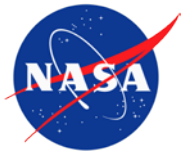


New Approaches for Direct Current (DC) Balanced SpaceWire

Session: SpaceWire Networks and Protocols, Short Paper

Prepared by:
Alex Kisin, AS&D, Inc. work performed for NASA GSFC
alexander.b.kisin@nasa.gov
and Glenn Rakow, NASA GSFC
glenn.p.rakow@nasa.gov

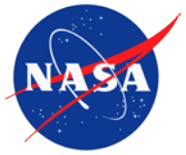
To be presented by Alex Kisin at the 7th International SpaceWire Conference, Yokohama, Japan, October 25-28, 2016.

List of Abbreviations and Acronyms

CMV	–	Common Mode Voltage
DC	–	Direct Current
DCF	–	Data Control Flag
EEP	–	Error End of Packet
ESC	–	Escape
EOP	–	End of Packet
FCT	–	Flag Control Character
FPGA	–	Field Programmable Gate Array
LFSR	–	Linear Feedback Shift Register
LVDS	–	Low Voltage Differential Signaling
PRS	–	Pseudo-Random Sequence
XOR	–	Exclusive OR

4b/5b, 4b/6b, 8b/10b, 8b/20b, 16b/30b – Different Data Encoding Schemes for DC Balancing

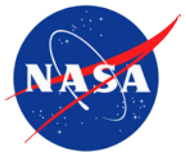
What is a DC Balanced Communication Line?



- **A DC balanced line is a communication line with "0" to "1" bits ratio over a certain time stretch equals to 1 (or very close to it)**
  - DC component of DC balanced line is 0V or very close to it
  - This will allow data to pass through capacitive and transformer isolation barriers

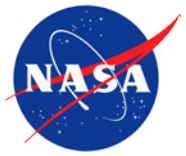
- **DC balanced lines allow using following types of SpaceWire interfaces:**
  - Classical SpaceWire interface with Data and Strobe lines
    - When using it – both Data and Strobe has to be DC balanced
      - This creates extra efforts to comply with balancing 2 lines
  - Reduced set of interface cable created by eliminating Strobe lines
    - Only Data lines has to be DC balanced
    - There are numerous DC balancing schemes available with different levels of implementation complexity

Advantages of Using DC Balanced SpaceWire



- **Greatly improved input common mode voltage (CMV) tolerance:**
  - State-of-the-art Low Voltage Differential Signaling (LVDS) receivers tolerate only +5/-4V
    - Some systems may require much higher tolerances
  - CMV is limited only by capacitor or transformer voltage ratings (can be in quite high) when using these isolation techniques
- **DC balanced lines allow using following types of SpaceWire interfaces:**
  - Classical SpaceWire interface with Data and Strobe
    - When using it – both Data and Strobe must be DC balanced
  - Reduced set of interface cables created by eliminating Strobe lines
    - Only Data lines has to be DC balanced
      - No special tricks to balance 2 lines at a time
    - There are numerous DC balancing schemes available with different levels of implementation complexity
    - Cable and on-board hardware are reduced to 2 full-duplex differential pairs instead of 4
    - Smaller diameter cable – better bending radius
    - Less area required for interface components on PWB

Disadvantages of Using DC Balanced SpaceWire



❖ Greater data overhead for interfaces with both Data and Strobe

- Balancing both Data and Strobe lines will require an increased amount of data traffic
  - Overhead amount greatly depends on used balancing methods
- Some characters must be changed to keep DC balance for Data and Strobe lines
  - Null and Flag Control Character (FCT) characters may take new encoded values in character level
  - End of Packet (EOP), Error End of Packet (EEP) and Time codes may keep their original values because of their rare occurrences

❖Easy clock extraction technique is gone for interfaces without Strobes

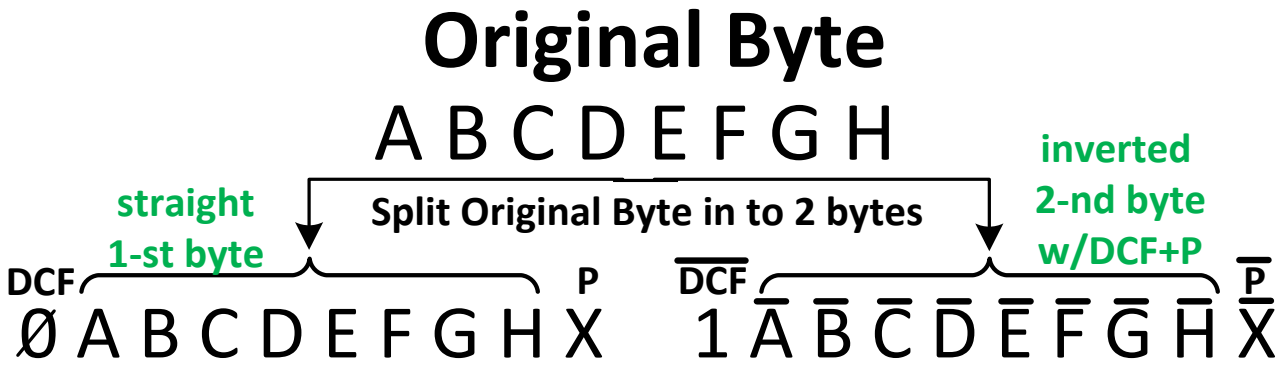
- No Strobe systems will require to extract clock from an incoming encoded Data
  - More complex extracting techniques are required

❖Easy on-a-fly data rate change is gone for interfaces without Strobes

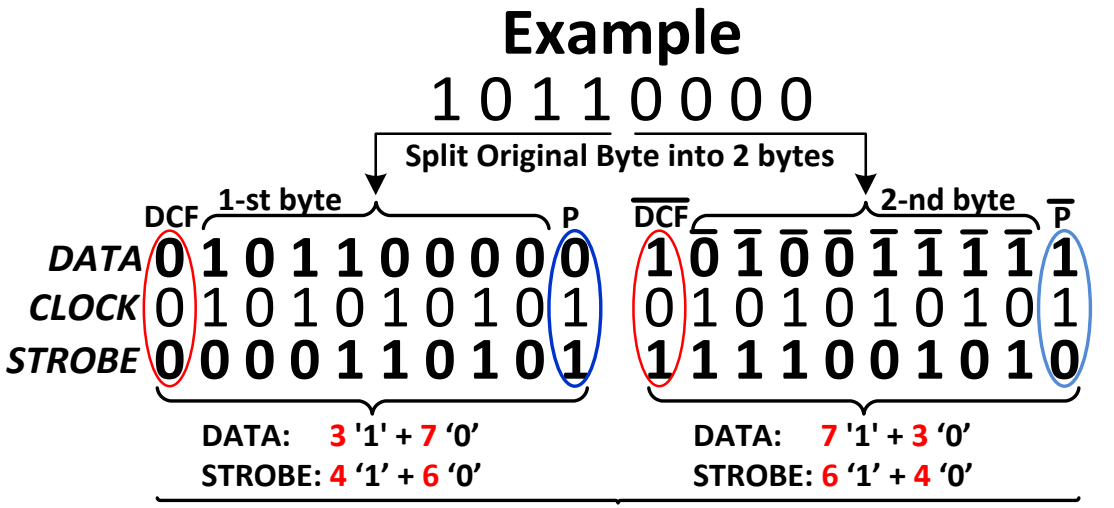
- System is supposed to know ahead of time at which rate it has to start
- On-a-fly data rate change will require changes in an initial handshake process
  - This will require change of protocols

DC Balanced SpaceWire with Data and Strobe: Dual Bytes Method

- **It is suggested to duplicate original Data byte with its inversed image**
  - 2 bytes will be sent instead of 1
  - Each byte will have its Data Control Flag (DCF) and Parity bit which covers DCF and byte itself
    - 2-nd inversed byte will have inversed DCF and inversed Parity to maintain perfect balance



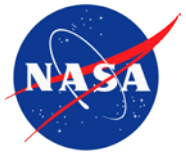
DC Balanced SpaceWire with Data and Strobe: Dual Bytes Method



But for 20 bits... DATA: 10 '1' + 10 '0'
STROBE: 10 '1' + 10 '0'
Good DC balance

- **It is obvious that Data will be 100% DC balanced, but how about Strobe?**
  - Strobe is balanced too:
    - Because on a stretch of 20 bits clock will always place its "0" or "1" under complementary bits
      - Their resulted Exclusive OR (XOR) products will be inversion of each other and also be complementary
      - See colored columns above

DC Balanced SpaceWire with Data and Strobe: Dual Bytes Method (Continued)



❖ Data and Strobe are balanced, but what to do with Control Characters?

- Control characters, except those used in handshake initialization, can either be:
  - Also inversely duplicated as described above (recommended), or
  - Used “as is”: they are so rare that small misbalance created by them will not be noticeable
- Control characters used in handshake initialization shall be both DC balanced for Data and Strobe:
  - Null can be send by an Originator for a long time when Responder is not ready
    - This might create significant DC bias on Strobe line because its Null image is misbalanced
  - Assign Null and FCT their new aliases when they are used in handshake protocol:
    - Substitute original Null character 01110100 with **10011100**
    - Its resulted Strobe image will be changed from 00100001 to a DC balanced **11001001**
    - Substitute original FCT character 0100 with **1100**
    - Its resulted Strobe image will be changed from 0001 to a DC balanced **1001**

DC Balanced SpaceWire with Data and Strobe: Dual Bytes Method (Continued)

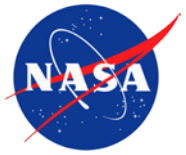
❖ What are Dual Bytes Method advantages?

- Very good DC balancing for Data and Strobe
- Easy implementation in FPGA
- "Single error correction" for an incoming data:
  - User can choose between 2 received bytes the one with valid parity
    - Only single bit errors can be detected, more errors may be masked
    - XOR both bytes to verify that there are no more than 1 bit error

❖ What are Dual Bytes Method disadvantages?

- Doubled data rate
  - Will limit SpaceWire theoretical highest rate to half of what is possible
- Using aliases for control characters to mitigate their misbalancing
  - Small handshake protocol changes are required

DC Balanced SpaceWire with Data and Strobe: Pseudo-Random Sequence (PRS) Method



❖ What is PRS Method?

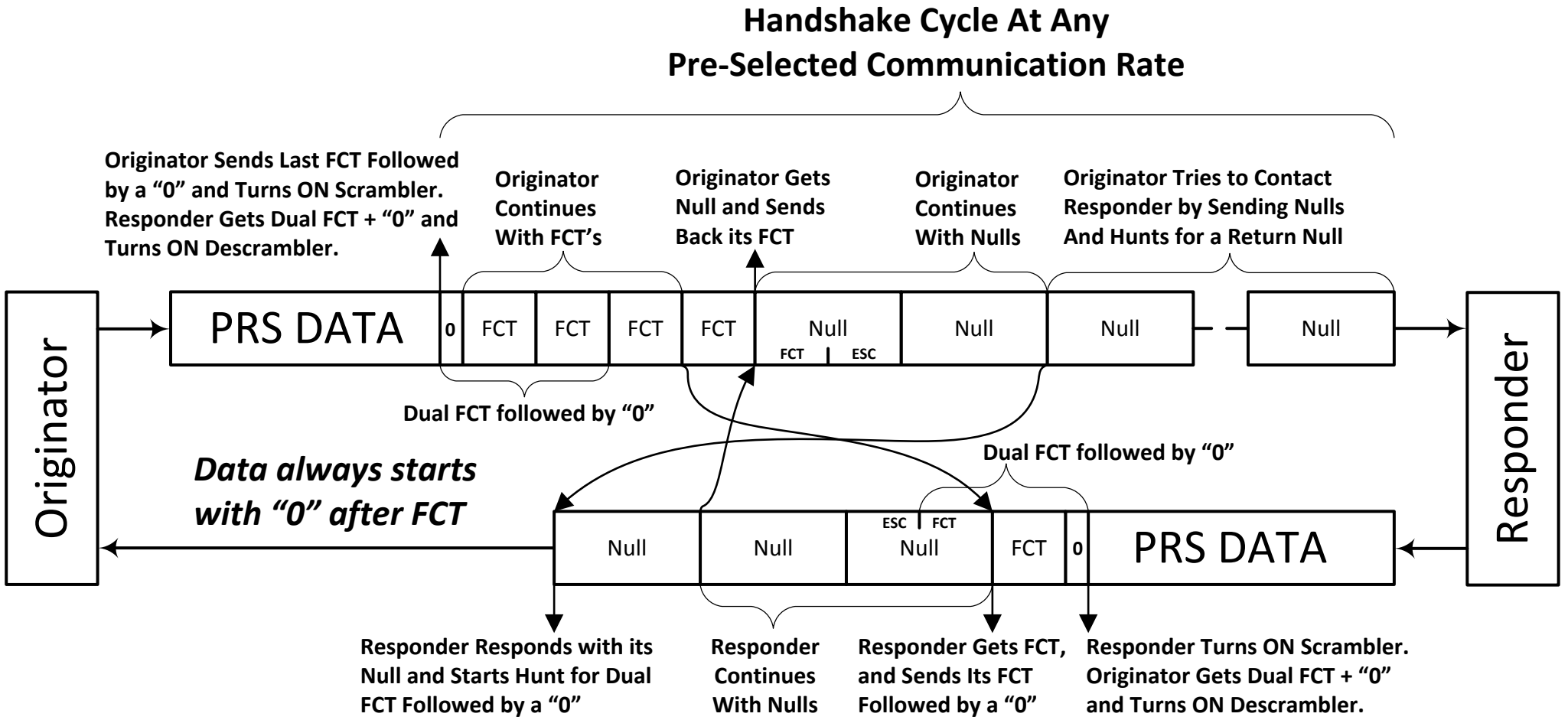
- The PRS method is intended to produce DC balanced data as a result of mixing random or pseudo-random bit sequence with real data stream
  - Mixing is done by the XOR function of both data streams on bit-by-bit basis
- Because it is difficult to produce a true random sequence – its pseudo-random version is used
  - PRS are usually generated by Linear Feedback Shift Registers (LFSR) (see backup slides)
- Resulted data stream is considered to be pseudo-random too (see backup slides)
  - All further XOR operations with a resulted PRS data stream will also produce PRS data stream

DC Balanced SpaceWire with Data and Strobe: Pseudo-Random Sequence (PRS) Method (Continued)

❖ PRS Method Implementation

- It is suggested to start handshaking sequence with a disabled PRS generator
  - It will be difficult for an ad-hoc Responder receiver to sync to PRS generated by an Originator
    - Responder will not know what is a current state of PRS
  - Use the same Null and FCT characters balanced aliases as in above described Dual Byte method
- PRS can be enabled after both Originator and Responder exchanged with their Nulls and FCTs
- However, there are some requirements:
  - Disabled PRS startup seed shall generate “0” at its 1-st bit to be mixed with data (see backup slides)
    - Because both Null and FCT start with “1” – first “0” will indicate start of data cargo
  - PRS will be enabled immediately after start of data cargo is detected
  - No changes or aliasing from now on are required for an original Control characters and Time codes
  - Both Originator and Responder will keep tracking of both RX and TX PRS streams

DC Balanced SpaceWire with Data and Strobe: Pseudo-Random Sequence (PRS) Method (Continued)



DC Balanced SpaceWire with Data and Strobe: Pseudo Random Sequence Method (Continued)

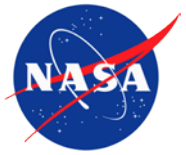
❖ What are PRS Method advantages?

- Good DC balancing for Data and Strobe
- No data rate overhead: exactly same rate as an original data
- No changes for DCF and Parity bits are required
- No substitutes and aliases for control characters, except during handshaking

❖ What are PRS Method disadvantages?

- Longer stretches of data to be consider balanced
- Moderate to complex implementation in FPGA
- PRS patterns matched with a same or inverted data patterns are possible
  - Probability of this is extremely low
  - May require LVDS receivers with wider CMV tolerances for DC offset mitigation
- Handshake protocol changes are required

DC Balanced SpaceWire with Data Only: Dual Nibbles 4b/6b Coding

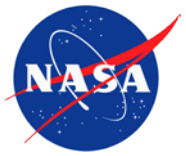


- **It is suggested to substitute two of Data byte nibbles with two 6-bit symbols:**
  - Each symbol will contain only 3 “0” and 3 ”1”
    - Number of permutations (P) for 3 “1” bits in 6-bit symbol for 64 symbols group is:

      P(3) = 6!/(3! x (6-3)!) = 20

      - It means that for each of 16 nibble’s combinations there is an unique 6-bit DC balanced symbol
  - There is no need anymore for Data Control Flag and Parity bit for this DC balancing method:
    - 4 remained symbols can be used as unique aliases for the original Control Characters
    - Data integrity can be checked by calculating number of “1” of a received 6-bit symbol: always 3
  - Overhead redundancy for this method (compared with the original SpaceWire) is only 20%:
    - Higher than for 4b/5b or 8b/10b methods (where overhead is 0%), but is much easier to implement
    - Lower than most of other Data Only methods
  - Requires lookup table of only 20 x 6 = 120 bits for transmitter, and 64 x 6 = 384 bits for receiver
    - If symbol doesn’t match table – it is an error
  - However, Strobe line can't be used because it is not DC balanced

DC Balanced SpaceWire with Data Only: Dual Nibbles 4b/6b Coding (Continued)

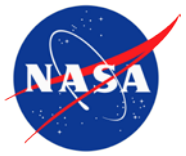


❖ **Suggested symbols assignment for each 16 nibble combinations and 4 control characters:**

Data+Cont	MSB	<--6-bit Symbol-->				LSB
0	0	1	0	1	1	0
1	1	0	0	1	0	1
2	1	0	0	1	1	0
3	1	0	0	0	1	1
4	1	0	1	1	0	0
5	0	0	1	1	0	1
6	0	0	1	1	1	0
7	0	0	1	0	1	1
8	1	1	0	1	0	0
9	1	1	0	0	0	1

Data+Cont	MSB	<--6-bit Symbol-->				LSB
A	1	1	0	0	1	0
B	0	1	0	0	1	1
C	0	1	1	1	0	0
D	0	1	1	0	0	1
E	0	1	1	0	1	0
F	1	0	1	0	0	1
EEP	1	1	1	0	0	0
EOP	0	0	0	1	1	1
FCT	1	0	1	0	1	0
ESC[ape]	0	1	0	1	0	1

DC Balanced SpaceWire with Data Only:
4b/6b, Dual Bytes and PRS w/ Fixed Data Rates Methods



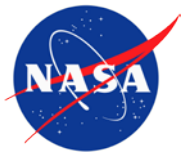
❖ What are Data Only w/ Fixed rates advantages?

- Only Data line balancing is required
- Reduced amount of cable
  - More space and less weight
  - Cheaper cables
- Sharper cable bend radius
  - A big plus for small satellites
- Less electronic hardware on-board:
  - Less power needed

❖ What are Data Only w/ Fixed rates disadvantages?

- More complex methods of clock synchronization (see paper for suggestions)
- Longer stretches of data to be consider balanced
- More complex FPGA implementations
- Only single frequency data rate
  - No on-a-fly data rates changes

DC Balanced SpaceWire with Data Only:
4b/6b, Dual Bytes and PRS w/ Variable Data Rates Methods



❖ What are Data Only w/ Variable rates advantages?

- Same advantages as for Fixed data rates
- Variable rates are possible:
  - Use double handshake initialization
  - After initial low rate handshake is established:
    - Originator and Responder "tell" each other about new desired rate
    - Responder breaks connection for no less then 6.4us
    - Both Responder and Originator adjust their clock generators
    - Responder starts new handshake with a new rate

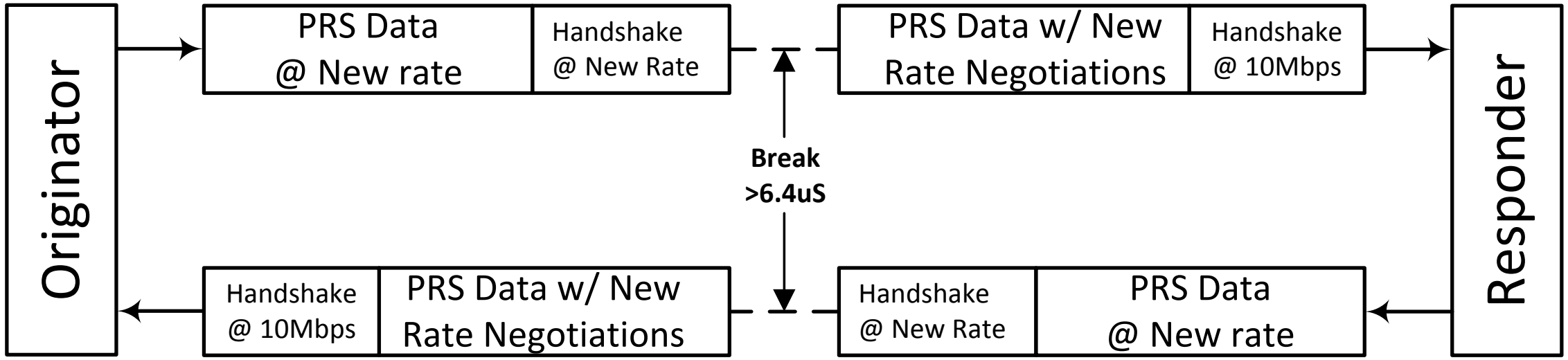
❖ What are Data Only w/ Variable rates disadvantages?

- Same disadvantages as for Fixed data rates
  - Except rate changes

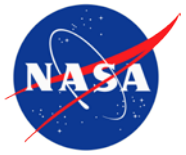
DC Balanced SpaceWire with Data Only:

4b/6b, Dual Bytes and PRS w/ Variable Data Rates Methods (Continued)

Handshake Cycles At Variable Communication Rates

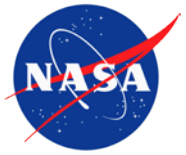


DC Balanced SpaceWire Methods: Short Summary



Method	Description	Lines	RX Clock	Data Rate	Overhead vs. SpW	DC Balance	DC Quality	FPGA Implementation
1	Standard SpaceWire	Data and Strobe	XOR-ed from D and S	Variable: as in original SpaceWire	0%	No	Terrible	Existing
2	Dual bytes (DB) encoding				100%	Yes	Very good	Easy
3	8b/20b				100%	Yes	Good	Moderate-Complex
4	16b/30b				50%	Yes	Good	Moderate-Complex
5	2 lines PRS modulation				0%	Yes	Good	Moderate
6	Fixed DB or PRS modulation	Data only	4-phase sampling, or others	Fixed: rate change requires dual handshaking	Same as in above D&S	Yes	Good	Moderate
7	Variable DB or PRS modulation					Yes	Good	Moderate
8	8b/10b or dual 4b/5b				0%	Yes	Very good	Moderate-Complex
9	Manchester modulation				100%	Yes	Excellent	Easy-Moderate
10	Dual nibble 4b/6b				20%	Yes	Very good	Easy-Moderate

Backup Slides



LFSR Polynomial Generators for PRS

Here is a table of maximum-cycle Linear Feedback Shift Register (LFSR) taps. The bit numbering starts from $n \ldots 1$ with n being the input bit and 1 the output bit. Figure 1 shows an 8-stage maximum-cycle LFSR. LFSR-2 refers to two tap LFSRs, LFSR-4 to LFSRs. Blanks indicate no solution exists.

Table 1: Shift Registers with Cycle Size $2^n - 1$

n	LFSR-2	LFSR-4	n	LFSR-2	LFSR-4	n	LFSR-2	LFSR-4
2	2, 1		24		24, 23, 21, 20	46		46, 40, 39, 38
3	3, 2		25	25, 22	25, 24, 23, 22	47	47, 42	47, 46, 43, 42
4	4, 3		26		26, 25, 24, 20	48		48, 44, 41, 39
5	5, 3	5, 4, 3, 2	27		27, 26, 25, 22	49	49, 40	49, 45, 44, 43
6	6, 5	6, 5, 3, 2	28	28, 25	28, 27, 24, 22	50		50, 48, 47, 46
7	7, 6	7, 6, 5, 4	29	29, 27	29, 28, 27, 25	51		51, 50, 48, 45
8		8, 6, 5, 4	30		30, 29, 26, 24	52	52, 49	52, 51, 49, 46
9	9, 5	9, 8, 6, 5	31	31, 28	31, 30, 29, 28	53		53, 52, 51, 47
10	10, 7	10, 9, 7, 6	32		32, 30, 26, 25	54		54, 51, 48, 46
11	11, 9	11, 10, 9, 7	33	33, 20	33, 32, 29, 27	55	55, 31	55, 54, 53, 49
12		12, 11, 8, 6	34		34, 31, 30, 26	56		56, 54, 52, 49
13		13, 12, 10, 9	35	35, 33	35, 34, 28, 27	57	57, 50	57, 55, 54, 52
14		14, 13, 11, 9	36	36, 25	36, 35, 29, 28	58	58, 39	58, 57, 53, 52
15	15, 14	15, 14, 13, 11	37		37, 36, 33, 31	59		59, 57, 55, 52
16		16, 14, 13, 11	38		38, 37, 33, 32	60	60, 59	60, 58, 56, 55
17	17, 14	17, 16, 15, 14	39	39, 35	39, 38, 35, 32	61		61, 60, 59, 56
18	18, 11	18, 17, 16, 13	40		40, 37, 36, 35	62		62, 59, 57, 56
19		19, 18, 17, 14	41	41, 38	41, 40, 39, 38	63	63, 62	63, 62, 59, 58
20	20, 17	20, 19, 16, 14	42		42, 40, 37, 35	64		64, 63, 61, 60
21	21, 19	21, 20, 19, 16	43		43, 42, 38, 37	65	65, 47	65, 64, 62, 61
22	22, 21	22, 19, 18, 17	44		44, 42, 39, 38	66		66, 60, 58, 57
23	23, 18	23, 22, 20, 18	45		45, 44, 42, 41	67		67, 66, 65, 62

LFSR Hardware Implementation for PRS

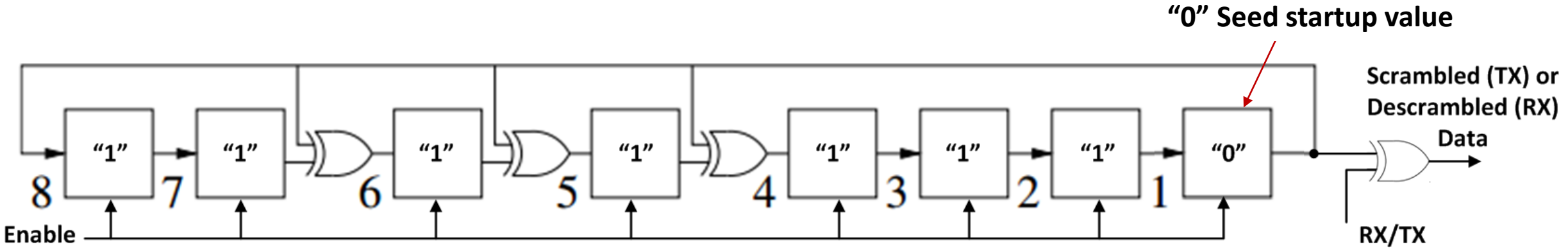


Figure 1: An 8-stage Galois LFSR with cycle size 255. This LFSR has taps at positions 8,6,5 and 4.

- **Requirements to LFSR generator:**
  - Start seed shall have a "0" at Data XOR mixer (shown above) when LFSR is disabled
  - All "0" seeds are prohibited
  - All "1" seeds should be skipped to improve DC balance
    - Detect LFSR state prior to all "1" state
    - On a next clock cycle upload LFSR with state that follows after all "1"
      - This will skip unwanted all "1" state

Proof of Producing New PRS When Mixing PRS w/Data

LEMMA

(by Dr. A.V. Marchenko)

Let A be an array of N bits containing the message and R be a random bit array of the same length with individual bits being independent and taking values 0 and 1 with probability of ½. Then a bitwise XOR operation (denoted by ^) applied to A and R will result in an array X = A^R = R^A which is random and distributed the same way as R (i.e. its individual bits are independent and take values 0 and 1 with a same probability of ½).

PROOF

Consider arbitrary bit Xk = Ak ^ Rk = (Ak + Rk) mod (2). Resulted conditional probabilities for various Ak and Rk are:

P{Xk = 0| Ak = 0} = P{Rk = 0} = ½
P{Xk = 0| Ak = 1} = P{Rk = 1} = ½
P{Xk = 1| Ak = 0} = P{Rk = 1} = ½
P{Xk = 1| Ak = 1} = P{Rk = 1} = ½

Therefore, resulted probabilities of P{Xk = 0} = P{Xk = 1} = ½.

Consider now arbitrary but different 2 bits Xm and Xn. We have:

P{Xm = 0, Xn = 0| Am = 0, An = 0} = P{Rm = 0, Rn = 0} = P{Rm = 0}* P{ Rn = 0}= ¼ due to the independence of Rm and Rn.

Repeating this calculation 15 more times for all possible combinations of values of Xm, Xn, Am, and An, we can find that all the mutual probabilities P{Xm = U, Xn = V} with U,V = 0 or 1 are equal to ¼. So, for any combination of U and V we have:

P{Xm = U, Xn = V} = ¼ = ½ * ½ = P{Xm = U} * P{Xn = V}.

CONCLUSIONS

Therefore, Xm and Xn are independent and a whole resulted bit array X is random. All further XOR mixing of random bit array X with bit arrays similar to A will also produce random bit arrays X'.

In practice pseudorandom array PR does not strictly satisfies these conditions, so the result is valid to the extent of PR being close to random vector of independent bits taking values 0 and 1 with probability ½. The longer generated PR sequence – the closer it approaches to R. LFSR registers can be good example of PR generators.

HISTORY

The above concept of mixing together general data with random data and still get random data was first suggested by Frank Miller in 1882 and "re-invented" and patented in 1917 by Gilbert S. Vernam as …. a secure interceptible spy communications method, aka "One Time Pad". Refer to: https://en.wikipedia.org/wiki/One-time_pad