**Recommended top-level architecture**

1. **I²C Upstream Slave Front-End (USFE)**
   * Listens on SDA/SCL from the bus master.
   * Detects START/STOP, captures the **address + R/W** byte, ACK/NAK handling, receives/transmits data bytes.
   * Provides byte-level events to the Transaction Manager:
     + rx\_addr[7:1], rx\_rw, addr\_match, start\_seen, stop\_seen, byte\_in, byte\_in\_valid, byte\_out\_ready, byte\_out, ack\_in, ack\_out.
   * Supports **clock stretching** (drive SCL low) when the downstream bus needs more time.
2. **Transaction Manager (TM)**
   * The “brain”. Implements the protocol translation and data pump.
   * If addr\_match == “virtual address for B” → perform downstream transaction to the real device’s **physical address**.
   * Handles multi-byte read/write sequencing, repeated START, and STOP.
   * Buffers one byte in each direction using simple FIFOs or shift registers.
   * Decides when to stretch upstream SCL (e.g., while waiting on downstream).
3. **I²C Downstream Master Engine (DME)**
   * Generates START/STOP on the target bus.
   * Sends address + R/W (physical address), transmits/receives bytes, samples ACKs.
   * Exposes clean handshakes to TM:
     + dme\_cmd (START/STOP/WRITE/READ), dme\_busy, dme\_done, dme\_ack, dme\_byte\_out, dme\_byte\_in, dme\_byte\_in\_valid.
4. **Address Map & Config**
   * Registers holding:
     + virtual\_addr (e.g., 0x49) — **what the master sees**.
     + physical\_addr (e.g., 0x48) — **actual target address**.
     + Optionally a bypass\_enable or a device-select bit if you add the bonus runtime configurability.
   * For the base task, constants via generics/parameters or DIP switches is fine.
5. **Timing & Filters**
   * Use a single FPGA clock (e.g., 50–100 MHz).
   * Implement **digital filters** / metastability synchronizers on SDA/SCL inputs (2–3-FF synchronizers + optional short glitch filter).
   * Bit timing for DME by counters derived from the FPGA clock. Support 100 kHz and 400 kHz via a divider register/parameter.
   * USFE is event-driven by edges on SCL; you still synchronize those edges to your FPGA clock.
6. **Open-Drain I/O handling**
   * SDA and SCL are open-drain: drive ‘0’ actively, release (‘Z’) for ‘1’.
   * Upstream and downstream sides are **electrically separate** (two busses). Do not short them together—this translator is a protocol bridge, not a wired repeater.

**FSMs you’ll likely need**

**A) Upstream Slave Bit-Level → Byte-Level FSM (USFE)**

* **IDLE** → wait for START (SDA falling while SCL high).
* **ADDR** → shift in 8 bits (7-bit address + R/W).
* **ACK\_ADDR** → if address = virtual\_addr, drive ACK; else release to NAK (and optionally ignore until next STOP).
* **RX\_BYTE** (for write transactions) → shift in 8 bits, ACK on completion, present byte\_in\_valid.
* **TX\_BYTE** (for read transactions) → wait for byte\_out\_valid from TM, shift it out, sample master ACK/NAK.
* **STOP\_WAIT** → watch for STOP and clean up.
* At any time, **clock stretch** if TM asserts usfe\_stretch.

**B) Downstream Master Engine (DME)**

* **IDLE**
* **START** → generate START on downstream bus.
* **SEND\_ADDR** → send physical address + R/W, sample ACK.
* **WRITE\_BYTE** → send data byte, sample ACK.
* **READ\_BYTE** → sample data byte, then drive ACK/NAK depending on whether more bytes are needed.
* **STOP** → generate STOP.
* Provides dme\_busy/done/ack signals to TM.

**C) Transaction Manager (TM)**

* **UP\_RX\_ADDR** → on upstream address match, decide R/W direction.
* **FWD\_WRITE** (upstream write to device):
  1. Command DME: START + (physical\_addr, W)
  2. Loop: for each upstream data byte → issue DME WRITE → upstream ACK → continue until STOP or repeated START.
  3. If repeated START with R, switch to **FWD\_READ**.
  4. STOP on upstream STOP; mirror STOP downstream if needed.
* **FWD\_READ** (upstream read from device):
  1. Command DME: START + (physical\_addr, R)
  2. Loop: for each byte from device → provide to USFE TX path → mirror master ACK/NAK to DME.
  3. On upstream NAK or STOP → DME STOP, then exit.
* **ERROR/RECOVERY** → handle NAKs/timeouts gracefully: upstream should see a NAK if downstream failed.

**Address translation rule (simple & robust)**

* Upstream USFE **only** acknowledges address **virtual\_addr**.
* When it does, TM launches a downstream transaction with **physical\_addr** and the **same R/W** bit.
* All **payload bytes** are passed through unmodified, in order, until STOP (or repeated START changes direction).
* If the upstream address doesn’t match, USFE releases SDA for the address ACK bit (i.e., it doesn’t respond). Device A at 0x48 remains unaffected and will still ACK its own address directly.

**Timing & SCL rates (100/400 kHz)**

* Choose a fast FPGA sysclk (e.g., 50 MHz).
* **USFE** works from synchronized SCL edges; you don’t generate timing, you **observe**. Use an edge detector to shift bits on SCL rising edges and sample START/STOP on SDA transitions while SCL high.
* **DME** must generate SCL using a divider to meet either 100 kHz or 400 kHz.
* To bridge between different rates and to cover downstream latency, the USFE may **clock stretch** upstream by keeping SDA prepared but holding SCL low (drive 0) until the DME completes its current byte operation.

**Simulation plan (meets your “two devices” requirement)**

**Testbench components (behavioral, your own code):**

1. **Upstream Master Model**
   * Issues transactions at ~100 kHz.
   * Scenarios:
     + Write 2 bytes to 0x49 (virtual, R/W=0).
     + Read 3 bytes from 0x49 (R/W=1).
     + Access 0x48 directly (device A) to prove it is unaffected.
2. **Device A Model** (physical 0x48)
   * Simple memory/register model to echo reads and latch writes.
3. **Device B Model** (also physical 0x48!)
   * Identical to A but connected only to **downstream** side of your translator.
   * Upstream bus should never “see” B directly; only via translator responding at 0x49.
4. **Your Translator DUT**
   * virtual\_addr = 7'h49, physical\_addr = 7'h48.

**Must-check waveforms**

* START/STOP correctly detected and mirrored.
* For transactions to **0x49** upstream:
  + USFE ACKs the address.
  + DME generates a downstream START with **0x48**.
  + Payload bytes match byte-for-byte.
  + Repeated STARTs work (write → read sequences).
* For transactions to **0x48** upstream:
  + Translator **does not ACK**.
  + Device A ACKs and responds normally.
* **ACK/NAK propagation**: if downstream NAKs (e.g., device B missing), upstream sees a NAK.
* **Clock stretching** visible when DME isn’t ready.

**EDA Playground setup (public link per task)**

* Choose Verilog or VHDL flavor + simulator (e.g., Icarus/Questa).
* Add four files: tb\_top, i2c\_master\_model.sv, i2c\_device\_model\_a.sv, i2c\_device\_model\_b.sv, plus your RTL files.
* Include assert statements on:
  + Data integrity (write→read back).
  + No translator ACK on 0x48.
  + Translator ACK on 0x49.
  + Address bytes seen downstream equal 0x48 when upstream used 0x49.

**Resource & timing report (tool-agnostic checklist)**

Use **any** FPGA toolchain you’re comfortable with (Vivado/Quartus/Radiant/etc.):

* Top-level I/O:
  + scl\_up, sda\_up (inout, open-drain)
  + scl\_dn, sda\_dn (inout, open-drain)
  + clk, rst\_n
  + Optional: mode\_fast (100/400 kHz select), DIP switches for addresses
* Synthesize with typical constraints:
  + create\_clock on clk (e.g., 50 MHz)
  + I²C pads as open-drain OBUFTs or vendor primitives
  + Multicycle/false paths are generally **not** needed if all logic synchronous
* Export:
  + **Utilization** (LUTs/FFs/BRAM)
  + **Timing summary** (WNS/TNS; ensure positive slack at sysclk)
  + **Fmax** of sysclk comfortably above requirement
  + (Optional) Power estimate

**Documentation outline (what to include)**

1. **Architecture overview**
   * Block diagram: USFE, TM, DME, Address Map, Filters.
   * Explain the two separate I²C domains (upstream vs downstream).
2. **FSM/logic explanation**
   * State diagrams with transitions (as listed above).
   * When and why you assert clock stretching.
3. **Address translation**
   * Upstream sees virtual\_addr; downstream uses physical\_addr.
   * Only the address byte is changed; all payload bytes pass through.
4. **Design challenges** (discuss at least these)
   * Synchronizing asynchronous SCL/SDA to FPGA clock.
   * START/STOP robustness with glitches.
   * Correct ACK/NAK propagation across domains.
   * Handling repeated START (write→read combined format).
   * Timing closure with open-drain tri-states.
5. **How to run the sim**
   * Exact steps for your EDA Playground link: compile, run, view waves, what to look for.
6. **(Bonus) Runtime configurability**
   * Simple register interface (e.g., a tiny sideband I²C control port at a distinct config address) to set virtual\_addr, physical\_addr, and mode.

**Test scenarios you should implement (copy this into your test plan)**

1. **Write pass-through**
   * Master → 0x49 (W): bytes {0xAA, 0x55}
   * Expect downstream: 0x48 (W), same bytes, both ACKed, STOP seen at both sides.
2. **Read pass-through**
   * Device B preloaded memory at address 0x10 = {0xDE, 0xAD, 0xBE}.
   * Master → 0x49 (W): {0x10}, repeated START → 0x49 (R), expect upstream read {0xDE, 0xAD, 0xBE}.
   * Downstream must show 0x48 (W) then 0x48 (R) with matching data.
3. **Address isolation**
   * Master → 0x48 (R): Device A responds; translator stays silent (no ACK on the upstream address byte).
   * Ensure Device A’s data path unaffected by translator transactions.
4. **Downstream NAK propagation**
   * Force Device B to NAK a data byte; verify upstream master observes NAK in the mirrored position.
5. **Clock stretching**
   * Make DME intentionally slow (e.g., insert a few cycles of delay before dme\_done).
   * Confirm USFE holds SCL low (stretch) until the byte is ready.
6. **400 kHz mode**
   * Rerun (1) and (2) with your 400 kHz divider—verify setup/hold and clean ACKs.

**Practical implementation tips**

* **Edge detection:** Sample SDA and SCL through 2-FF synchronizers; detect START when SDA\_falls && SCL\_high; STOP when SDA\_rises && SCL\_high.
* **Shift registers:** Shift on SCL rising edges (for input) and drive SDA on SCL falling edges (for output) to meet I²C timing.
* **ACK timing:** The ACK bit is driven by the receiver during the 9th clock—ensure SDA is valid while SCL is high on that bit.
* **Open-drain buffers:** In HDL, model as output enable + ‘0’; never drive ‘1’.
* **Reset behavior:** Return to IDLE on STOP or reset; clear FIFOs.
* **Reusable counters:** One bit counter for 0..7 within a byte; a divider for DME SCL generation.