SpaceWire
RMAP checksum calculation

Torbjörn Hult

19 July 2005

This document or software is confidential to Saab Ericsson Space and may not:

a) be used for any purpose other than for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space.

Saab Ericsson Space

Checksum type?

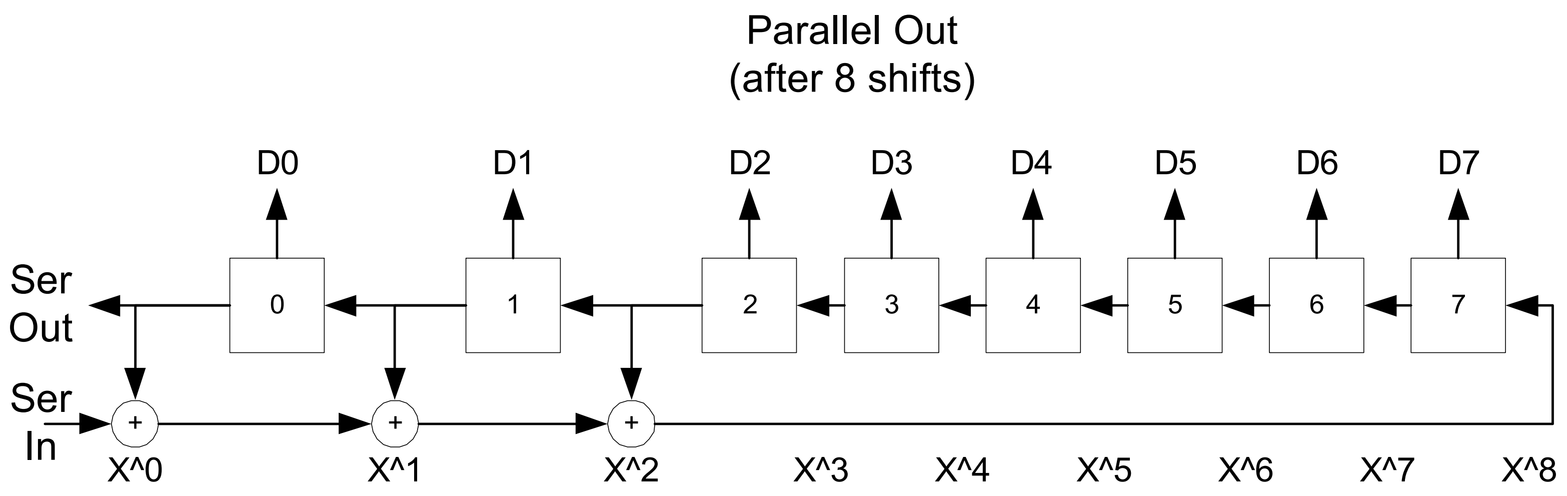
- CRC or longitudinal parity?
- Remember that SpaceWire already has byte parity!
- If CRC selected the algorithm <u>and</u> implementation must be specified
- Polynomial: $g(x) = x^8 + x^2 + x^1 + 1$

This document or software is confidential to Saab Ericsson Space and may not:

a) be used for any purpose other than for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space.

Saab Ericsson Space

Fibonacci implementation



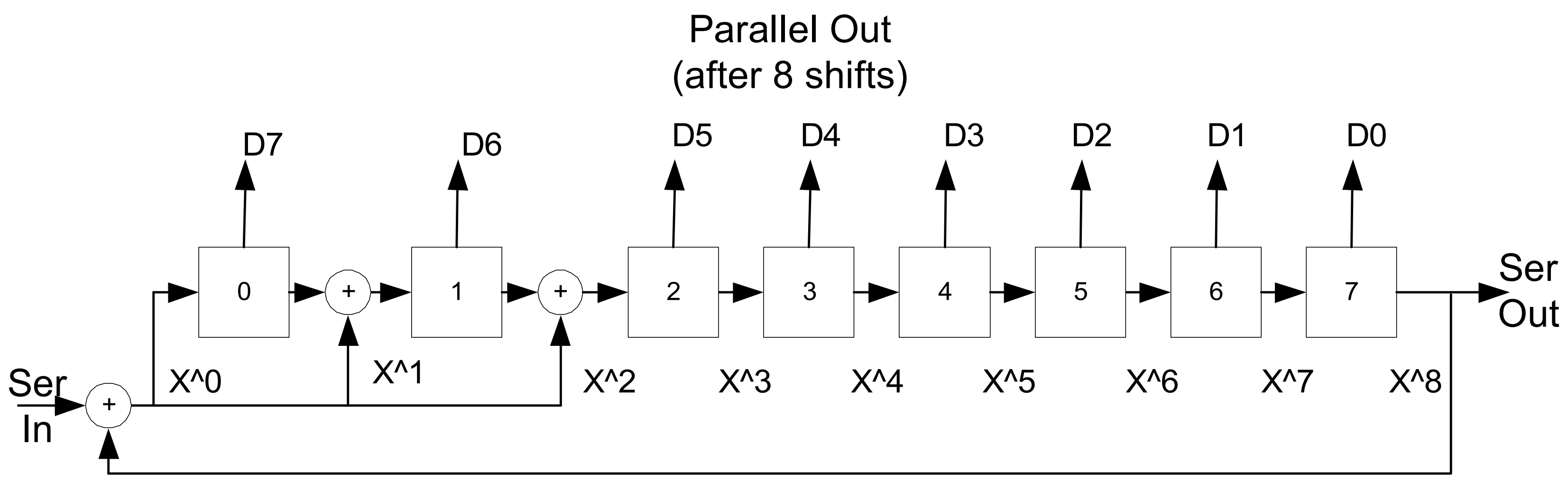
- Forward CRC

This document or software is confidential to Saab Ericsson Space and may not:

a) be used for any purpose other than for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space.

Saab Ericsson Space

Galois implementation



- Reverse CRC
- Produces zero result if a checksum is summed with itself
- Note the byte order numbering based on bit 0 (LSB) entered first

This document or software is confidential to Saab Ericsson Space and may not:

a) be used for any purpose other than for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space.

Saab Ericsson Space

Galois implementation, VHDL code

```
-------------------------------------------------------------
-- Purpose : Generate CRC checksum function
--
-- A is the input byte
-- StartValue is the accumulated CRC checksum
-------------------------------------------------------------
function CRC8(A : Byte_T; StartValue : Byte_T) return Byte_T is
   variable NextStart : Byte_T;
   variable CRCloop   : std_ulogic;
  begin
    NextStart := StartValue;

    for I in 0 to 7 loop -- For serial transfer with LSB first(SPW)
      CRCloop   := NextStart(0) xor A(I);
      NextStart := CRCloop &
                   NextStart(7) xor CRCloop &
                   NextStart(6) xor CRCloop &
                   NextStart(5 downto 1);
    end loop;
    return NextStart; -- CRC checksum
  end function;
```

This document or software is confidential to Saab Ericsson Space and may not:

a) be used for any purpose other than for which it was supplied;
b) be copied or reproduced in whole or in part without the prior written consent of Saab Ericsson Space;
c) be disclosed to any third party without the prior written consent of Saab Ericsson Space.

Saab Ericsson Space