacenara esa informacion confidencial en la cache, permitiendo que el atacante lo extraiga mediante mediciones de tiempo de acceso.

4.1.2 *Spectre-BTB*

En esta variante, el atacante manipula la Branch Target Buffer (BTB), que es la estructura del procesador encargada de predecir el destino de saltos indirectos. El atacante entrena el BTB para que un salto indirecto en el código del programa víctima se redirija erróneamente a una ubicación de código que el atacante elige. Esta ubicación puede contener una secuencia de instrucciones (un ”gadget”) que filtra información confidencial a través de la caché.

4.2 Meltdown

Meltdown es una vulnerabilidad que permite a un atacante acceder a la memoria protegida del sistema, incluyendo la memoria del kernel, rompiendo los mecanismos de aislamiento entre aplicaciones y el sistema operativo.
Fue descubierto por tres equipos diferentes: Jann Horn (Google Project Zero), Werner Haas y Thomas Prescher (Cyberus Technology) y Daniel Gruss, Moritz Lipp, Stefan Mangard y Michael Schwarz (Graz University of Technology).
Esta vulnerabilidad afecta principalmente a los procesadores Intel fabricados desde 1995, aunque ciertas variantes también pueden impactar a algunos modelos de ARM y AMD.
Funciona de la siguiente manera:
Primero, el atacante manipula el programa para que el procesador cargue en un registro el contenido de una dirección de memoria secreta, a la que normalmente no tendría acceso directo. Luego, debido a la ejecución especulativa, el procesador accede anticipadamente a la memoria caché basándose en ese contenido secreto, dejando un rastro en la caché. Finalmente, el atacante explota este rastro utilizando la técnica Flush+Reload, que le permite monitorear la caché y detectar qué datos han sido accedidos recientemente, revelando así información confidencial.
A continuacion, presentaremos 3 variantes de Meltdown:

4.2.1 ***Meltdown-P***

Este ataque afecta a los procesadores Intel y explota una vulnerabilidad en el sistema SGX, que es una tecnologia que crea espacios seguros en la memoria donde se pueden almacenar datos confidenciales..
El ataque se basa en manipular la tabla de paginas, que es un mapa que le indica al procesador que direccion tiene cada pieza de informacion. Cuando un atacante cambia alguna de estas direcciones en la tabla, se provoca un fallo de protección. Este fallo debería impedir el acceso a los datos y bloquear el sistema, pero debido a la ejecución especulativa, el procesador sigue trabajando aunque haya ocurrido un error. A pesar de detectar el fallo, el procesador continúa ejecutando instrucciones usando direcciones de memoria incorrectas, lo que hace que los datos se carguen temporalmente en la caché L1. Luego el atacante puede usar la tecnica flush+reload para extraer esta informacion.

4.2.2 ***Meltdown-NM***

Este ataque aprovecha cómo la CPU maneja de manera ineficiente los registros de la unidad de punto flotante (FPU) al cambiar de un proceso a otro. En lugar de guardar los registros de inmediato cuando se cambia de proceso, la CPU los marca como ”no disponibles” y los guarda solo cuando se vuelven a utilizar.
El ataque se desarrolla en tres pasos:
1- La víctima carga datos en la FPU.
2- La CPU cambia al atacante y marca la FPU como ”no disponible”.
3- El atacante ejecuta una instrucción que usa la FPU, lo que genera una excepción. Sin embargo, antes de que la CPU detenga la ejecución, ya ha procesado temporalmente instrucciones usando datos de la víctima.
De esta manera, el atacante puede aprovechar los datos no guardados y extraer información sensible que se encontraba en los registros de la FPU, sin tener acceso directo a ella.

4.2.3 ***Meltdown-PK***

Los procesadores Intel Skylake-SP para servidores incluyen una función de protección de claves de memoria (PKU), que permite modificar los permisos de acceso a páginas de memoria dentro de un proceso sin necesidad de realizar una llamada al sistema o al hipervisor. Esto facilita un aislamiento eficiente de datos sensibles.
Meltdown-PK es un ataque que evade las restricciones de lectura y escritura impuestas por PKU. Aunque un atacante no tenga permisos para modificar los permisos de acceso con la instrucción wrpkru, puede explotar la ejecución especulativa del procesador para acceder temporalmente a datos protegidos y filtrarlos a través de un canal encubierto.

5 MITIGACIONES

Las vulnerabilidades de Spectre y Meltdown surgen como consecuencia directa de optimizaciones en los procesadores modernos. Aunque estas tecnicas se enfocan en mejorar el rendimiento, tambien han introducido riesgos de seguridad que han permitido el desarrollo de ataques capaces de comprometer la seguridad de los sistemas.

Dado que el problema esta relacionado con el diseño de la arquitectura de los procesadores, no tendremos una solucion definitiva hasta que el hardware sea modificado. Sin embargo, debido al tiempo que toma el desarrollo y fabricacion de nuevos chips, podrian pasar varios años antes de que las futuras generaciones de CPUs solucionen completamente el problema.

Mientras tanto, se han desarrollado diversas mitigaciones a nivel de software y hardware que pueden controlar algunas variantes de estas vulnerabilidades y minimizar el riesgo de ataques en sistemas actuales, en este informe vamos a listar algunas de las mas relevantes.

5.1 Aislamiento de Tablas de Paginas y Aleatoridad

El kernel se encarga de establecer tablas de páginas para controlar la asignación entre una dirección virtual y una dirección física. El acceso entre el núcleo y el espacio de usuario también se controla mediante las tablas de páginas. Una página que está mapeada para el kernel no es accesible desde el espacio de usuario, aunque el núcleo normalmente puede acceder al espacio de usuario.

Traducir estos mapas puede ser costoso, por lo que el hardware tiene un búfer de traducción de direcciones (TLB) para almacenar asignaciones. A veces es necesario eliminar las asignaciones antiguas (invalidar el TLB), pero hacerlo es costoso, por lo que el código se escribe para minimizar tales llamadas. Un truco aquí es mantener siempre las tablas de páginas del núcleo mapeadas cuando los procesos de usuario se están ejecutando. Los permisos de la tabla de páginas impiden que el espacio de usuario acceda a las asignaciones del kernel, pero el núcleo puede acceder a ellas inmediatamente cuando se realiza una llamada al sistema.
Exploits como Meltdown demostraron que este método de mantener las tablas de páginas del kernel disponibles en

el espacio de usuario, si bien no permite directamente la filtración de información del núcleo, puede ser explotado por atacantes. Estos generan intentos de lectura sobre dichas páginas; al no contar con los permisos necesarios, la CPU bloquea el programa por seguridad (ya que se está ejecutando en modo usuario). Sin embargo, la CPU prebusca los datos de todas formas. Este comportamiento es aprovechado por los atacantes para medir el tiempo de acceso a los datos prebuscados y, de esta manera, llevar a cabo un ataque de canal lateral basado en caché.

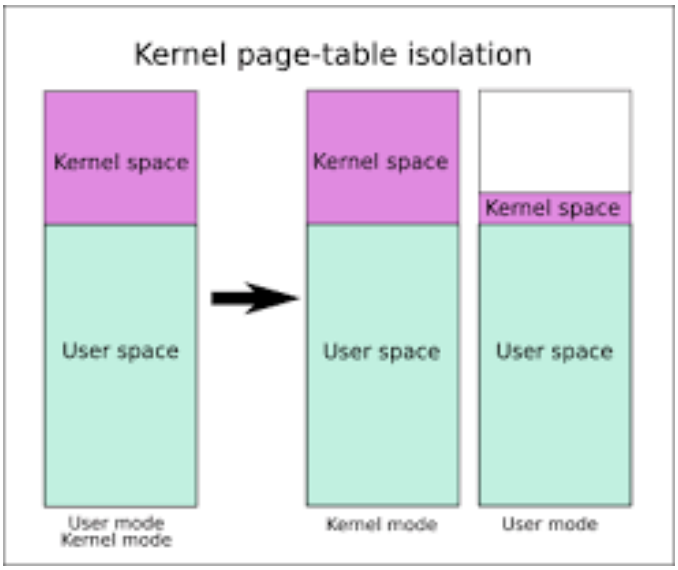

Fig. 3. Imagen representativa de como se modifican las tablas del modo usuario y kernel, luego de aplicarle el aislamiento.

5.2 Retpoline

El nombre Retpoline es un acrónimo de "return" (retorno) y "trampoline" (trampolín). En este contexto, un trampolin es un pequeño fragmento de codigo invocado como funcion, que realiza una tarea breve y, a diferencia de una llamada a funcion comun, no regresa a su llamador despues de la ejecución, sino que salta a otra ubicacion, como se muestra en la Figura 3. Si el destino del salto del trampolin es una funcion, al retornar, dicha funcion regresara al llamador original del trampolin.

Retpoline fue desarrollado por Google para mitigar los ataques Spectre-BTB. Su objetivo es evitar que este tipo de ataques manipulen el Branch Target Buffer (BTB) y desvien la ejecucion especulativa hacia una direccion arbitraria. Para ello, Retpoline reemplaza las bifurcaciones indirectas con una secuencia especial de instrucciones basada en llamadas y retornos, asegurando que la ejecucion especulativa permanezca en un bucle inofensivo dentro del Return Stack Buffer (RSB). Esto impide que un atacante manipule las predicciones de bifurcaciones para ejecutar codigo malicioso.

Para lograr esto, Retpoline aprovecha la forma en que los procesadores gestionan las llamadas y retornos de funciones. Al utilizar una instruccion call hacia un trampolin, se llena el RSB con una direccion de retorno controlada. Cuando el procesador especula sobre el retorno, en lugar

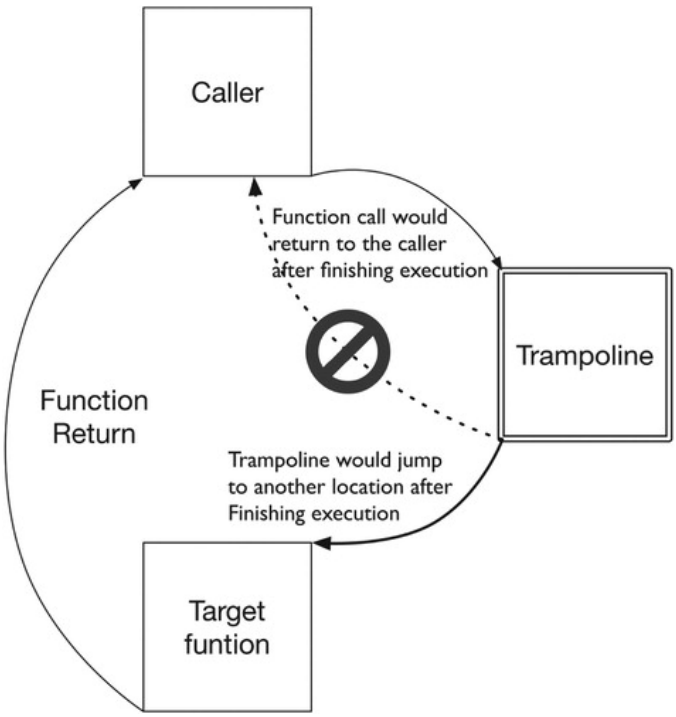

Fig. 4. Trampoline se refiere a un fragmento de código que saltaría a otra ubicación después de terminar la ejecución en lugar de regresar al llamador original.

de seguir una prediccion potencialmente manipulada desde el BTB, sigue la direccion almacenada en el RSB y ejecuta un bucle de instrucciones seguras.

6 Where to get the IEEEtran Templates

The **IEEE Template Selector** will always have the most up-to-date versions of the LaTeX and MSWord templates. Please see: https://template-selector.ieee.org/ and follow the steps to find the correct template for your intended publication. Many publications use the IEEETran LaTeX templates, however, some publications have their own special templates. Many of these are based on IEEEtran, but may have special instructions that vary slightly from those in this document.

7 Where to get LaTeX help - user groups

The following on-line groups are very helpful to beginning and experienced LaTeX users. A search through their archives can provide many answers to common questions.

http://www.latex-community.org/
https://tex.stackexchange.com/

8 Document Class Options in IEEEtran

At the beginning of your LaTeX file you will need to establish what type of publication style you intend to use. The following list shows appropriate documentclass options for each of the types covered by IEEEtran.

Regular Journal Article
`\documentclass[journal]IEEEtran`

Conference Paper

```
\documentclass[conference]IEEEtran
```

Computer Society Journal Article
```
\documentclass[10pt,journal,compsoc]IEEEtran
```

Computer Society Conference Paper
```
\documentclass[conference,compsoc]IEEEtran
```

Communications Society Journal Article
```
\documentclass[journal,comsoc]IEEEtran
```

Brief, Correspondence or Technote
```
\documentclass[9pt,technote]IEEEtran
```

There are other options available for each of these when submitting for peer review or other special requirements. IEEE recommends to compose your article in the base 2-column format to make sure all your equations, tables and graphics will fit the final 2-column format. Please refer to the document "IEEEtran_HOWTO.pdf" for more information on settings for peer review submission if required by your EIC.

9 How to Create Common Front Matter

The following sections describe general coding for these common elements. Computer Society publications and Conferences may have their own special variations and will be noted below.

9.1 Paper Title

The title of your paper is coded as:

```
\title{The Title of Your Paper}
```

Please try to avoid the use of math or chemical formulas in your title if possible.

9.2 Author Names and Affiliations

The author section should be coded as follows:

```
\author{Masahito Hayashi
\IEEEmembership{Fellow, IEEE}, Masaki Owari
\thanks{M. Hayashi is with Graduate School
of Mathematics, Nagoya University, Nagoya,
Japan}
\thanks{M. Owari is with the Faculty of
Informatics, Shizuoka University,
Hamamatsu, Shizuoka, Japan.}
}
```

Be sure to use the \IEEEmembership command to identify IEEE membership status. Please see the "IEEEtran_HOWTO.pdf" for specific information on coding authors for Conferences and Computer Society publications. Note that the closing curly brace for the author group comes at the end of the thanks group. This will prevent you from creating a blank first page.

9.3 Running Heads

The running heads are declared by using the \markboth command. There are two arguments to this command: the first contains the journal name information and the second contains the author names and paper title.

```
\markboth{Journal of Quantum Electronics,
Vol. 1, No. 1, January 2021}
{Author1, Author2, }
\MakeLowercase{\textit{(et al.)}:
Paper Title}
```

9.4 Copyright Line

For Transactions and Journals papers, this is not necessary to use at the submission stage of your paper. The IEEE production process will add the appropriate copyright line. If you are writing a conference paper, please see the "IEEEtran_HOWTO.pdf" for specific information on how to code "Publication ID Marks".

9.5 Abstracts

The abstract is the first element of a paper after the \maketitle macro is invoked. The coding is simply:

```
\begin{abstract}
Text of your abstract.
\end{abstract}
```

Please try to avoid mathematical and chemical formulas in the abstract.

9.6 Index Terms

The index terms are used to help other researchers discover your paper. Each society may have it's own keyword set. Contact the EIC of your intended publication for this list.

```
\begin{IEEEkeywords}
Broad band networks, quality of service
\end{IEEEkeywords}
```

10 How to Create Common Body Elements

The following sections describe common body text elements and how to code them.

10.1 Initial Drop Cap Letter

The first text paragraph uses a "drop cap" followed by the first word in ALL CAPS. This is accomplished by using the \IEEEPARstart command as follows:

```
\IEEEPARstart{T}{his} is the first paragraph
of your paper. . .
```

10.2 Sections and Subsections

Section headings use standard LaTeX commands: \section, \subsection and \subsubsection. Numbering is handled automatically for you and varies according to type of publication. It is common to not indent the first paragraph following a section head by using \noindent as follows:

```
\section{Section Head}
\noindent The text of your paragraph . . .
```

Fig. 5. This is the caption for one fig.

10.3 Citations to the Bibliography

The coding for the citations are made with the LaTeX `\cite` command. This will produce individual bracketed reference numbers in the IEEE style. At the top of your LaTeX file you should include:

```
\usepackage{cite}
```

For a single citation code as follows:

```
see \cite{ams}
```

This will display as: see [1]

For multiple citations code as follows:

```
\cite{ams,oxford,lacomp}
```

This will display as [1], [2], [3]

10.4 Figures

Figures are coded with the standard LaTeX commands as follows:

```
\begin{figure}[!t]
\centering
\includegraphics[width=2.5in]{fig1}
\caption{This is the caption for one fig.}
\label{fig1}
\end{figure}
```

The [!t] argument enables floats to the top of the page to follow IEEE style. Make sure you include:

```
\usepackage{graphicx}
```

at the top of your LaTeXfile with the other package declarations.

To cross-reference your figures in the text use the following code example:

```
See figure \ref{fig1} ...
```

This will produce:
See figure 5 . . .

TABLE 1
A Simple Table Example.

Order of filter	Arbitrary coefficients e_m	coefficients b_{ij}
1	$b_{ij} = \hat{e}.\hat{\beta_{ij}}$,	$b_{00} = 0$
2	$\beta_{22} = (\ 1, -1, -1,\ \ 1,\ \ 1,\ \ 1)$	
3	$b_{ij} = \hat{e}.\hat{\beta_{ij}}$,	$b_{00} = 0$,

10.5 Tables

Tables should be coded with the standard LaTeX coding. The following example shows a simple table.

```
\begin{table}
\begin{center}
\caption{Filter design equations  ...}
\label{tab1}
\begin{tabular}{| c | c | c |}
\hline
Order & Arbitrary coefficients &
coefficients\\
of filter & $e_m$ &   $b_{ij}$ \\
\hline
1& $b_{ij}=\hat{e}.\hat{\beta_{ij}}$,
& $b_{00}=0$\\
\hline
2&$\beta_{22}=(~1,-1,-1,~~1,~~1,~~1)$ &\\
\hline
3& $b_{ij}=\hat{e}.\hat{\beta_{ij}}$,
& $b_{00}=0$,\\
\hline
\end{tabular}
\end{center}
\end{table}
```

To reference the table in the text, code as follows:

```
Table~\ref{tab1} lists the closed-form...
```

to produce:
Table 1 lists the closed-form . . .

10.6 Lists

In this section, we will consider three types of lists: simple unnumbered, numbered and bulleted. There have been numerous options added to IEEEtran to enhance the creation of lists. If your lists are more complex than those shown below, please refer to the "IEEEtran_HOWTO.pdf" for additional options.

A plain unnumbered list

bare_jrnl.tex
bare_conf.tex
bare_jrnl_compsoc.tex
bare_conf_compsoc.tex
bare_jrnl_comsoc.tex

coded as:

```
\begin{list}{}{}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
```

```
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{list}
```

A simple numbered list

1) bare_jrnl.tex
2) bare_conf.tex
3) bare_jrnl_compsoc.tex
4) bare_conf_compsoc.tex
5) bare_jrnl_comsoc.tex

coded as:

```
\begin{enumerate}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{enumerate}
```

A simple bulleted list

- bare_jrnl.tex
- bare_conf.tex
- bare_jrnl_compsoc.tex
- bare_conf_compsoc.tex
- bare_jrnl_comsoc.tex

coded as:

```
\begin{itemize}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{itemize}
```

10.7 Other Elements

For other less common elements such as Algorithms, Theorems and Proofs, and Floating Structures such as page-wide tables, figures or equations, please refer to the "IEEEtran_HOWTO.pdf" section on "Double Column Floats."

11 How to Create Common Back Matter Elements

The following sections demonstrate common back matter elements such as Acknowledgments, Bibliographies, Appendicies and Author Biographies.

11.1 Acknowledgments

This should be a simple paragraph before the bibliography to thank those individuals and institutions who have supported your work on this article.

```
\section{Acknowledgments}
\noindent Text describing those who
supported your paper.
```

11.2 Bibliographies

References Simplified: A simple way of composing references is to use the `\bibitem` macro to define the beginning of a reference as in the following examples:

[6] H. Sira-Ramirez. "On the sliding mode control of nonlinear systems," *Systems & Control Letters*, vol. 19, pp. 303–312, 1992.
coded as:

```
\bibitem{Sira3}
H. Sira-Ramirez. ``On the sliding mode
control of nonlinear systems,''
\textit{Systems \& Control Letters},
vol. 19, pp. 303--312, 1992.
```

[7] A. Levant."Exact differentiation of signals with unbounded higher derivatives," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, California, USA, pp. 5585–5590, 2006.
coded as:

```
\bibitem{Levant}
A. Levant. ``Exact differentiation of
signals with unbounded higher
derivatives,''  in \textit{Proceedings
of the 45th IEEE Conference on
Decision and Control}, San Diego,
California, USA, pp. 5585--5590, 2006.
```

[8] M. Fliess, C. Join, and H. Sira-Ramirez. "Non-linear estimation is easy," *International Journal of Modelling, Identification and Control*, vol. 4, no. 1, pp. 12–27, 2008.
coded as:

```
\bibitem{Cedric}
M. Fliess, C. Join, and H. Sira-Ramirez.
``Non-linear estimation is easy,''
\textit{International Journal of Modelling,
Identification and Control}, vol. 4,
no. 1, pp. 12--27, 2008.
```

[9] R. Ortega, A. Astolfi, G. Bastin, and H. Rodriguez. "Stabilization of food-chain systems using a port-controlled Hamiltonian description," in *Proceedings of the American Control Conference*, Chicago, Illinois, USA, pp. 2245–2249, 2000.
coded as:

```
\bibitem{Ortega}
R. Ortega, A. Astolfi, G. Bastin, and H.
Rodriguez. ``Stabilization of food-chain
systems using a port-controlled Hamiltonian
description,'' in \textit{Proceedings of the
American Control Conference}, Chicago,
Illinois, USA, pp. 2245--2249, 2000.
```

11.3 Accented Characters in References

When using accented characters in references, please use the standard LaTeX coding for accents. **Do not use math coding for character accents**. For example:

```
\'e, \"o, \`a, \~e
```

will produce: é, ö, à, ẽ

11.4 Use of BibTeX

If you wish to use BibTeX, please see the documentation that accompanies the IEEEtran Bibliography package.

11.5 Biographies and Author Photos

Authors may have options to include their photo or not. Photos should be a bit-map graphic (.tif or .jpg) and sized to fit in the space allowed. Please see the coding samples below:

```
\begin{IEEEbiographynophoto}{Jane Doe}
Biography text here without a photo.
\end{IEEEbiographynophoto}
```

or a biography with a photo

```
\begin{IEEEbiography}[{\includegraphics
[width=1in,height=1.25in,clip,
keepaspectratio]{fig1.png}}]
{IEEE Publications Technology Team}
In this paragraph you can place
your educational, professional background
and research and other interests.
\end{IEEEbiography}
```

Please see the end of this document to see the output of these coding examples.

12 Mathematical Typography and Why It Matters

Typographical conventions for mathematical formulas have been developed to **provide uniformity and clarity of presentation across mathematical texts**. This enables the readers of those texts to both understand the author's ideas and to grasp new concepts quickly. While software such as LaTeX and MathType® can produce aesthetically pleasing math when used properly, it is also very easy to misuse the software, potentially resulting in incorrect math display.

IEEE aims to provide authors with the proper guidance on mathematical typesetting style and assist them in writing the best possible article.

As such, IEEE has assembled a set of examples of good and bad mathematical typesetting. You will see how various issues are dealt with. The following publications have been referenced in preparing this material:

Mathematics into Type, published by the American Mathematical Society
The Printing of Mathematics, published by Oxford University Press
The LaTeXCompanion, by F. Mittelbach and M. Goossens
More Math into LaTeX, by G. Grätzer
AMS-StyleGuide-online.pdf, published by the American Mathematical Society

Further examples can be seen at http://journals.ieeeauthorcenter.ieee.org/wp-content/uploads/sites/7/IEEE-Math-Typesetting-Guide.pdf

12.1 Display Equations

A simple display equation example shown below uses the "equation" environment. To number the equations, use the `\label` macro to create an identifier for the equation. LaTeX will automatically number the equation for you.

$$x = \sum_{i=0}^{n} 2iQ. \tag{1}$$

is coded as follows:

```
\begin{equation}
\label{deqn_ex1}
x = \sum_{i=0}^{n} 2{i} Q.
\end{equation}
```

To reference this equation in the text use the `\ref` macro. Please see (1)
is coded as follows:

```
Please see (\ref{deqn_ex1})
```

12.2 Equation Numbering

Consecutive Numbering: Equations within an article are numbered consecutively from the beginning of the article to the end, i.e., (1), (2), (3), (4), (5), etc. Do not use roman numerals or section numbers for equation numbering.

Appendix Equations: The continuation of consecutively numbered equations is best in the Appendix, but numbering as (A1), (A2), etc., is permissible.

Hyphens and Periods: Hyphens and periods should not be used in equation numbers, i.e., use (1a) rather than (1-a) and (2a) rather than (2.a) for sub-equations. This should be consistent throughout the article.

12.3 Multi-line equations and alignment

Here we show several examples of multi-line equations and proper alignments.
A single equation that must break over multiple lines due to length with no specific alignment.

$$\begin{multline}\text{The first line of this example}\\ \text{The second line of this example}\\ \text{The third line of this example}\end{multline} \tag{2}$$

is coded as:

```
\begin{multline}
\text{The first line of this example}\\
\text{The second line of this example}\\
\text{The third line of this example}
\end{multline}
```

A single equation with multiple lines aligned at the = signs

$$a = c + d \tag{3}$$
$$b = e + f \tag{4}$$

is coded as:

```
\begin{align}
```

```
a &= c+d \\
b &= e+f
\end{align}
```

The `align` environment can align on multiple points as shown in the following example:

$$x = y \qquad X = Y \qquad a = bc \tag{5}$$
$$x' = y' \qquad X' = Y' \qquad a' = bz \tag{6}$$

is coded as:

```
\begin{align}
x &= y & X & =Y & a &=bc\\
x' &= y' & X' &=Y' &a' &=bz
\end{align}
```

12.4 Subnumbering

The amsmath package provides a `subequations` environment to facilitate subnumbering. An example:

$$f = g \tag{7a}$$
$$f' = g' \tag{7b}$$
$$\mathcal{L}f = \mathcal{L}g \tag{7c}$$

is coded as:

```
\begin{subequations}\label{eq:2}
\begin{align}
f&=g \label{eq:2A}\\
f' &=g' \label{eq:2B}\\
\mathcal{L}f &= \mathcal{L}g \label{eq:2c}
\end{align}
\end{subequations}
```

12.5 Matrices

There are several useful matrix environments that can save you some keystrokes. See the example coding below and the output.

A simple matrix:

$$\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \tag{8}$$

is coded as:

```
\begin{equation}
\begin{matrix}  0 &  1 \\
1 &  0 \end{matrix}
\end{equation}
```

A matrix with parenthesis

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{9}$$

is coded as:

```
\begin{equation}
\begin{pmatrix} 0 & -i \\
 i &  0 \end{pmatrix}
\end{equation}
```

A matrix with square brackets

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{10}$$

is coded as:

```
\begin{equation}
\begin{bmatrix} 0 & -1 \\
1 &  0 \end{bmatrix}
\end{equation}
```

A matrix with curly braces

$$\begin{Bmatrix} 1 & 0 \\ 0 & -1 \end{Bmatrix} \tag{11}$$

is coded as:

```
\begin{equation}
\begin{Bmatrix} 1 &  0 \\
0 & -1 \end{Bmatrix}
\end{equation}
```

A matrix with single verticals

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \tag{12}$$

is coded as:

```
\begin{equation}
\begin{vmatrix} a &  b \\
c &  d \end{vmatrix}
\end{equation}
```

A matrix with double verticals

$$\begin{Vmatrix} i & 0 \\ 0 & -i \end{Vmatrix} \tag{13}$$

is coded as:

```
\begin{equation}
\begin{Vmatrix} i &  0 \\
0 & -i \end{Vmatrix}
\end{equation}
```

12.6 Arrays

The `array` environment allows you some options for matrix-like equations. You will have to manually key the fences, but you'll have options for alignment of the columns and for setting horizontal and vertical rules. The argument to `array` controls alignment and placement of vertical rules.

A simple array

$$\left(\begin{array}{cccc} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 134 \end{array} \right) \tag{14}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{cccc}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

A slight variation on this to better align the numbers in the last column

$$\left(\begin{array}{cccc} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 134 \end{array} \right) \tag{15}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{cccr}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

An array with vertical and horizontal rules

$$\left(\begin{array}{c|c|c|r} a+b+c & uv & x-y & 27 \\ \hline a+b & u+v & z & 134 \end{array} \right) \tag{16}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{c|c|c|r}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

Note the argument now has the pipe "|" included to indicate the placement of the vertical rules.

12.7 Cases Structures

Many times we find cases coded using the wrong environment, i.e., `array`. Using the `cases` environment will save keystrokes (from not having to type the `\left\lbrace`) and automatically provide the correct column alignment.

$$z_m(t) = \begin{cases} 1, & \text{if } \beta_m(t) \\ 0, & \text{otherwise.} \end{cases}$$

is coded as follows:

```
\begin{equation*}
{z_m(t)} =
\begin{cases}
1,&{\text{if}}\ {\beta }_m(t),\\
{0,}&{\text{otherwise.}}
\end{cases}
\end{equation*}
```

Note that the "&" is used to mark the tabular alignment. This is important to get proper column alignment. Do not use `\quad` or other fixed spaces to try and align the columns. Also, note the use of the `\text` macro for text elements such as "if" and "otherwise".

12.8 Function Formatting in Equations

In many cases there is an easy way to properly format most common functions. Use of the \ in front of the function name will in most cases, provide the correct formatting. When this does not work, the following example provides a solution using the `\text` macro.

$$d_R^{KM} = \underset{d_l^{KM}}{\arg\min}\{d_1^{KM}, \ldots, d_6^{KM}\}.$$

is coded as follows:

```
\begin{equation*}
 d_{R}^{KM} = \underset {d_{l}^{KM}}
 {\text{arg min}} \{ d_{1}^{KM},
 \ldots,d_{6}^{KM}\}.
\end{equation*}
```

12.9 Text Acronyms inside equations

This example shows where the acronym "MSE" is coded using `\text{}` to match how it appears in the text.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

```
\begin{equation*}
 \text{MSE} = \frac {1}{n}\sum _{i=1}^{n}
(Y_{i} - \hat {Y_{i}})^{2}
\end{equation*}
```

12.10 Obsolete Coding

Avoid the use of outdated environments, such as `eqnarray` and $$ math delimiters, for display equations. The $$ display math delimiters are left over from PlainTeX and should not be used in LaTeX, ever. Poor vertical spacing will result.

12.11 Use Appropriate Delimiters for Display Equations

Some improper mathematical coding advice has been given in various YouTube™ videos on how to write scholarly articles, so please follow these good examples:

For **single-line unnumbered display equations**, please use the following delimiters:

```
\[ . . . \] or
\begin{equation*} . . . \end{equation*}
```

Note that the * in the environment name turns off equation numbering.

For **multiline unnumbered display equations** that have alignment requirements, please use the following delimiters:

```
\begin{align*} . . . \end{align*}
```

For **single-line numbered display equations**, please use the following delimiters:

```
\begin{equation} . . . \end{equation}
```

For **multiline numbered display equations**, please use the following delimiters:

```
\begin{align} . . . \end{align}
```

13 LaTeX Package Suggestions

Immediately after your documenttype declaration at the top of your LaTeX file is the place where you should declare any packages that are being used. The following packages were used in the production of this document.

```
\usepackage{amsmath,amsfonts}
\usepackage{algorithmic}
\usepackage{array}
\usepackage[caption=false,font=normalsize,
   labelfont=sf,textfont=sf]{subfig}
\u00sepackage{textcomp}
\usepackage{stfloats}
\usepackage{url}
\usepackage{verbatim}
\usepackage{graphicx}
\usepackage{balance}
```

14 Additional Advice

Please use "soft" (e.g., `\eqref{Eq}`) or `(\ref{Eq})` cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Please do not use `\nonumber` or `\notag` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

15 A Final Checklist

1) Make sure that your equations are numbered sequentially and there are no equation numbers missing or duplicated. Avoid hyphens and periods in your equation numbering. Stay with IEEE style, i.e., (1), (2), (3) or for sub-equations (1a), (1b). For equations in the appendix (A1), (A2), etc..
2) Are your equations properly formatted? Text, functions, alignment points in cases and arrays, etc.
3) Make sure all graphics are included.
4) Make sure your references are included either in your main LaTeX file or a separate .bib file if calling the external file.

References

[1] *Mathematics into Type*, American Mathematical Society. Online available:
[2] T.W. Chaundy, P.R. Barrett and C. Batey, *The Printing of Mathematics*, Oxford University Press. London, 1954.
[3] *The LaTeXCompanion*, by F. Mittelbach and M. Goossens
[4] *More Math into LaTeX*, by G. Grätzer
[5] *AMS-StyleGuide-online.pdf,* published by the American Mathematical Society
[6] H. Sira-Ramirez. "On the sliding mode control of nonlinear systems," *Systems & Control Letters*, vol. 19, pp. 303–312, 1992.
[7] A. Levant. "Exact differentiation of signals with unbounded higher derivatives," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, California, USA, pp. 5585–5590, 2006.
[8] M. Fliess, C. Join, and H. Sira-Ramirez. "Non-linear estimation is easy," *International Journal of Modelling, Identification and Control*, vol. 4, no. 1, pp. 12–27, 2008.
[9] R. Ortega, A. Astolfi, G. Bastin, and H. Rodriguez. "Stabilization of food-chain systems using a port-controlled Hamiltonian description," in *Proceedings of the American Control Conference*, Chicago, Illinois, USA, pp. 2245–2249, 2000.

Jane Doe Biography text here without a photo.

IEEE Publications Technology Team In this paragraph you can place your educational, professional background and research and other interests.