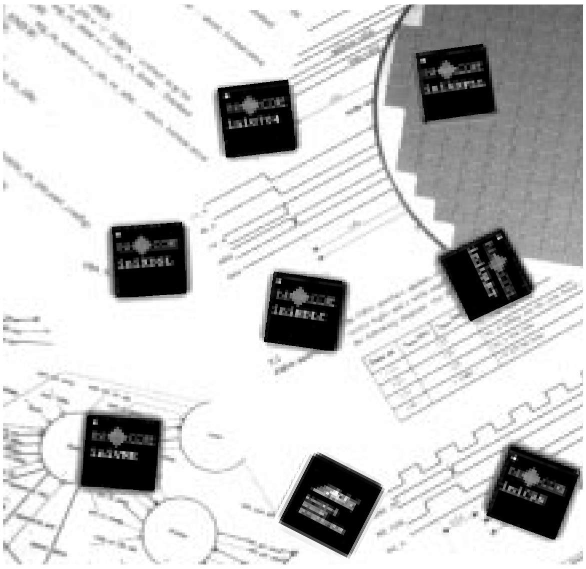
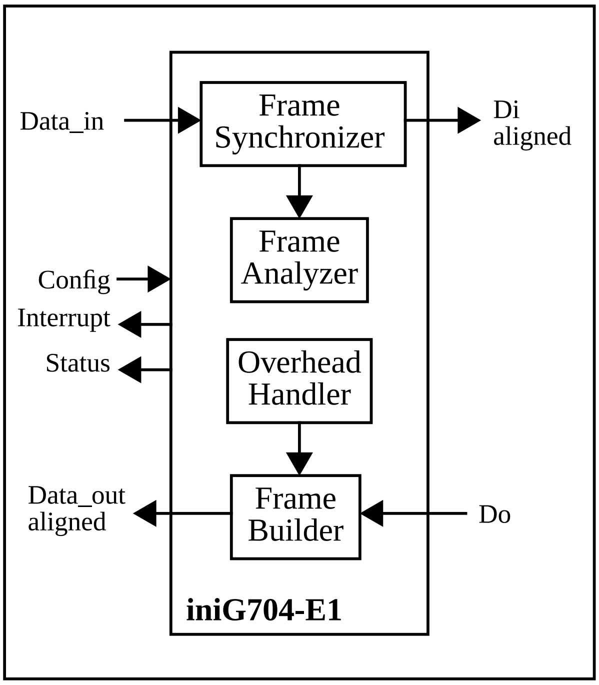
iniG704-E1 data sheet



Features:

- Asymmetric Application (e.g. E1 - ATM, nx64 - E1, E1 - 30 BRI-Channels)
- Symmetric Application (e.g. E1-E1)
- Multi G704 on Chip (e.g E1-E2)
- Basic and Multiframe Alignment
- Alarm Bit Processing
- Customizable Error Counters
- Selectable Conditions for Loss of Sync
- CRC4 Error Checking and Monitoring
- Structural VHDL RTL Description
- Certified Core by the Standard Verification Bureau
- Gate Count ~7k (Main Building Blocks)
- Sample FPGAs Available

Structure of iniG704-E1:



INICORE - the reliable Core and System Provider. We provide high quality IP, design expertise and leading edge silicon to the industry.

The **iniG704-E1** is a flexible implementation of INICORE's G704 E1-Framer function. This flexibility is due to the structural partition of the core into the various building blocks. Thanks to INICORE's methodology, the core remains same in different applications. Application specific blocks (e.g. InPort, OutPort, Interrupt Controller, Microprocessor interface) don't influence the main core and its function is still guaranteed.

INICORE offers the structural VHDL G704 simulation/synthesis model for the target technology of your choice.

INICORE's strategy is not to compete with the standard chip manufacturers, but to use the ASIC technologies for the 'system on chip' design, which demands standards like G704 with highly specific data processing capabilities. Due to its flexible interfaces, the iniG704-E1 can be combined easily with innovative architectures that lead to new competitive products.

To date, INICORE has delivered G704 cores to several telecommunication companies. The iniG704-E1 is certified by the standard verification bureau.

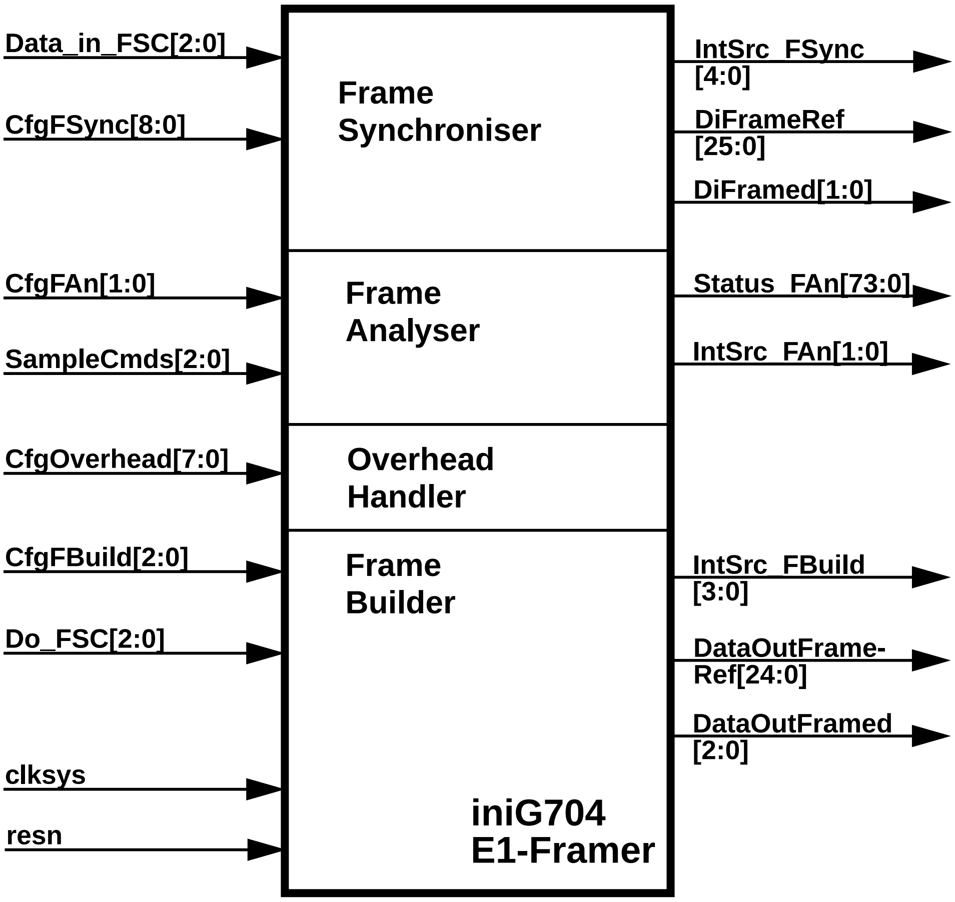
US Sales Office:
INICORE INC.
5600 Mowry School Road, Suite 180,
Newark, CA 94560
Tel: 510 445 1529 Fax: 510 656 0995
E-mail: ask_us@inicore.com
Web: www.inicore.com

INICORE AG
Mattenstrasse 6a,CH-2555 Brügg, Switzerland
Tel: ++41 32 374 32 00, Fax: ++41 32 374 32 01
E-mail: ask_us@inicore.ch
Web: www.inicore.ch

iniG704-E1: The Universal G704 E1-Framer Core

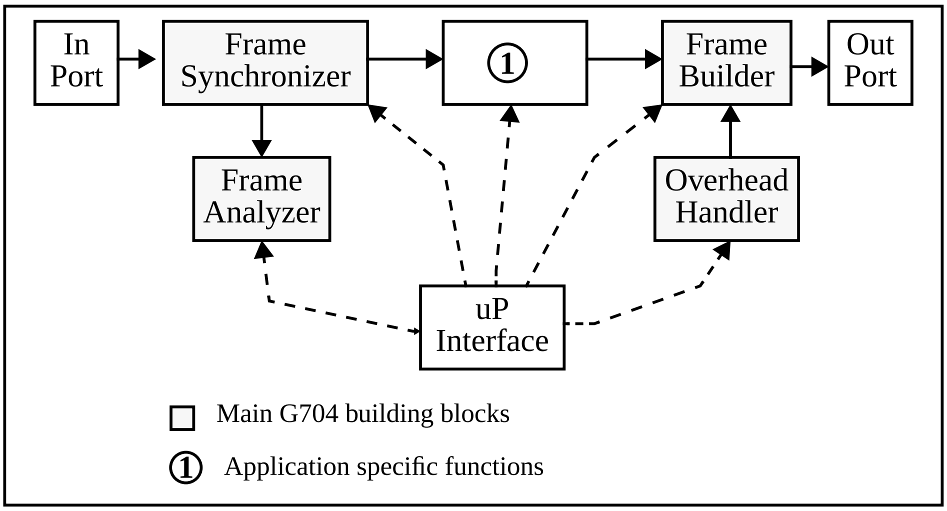
1 Overview

This is a short description of the g704 E1-Framer core. In the following, the term G704 refers to framing at 2048 kbit/s (E1) only.
The following picture shows all inputs and outputs.



2 Structure

INICORE’s iniG704-E1 core is partitioned in four main building blocks. Application specific functions can be added as necessary. An example is shown bellow.



The main building blocks of the g704 E1-Framer are:

- Frame Synchroniser
- Frame Analyser
- Overhead Handler
- Frame Builder

Further example applications will be found on page 9.

3 IO description

The following part lists the input and output ports of the INICORE G704 E1-Framer core and gives a short overview of its functionality.

3.1 General inputs

These pins are use to clock and initialize the whole iniG704 core . There are no other clocks in this core. The recommended minimal clock frequency is 4.048 MHz.

pin name	type	description
ClkSys	in	System clock
resn	in	asynchronous system reset, active low

3.2 Configuration

The configuration pins are used to select the operating mode. They are static.

Table 1: Configuration of Frame Synchroniser

pin name	type	description
CfgFSync .SyncMode[1:0]	in	Configuration of Frame Synchroniser “00”: transparent (no FSync generated) “01”: free run (generate dummy FSync) “10”: use external FSync “11”: fully automatic sync (G.706)
CfgFSync .ForceResync	in	User controlled resync: When toggled form ‘0’ to ‘1’, a resync is initialised
CfgFSync .AutoResync	in	Automatic resync after loss of sync: ‘1’: Automatic resync ON (‘0’: OFF)

Table 1: Configuration of Frame Synchroniser

pin name	type	description
CfgFSync .FAImprove_411	in	‘0’: Improved BasicFrame alignment desabled ‘1’: Use improved BasicFrame alignment procedure as in $4.1.1 of G.704/Note 1 (check FA bit 2 of nFAS frames)
CfgFSync .MF_Mode	in	‘0’: MultiFrame alignment search disabled (only Basic-Frame search) ‘1’: MultiFrame alignment search enabled
CfgFSync .MF_SyncMode	in	‘0’: use parallel BFA search (G.706) ‘1’: reuse primary BFA search (PTT simplified search path)
CfgFSync .MFA_Check	in	‘0’: MultiFrame alignment loss checking process disabled ‘1’: MultiFrame alignment checking process enabled (if 3 consecutive MFA not found while MFSyncState = InSync, then MFSyncState <= Hunt)
CfgFSync .CRC4_Mode	in	‘0’: CRC4 Error limit of <= 915 disabled ‘1’: CRC4 Error limit checking enabled

Table 2: Configuration of Frame Analyser

pin name	type	description
CfgFan .CRC_CountEbl	in	‘0’: CRC4 Error counter enabled ‘1’: CRC4 Error counter disabled
CfgFan .FAS_CountEbl	in	‘0’: FAS Error counter enabled ‘1’: FAS Error counter disabled
CfgFan .E_CountEbl	in	‘0’: E-Bit counter enabled ‘1’: E-Bit counter disabled

Table 3: Configuration of Overhead Handler

pin name	type	description
CfgOverhead .Si_E1	in	E1 / Si of FAS frame
CfgOverhead .Si_E2	in	E2 / Si of non FAS frame
CfgOverhead .A-Bit	in	A-Bit
CfgOverhead .InsertEbl[4:0]	in	Insert enable pattern related to Sa bits
CfgOverhead .saBitsMF[1)] [1:8][4:0]	in	Sa4...Sa8 bits Note: When FrameRef.MF.MFSyncState NOT = InSync, then only SaBitsMF(1) will be inserted.

[1)] SaBits of 8 DoubleFrames of a MultiFrame

Table 4: Configuration of Frame Builder

pin name	type	description
CfgFBuild .CRC4_MFMode	in	‘0’: CRC4 MultiFrame Mode disabled ‘1’: CRC4 MultiFrame Mode enabled

Table 4: Configuration of Frame Builder

pin name	type	description
CfgFBuild .BuildMode[1:0]	in	“0x”: transparent “10”: synchronise BFA phase “11”: generate BFA phase (Note: when NOT transparent, a new MultiFrame is generated

3.3 Interrupt sources

For event communication several interrupts are used. An interrupt is active ‘1’ for only one clock cycle.

The abbreviation used in the table below are:

BFA	Basic Frame Alignment
MFA	Multi Frame Alignment

Table 5: Interrupt sources Frame Synchroniser

pin name	type	description
IntSrc_FSync .BFA.Sync	out	Pulse @ ‘1’ when entering state InSync
IntSrc_FSync .BFA.SyncLoss	out	Pulse @ ‘1’ when leaving state InSync
IntSrc_FSync .MFA.SyncEntry	out	Pulse @ ‘1’ when entering state InSync
IntSrc_FSync .MFA.MFSync	out	Pulse @ ‘1’ at begin of each MultiFrame
IntSrc_FSync .MFA.SyncLoss	out	Pulse @ ‘1’ when leaving state InSync

Table 6: Interrupt sources Frame Analyser

pin name	type	description
IntSrc_FAn .AlarmByteCaptured	out	Pulse @ ‘1’ when 8 A-bits have been captured (only when BFSyncState = InSync and MFSyncState /= InSync)
IntSrc_FAn .A_Bit_detected	out	Pulse @ ‘1’ when A-Bit sequence “001” has been received

Table 7: Interrupt sources Frame Builder

pin name	type	description
IntSrc_FSync .BFB.Sync	out	Pulse @ ‘1’ when entering state InSync
IntSrc_FSync .BFB.SyncLoss	out	Pulse @ ‘1’ when leaving state InSync
IntSrc_FSync .MFB.SyncEntry	out	Pulse @ ‘1’ when entering state InSync
IntSrc_FSync .MFB.MFSync	out	Pulse @ ‘1’ at begin of each MultiFrame

3.4 Status indication and sample command

Sample-command logic:
The readout of performance counters poses always a technical problem. The counter value should be held stable during the microprocessor access and it should automatically be reset after readout. Both is not easy with an asynchronous microprocessor interface and events occurring by chance exactly during the microprocessor access may be lost.
There is a safe and simple INICORE-standard solution: the ‘SampleCmd’:

1. Before reading the monitor value, the microprocessor must write to the related ‘read-only’ address (data is don’t care). This generates an internal ‘SampleCmd’.
2. When the SampleCmd occurs, the monitor value is synchronously copied into the readout register and in the same clock-cycle, the counter is restarted.
3. Now the microprocessor can read the value. The value is stable by design and it is impossible to miss an event.

Table 8: Status of Frame Analyser

pin name	type	description
Status_FAn .LOS_2M	out	2MBit Loss detection
Status_FAn .AIS	out	AIS detection result
Status_FAn .TS0Data	out	TS0 Data, OTHERS => ‘0’ when not InSync
Status_FAn .AlarmHistory	out	Sampled history of last 8 A-bits
Status_FAn.Error CountCRC [7:0]	out	Sampled state of CRC error counter
Status_FAn.Error CountFAS [7:0]	out	Sampled state of FAS/nFAS error counter
Status_FAn.Error CountE [7:0]	out	Sampled state of E-Bit counter

Table 9: Sample Commands of Frame Analyser

pin name	type	description
SampleCmds. SampleCRC	in	Active ‘1’ during one clock cycle, when up write access has been detected to the read-only port of the related counter
SampleCmds. SampleFAS	in	Active ‘1’ during one clock cycle, when up write access has been detected to the read-only port of the related counter
SampleCmds. SampleE	in	Active ‘1’ during one clock cycle, when up write access has been detected to the read-only port of the related counter

3.5 Frame Synchroniser

This module contains the synchronization circuits. The resulting frame reference pointer (DiFrameRef) identifies every bit in a frame and multiframe. This allows easy access to the needed data bits or for example to the TS16-Datas.

Table 10: I/O’s Frame Synchroniser

pin name	type	description
Data_in_FSC .Data	in	Binary NRZ data
Data_in_FSC .DataEbl	in	Data enable

Table 10: I/O’s Frame Synchroniser

pin name	type	description
Data_in_FSC .FSync	in	(FSC) Frame Sync pulse
DiFramed. Data	out	Binary NRZ Data, synchronized to local clock domain
DiFramed. DataEbl	out	Data enable
DiFrameRef [24:0]	out	Frame reference pointer

3.6 Frame Builder

This module is able to build a new frame and multiframe structure, including CRC4 calculation and insertion.This module always needs a basic frame sync pulse (FSC) at its inputs, since it does not in corporate a FAS synchroniser function.

Table 11: I/O’s Frame Builder

pin name	type	description
Do_FSC .Data	in	Binary NRZ data
Do_FSC .DataEbl	in	Data enable
Do_FSC .FSync	in	(FSC) Frame Sync pulse
DataOut_Framed .Data	out	Binary NRZ Data, synchronized to local clock domain
DataOut_Framed .DataEbl	out	Data enable
DataOut_Frame-Ref[24:0]	out	Frame reference pointer

4 Main Building Blocks

The following paragraph describes the functionality of the main building blocks of the G704 E1-Framer.

4.1 Frame Synchroniser

This module provides event outputs, which signal events like loss of sync, CRC error, etc. Following functions are part of the Frame Builder:

- basic frame alignment
- multi frame alignment
- parallel basic frame alignment for the case, that no multiframe can be found at the primary basic frame alignment phase. (If required, parallel frame alignment can be disabled. Then the function is compatible with older circuits present in existing installations, which do not contain the parallel alignment function.)
- CRC4 calculator
- CRC4 history for checking, if more than 915 CRC errors have occurred within the last 1000 multiframes. This is a running statistic such that detection time of CRC failure does not exceed 1 second independent of the point in time, where CRC errors start, and independent of which CRC error rate was present before.

4.2 Frame Analyser

This module accesses the overhead bit in the first octet (A, E, Sa_4-Sa_8). It includes detection algorithms for the A-bit and counters for error events. Further, microprocessor interrupt signals are provided too.

- A-Bit processing
  The detection of the alarm bit is complex, because there are many different standards (e.g. majority of three, hysteresis of 8, etc.). The recommended solution is to provide a history of the last 8 alarmbits (either aligned to the multiframe interval or to an arbitrary 16-frame interval). Based on this, the A-bit detection can be implemented in software.
- Error Counters
  The following counters are implemented:
  - E-Bit
  - CRC-4 error
  - Syncword
  Each of these uses a 8-bit counter with a separate enable flag.
- Signalling CRC error to Frame Overhead generator
  Some applications demand the automatic insertion of an E-bit on the Frame Builder side for each CRC error detected by the Synchroniser. In this case, the Analyser produces a 2-bit signal, which advances in a gray code sequence each time, a CRC error has been detected. This way, the error event can easily cross the clock domain boundary to the Frame Builder side and even large phase-shifts between Synchroniser and Builder can not cause the loss of a CRC error event.
- Si and Sai bit termination
  The received A, E and $Sa_{(4\text{-}8)}$ bits are stored in a 7x8 bit register file with a shadow readout register for a multiframe synchronous access to the overhead bits.

4.3 Overhead Handler

This process contains the required microprocessor interface circuits for the insertion of the overhead bits. The microprocessor interface itself is not a part of this process. The information is read from the buffers by means of the frame reference delivered by the Frame Builder.

- Si and Sai bit insertion
  A 7x8 bit register file with shadow readout register for a multiframe synchronous is used to access to the overhead bits.
  A 8-bit mask is used in order to selectively enable the insertion of A, E and

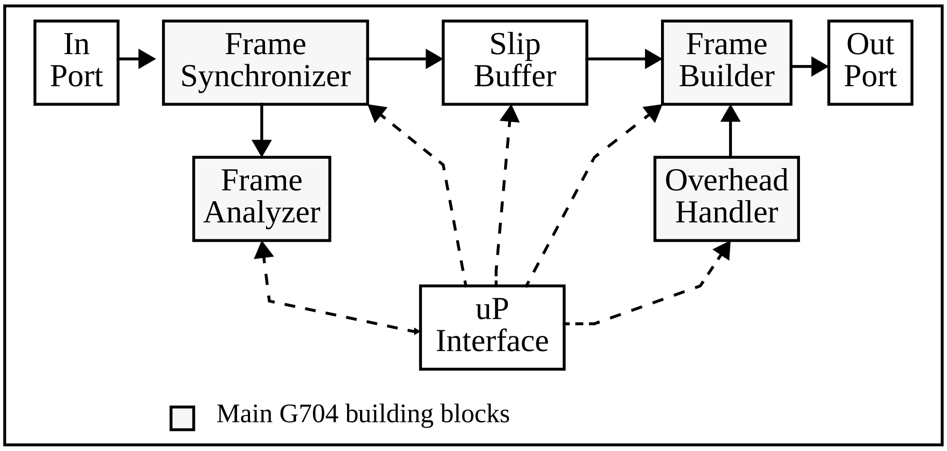
Sai bits. In this way, for each bit there is a choice to insert it under software control or pass it through unchanged.
Note: FAS and nFAS bit patterns are always inserted. This way, upstream bit errors on the FAS/nFAS pattern are corrected by the frame builder.

4.4 Frame Builder

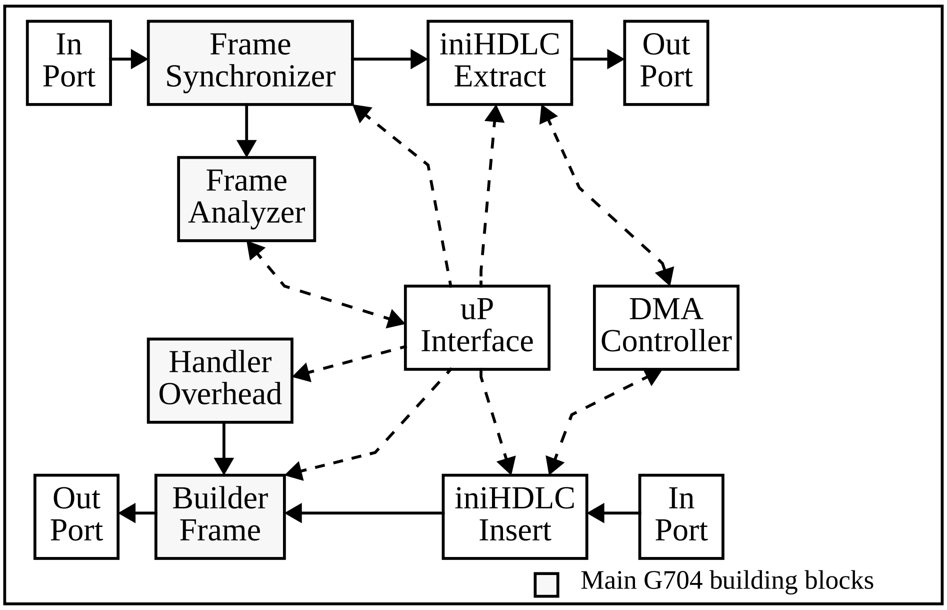
This module is able to build a new frame and multiframe structure, including CRC4 calculation and insertion. It allows the replacement of individual overhead bits. It may be set to a transparent mode. Rebuilding is required, if a slip buffer inserts or removes frames, or if the state of the overhead bits (E, A, Sai) have been changed.
This module provides a frame reference output, which identifies every bit in a frame and multiframe.
On the input side, this module always needs a basic frame sync pulse (FCS), since it does not incorporate a FAS synchroniser function. However, by configuration it can be forced to align to the FAS/nFAS phase of the incoming frames (This only works, if there is no upstream slip-buffer, because a slip-buffer disturbs the FAS/nFAS sequence). When the Frame Builder is aligned to the FAS/nFAS phase, some overhead bits may transparently pass through the Frame Builder. This might be useful for passing the A-bit through.
Warning: Passing through Sai bits in multiframe mode does not help, if the Sai bit link uses a protocol, which is aligned to the multiframe sequence, because the Frame Builder creates its own new multiframe.

5 Example Applications

Using a Slip Buffer within a G704-Framer application

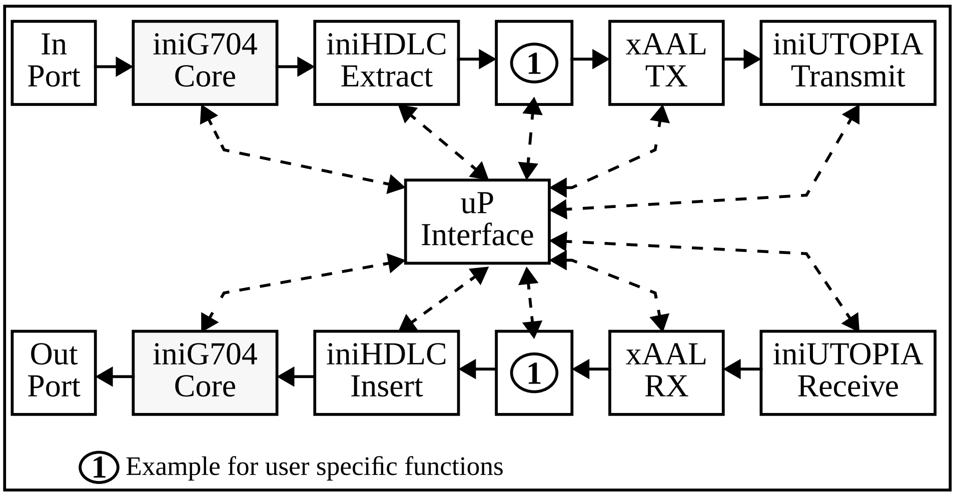


G704-Framer and HDLC Data Link Controller (ISDN Terminator)



iniHDLC-Extract/Insert are two core functions which are also available from INICORE.

G704-Framer and ...

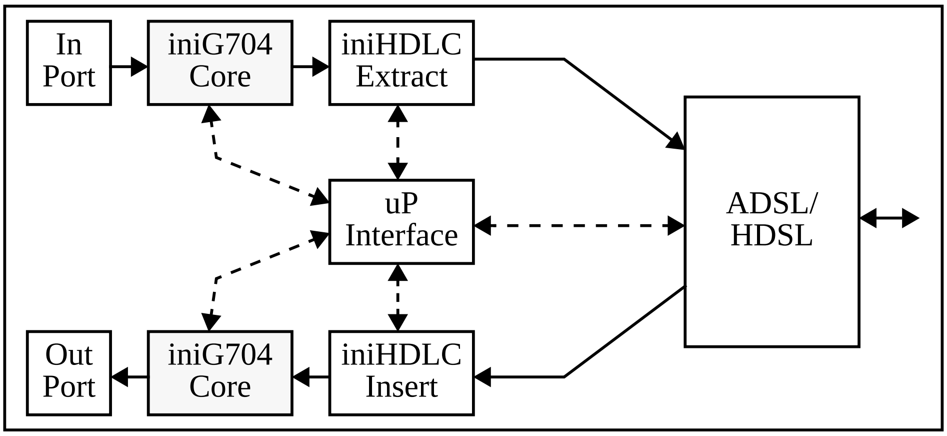


xAAL is an universal ATM AALx framer function.

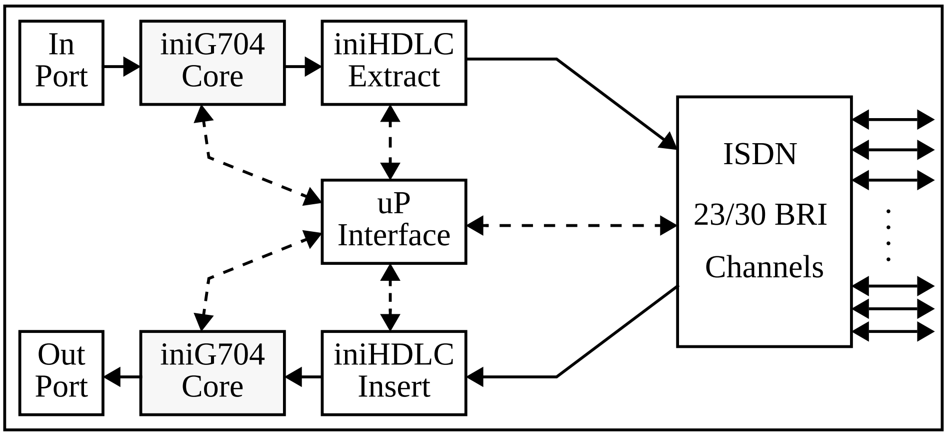
Following INICORE core functions are used in the examble above:

- iniUTOPIA: UTOPIA level II interface
- iniHDLC: HDLC Data link controller
- iniG704: G704 E1- or DS1-Framer

... or ...



... or ...



6 Performance

The following chapter provides a short overview of the performance of INICORE’s iniG704 E1-Framer core.

6.1 ACTEL 3200DX

Device	Max clock frequency*	Utilisation sequential	Utilisation logic
32200DX	> 12 MHz	36 %	80 %
32300DX	> 12 MHz	26 %	52 %
32400DX	n.a.	n.a.	n.a.

*) the maximum clock frequency is based on -1 speed grade

The following resources are used:

- 0 Quadrant clock network
- 1 global clock network
- 0 SRAM modules
- 0 wide decode circuitry
- xx user I/Os

The number of user I/Os is not available because it depends on the application.

7 Reference

ITU (CCIITT) Recommendation G.704
+ Amendments June 90

- Section 2.3: Basic frame structure at 2048 kbit/s
- Annex A.3: CRC4 procedure for interface at 2048 kbit/s

ITU (CCITT) Recommendation G.706
+ Amendments June 90

- Section 4: Frame alignment and CRC procedures at 2048 kbit/s interface
- Annex B: .. automatic interworking
  Figure B1/G.706 Framing algorithm