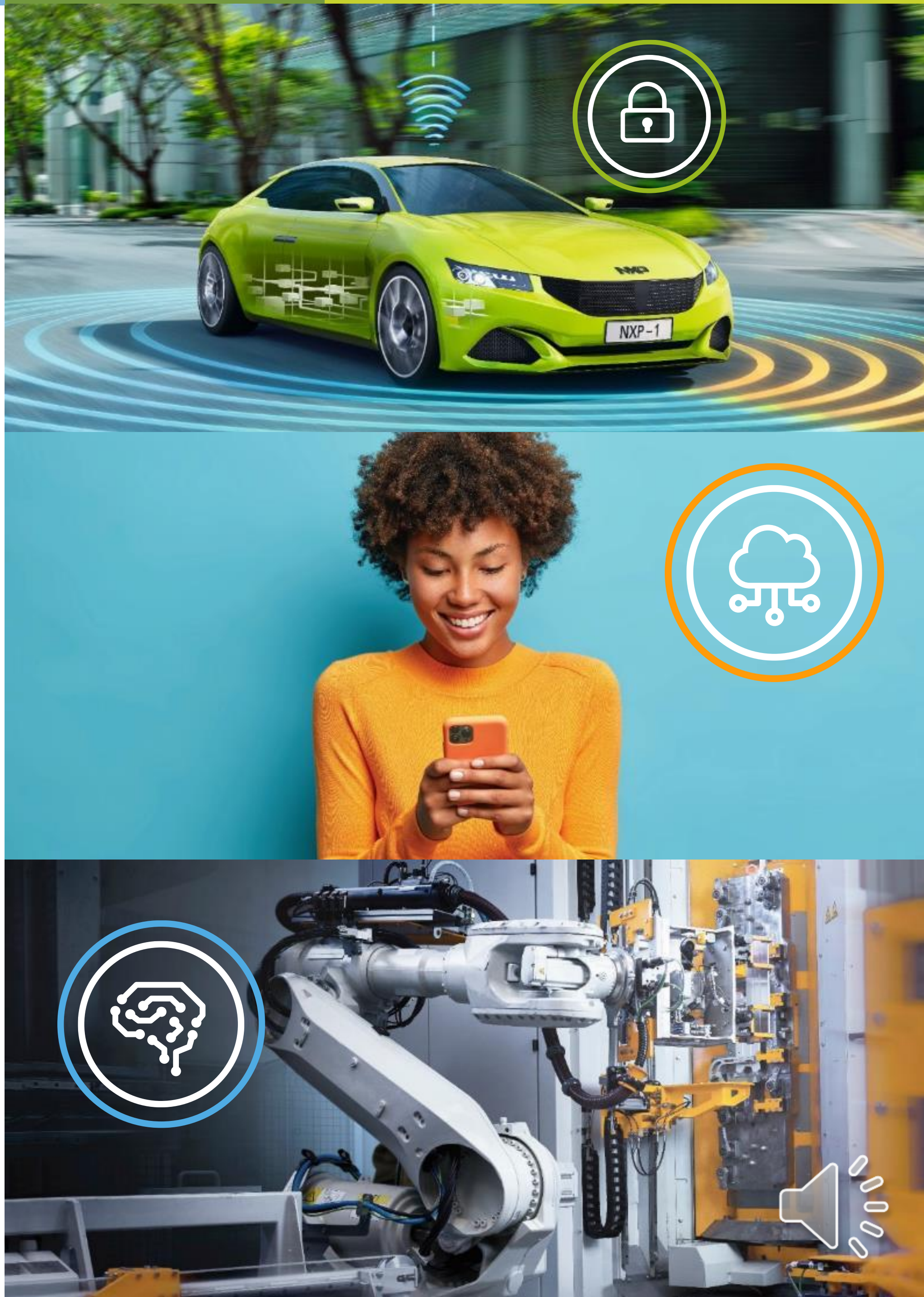
INTRODUCTION TO MIPI I3C

Sally Huang
FAE

NOVEMBER 2022



SECURE CONNECTIONS FOR A SMARTER WORLD

PUBLIC

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V.
ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2022 NXP B.V.

AGENDA

- MIPI I3C Introduction
- I3C vs I^{2}C - Benefits and Use Cases
- MIPI I3C Signaling and Protocol
- Device Identifier – Provisional-ID
- Common Command Codes (CCC)
- Dynamic Address Assignment (DAA) Procedure
- I3C Specification Updates

MIPI I3C = NEXT GENERATION FROM I^{2}C

- MIPI I3C is a follow on to I^{2}C
  - Has major improvements in use and power and performance
  - Optional alternative to SPI for mid-speed (equivalent to ~30MHz)
- Background
  - NXP (Philips legacy) is I2C leader and spec owner
  - I^{2}C is used predominantly as control and communication interface with a focus in sensors (>90% according to 2013 MIPI Alliance survey)
  - MIPI Alliance Sensor Interface Workgroup initiated an upgrade of requirements in 2013
- Rationale for upgrade
  - In-band interrupt to reduce # of GPIO wires on SoC, as # of sensors increase on the mobile devices
  - I^{2}C speed has become limiting, as amount of data increases on the bus
  - Upgrade/feature Constraints – maintain simplicity, multi-drop (unlike SPI), and concerns about backwards compatibility
    - Due to adoption curve, backwards compatibility helps enable a smooth transition from I^{2}C to MIPI I3C
- MIPI I3C Spec Contributors
  - Primary Spec authoring: NXP (Paul Kimelman), Qualcomm, Intel, other contributors: Invensense, TI, STM, Synopsys, Cadence, Mentor, Sony, Knowles, Lattice

ADVANCED ANALOG

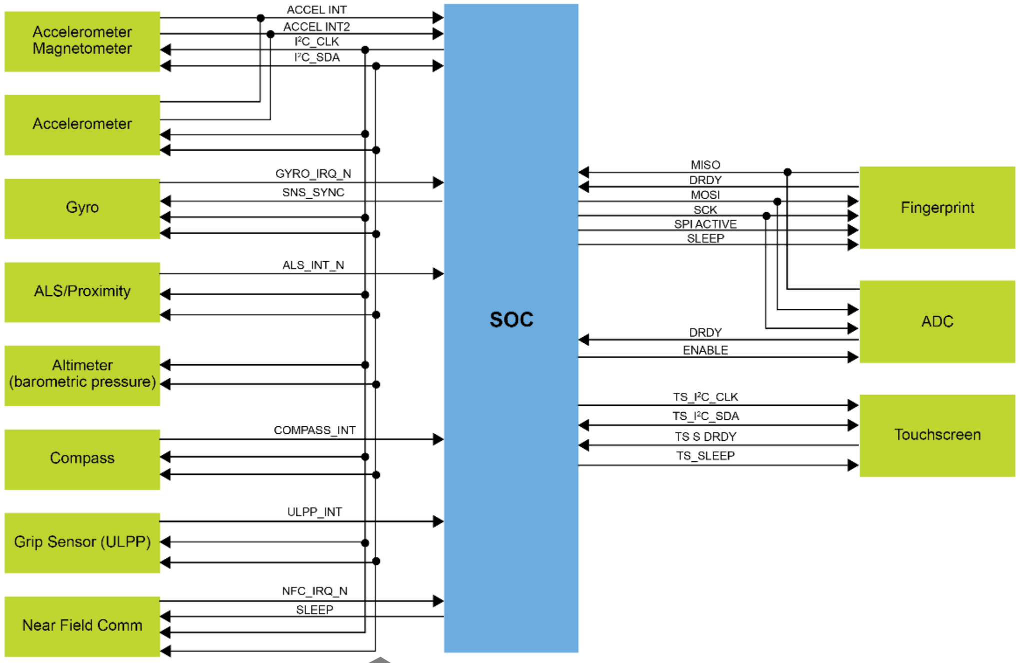
WE BRING DIGITAL TO LIFE

I3C vs I^{2}C - Benefits and Use Cases

SENSOR INTERFACE BLOCK DIAGRAM

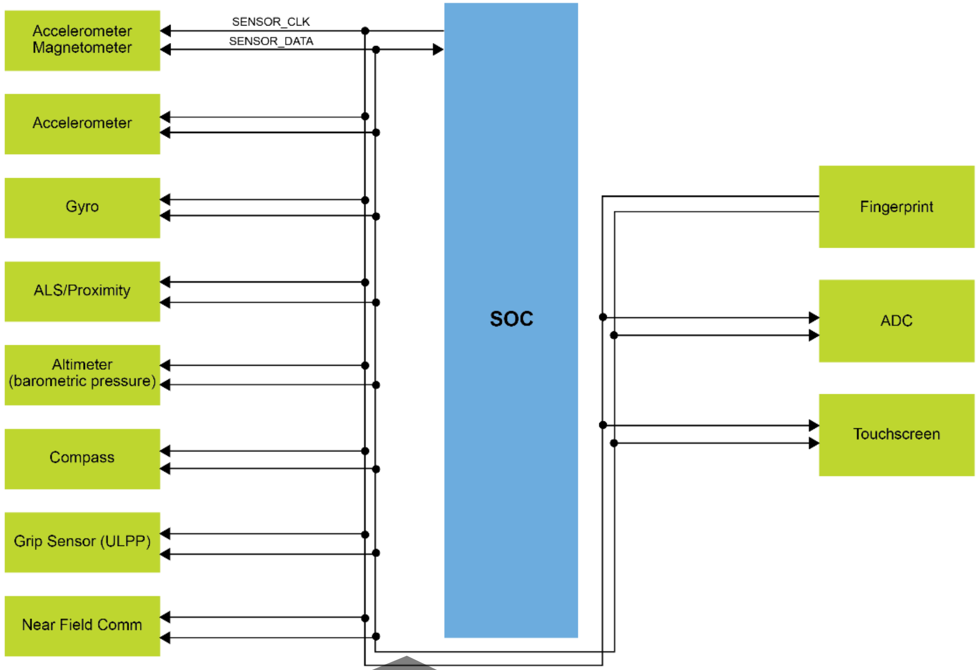
- Adding more Slaves adds high cost: In addition to data rate limits, side-band pins/traces such as dedicated interrupts, enable/reset, and sleep signals usually are needed
- Increased number of GPIOs is adding **system cost** in the form of added SoC package pins and PCB layer count

Current Scenario



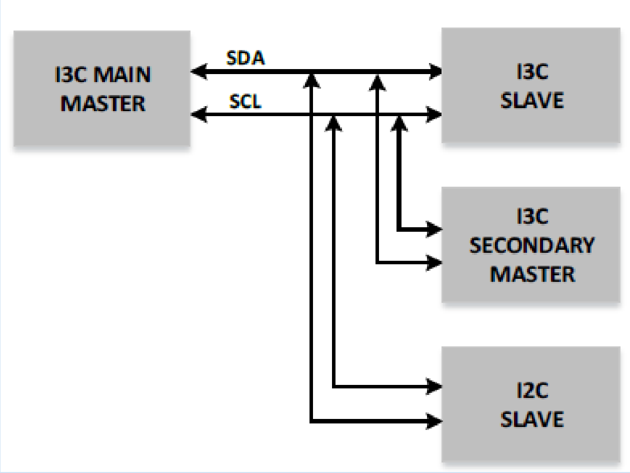
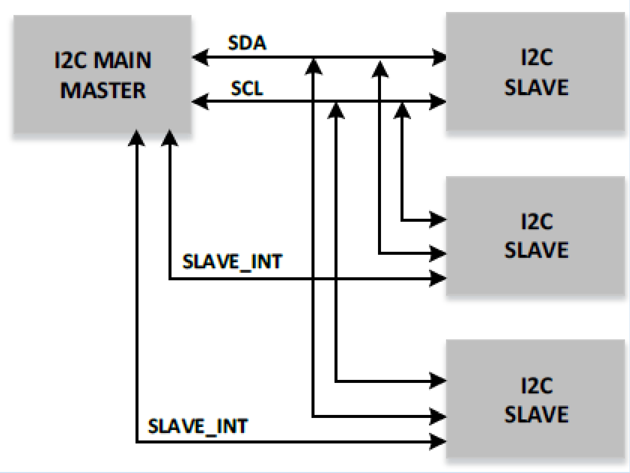
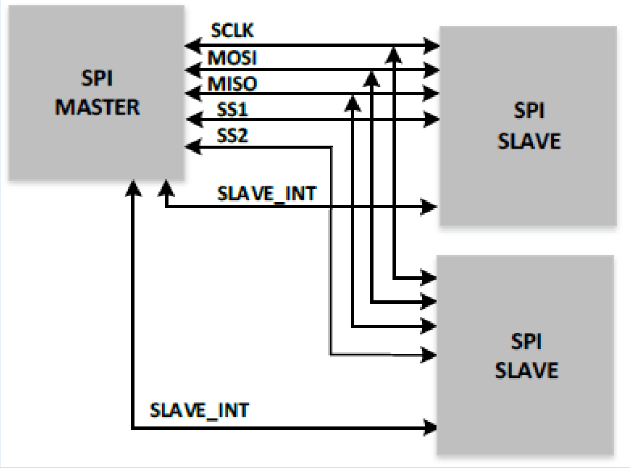
I^{2}C and SPI devices with side band channels EN or RSTn, INT, etc

Desired Scenario



MIPI I3C with in-band interrupt, Common Command Codes for device control

SENSOR INTERFACE BLOCK DIAGRAM FOR MIPI I3C VS. I²C & SPI

Parameter	MIPI I3C	I²C	SPI
Overview	 I3C MAIN MASTER SDA SCL I3C SLAVE I3C SECONDARY MASTER I2C SLAVE	 I2C MAIN MASTER SDA SCL I2C SLAVE I3C SLAVE SLAVE_INT SLAVE_INT I2C SLAVE	 SPI MASTER SCLK MOSI MISO SS1 SS2 SLAVE_INT SPI SLAVE SPI SLAVE SLAVE_INT
# of lines	2-wire	2-wire (plus separate wires for each required interrupt signal)	4-wire (plus separate wires for each required interrupt signal and additional selects). Not multi-master
Effective Real-Data Bitrate	30 Mbps max at 12.5 MHz (Typical:10.6 Mbps at 12 MHz SDR)	3 Mbps max at 3.4 MHz (Hs) 0.8 Mbps max at 1 MHz (Fm+) 0.35 Mbps max at 400 KHz (Fm)	Approx. 60 Mbps max at 60 MHz for conventional implementations (Typical: 10 Mbps at 10 MHz)

From MIPI I3C White paper: http://resources.mipi.org/MIPI I3C-sensor-whitepaper-from-mipi-alliance

MIPI I3C VERSUS I²C AT-A-GLANCE

	I²C	MIPI I3C
Clock Speed & Data Rate	Fast mode: 400kb/s Fast Mode+: 1Mb/s High speed: 3.4Mb/s (after special header) Actual Data: computed 8/9th – 1 byte	SDR: up to 11Mbps Actual Data rate (8/9th – per 1 byte) HDR-DDR: Actual Data Rate 20Mbps HDR-TSP: Actual Data Rate to ~30Mbps

# wires	2 – multi-drop (OpenDrain IF) 	
	SCL: clock – from Master(s), Slaves stretch
	SDA: data – bidirectional (OpenDrain)
		2 – multi-drop (SCL is push-pull, SDA OpenDrain and push-pull)
		SCL = clock (except for HDR-TSP) - from *current* Master only
		SDA = data – bidirectional (OpenDrain and push-pull)
		Note: Support for optional 2 and 4 data lines (ML) coming in v1.1

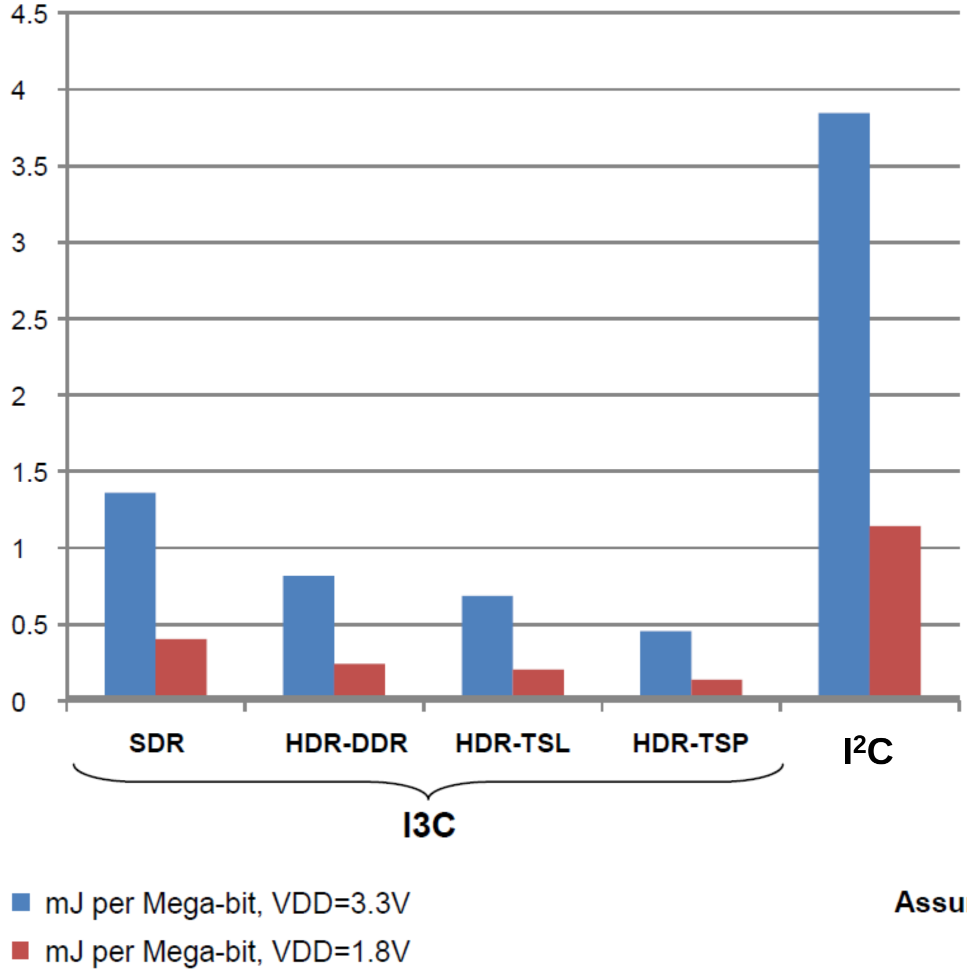
Power	High due to open-drain SCL , SDA with strong pullups	Lower due to SCL being push-pull only and SDA working in push-pull most of the time
Slave Read termination	Master has to end Read (so has to know length in advance)	Slave ends Read, but Master may terminate early
In-Band Interrupts	None – use a separate wire/pin per slave	Integrated, prioritized, and may include a byte (or more) of context
Hot-Plug	None – proprietary systems only	Built-in. Same mechanism as in-band-interrupt

MIPI I3C VERSUS I^{2}C AT-A-GLANCE

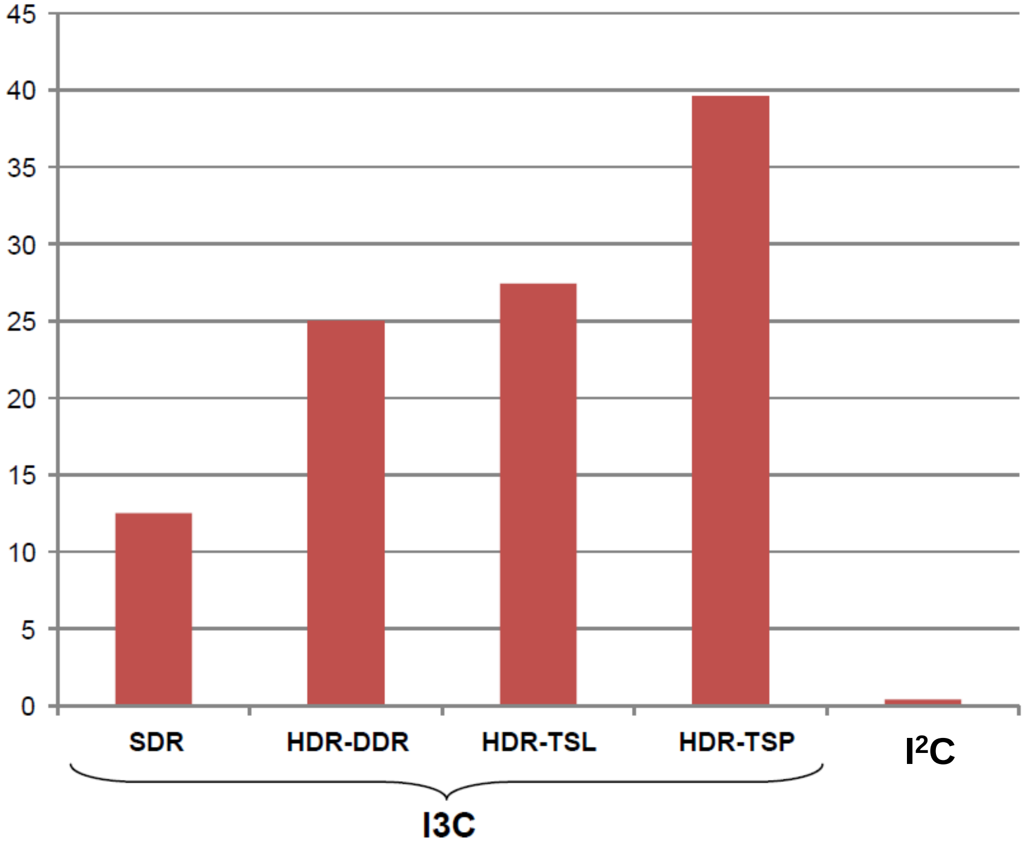
	I^{2}C	MIPI I3C
Error detection	No protocol inherent error detection	Master and slave side error detection standardized/mandated
Time stamping	Has to be done by master once separate INT signal is triggered	Is an essential part of the MIPI I3C spec – no dedicated INT signal required. Coming to I3C Basic v1.1
Built-in Commands	None. Proprietary messages only	Built-in for control, discovery, bus management, etc. Expandable: e.g. Time Control, IO Expander. Out of band of normal messages (so does not interfere), Vendor/std specific
Master / Slave	Master-Slave, Multi-master optional	Master-Slave; Master handoff (by request or M initiated)
IO pads	I^{2}C special pads (e.g. 50ns spike filter)	Standard pads 4 mA drive, no spike filter
Slave address	Static	Dynamically assigned during initialization. Slaves may have i^{2}c static address at start (work on i^{2}c buses)
Clocking	Slaves use inbound clock or oversampling	Slaves use inbound clock (allows slow/no internal clock)
Complexity	Low for Slaves. Higher for masters, especially around multi-master	Full Slaves as small as 2 K gates Masters as small as 2.5 K gates State machine and processor supporting implementations

ADVANTAGES IN ENERGY AND DATA RATE

Energy Consumption
milliJoules per Megabit for I3C Data Modes (100pF)
vs I²C (100pF, 3.54KOhm)



Raw Bitrate
Mbps for I3C Data Modes (@12.5MHz)
vs I²C (@400KHz)



Assumptions: 1) All symbols in each mode have equal probability for use.

2) Energy consumption is the energy delivered by pull-up devices to the bus (which includes drivers and resistors).

ADVANCED ANALOG

WE BRING DIGITAL TO LIFE

MIPI I3C Signaling and Protocol

WHAT DOES AN MIPI I3C MESSAGE LOOK LIKE?

MIPI I3C SDR looks almost the same as I^{2}C:

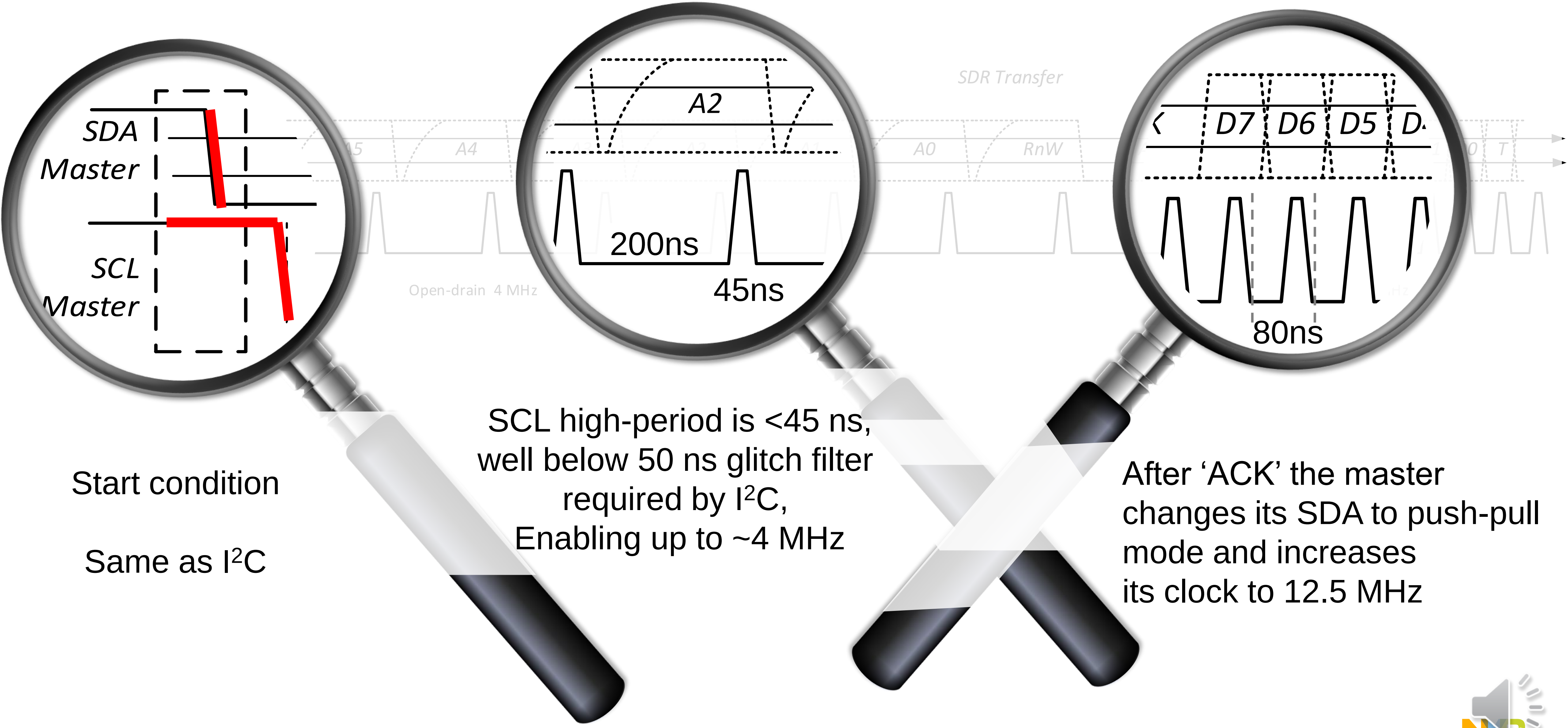
- E.g. Write data

	1-bit	8-bits	1-bit	8-bits	1-bit	…	1-bit
MIPI I3C	S or Sr	Addr+W	ACK/ NACK	1 Byte data	T bit = Parity	More data	Sr or P
I2C	same	same	same	same	ACK/ NACK	same	same

- E.g. Read data (typical approach):

	1-bit	8-bits	1-bit	8-bits	1-bit	1-bit	8-bit	1-bit	8-bit	1-bit	…	1-bit
MIPI I3C	S or Sr	Addr+ W	ACK / NACK	1 Byte data	T bit = Parity	Sr	Addr + R	ACK / NACK	1 Byte from Slave	T bit = ‘1 then Z’ to continue T bit = ‘0’ Slave ends transmission	More data	Sr or P
I2C	same	same	same	same	ACK/ NACK	same	same	same	same	ACK/ NACK **Slave can’t abort read**	same	Master ends read

MIPI I3C BUS SIGNAL IN SDR MODE AFTER DYNAMIC ADDRESS ASSIGNMENT



Start condition

Same as I^{2}C

SCL high-period is <45 ns, well below 50 ns glitch filter required by I^{2}C, Enabling up to ~4 MHz

After ‘ACK’ the master changes its SDA to push-pull mode and increases its clock to 12.5 MHz

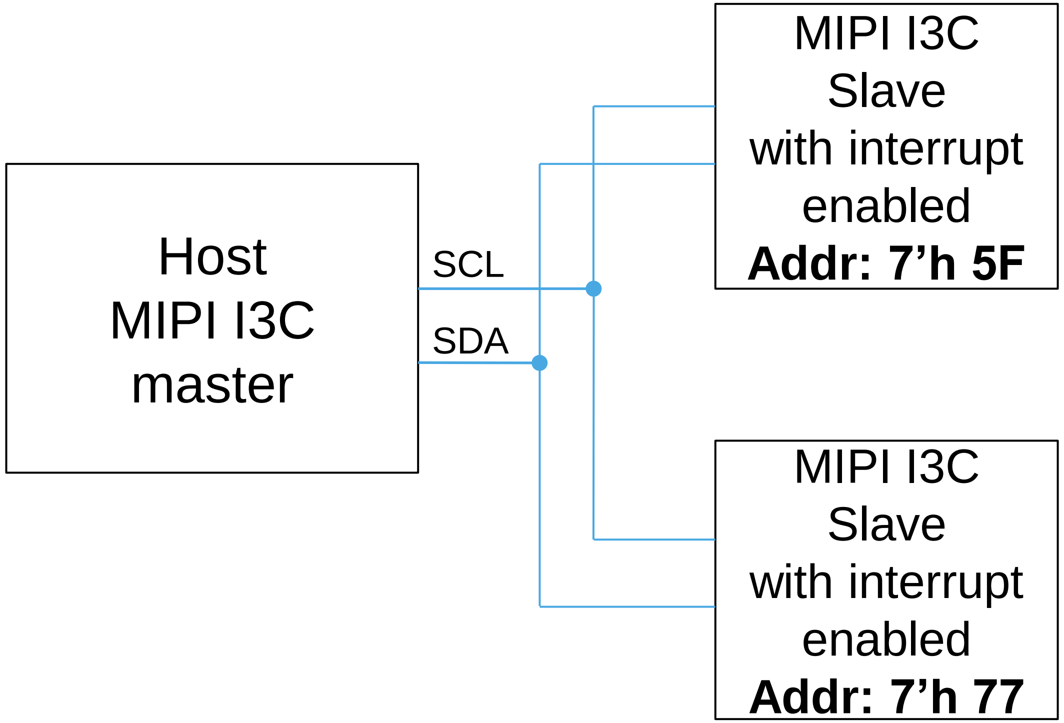
WHY IS ADDRESS ARBITRATION IMPORTANT IN MIPI I3C

Address arbitration (OD) is used for multiple functions in the MIPI I3C specification:

- In-Band Interrupt (IBI)
  - Slaves can trigger the master by pulling SDA low during a quiet period and the master will start its SCL (start condition)
  - Slaves drive their own address during address header; 0s win, so lowest number slave wins the arbitration and Master can ACK/NACK the IBI
- Hot-Join
  - IBI with special address=7’h02 (lowest possible)
- Bus initialization to assign Dynamic Addresses
  - IDs of Slaves arbitrated in same way so all get a Dynamic Address assigned
- Multi-master request
  - Secondary Master can request to become Master using similar method as IBI

ADDRESS ARBITRATION

- System example:
  - MIPI I3C-only system
  - 2 slaves with In-Band Interrupt enabled
  - BOTH slaves trigger an interrupt at the same time



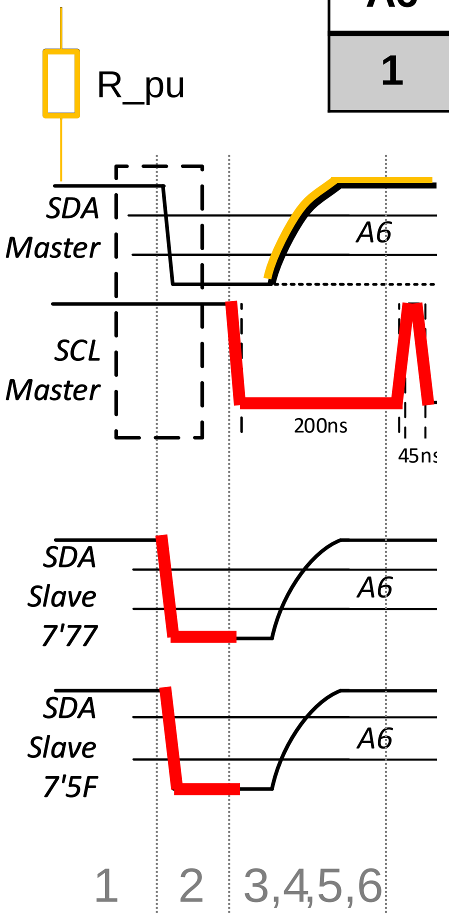
A6	A5	A4	A3	A2	A1	A0
1	0	1	1	1	1	1

A6	A5	A4	A3	A2	A1	A0
1	1	1	0	1	1	1

ADDRESS ARBITRATION

EXAMPLE: INTERRUPT TRIGGERED BY SLAVE IN A SYSTEM WITH 2 IBI CAPABLE SLAVES

A6	A5	A4	A3	A2	A1	A0
1	-	-	-	-	-	-



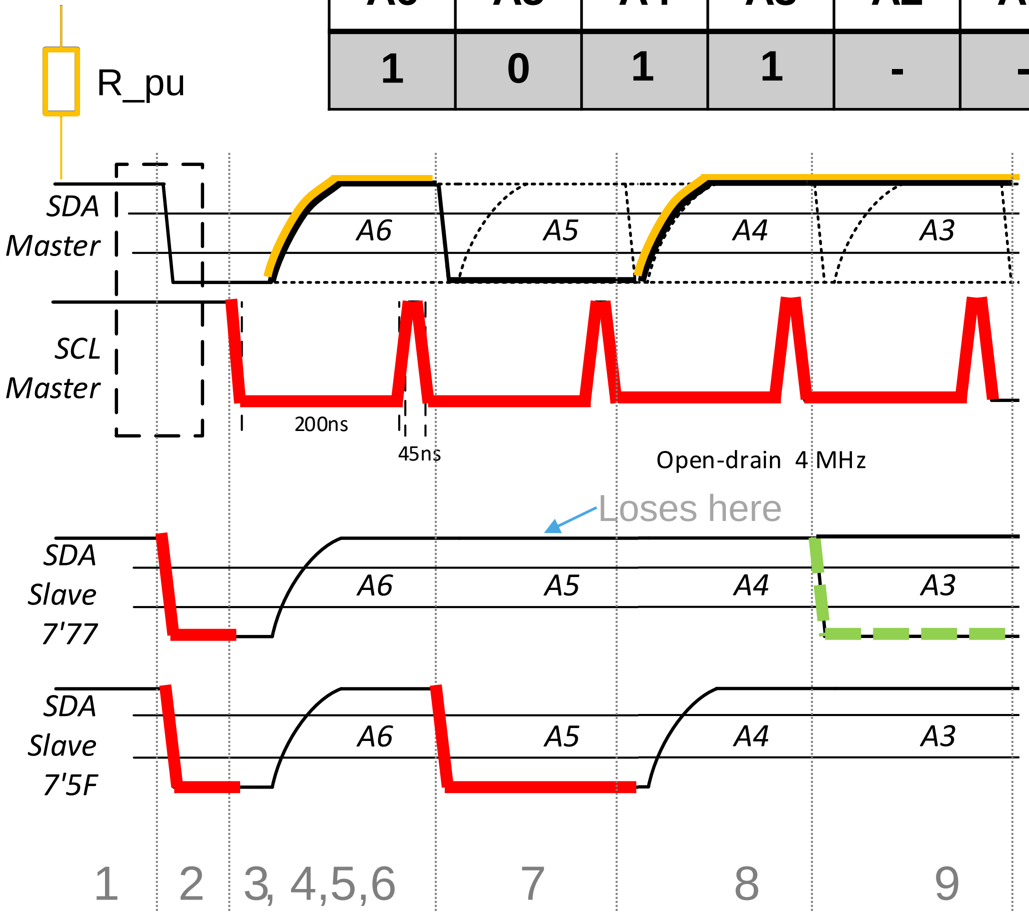
1. Master is idle with SCL stopped and SDA being pulled high by resistor
2. BOTH Slaves trigger an interrupt by pulling SDA low
3. Master starts SCL, pulling it low
4. Slave releases SDA
5. SDA is pulled high by R_pu
6. SCL pulse to latch address bit A6

| Actively driving device | signal on bus

ADDRESS ARBITRATION

EXAMPLE: INTERRUPT TRIGGERED BY SLAVE IN A SYSTEM WITH 2 IBI CAPABLE SLAVES

A6	A5	A4	A3	A2	A1	A0
1	0	1	1	-	-	-



🟥 Actively driving

signal on bus

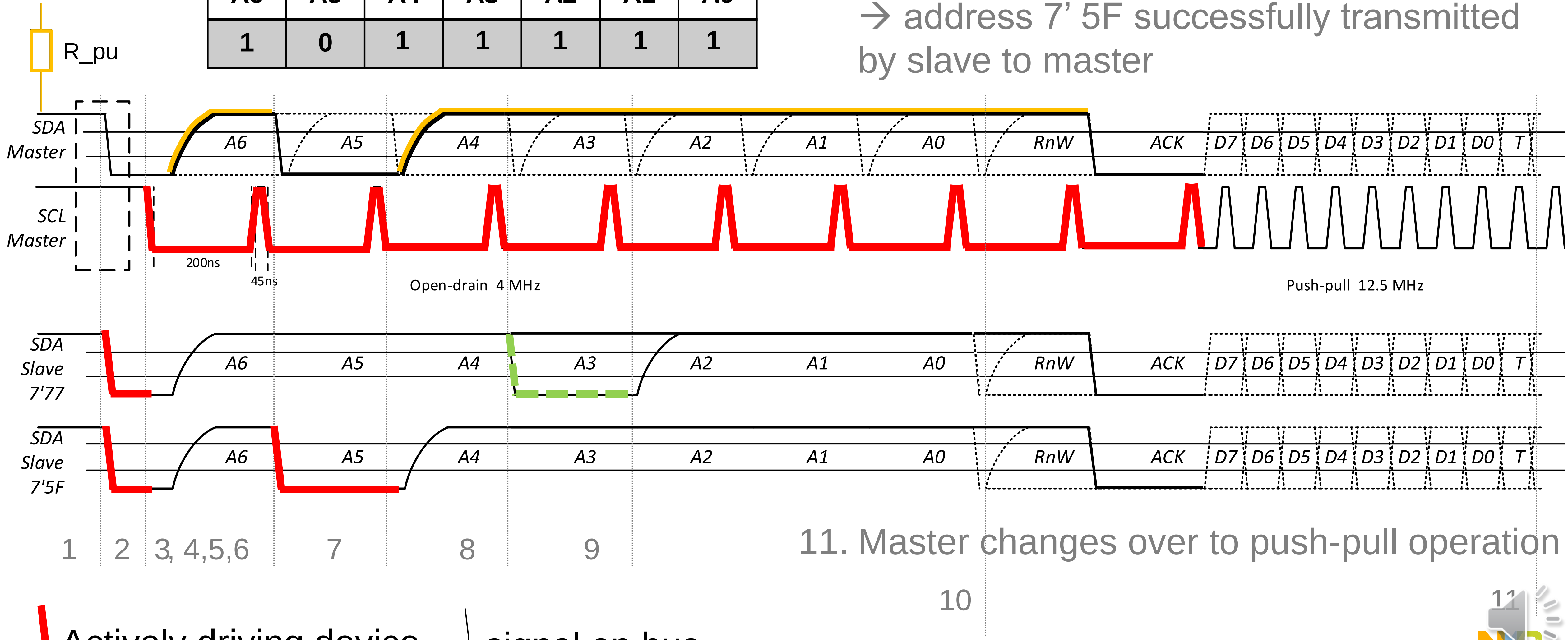
1. Master is idle with SCL stopped and SDA being pulled high by resistor
2. BOTH Slaves trigger an interrupt by pulling SDA low
3. Master starts SCL, pulling it low
4. Slave releases SDA
5. SDA is pulled high by R_pu
6. SCL pulse to latch address bit A6
7. Slave 7'5F pulls A5 low. Latched with next SCL pulse – slave 7'77 keeps listening
8. Slave 7'5F releases SDA so A4 is '1'
9. Slave **7'77** does NOT communicate since A5 deviates from it's address so it 'lost' the arbitration and abstains from communication until next Start condition

ADDRESS ARBITRATION

EXAMPLE: INTERRUPT TRIGGERED BY SLAVE IN A SYSTEM WITH 2 IBI CAPABLE SLAVES

A6	A5	A4	A3	A2	A1	A0
1	0	1	1	1	1	1

10. [A2:A0] are latched by master ('111')
→ address 7' 5F successfully transmitted by slave to master



11. Master changes over to push-pull operation

| Actively driving device | signal on bus

IBI (IN-BAND INTERRUPT)

In-Band interrupt allows Slaves to notify the master

- Can be used as an equivalent function compared to a separate GPIO
  - additionally, the IBI data frame can also be directly data bearing
  - and an IBI is prioritized. The lowest dynamic address slave will gain highest priority during arbitration
- Interrupts can be started even when Master is not active on the bus
  - No free running clock required (lower power)
- Time-stamping option to allow resolution of time of initial event
  - Multiple ways to do: all relate to when actual IBI gets through to Master

Open Drain			**Hand Off**	**Push-Pull**	**Drive High or Low, and then High-Z**	**Optional (push-pull)**	**Push-Pull**
S	Slave_addr_as_IBI/R	**Master_ACK**	SCL High	Slave_byte (‘mandatory byte’)	T	More bytes	**Sr**

HDR MODES FOR HIGHER THROUGHPUT

High data rate (HDR) modes are optional

- Not faster clock, but more bits for same frequency (and so more efficiency)
- Optional to support for Master and Slave
  - Incapable slaves know how to ignore, so others may use safely
- HDR-DDR format
  - About 2x the data rate of SDR (so about 20 Mbps net using 12.5 MHz SCL)
  - Also includes CRC and parity for both Read and Write
  - Uses same SCL clocking, so small adder to Slave logic
- HDR-BT (Bulk transport)
  - 97Mbps real-data rate over 4 wire ML, 48.5Mbps over 2 wire ML, 24.3Mbps single
  - Optimized for large data such as firmware load, camera image (AO), etc.

ERROR DETECTION AND RECOVERY METHODS

- The MIPI I3C specification details error detection/recovery Master and Slave over each mode
  - These are provided in order to avoid fatal conditions when errors occur.
- A set of 6 mandated methods and 1 optional method are specified for MIPI I3C Slave Devices, and a separate set of required methods is specified for MIPI I^3C Master Devices.
- In v1.1, a Slave Reset mechanism is provided so SRSTn pin can be skipped
  - The Slave Reset mechanism can be used to reset broken slaves – special pattern always seen
  - Slave Reset can be controlled for specific slaves and specific resets

Side note:
Clock stretching by slaves is NOT permitted, so bus hang is not an issue
(→ SCL is driven via push-pull by the master)

ADVANCED ANALOG

WE BRING DIGITAL TO LIFE

Device Identifier – Provisional-ID

DEVICE IDENTIFIER - MIPI I3C SLAVE ADDRESSES

Device Identifier

In order to support the Dynamic Address Assignment procedure, each MIPI I3C Device to be connected to an MIPI I3C Bus shall be uniquely identifiable in **one** of two ways, before starting the procedure.

1. *The Device may have an i^2c type <u>Static Address</u>, in which case the Master may use that Static Address to assign it the Dynamic Address more quickly.*
2. The Device shall **in all cases** have a <u>48-bit Provisional ID</u> (unique ID)
   The Master shall rely on this 48-bit Provisional ID, unless the Device has a Static Address used by the master.

The 48-bit Provisional ID is composed of the following parts:

Bits [47:33]	Bit [32]	Bits [31:00]		
		[31:16] 16-bits	[15:12] 4-bits	[11:0] 12-bit
MIPI Manufacturer ID (Note: MSB is discarded) Whether MIPI member or not	Provisional ID Type Selector 1'b1: Random 1'b0: Fixed (normal)	Part ID: The meaning of this 16-bit field is left to the Device vendor to define	Instance ID: Value to identify the individual chip on bus (when multiple of same type): using straps, fuses, non-volatile memory, or another appropriate method	This is left for definition with additional meaning. For example: deeper Device Characteristics, which could optionally extend Device Characteristic Register values
	If Bit [32] = 1'b1: Random Value: **Bits [31:0]: 32-bit value randomly generated by the Device.**			

ADVANCED ANALOG

WE BRING DIGITAL TO LIFE

Common Command Codes (CCC)

COMMAND SPACE (CCC – COMMON COMMAND CODES)

- Built-in Commands (>40) in separate “space” to avoid collision with normal Master → Slave messages
  - Controls bus behavior, modes and states, low power state, enquiries, etc.
    - SET, GET, and SET-GET commands
    - Some required and some optional
  - Has additional room for new built-in commands to be used by other groups
    - E.g. JEDEC, VESA, ETSI, Sim Alliance, PCI-Sig, etc.
    - Vendor space as well for private uses
  - May be direct communication with specific slave(s) and/or broadcast to all

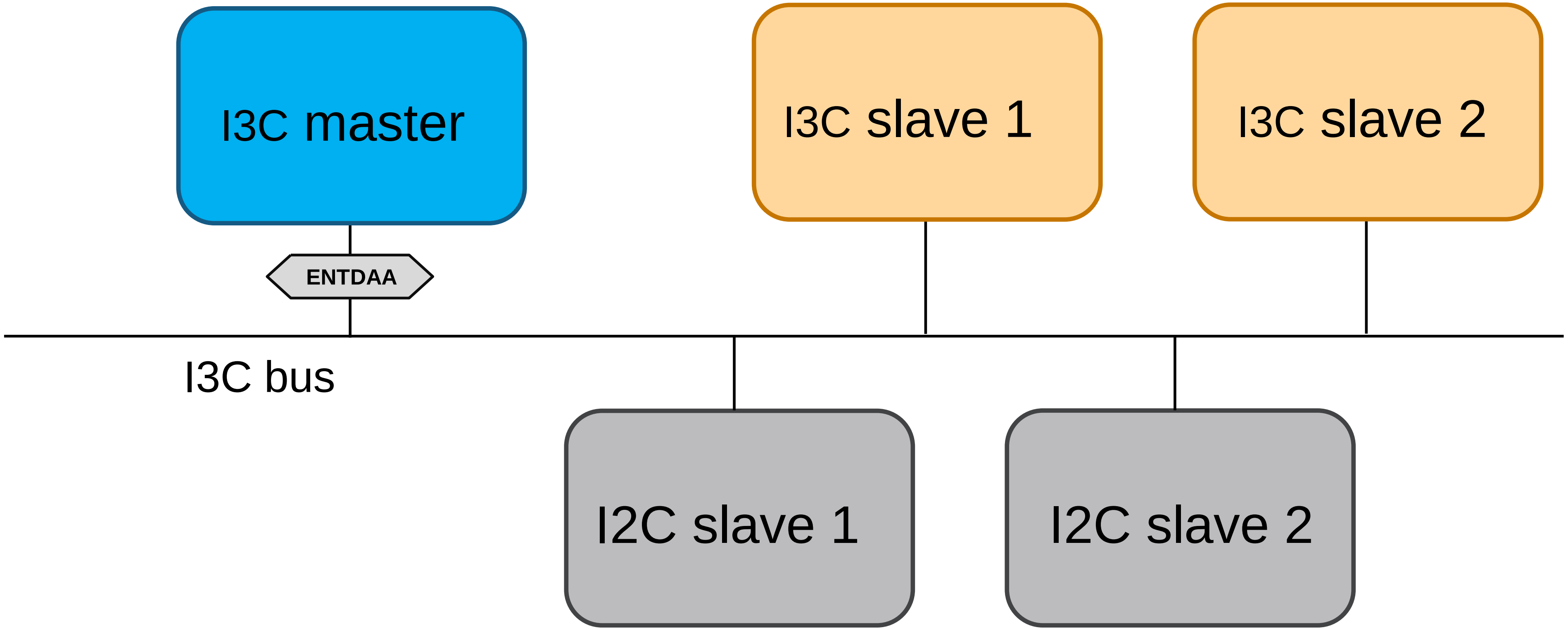
ADVANCED ANALOG

WE BRING DIGITAL TO LIFE

Dynamic Address Assignment (DAA)

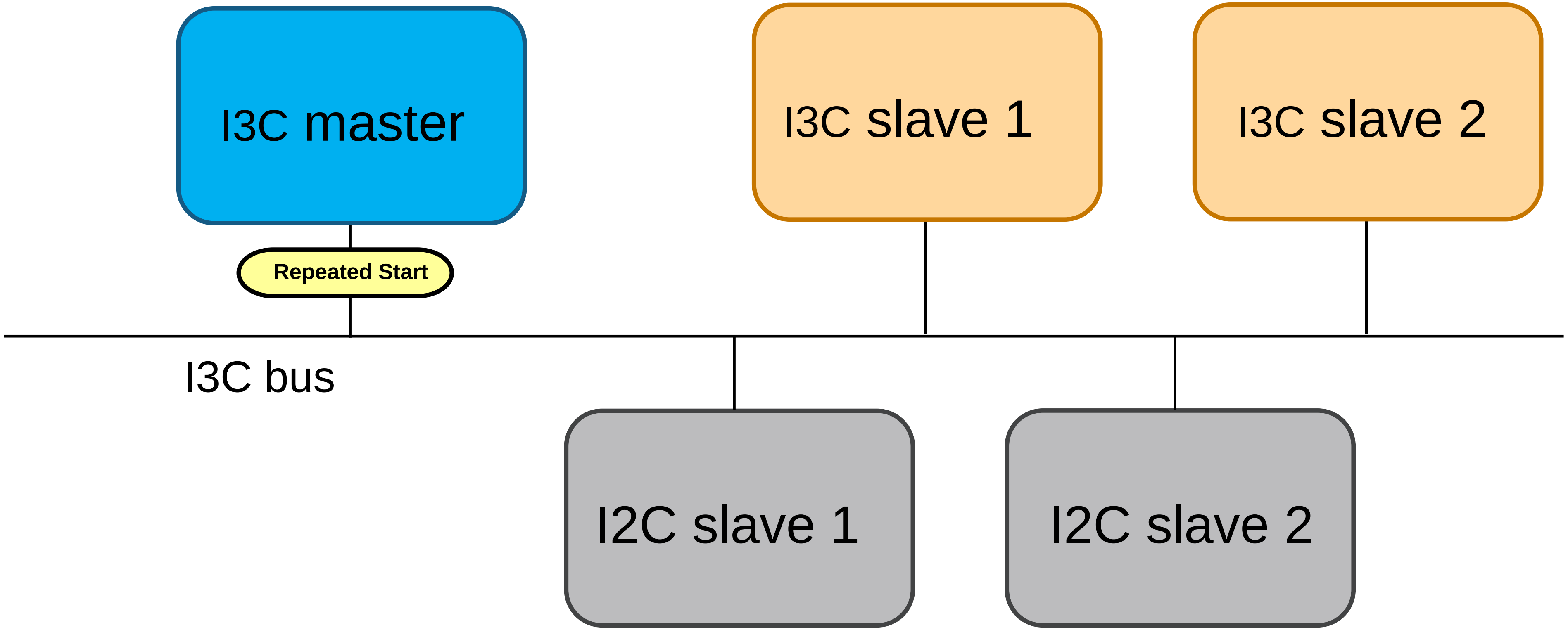
DYNAMIC ADDRESS ASSIGNMENT (DAA)

- I3C master writes ENTDAA CCC Command (7Eh, 07h) to enter “Dynamic Address Assignment” process



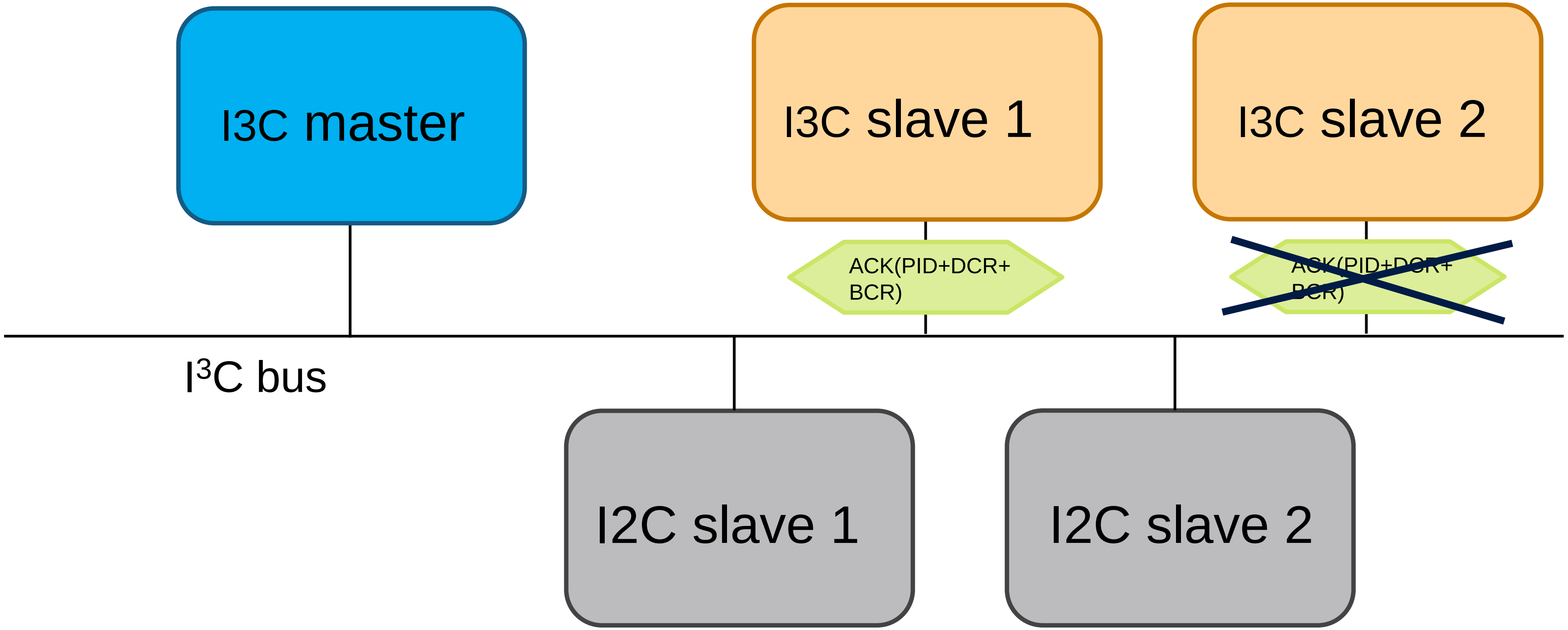
DYNAMIC ADDRESS ASSIGNMENT (DAA) - CONT.

- I3C master writes “Repeated Start” condition



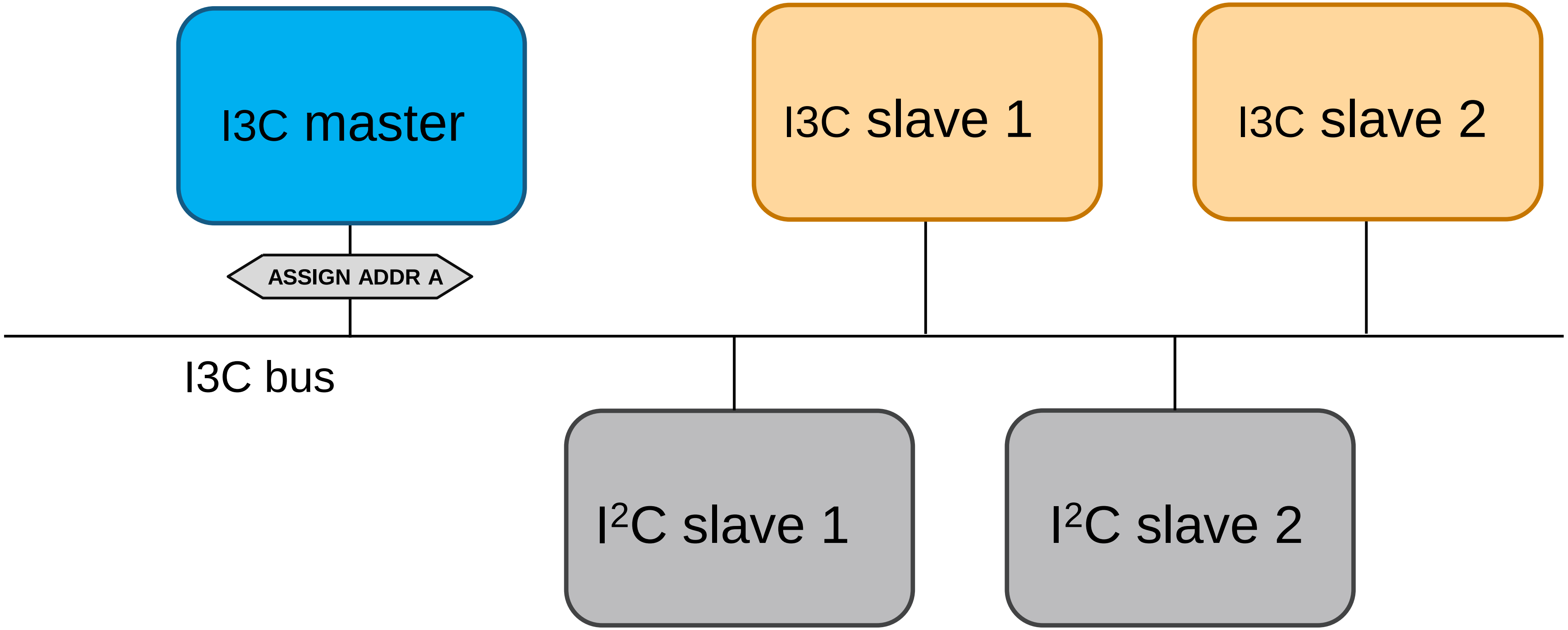
DYNAMIC ADDRESS ASSIGNMENT (DAA) - CONT.

- One of I^{3}C slave wins arbitration



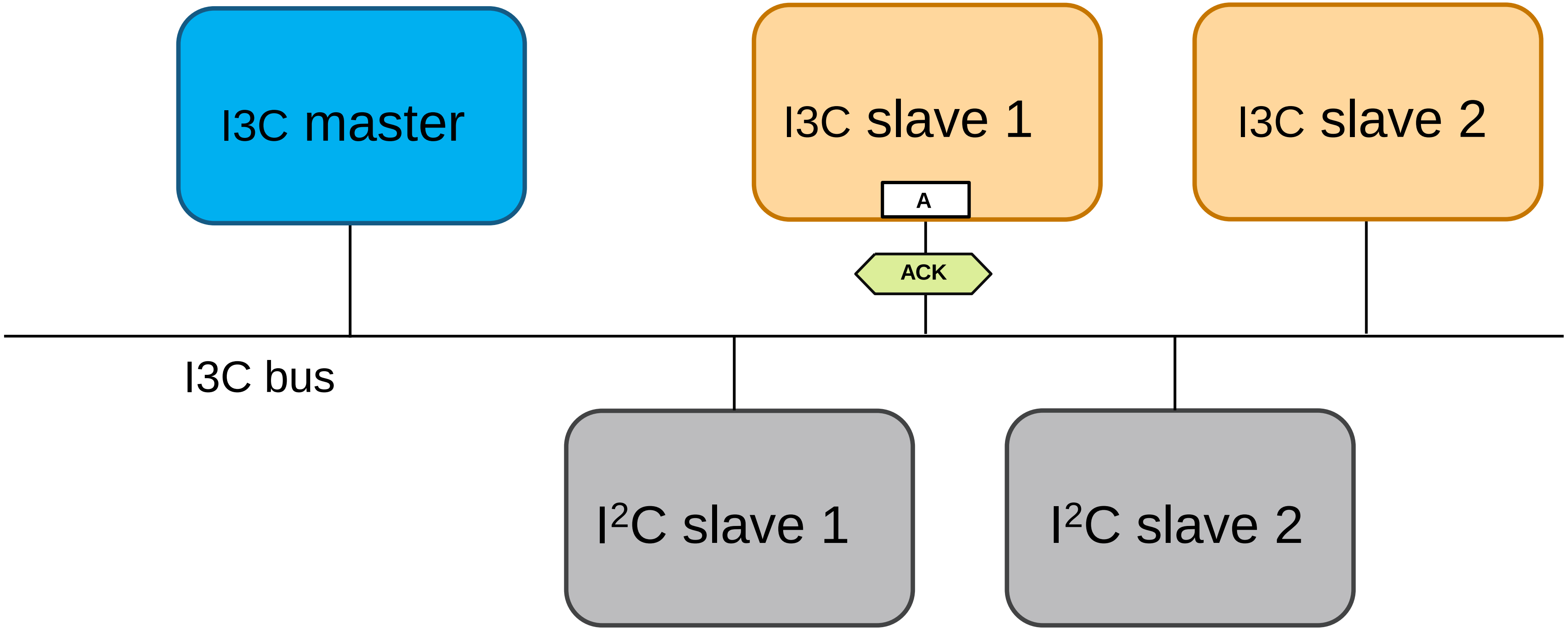
DYNAMIC ADDRESS ASSIGNMENT (DAA) - CONT.

- I3C master assign a dynamic address “A” to the winning I3C slave



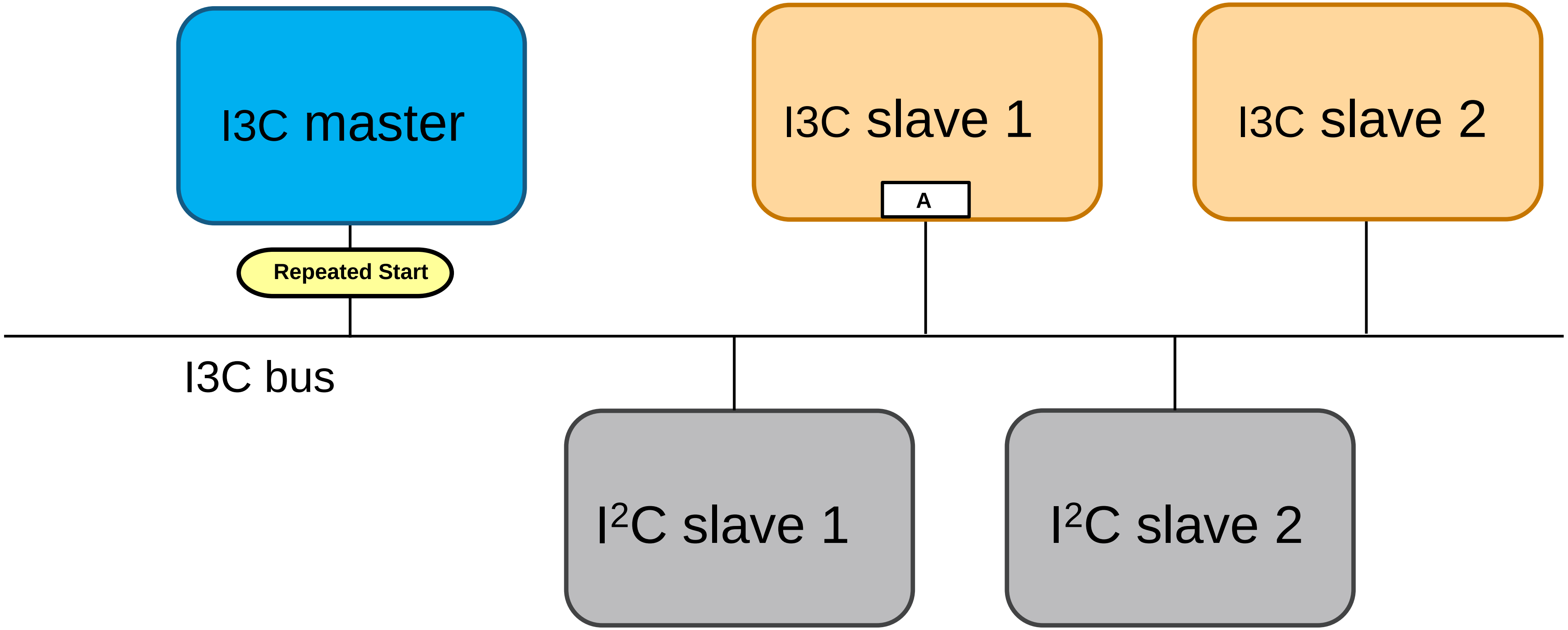
DYNAMIC ADDRESS ASSIGNMENT (DAA) - CONT.

- Winning I3C slave ACK to the dynamic slave address



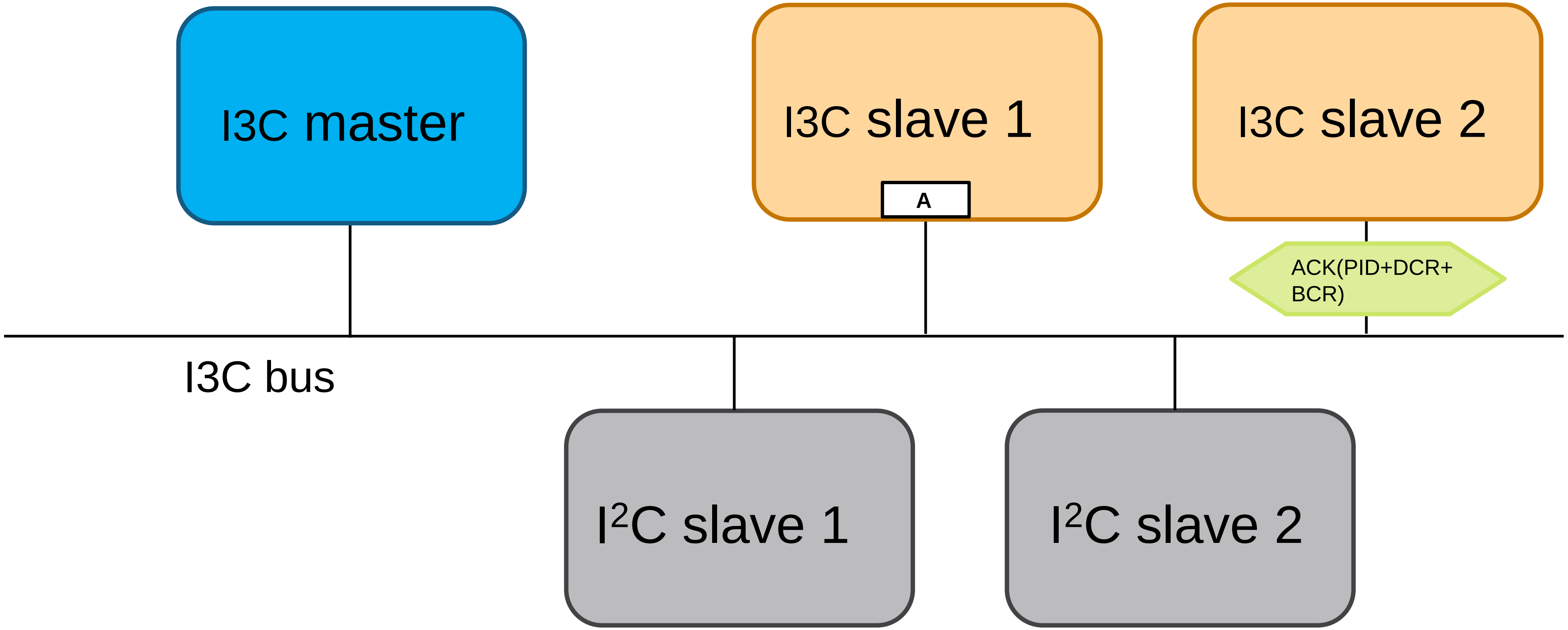
DAA (DYNAMIC ADDRESS ASSIGNMENT) - CONT.

- I3C master writes “Repeated Start” condition onto the I3C bus



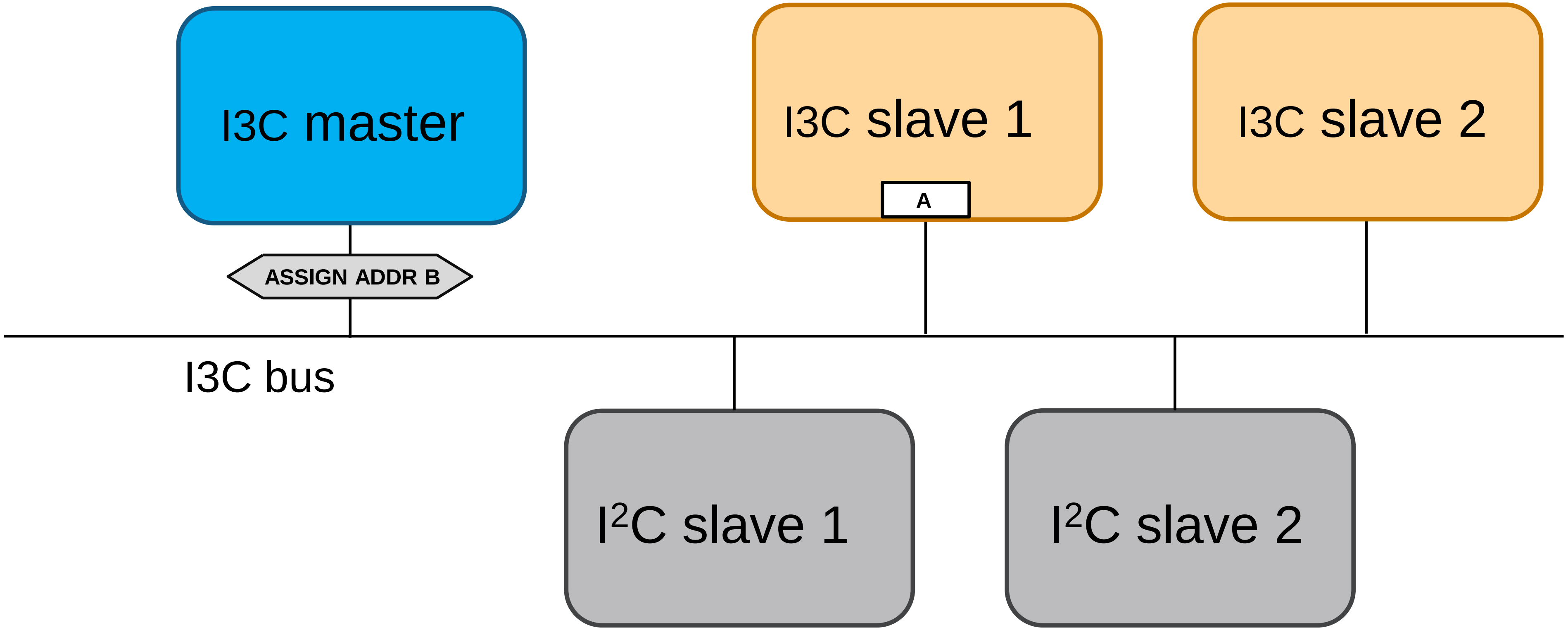
DYNAMIC ADDRESS ASSIGNMENT (DAA) - CONT.

- Second I3C slave device wins arbitration



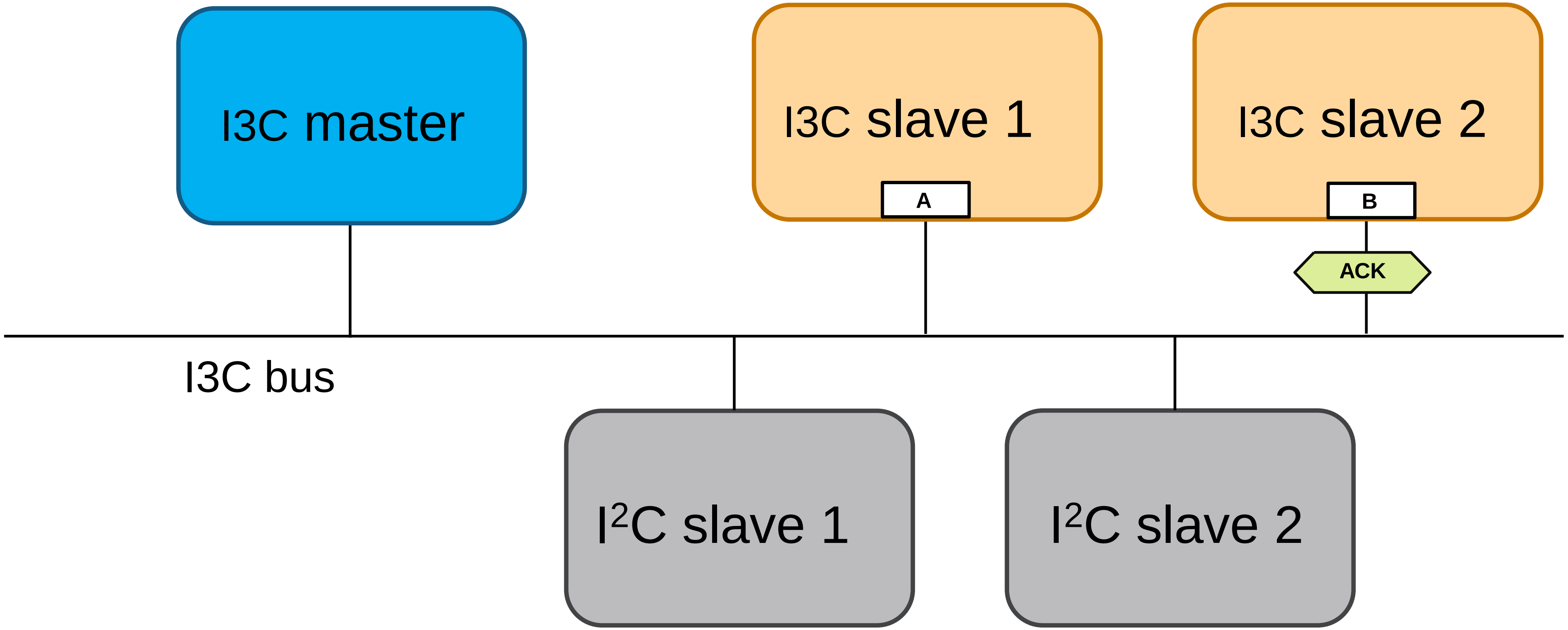
DYNAMIC ADDRESS ASSIGNMENT (DAA) - CONT.

- I3C master assign a dynamic address “B” to the winning I3C slave



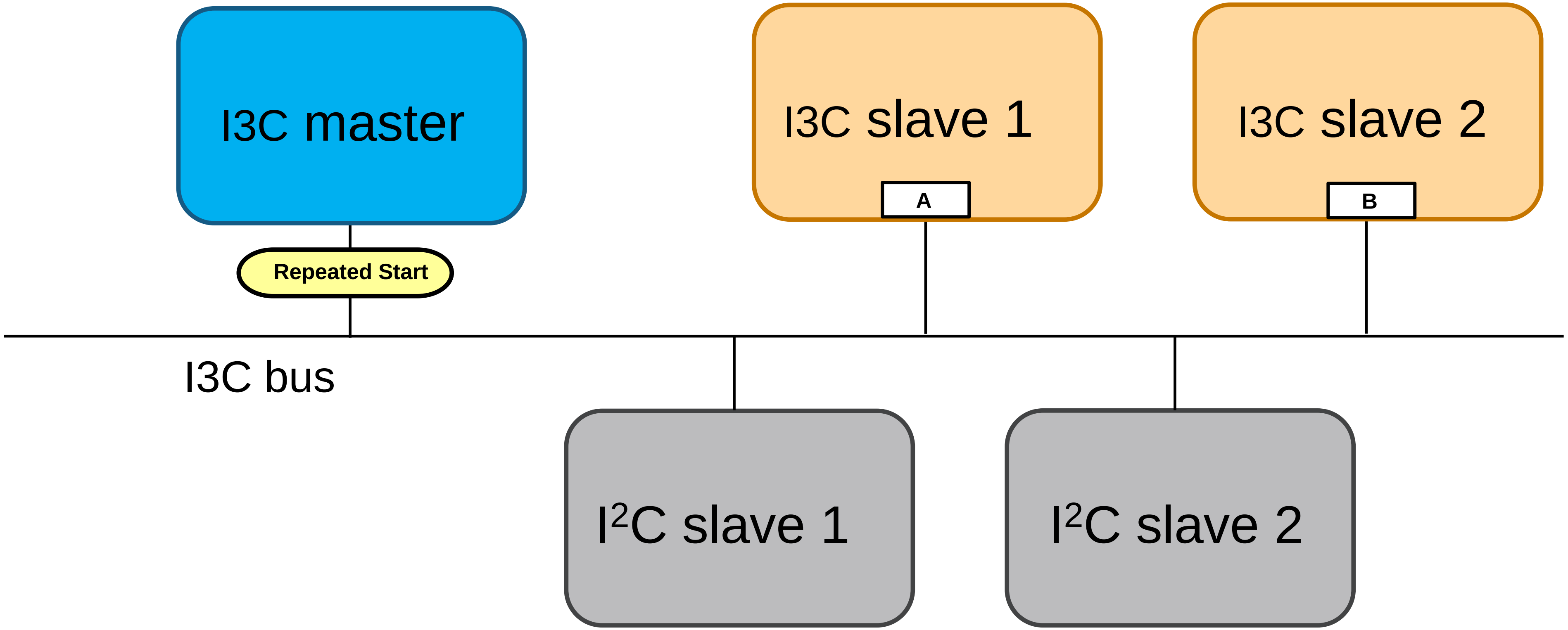
DYNAMIC ADDRESS ASSIGNMENT (DAA) - CONT.

- Winning I3C slave ACK to the dynamic slave address



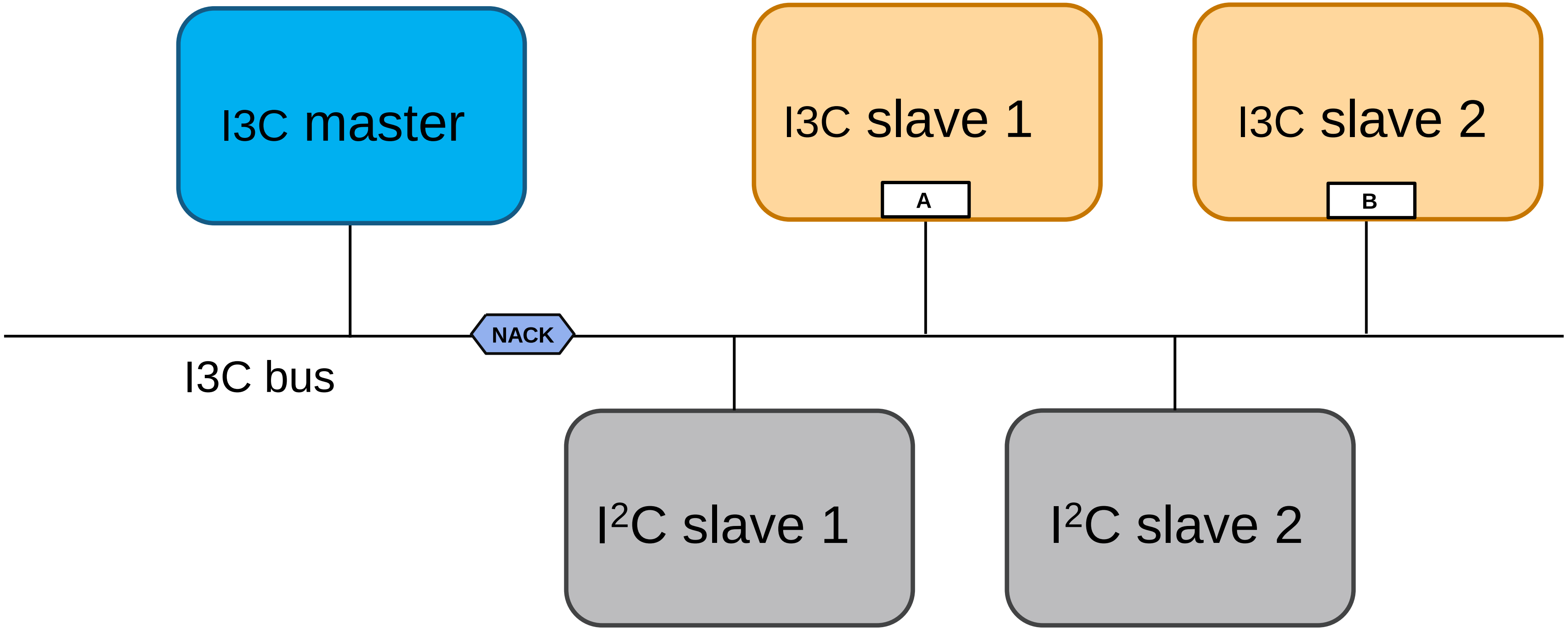
DYNAMIC ADDRESS ASSIGNMENT (DAA) - CONT.

- I3C master writes “Repeated Start” condition onto the I3C bus



DYNAMIC ADDRESS ASSIGNMENT (DAA) - CONT.

- None of I3C slave ACK to the PID/BCR/DCR read



ADVANCED ANALOG

WE BRING DIGITAL TO LIFE

I3C Specification

MIPI I3C V1.1.1 AND I3C BASIC V1.1.1 IN JUNE 2021

- Slave Reset using variant of HDR Exit pattern – independent of I3C state machine
  - Can live in always-on part of device so can be used for low-cost wakeup
- Group addressing
  - Similar to PMBus. Used for multi-cast to a group. Slave may belong to 1 or more
- New HDR-BT (bulk transport) – see next slide
  - Supplying up to 97Mbps real data at 12.5MHz and 4 data lines (same as SPI)
- HDR-DDR mode Slave side Write termination, CRC emitted on Terminate
- Device-to-device tunnel mechanism
- Bus context CCC: so Slave knows details of bus
  - E.g. can identify if on a specific Standard (e.g. JEDEC) and what I3C version

HDR-BT: HDR MODE FOCUSED ON LARGER MESSAGES

- Aimed at max entitled speeds, while allowing ‘flow control’ (terminate):
  - 97Mbps at 12.5MHz : Quad Lane (5 total wires – SCL + SDA[3:0])
  - 48.5Mbps at 12.5MHz : Bi Lane (3 total wires – SCL + SDA[1:0])
  - 24.25Mbps at 12.5MHz: Single Lane (2 total wires – SCL + SDA[0])
- Moves in blocks of 32 bytes, but allows last to be ragged end
- Designed for direct to/from internal SRAM or internal bus for Slave and Master
  - Allows width of 8b, 16b, 32b, 64b, or 128b interface (e.g. AXI/OCP, RAM, etc.)
- Designed to be very easy to implement with no special HW
  - All registered logic, no special pads – should be <1K gates in any process
- Designed to work with existing SW models – as is
- Other: CRC16/32, opt Read clock from Slave, Receiver Verify after CRC, RX stall
  - Supports long wire, pipelined pump of data with verify inline, crypto/auth

I3C BASIC AND IPR

- I3C Basic v1.0 release in late 2018
  - Public spec available to all (MIPI members or not)
  - RAND-Z for all users – using a mutual RANDZ model
  - Being taken up by JEDEC, VESA, SIM Alliance, etc.
- I3C Basic v1.1 release by mid 3Q19
  - Will add Slave Reset, HDR-DDR, HDR-BT, Groups, Async timing control, refinements
- Also, starting March 2019, MIPI IPR rules have changed
  - All specifications, new and old, are RANDZ for all MIPI members in *all* markets
    - No more ”mobile terminal”
  - It is possible QC will relent and I3C will be RANDZ for Everyone starting in 2019
    - I3C Basic would then only exist as “public” document with subset needed by other standards
- I3C Basic v1.1.1 release by mid June 2021

SECURE CONNECTIONS
FOR A SMARTER WORLD

NXP, THE NXP LOGO AND NXP SECURE CONNECTIONS FOR A SMARTER WORLD ARE TRADEMARKS OF NXP B.V. ALL OTHER PRODUCT OR SERVICE NAMES ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. © 2022 NXP B.V.