mipi DEVCON
MIPI ALLIANCE DEVELOPERS CONFERENCE

28-29 SEPTEMBER 2021

Alexander Weiss & Thomas Preußer
Accemic Technologies

MIPI HTI(SM), PTI(SM) and STP(SM) – The Bases for Next-Generation Analyses of Multicore Processors

mipi® DEVCON
MIPI ALLIANCE DEVELOPERS CONFERENCE

28-29 SEPTEMBER 2021

Motivation

Motivation: Complex Systems Challenges

Burkacky 2018 (McKinsey Center for Future Mobility®):
"Snowballing complexity is causing significant software-related quality issues, as evidenced by millions of recent vehicle recalls."

Where software is, there are errors.

The defect potential increases with system complexity.
The defect removal efficiency decreases with increasing system complexity.

Growing and self-reinforcing problem

Key capability: **System observation**

Embedded System Observation

Challenges

- Real-time Control
- Concurrency
- Elusive *Heisenbugs*
- Rare *Mandelbugs*

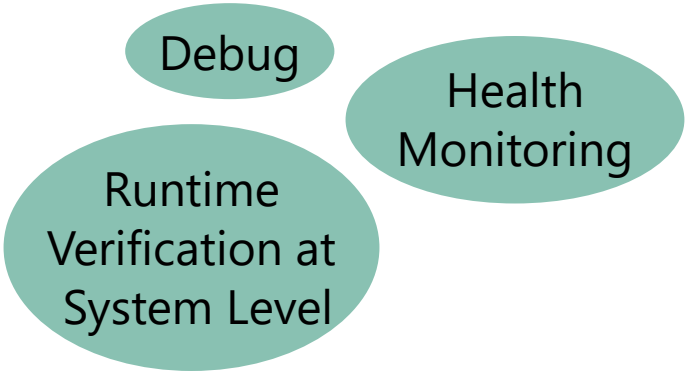
Key 1: Non-Intrusiveness

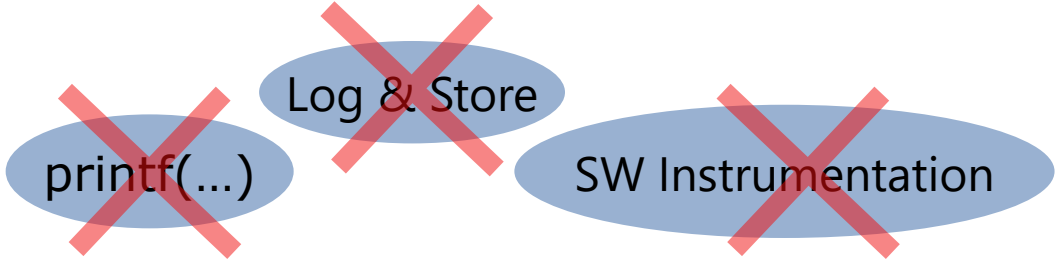
- Maintain application timing
- Avoid phantom synchronization

~~Probe Effect~~

Key 2: Continuity

- No systematic observation limits
- Online trace data processing





mipi® DEVCON

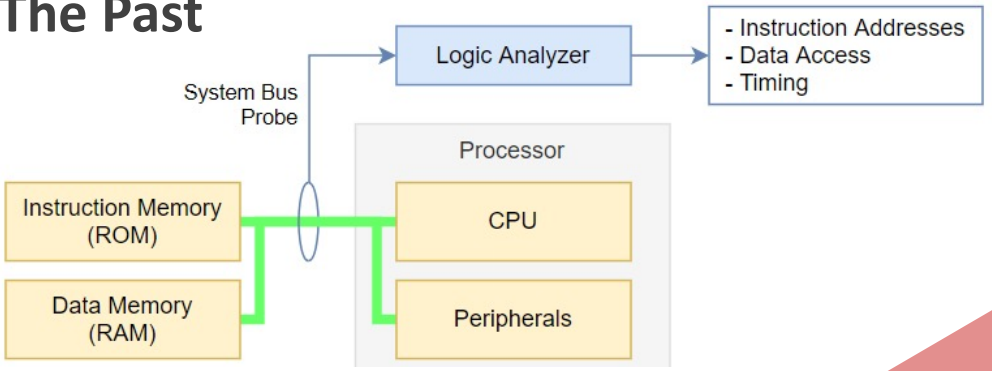
MIPI ALLIANCE DEVELOPERS CONFERENCE

28-29 SEPTEMBER 2021

State of Affairs

The Crux of System Integration

The Past



Need

- On-Chip Probes
- Data Filters
- High-Speed Interfaces

The Present

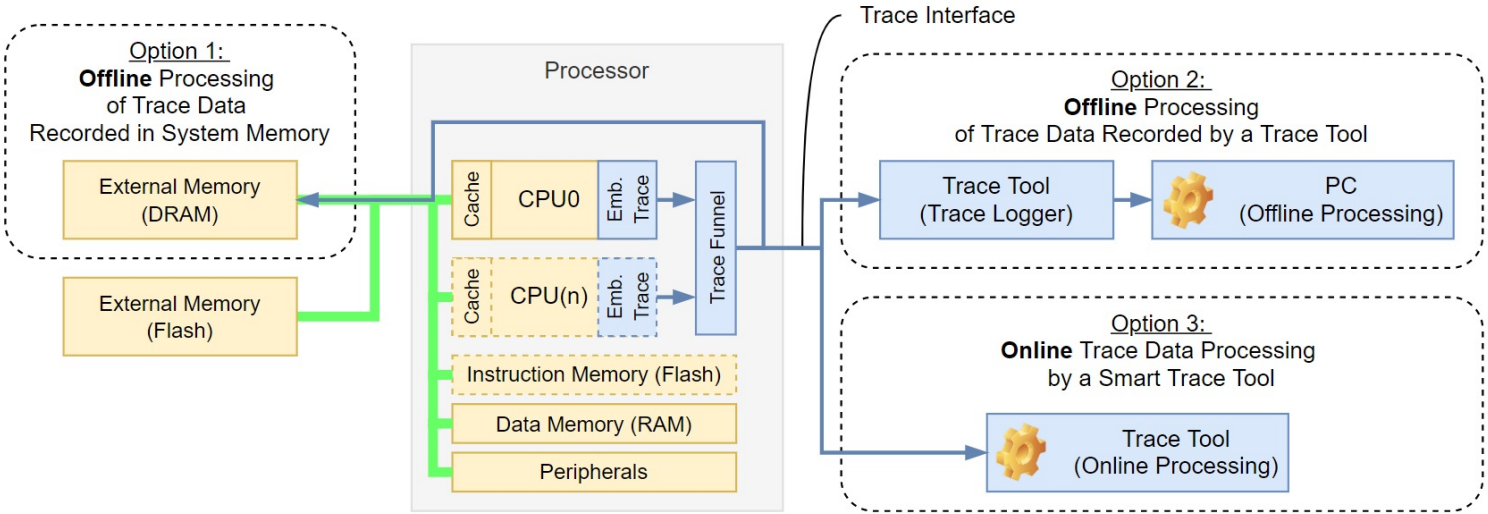
Data Volume

- Multiple players
- Working at GHz speeds.

Clueless System Bus

- Caches hide memory access.
- On-chip memory swallows all access info.
- Multicore systems prevent effect attribution.

Embedded Trace Options



Option 1 - Intrusive

- Compete with application for memory bandwidth and space.

Option 2 - Bounded

- Size of trace buffer limits time.

Option 3 - Hard

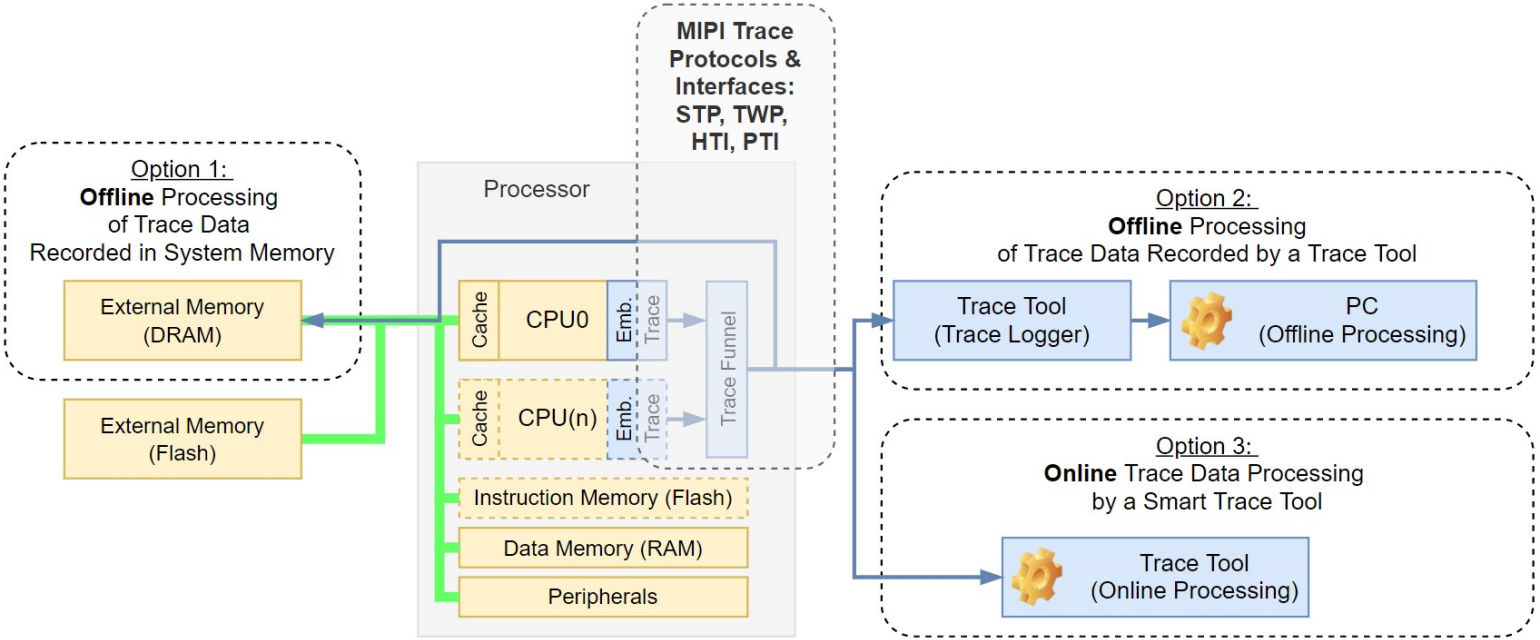
- Processing must match line rate.

mipi® DEVCON
MIPI ALLIANCE DEVELOPERS CONFERENCE

28-29 SEPTEMBER 2021

MIPI Protocols in Action

Carrying Execution Trace Data Off-Chip



STP	MIPI System Trace Protocol	TWP(SM)	MIPI Trace Wrapper Protocol
HTI	MIPI High-Speed Trace Interface	PTI	MIPI Parallel Trace Interface

MIPI PHY + Link Layer Technologies

PTI

- Parallel little-endian TRC_DATA (up to 32)
- Synchronous to shared TRC_CLK (up to ~300 MHz)
- Mechanisms for:
  - Signal skew calibration,
  - Physical, by-pin slicing for parallel point-to-point streams, and
  - Data striping for mismatched protocol/interface bit widths.

HTI

- Trace transmission over (multiple) high-speed serial lanes.
- **Aurora** 8B/10B simplex with up to 6 lanes (HTI) or 8 lanes (HTIv1).
- Up to 12.5 Gbps per lane.
- UFC Messages for sideband info: trigger, overflow, link health, aux.
- Session management (incl. disabled, init, idle, run, test).

MIPI Transport Layer Technologies

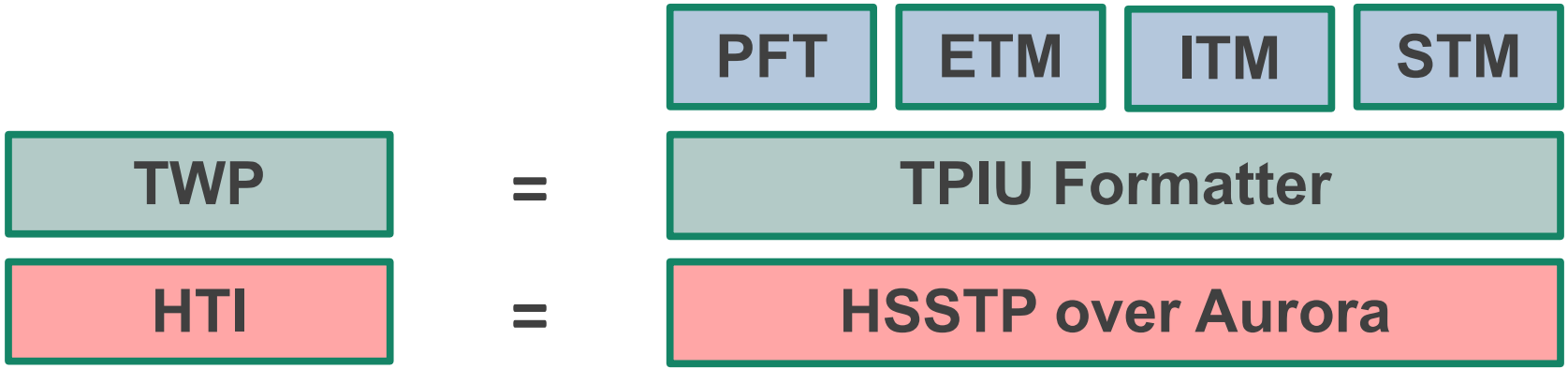
STP – System Trace Protocol

- Multiplexing among up to 65536 sources at the granularity of 4-bit nibbles.
- Autarchic timestamping and syncing.
- Sideband packets for user payload and flagging.
- Stream synchronization facilities.
- Data integrity protection.

TWP – Trace Wrapper Protocol

- Multiplexing among up to 111 sources at byte granularity.
- Designed for very low overhead.

Peering with the ARM CoreSight Stack



Recap: Key Embedded Trace Properties

Key 1: Non-Intrusiveness

- Maintain application timing
- Avoid phantom synchronization

- MIPI Trace protocols and interfaces ☑
- Standard interfaces for (non-intrusive) trace output (USB, Ethernet, PCIe)

Key 2: Continuity

- No systematic observation limits
- Online trace data processing

mipi DEVCON
MIPI ALLIANCE DEVELOPERS CONFERENCE

28-29 SEPTEMBER 2021

Enabling Online Trace Data Processing

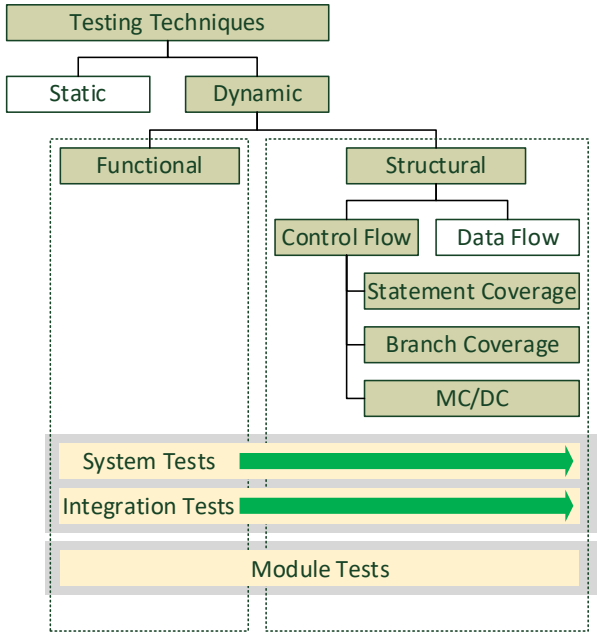
Benefits & Use Cases

Benefits

- Evade all observation time limits.
- Provide instantaneous reaction opportunities.
- Avoid intrusive SW instrumentation.

Use Cases

- Continuous runtime verification:
  - Formal constraints-based anomaly detection.
  - Complex triggers on rare Mandelbug occurrences.
- Structural coverage measurements of high-level integration & system tests.



A Taste of the Challenge

```
// Count all encounters of 42
static unsigned  matches = 0;
static void count42(int const  a) {
  if(a == 42)  matches++;
}

int main() {
  ...
  for(int  i = 0; i < n; i++) {
    int  arg = f(i, ...);
    count42(arg);
  }
  printf("Matches: %0d\n", matches);
}
```

```
0x400100        cmp   $42, %edi
0x400103        je    L0
0x400105        ret
0x400106  L0:   addl  $1, matches
0x40010D        ret
```

```
0x400200        ...
0x400210  L1:   ...
0x400220        mov   arg, %edi
0x400223        callq 0x400100
0x400228        subl  $1, i
0x40022C        jnz   L1
```

Resolved Return		Compressed Return
PSB		PSB
FUP 0x400220	– Synchronization –	FUP 0x400220
⋮		⋮
PSBEND		PSBEND
TNT N	– Conditional branch @0x400103 not taken –	TNT NTT
TIP 0x__0228	– Return –	
TNT T	– Conditional branch @0x40022C taken (loop) –	
⋮		⋮

- *Occasional* synchronization
- Only branches and diversions:
  - Taken / not taken indicators
  - Delta-encoded indirect targets

Setup for Measuring the Structural Coverage

DuT
(Intel, ARM, PPC, Aurix)

Periphery trace
Per
Mem
Mem

CPU0 Instr / Data Trace
CPU1 Instr / Data Trace
CPU2 Instr / Data Trace
CPUn Instr / Data Trace

Trace Buffer and Concentrator

Trace Port

CEDARtools® Hardware Platform
(FPGA Board)

high-speed serial

Debug Server

Trace Data Pre-processing and Control Flow Reconstruction

Branches

Branch Counter

Branch counts

Post processing

Object code coverage

Binary

Configuration

Source Code

Compiler

Debug information

Mapping

Source code coverage

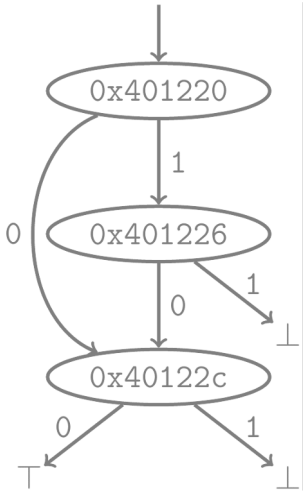
Structural Coverage Reporting

```
7  |    0040090f:  mov   qword ptr [rbp - 8], rax
7  |    00400913:  mov   rax, qword ptr [rbp - 8]
7  |    00400917:  mov   qword ptr [rax + 8], 0
7  |    0040091f:  mov   rax, qword ptr [rbp - 8]
7  |    00400923:  mov   rdx, qword ptr [rax + 8]
7  |    00400927:  mov   rax, qword ptr [rbp - 8]
7  |    0040092b:  mov   qword ptr [rax + 0x10], rdx
7  |    0040092f:  mov   rax, qword ptr [rbp - 8]
7  |    00400933:  mov   edx, dword ptr [rbp - 0x1c]
7  |    00400936:  mov   dword ptr [rax], edx
7  |    00400938:  mov   rax, qword ptr [rbp - 0x18]
7  |    0040093c:  mov   rdx, qword ptr [rbp - 8]
7  |    00400940:  mov   qword ptr [rax], rdx
   |_ 7  00400943:  jmp   0x400993
10 |    00400945:  mov   rax, qword ptr [rbp - 0x18]
10 |    00400949:  mov   rax, qword ptr [rax]
10 |    0040094c:  mov   eax, dword ptr [rax]
10 |    0040094e:  cmp   dword ptr [rbp - 0x1c], eax
   |_ 5  00400951:  jge   0x40096d
5  |    00400953:  mov   rax, qword ptr [rbp - 0x18]
5  |    00400957:  mov   rax, qword ptr [rax]
5  |    0040095a:  lea   rdx, [rax + 0x10]
5  |    0040095e:  mov   eax, dword ptr [rbp - 0x1c]
5  |    00400961:  mov   esi, eax
5  |    00400963:  mov   rdi, rdx
   |_ 5  00400966:  call  0x4008e2
   |_ 5  0040096b:  jmp   0x400993
5  |    0040096d:  mov   rax, qword ptr [rbp - 0x18]
5  |    00400971:  mov   rax, qword ptr [rax]
5  |    00400974:  mov   eax, dword ptr [rax]
5  |    00400976:  cmp   dword ptr [rbp - 0x1c], eax
```

Line	Branch	Instr.	Count	Source
1				`#ifdef CEDAR`
2				`#include "cedar_ipt.h"`
3				`#endif`
4				
5				`#include <stdlib.h>`
6				`#include <stdio.h>`
7				
8				`struct bin_tree {`
9				`  int data;`
10				`  struct bin_tree * right, *left;`
11				`};`
12				`typedef struct bin_tree node;`
13				
14				`void insert(node ** tree, int val)`
15		100%	17x	`{`
16		100%	17x	`  node *temp = NULL;`
17	2/2	100%	17x	`  if (!(*tree))`
18				`  {`
19		100%	7x	`    temp = (node *)malloc(sizeof(node));`
20		100%	7x	`    temp->left = temp->right = NULL;`
21		100%	7x	`    temp->data = val;`
22		100%	7x	`    *tree = temp;`
23		100%	7x	`    return;`
24				`  }`
25				
26	2/2	100%	10x	`  if (val < (*tree)->data)`
27				`  {`
28		100%	5x	`    insert(&(*tree)->left, val);`
29				`  }`

Higher-Level Coverage Metrics for Safety Certification

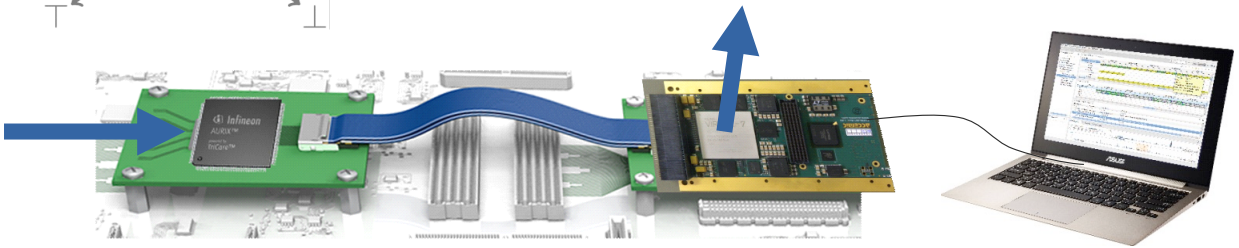
```
...
if((a&&b) || c) {
  ...
}
...
```



- What branches constitute a decision?
- What **paths** were part of an execution?
- Does the path set satisfy MC/DC coverage?

a b *	1	√
a b c	0	√
a * c	1	√
a * c	0	√

```
40121C: cmpl   $0x0,-0x24(%rbp)
401220: je     401228 <main+0x106>
401222: cmpl   $0x0,-0x28(%rbp)
401226: jne    40122e <main+0x10c>
401228: cmpl   $0x0,-0x2c(%rbp)
40122C: je     401234 <main+0x112>
```



mipi®
DEVCON
MIPI ALLIANCE DEVELOPERS CONFERENCE

28-29
SEPTEMBER
2021

Conclusion

Importance of MIPI Standards

- Capable vehicles for architecture-independent trace transport.
- Enable **reuse** of lower-level trace layers.
- Allow vendors to **focus** efforts on added high-level functionality.
- Accelerate innovation.
- Reduce time to market.

Challenges for Online Trace Data Processing

- Standardisation of COTS-based trace interfaces
- Increase available trace bandwidth
- Unbounded timing slack between multiplexed streams (TWP/STP)
- Defined buffering limits on carried protocols would be helpful

ADDITIONAL RESOURCES

- Ondrej Burkacky et al.: Rethinking Car Software and Electronics Architecture, McKinsey&Co., 2018. https://lmy.de/2uzLC
- C. Jones, Software Engineering Best Practices, 1st ed. USA: McGraw-Hill, Inc., 2009. https://lmy.de/rGvnE
- C. Jones and O. Bonsignour, The Economics of Software Quality. Addison-Wesley, 2011. https://lmy.de/5gXVT
- Alexander Weiss et al.: Understanding and Fixing Complex Faults in Embedded Cyberphysical Systems, IEEE Computer, Vol. 54, Issue 1, pp. 49-60, 01/2021. https://lmy.de/fT8rk
- Thomas Preußer et al.: Everything You Always Wanted to Know About Embedded Trace, accepted by IEEE Computer, preprint available at https://lmy.de/l4Hs5
- The CEDARtools online monitoring system: https://lmy.de/9vAuL

mipi®
DEVCON
MIPI ALLIANCE DEVELOPERS CONFERENCE

28-29
SEPTEMBER
2021

THANK YOU!