



PAL

P1394b Accelerations Status

Dave LaFollette
3/17/97

For discussion purposes only

Agenda

- Overview of new draft
- Request types
- Mixed bus discussion
- AOB

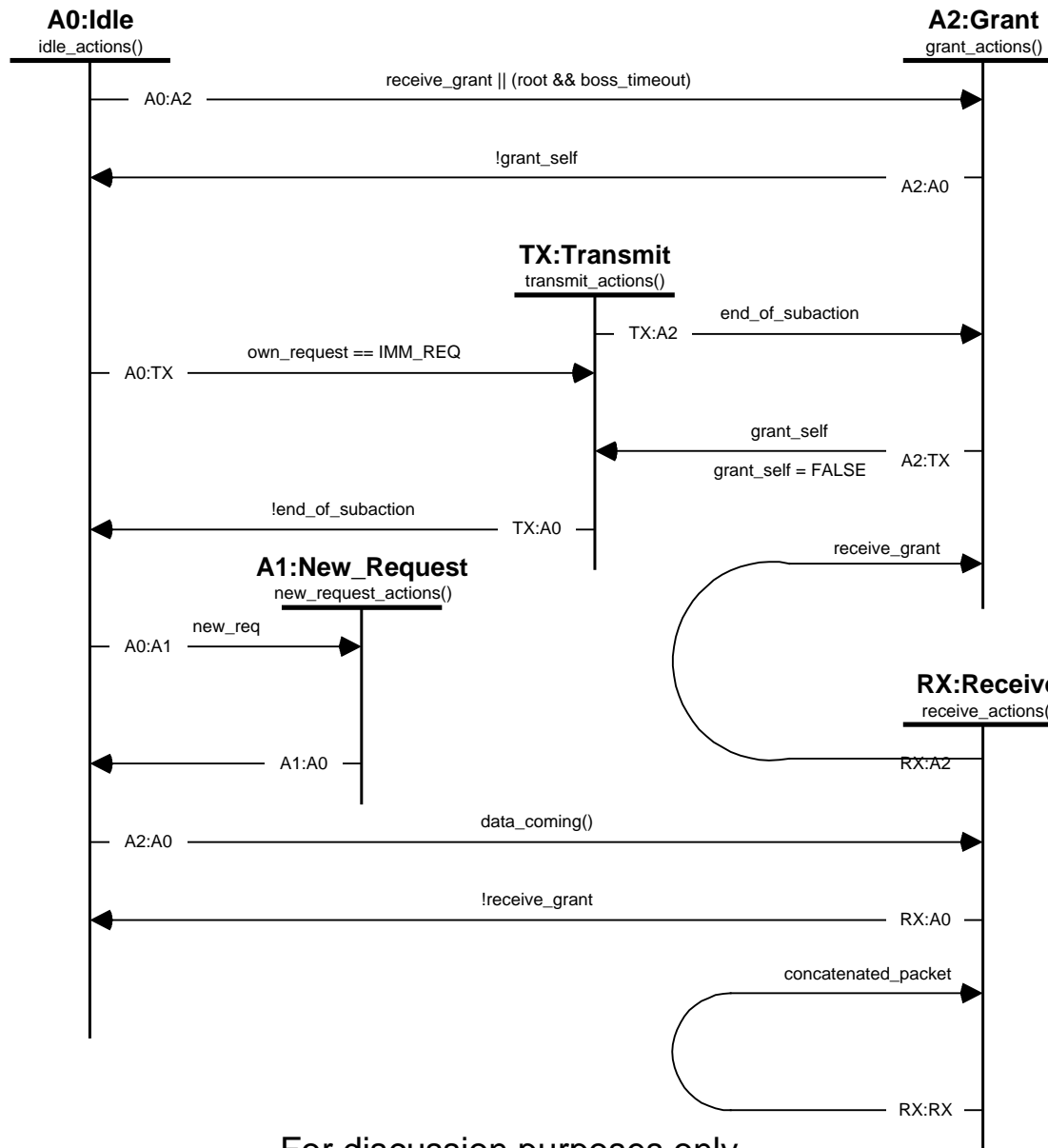
For discussion purposes only



INTEL
ARCHITECTURE LABS

PAL

Beta Mode Arbitration State Machine



For discussion purposes only



Beta Mode Arbitration Actions and Conditions

```
int out_req(int port){
    //combine best asynch and isoch request from other ports and
    //own request
    //return(request)
}
```

```
void idle_actions() {
    //while !(new_req || IMM_REQ || receive_grant || data_coming)
    // for (i=1; i<NPORT; i++)
    // transmit out_req(i) on each port
}
```

```
void new_request_actions() { //new request received from link
    //update own_request with new request from link
    //combine latest isoch and asynch requests in own_request
}
```

```
void transmit_actions() {
    //transmit packet on all ports
    //check whether end of subaction
    //decode phy packets
    //if no asynch requests for current interval
    // arb_reset==TRUE
}
```

```
void grant_actions() {
    //if (root)
    // wait for a request
    //if (own_req > best_request)
    // grant_self = TRUE;
    //else if (best_request > 0)
    // grant the port with the best request
    //else
    // grant parent port
    //send request code on all other ports
    //if arb_reset
    // send arb_reset on all ports
}
```

```
void receive_actions() {
    //send received packet to link layer
    //repeat packet out other ports
    //send out_req(receive_port) on receive_port
    //check for grant or arb reset notification at end of
    // packet
    //decode phy_packets
}
```

For discussion purposes only

Request Types

- **Isoch**

- ◆ current cycle
 - ≥ 800
 - ≤ 400
- ◆ next cycle
 - ≥ 800
 - ≤ 400
- ◆ none

- **Asynch**

- ◆ cycle start
- ◆ current interval
 - ≥ 800
 - ≤ 400
- ◆ next interval
 - ≥ 800
 - ≤ 400
- ◆ none

- **Best isoch type is combined with best asynch type**

- ◆ 5 isoch types and 6 asynch types lead to 30 type combinations
- ◆ allows requests such as:
 - cycle start and isoch ≥ 800 for next cycle
 - asynch ≥ 800 for next interval and isoch ≤ 400 for this cycle

- **Mixed nodes determine legacy request types by timing**

- ◆ Isoch ≤ 400 between cycle start and 1st subaction gap
- ◆ Asynch ≤ 400 between 1st subaction gap and next cycle start

For discussion purposes only

Request Type Prioritization

- **Cycle start must be highest priority**
- **All isoch traffic must precede any asynch traffic**
 - ◆ if asynch traffic slipped in, it could consume too much bandwidth
 - ◆ legacy isoch is not done until subaction gap occurs
 - this gap may occur during ≥ 800 traffic
- **general prioritization for ≥ 800 or ≤ 400 ?**
 - ◆ legacy interval is not done until an arbitration reset gap occurs

For discussion purposes only

Mixed Bus Considerations

- **Subaction gaps**
 - ◆ Legacy bus must see subaction gaps to guarantee all nodes send their asynch arbitration requests
 - ◆ Legacy bus must not see a subaction gap until all $\leq S400$ isoch traffic is complete
 - ◆ gaps may be synthesized during ≥ 800 traffic
 - ◆ subaction gaps must be punctuated by null packets to prevent unintentional arbitration reset gaps
 - ◆ Beta bus must account for the presence of random null packets on legacy busses
- **Arbitration reset gaps**
 - ◆ Legacy bus must see an arbitration reset gap to recognize the next fairness interval
 - ◆ The only way to know the legacy bus has sent all its current interval requests is to see an arbitration reset gap
 - ◆ gaps may be synthesized during ≥ 800 traffic

More Mixed Bus Considerations

- **Must wait a subaction gap after a ≤ 400 asynchronous packet except ACK**
 - ◆ just like ack-accelerated arbitration
- **What if legacy is root or at least parent?**
 - ◆ Beta nodes can try to become root
 - If beta domains are separated by legacy nodes, some beta nodes must have a legacy parent
 - ◆ Beta nodes could still operate in beta mode when they win the bus
 - They would have to release the bus in time for the cycle start packet
 - They would have to observe arbitration reset gaps
- **Beta nodes have different needs for gap count**
 - ◆ timeouts
 - ◆ legacy domains could have individual gap counts

Possibly Useful Tricks

- **Sort traffic by speeds to give legacy bus gaps during $\geq S800$ traffic**
- **Mixed nodes assume legacy requests until gaps indicate otherwise**
 - ◆ This could help prevent beta nodes from having to wait for gaps
 - ◆ Beta bus can proceed whenever there are no legacy requests
- **Separate fairness intervals for legacy and beta**
 - ◆ Legacy gets higher priority
 - ◆ Priorities change
 - ◆ Beta takes the bus during gaps
- **Use of more request types**
 - ◆ differentiate true legacy from beta ≤ 400
 - ◆ mark short or long packets
 - ◆ include legacy bus state information
 - ◆ others?