Proposal for an Open Loop PHY Rate Control Mechanism

Shimon Muller
Ariel Hendel

Sun Microsystems Inc.
Computer Systems

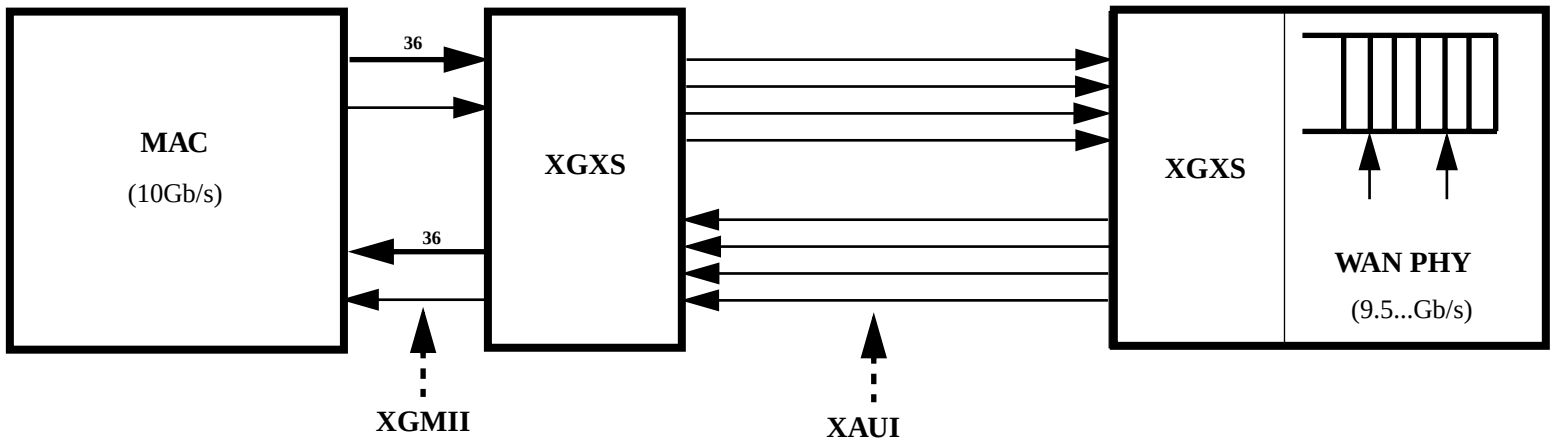
July 11, 2000

Outline

- Introduction
  - Why is a Rate Control Mechanism Necessary for 802.3ae
- MAC<->PHY Rate Control Alternatives
- MAC Self-Pacing Proposal
  - Concept
  - Implementation Implications
  - Standard Implications
- Summary

Introduction --- Why Rate Control for 802.3ae?

- At the November 1999 meeting, the HSSG adopted the following objectives for 802.3ae:
  - Support a speed of 10.0000 Gb/s at the MAC/PLS service interface
  - Define two families of PHYs:
    - A LAN PHY, operating at a data rate of 10.0000 Gb/s
    - A WAN PHY, operating at a data rate compatible with the payload rate of OC-192c/SDH VC-4-64c
  - Define a mechanism to adapt the MAC/PLS data rate to the data rate of the WAN PHY



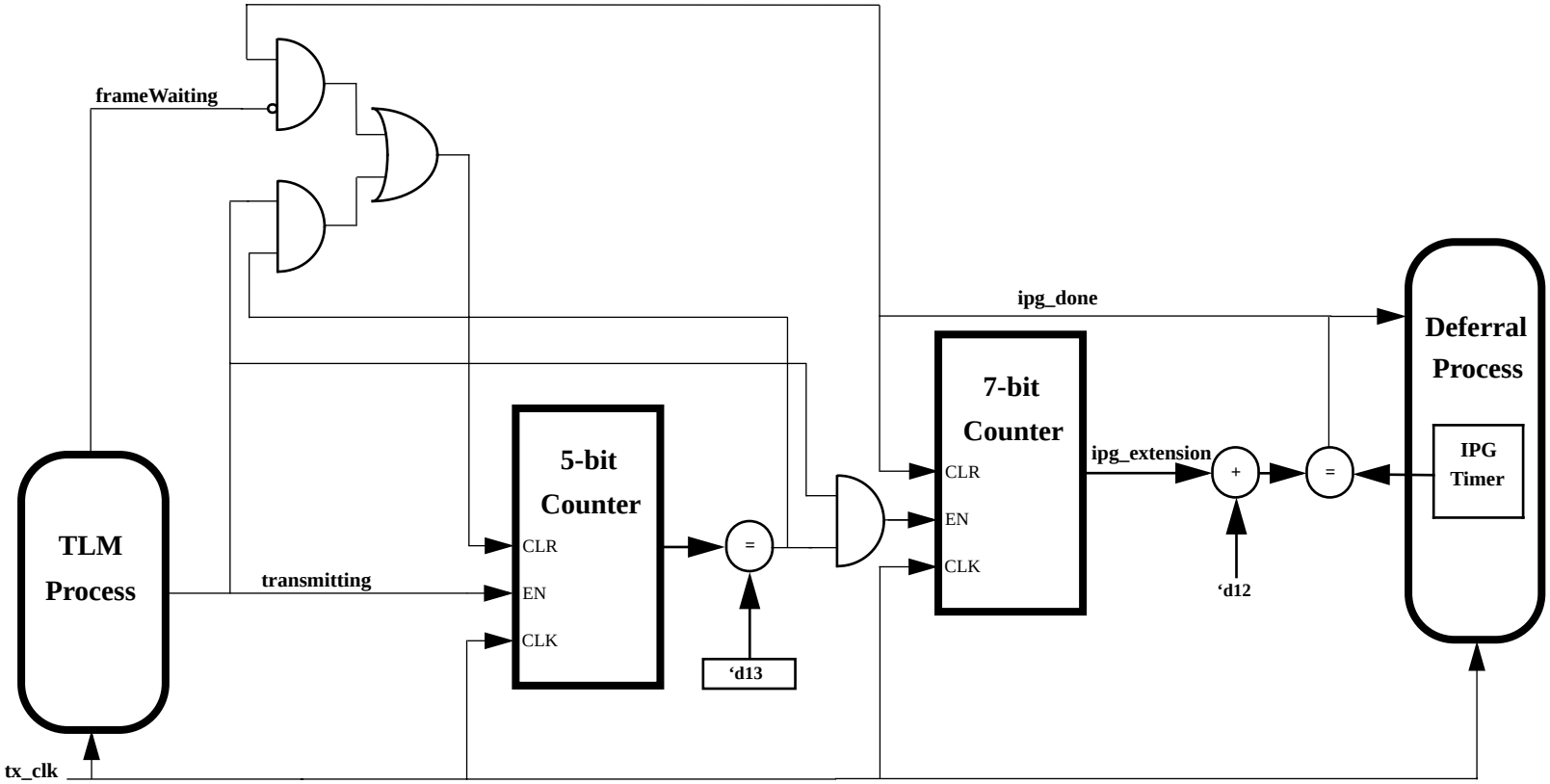
MAC<->PHY Rate Control Alternatives

- Fine granularity rate control
  - Word-by-Word hold signalling
  - Clock stretching
- Packet granularity rate control
  - Frame-based
  - Carrier Sense based
  - Busy Idle
  - Self pacing in the MAC

MAC Self-Pacing Proposal

- **Concept**
  - **The MAC "knows" that the PHY is slower and by how much**
  - **The MAC adapts its average data rate by extending the IPG after each frame transmission**
    - **This guarantees that the MAC never exceeds the average data rate in the PHY, with packet granularity**
  - **The IPG extension is "dynamic"**
    - **Depends on the size of the previously transmitted frames**
  - **The PHY is only required to sustain the transmission of one maximum size packet**
    - **Requires a rate adaptation fifo in the PHY of ~64 bytes (plus framer overhead)**

MAC Self-Pacing Proposal --- Implementation



Notes:

*** transmitting --- signal that frames the transmission of a frame in the MAC**

*** ipg_done --- signal that indicates the completion of IPG transmission**

*** frameWaiting --- signal that indicates that a frame was passed to the MAC for transmission**

MAC Self-Pacing Proposal --- Pascal Changes

- **Transmit state variables (4.2.7.2)**
  - ***const***

    ifsExtensionRatio = ...; {In bits, determines the number of bits in a frame that will require one octet of interFrameSpacing extension, see 4.4}
  - ***var***

    paceMode: Boolean; {Indicates the desired mode of operation, ... static variable}
    ifsExtensionCount: 0... {In bits, running counter that counts the number of bits during frame transmission that will be considered for minimum interFrameSpacing extension}
    ifsExtensionSize: 0... {In octets, running counter that counts the integer number of octets to be added to minimum interFrameSpacing}

- **State variable initialization (4.2.7.5)**
  - *procedure* Initialize;

    begin

    .....

    paceMode := ...;
    ifsExtensionCount := 0;
    ifsExtensionSize := 0;
    while carrierSense *or* receiveDataValid *do* nothing

    end;

MAC Self-Pacing Proposal --- Pascal Changes (cont.)

- **Frame transmission (4.2.8)**
  - *function* TransmitLinkMgmt: TransmitStatus;

```
begin
                .....
                .....
   begin {loop}
      if bursting then frameWaiting := true
      else
         begin
            if attempts > 0 then BackOff;
            if halfDuplex then frameWaiting := true;
            .....
            .....
         end;
      lateCollision := false;
      StartTransmit;
      frameWaiting := false;
      if halfDuplex then
         begin
            frameWaiting := false;
         .....
         .....
         end {half duplex mode}
      else while transmitting do nothing
   end; {loop}
         .....
         .....
end; {TransmitLinkMgmt}
```

MAC Self-Pacing Proposal --- Pascal Changes (cont.)

```
process BitTransmitter;
  begin
    cycle {outer loop}
      if transmitting then
        begin {inner loop}
          extendError := false;
          PhysicalSignalEncap;
          while transmitting do
            begin
              if (currentTransmitBit > lastTransmitBit) then TransmitBit(extensionBit)
              else if extendError then TransmitBit(extensionErrorBit)
              else
                begin
                  TransmitBit(outgoingFrame[currentTransmitBit]);
                  ifsExtensionCount := ifsExtensionCount + 1;
                  if ((ifsExtensionCount mod 8) = 0) then
                    if ((ifsExtensionCount mod ifsExtensionRatio) = 0) then
                                                  ifsExtensionSize := ifsExtensionSize + 1
                end;
              if newCollision then StartJam else NextBit
            end;
          if bursting then
            begin
              InterFrameSignal;
              if extendError then
                if transmitting then transmitting := false
                else IncLargeCounter(lateCollision);
                bursting := bursting and (frameWaiting or transmitting)
            end
        end {inner loop}
    end {outer loop}
  end; {BitTransmitter}
```

MAC Self-Pacing Proposal --- Pascal Changes (cont.)

```
process Deference;
  begin
    if halfDuplex then cycle {half duplex loop}
      .....
      .....
    end {half duplex loop}
    else cycle {full duplex loop}
      while not transmitting do nothing;
      deferring := true;
      while transmitting do nothing;
      StartRealTimeDelay;
      while RealTimeDelay(interFrameSpacing) do nothing;
      while paceMode and (ifsExtensionSize > 0) do
        begin
          Wait (8);
          ifsExtensionSize := ifsExtensionSize - 1
        end;
      if frameWaiting then ifsExtensionCount := ifsExtensionCount mod ifsExtensionRatio
      else ifsExtensionCount := 0;
      deferring := false
    end {full duplex loop}
  end; {Deference}
```

Summary

- The open loop rate control achieves rate adaptation by extending the IPG between frames, controlled by the MAC
- This method for rate adaptation, as proposed, has the following advantages:
  - Simple
  - Cheap
  - Very precise
    - Worst case imprecision is less than 0.05751%
  - Independent of the PHY and the MAC/PHY interconnect
  - The self contained nature of this mechanism provides a robust solution