Alternate I2C Proposal

And use cases

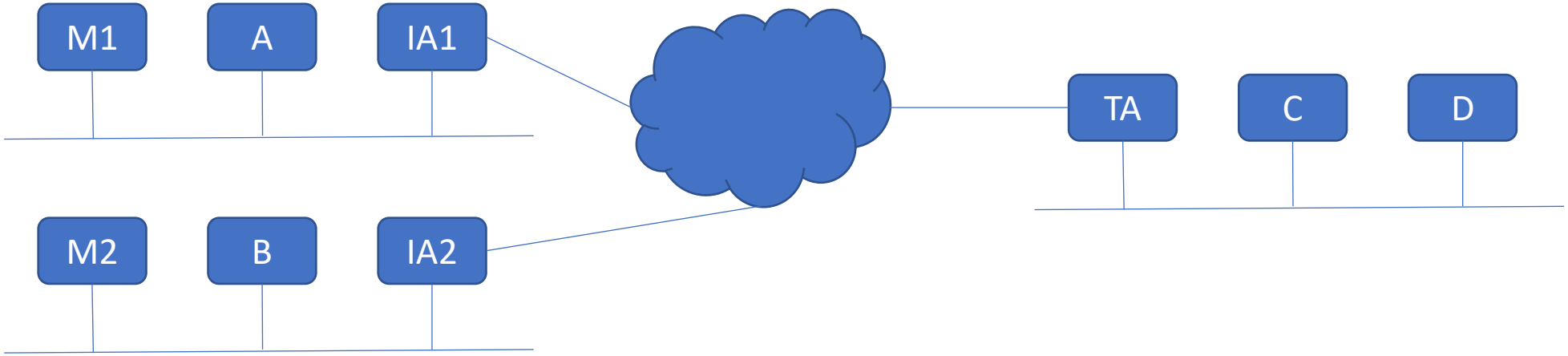
Brian Edem

Marvell Inc

I2C Transactions Types

- Transparent Access
  - A type of I2C bus bridging
  - Original requestor is not aware of the 1722 bridge
    - But must support clock stretching
  - Each bus has a 1722 agent – Initiator Agent or target agent
  - I2C transaction is broken down into three or more parts
  - Can be implemented using standard I2C IP and firmware
  - Provides Ethernet based alternative to existing I2C bridges
    - i.e., LVDS SERDES reverse channel
- Proxy Access
  - A type of I2C device access
    - Original requestor is aware of 1722
    - Requestor will generate packets directly – Initiator Agent implemented in software
  - Remote bus would have a 1722 Target Agent
    - Target Agent could be software driving an I2C bus master interface
  - Transactions are atomic – target bus is left in idle state at end of transaction
  - Better performance than Transparent Access
  - Can coexist with Transparent Access

Transparent Access



- M1 and M2 are I2C masters
- IA1 and IA2 are 1722 Initiator Agents
- TA is a 1722 Target Agent
- A, B, C, and D are end stations (masters or targets)

Transparent Access

- 1722 Requests originate from Initiator Agent
- Breaks down I2C transaction into segments:
  - Start Write - generate START, ADDR, WRITE, return ACK
  - Continue Write - generate DATA, return ACK
  - End Write - generate STOP
  - Start Read - generate START/RESTART, ADDR, READ, return ACK, DATA
  - Continue Read - generate ACK, return DATA
  - End Read - generate NACK, STOP
- Request includes:
  - Segment Type
  - Bus ID – for multi-initiator arbitration
  - Maximum completion time – milliseconds
  - ADDR, DATA
- Each request will generate a response
  - Provides a way to rate limit the requests

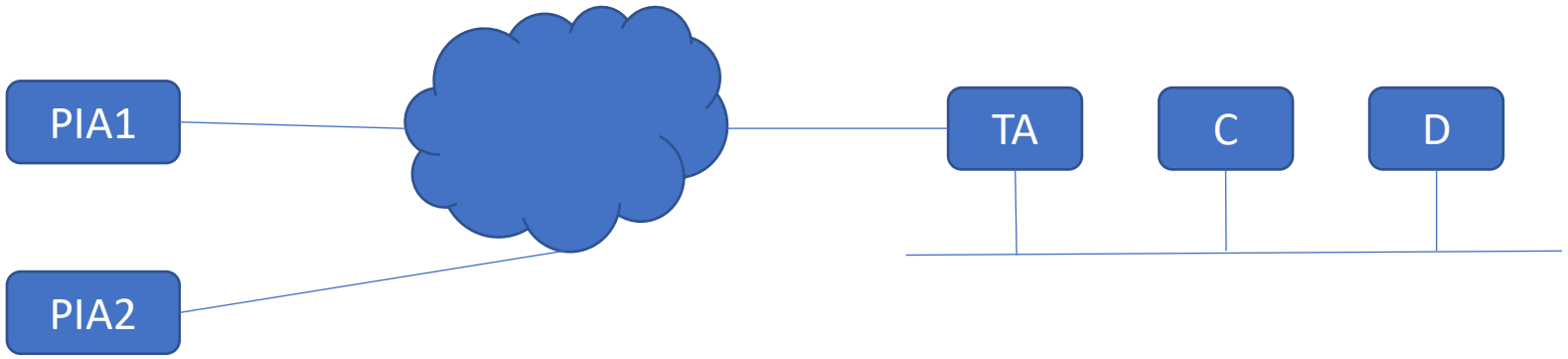
Transparent Access (cont)

- Response includes:
  - Response type
    - Success/failure of associated request
    - State of Target Agent
  - Response data from Read Request
  - Extended status upon Status Request or request failure
    - Failure code:
      - None
      - Bus busy timeout – transaction was unable to start in allocated time
      - Stall timeout – transaction did not complete in allocated time
      - Initiator conflict– Target Agent busy servicing another initiator's request
      - Sequence error – Continue or End with incorrect or missing Start
      - Address error – Continue or End address not matching Start address
    - Bus busy time – reports how long the bus had been busy
    - Competing initiator – bus ID of the initiator agent currently being serviced
    - Segment count – request segments processed for this I2C transaction, mod 4
    - Agent busy – Target Agent was busy processing Transparent Access Requests
    - Bus ID – Bus ID of the client that the Target Agent is currently servicing

Transparent Use Cases

- LVDS extender emulation
  - Only two buses being bridged
  - Target bus can have local bus masters, but not allowed to access 1722 channel
    - Target Agent required to maintain retry buffer if local master supported
- Multi-Bus
  - Masters on multiple I2C busses accessing a single target bus
  - Target Agent locked while busy servicing a request sequence
  - Target Agent may reject requests while busy with another initiator
    - Target Agent could provide additional queue to eliminate need to reject new requests while busy
    - Rejected Initiator Agent will need to retry request
      - This should be rare – only during initialization or error handling/recovery?
      - When does requester retry – timer?
      - Should Target Agent generate message when bus idle?

Proxy I2C Access



- PIA1 and PIA2 are end stations with Ethernet interfaces
- TA is the Target Agent
- C and D are end stations (masters or targets)

Proxy I2C Access

- Request originates from software that is aware of 1722
- Request triggers the Target Agent to perform a complete I2c access
  - Write
  - Write, Restart, Read
  - Read
- Request includes:
  - Device address – used for both write and read phases
  - Maximum completion time – milliseconds
  - Write length – zero indicates no write requested
  - Maximum read length – zero indicates no read requested
  - Write data – byte array used for write operation
- All requests generate a response

Proxy I2C Access (cont)

- Response includes:
  - Read data (if requested)
  - Extended status upon failure
    - Busy timeout indication – transaction was unable to start in allocated time
    - Stall timeout indication – transaction did not complete in allocated time
    - Bus busy time – how long has the bus been busy
- If access a target bus also supporting Transparent I2C
  - Agent busy rejection indication
  - Bus ID of the Initiator Agent currently being served

Proxy Use Cases

- Multiple initiators can access target bus
  - Multiple requests will be queued at Target Agent
  - There will be queue limits beyond which requests will be silently dropped
- Request to a Target Agent busy with a Transparent request can be rejected
  - Transparent and Proxy requests could be queued separately
  - Proxy requests should have priority over Transparent requests

Target Agent Capabilities

- Up to three request queues (requests can be rejected or not supported to reduce)
  - Immediate requests – non-start Transparent Requests, Status Request, Release Agent Request
    - highest priority, serviced immediately
  - Proxy requests
    - second highest priority, serviced when bus is idle
  - Transparent Starts
    - lowest priority
    - Could use immediate queue and reject if Target Agent busy with another Initiator Agent
- Each transaction processed will return basic status and read data (for reads)
  - Extended status returned upon error or if status requested
- Optionally performs I2C bus arbitration (necessary to support local masters)
  - All bytes driven by Target Agent to bus must be recorded to support arbitration
  - Transaction will be replayed after current transaction if arbitration is lost
- Clock speed/bus timing should be configurable

Target Agent Error Recovery

- Attempts to clear “stuck SDA” at start of transaction
- Target Agent releases SDA while waiting for next Transparent Request
  - Release Agent command will release SCL, ending transaction without STOP
  - For some devices, a write transaction without a STOP aborts the write
- Aborts transaction upon stall timeout
  - Continues to clock the bus until target releases SDA, then releases SCL
- Automatically generates STOP upon reading NACK

Transparent Initiator Agent Capabilities

- Monitor bus for transactions attempting to access the remote target
- Stall the master’s transaction by holding SCL low
- Generate request packets to remote Target Agent, wait for responses
- Continue transaction based on request type and response

Transparent Initiator Agent Error Handling

- Bus busy timeout – NACK?
- Stall timeout, Sequence error, or stuck SDA
  - During Start – NACK
  - During Continue Write – NACK
  - During Continue Read – return all ones data
- Initiator Conflict
  - Retry request while continuing to hold SCL low
  - NACK if retry is unsuccessful within allowed time limit
- Response timeout – either request or response packet lost
  - Request retransmission of last status
  - Status includes transaction segment count
    - If the same as the last status reported, request was lost
    - If one more than the last status reported, acknowledge was lost

Proxy Initiator Agent Capabilities

- Translates I2C device register access requests into ACF message
- Extracts results from ACF response message

Proxy Initiator Agent Error Handling

- Possible error types returned:
  - Unable to start access
    - Timeout while waiting for bus Idle
    - SDA stuck low and Target Agent was unable to clear
  - Target device did not respond
    - NACK received during start of Write or Read
    - Write will be non-zero for Read NACK
  - NACK returned during write phase
    - Transaction terminated by Target Agent with STOP
    - Write byte count reported
    - Read restart canceled if requested

Transparent Access Performance

- A two-byte I2C write transaction takes about 200us @100kHz
- As the bridge uses “store and forward”, the time is doubled to 400us
- The bridging requires 3 request packets and 3 responses
  - 64 bytes = 512 bits = 0.512us using 1G Ethernet
  - Six packets would add 3us of transport delay (one hop, ignoring cable delay)
  - Additional hops, queueing, and long cables could add an additional 100us
  - A firmware implementation of Agents would also add delay
    - I2C processing, packet generation/queuing/dequeuing/parsing, I2C generation
- Agents implemented in hardware would minimize additional delay

Bus Monitoring

- A third-party I2C bus monitoring implies multicast Ethernet
  - AcmPDUs are typically sent Unicast
  - These ACMPDUs could be sent Multicast
- Monitor will need to interleave messages
  - Requests have a corresponding Response
  - The response type will have an association to a Request type
- Nothing in this proposal prevents bus monitoring

Possible Message Formats

- Combined, single I2C message type for messages
  - Proxy vs Transparent, Request vs Response
    - Many unused fields for each message type – confusing
  - Error and Diagnostic Status
    - Includes fields not needed for normal operation
  - Large header size spills over into third quadlet (12 bytes)
- Three separate I2C message types
  - I2C Proxy message type – Request, success, fail sub types
  - I2C Transparent message type – 19 sub types that include encoded errors
  - I2C Status message type – 6 error types
- Two separate I2C message types
  - Request
  - Response

Combined I2C Message Header

Size	Name	Use	Description
7	acf_msg_type		
9	adf_msg_length		
2	pad length	Proxy	proxy write length is inferred from acf_msg_length and pad length
8	Bus ID		provides an identifier for the I2C bus on which the message originated
5	Request Type		See Request Types below
7	Device Address		7-bit I2C device address
9	max read length	Proxy Request	maximum length of proxy read
8	Max time	Request	Maximum time that agent should wait before indicating failure
8	transparent data	Transparent	Single byte value used by transparent commands
4	Error code	Response	See Error Types below
1	agent busy	Response	Indicates that the target agent is currently driving the bus
8	client bus id	Response	bus id of the client the target agent is currently servicing
2	segment count	Response	count of segments processed for I2C transaction
10	bus busy time	Response	Time in milliseconds that the bus has been busy, max value indicates more than 1023 milliseconds
88	<- total header length		
0-256	proxy data	Proxy	

Combined I2C Message Encoding

Code	Request Types	Use	Description
0	Proxy Request	proxy	Perform proxy write and/or read operation based on data lengths
1	Start Write Request	transparent	Generate START, ADDR, WRITE; read ACK
2	Continue Write Request	transparent	Generate DATA; read ACK
3	End Write Request	transparent	Generate STOP
4	Start Read Request	transparent	Generate START/RESTART, ADDR, READ; read ACK, DATA
5	Continue Read Request	transparent	Generate ACK, read DATA
6	End Read Request	transparent	Generate NACK, STOP
7	Last Response Request	transparent	Used to determine cause of timeout while waiting for response message
8	Release Agent Request		Provides status
16	Proxy Response	proxy	Provides read data, if requested
17	Write Response	transparent	Informational; Target Agent terminates I2C transaction
18	Read Response	transparent	DATA field valid
19	End Response	transparent	Informational
20	Bus Idle Indication	transparent	Provides status; only generated if initiator conflict message was generated
Code	Request Types	Use	Description
0	None		No errors
1	Bus busy timeout		Request was unable to start as the bus was busy
2	Stuck SCL timeout		SCL was stuck low for duration of request
3	Stuck SDA		Target agent was unable to clear stuck SDA
4	stall timeout		Request did not complete due to excessive clock stretching
5	Target NACK		Target device did not respond
6	Write NACK		Target device responded with NACK while writing
7	initiator conflict	transparent	agent was busy servicing a request for another initiator
8	Sequence Error	transparent	Continue or End request with missing or incorrect type of Start
9	Address error	transparent	device address mismatch from value used in Start
10	Type error	transparent	transaction type (read/write) mismatch from type used in Start

Proxy I2C Message Header

Size	Name	Use	Description
7	acf_msg_type		
9	adf_msg_length		
2	pad length		proxy write length is inferred from acf_msg_length and pad length
8	Bus ID		provides an identifier for the I2C bus on which the message originated
3	Proxy type		Indicates request to target
7	Device Address		7-bit I2C device address
9	max read length	request	maximum length of proxy read
8	Max time	request	Maximum time that agent should wait before indicating failure
8	write byte count	response	
61	<- total header length		
	0-256 bytes data payload		

Code	Proxy Type	Use	Description
0	Request Access	Request	Request to perform transaction
1	Normal Completion	Response	
2	No target response	Response	No device ACK in write or read phase
3	Access failed	Response	Unable to start or complete transaction

Transparent I2C Message Header

Size	Name	Use	Description
7	acf_msg_type		
9	adf_msg_length		
8	Bus ID		provides an identifier for the I2C bus on which the message originated
7	Device Address		7-bit I2C device address
5	Request Type		See Request Types below
8	data		Single byte value used by transparent commands
8	Max time	Request	Maximum time that agent should wait before indicating failure
1	Target Agent Busy	Response	Used to determine if request or response packet was lost
2	Segment Count	Response	Used to determine if request or response packet was lost
8	Client bus ID	Response	Used to determine if request or response packet was lost
63	<- total header length		
Code	Request Types	Use	
0	Start Write Request		Generate START, ADDR, WRITE; read ACK
1	Continue Write Request		Generate DATA; read ACK
2	End Write Request		Generate STOP
3	Start Read Request		Generate START/RESTART, ADDR, READ; read ACK, DATA
4	Continue Read Request		Generate ACK, read DATA
5	End Read Request		Generate NACK, STOP
6	Release Agent Request		Provides status
7	Last Response Request		Used when Initiating Agent times out waiting for response
8	Start Fail Response		Start timed out waiting for bus Idle or SDA stuck
9	Start NACK Response		Informational; Target Agent terminates I2C transaction
10	Write NACK Response		Informational; Target Agent terminates I2C transaction
11	Write ACK Response		Informational
12	Read Response		Read data byte available
13	End Response		Informational
14	Initiator Conflict Response		Recoverable error
15	Stall Timeout Response		Target clock streatch exceeded limit
16	Type Error		Continue/End type (read or write) does not match type of Start
17	Sequence Error		Continue or End without Start
18	Address Error		Continue/End device address does not match device address of Start

Status I2C Message Header

Size	Name	Use	Description
7	acf_msg_type		
9	adf_msg_length		
8	Bus ID		provides an identifier for the I2C bus on which the message originated
4	Error code		See Error Types below
7	device address		device address from request
8	transparent data		Value of last byte read from bus
1	agent busy		Indicates that the target agent is currently driving the bus
8	client bus id		bus id of the client the target agent is currently servicing
2	segment count		count of segments processed for I2C transaction
10	bus busy time		Time in milliseconds that the bus has been busy, max value indicates more than 1023 milliseconds
64	<- total header length		
Code	Msg Types		Description
0	None		No errors
1	Bus busy timeout		Request was unable to start as the bus was busy
2	Stuck SCL timeout		SCL was stuck low for duration of request
3	Stuck SDA		Target agent was unable to clear stuck SDA
4	stall timeout		Request did not complete due to excessive clock stretching
15	Status Request		