32nd Annual INCOSE
international symposium
hybrid event
Detroit, MI, USA
June 25 - 30, 2022

Presenter, Eric B. Dano, Ph.D.
BAE Systems, Electronic Systems (ES)
eric.b.dano@baesystems.com

Not export controlled per ES-CEMA-021521-0057

Using Design Structure Matrices (DSMs) to Derive System Architectures

What is a DSM ?

Design Structure Matrix (DSM) Definition (a.k.a. N^2 Dependency Matrix, Dependency Structure Matrix, etc.)

Tool used to optimize a grouping of Tasks (schedule or process applications), Components or functions (system architecture applications) or Teams (organizational applications) based on defined dependencies between elements to produce an optimal time sequence of activities or grouping of components for a given system application.

DSM Types

Static [1] – “Represent system elements existing simultaneously, such as components of a product architecture or groups in an organization. Usually solved using clustering algorithms.”

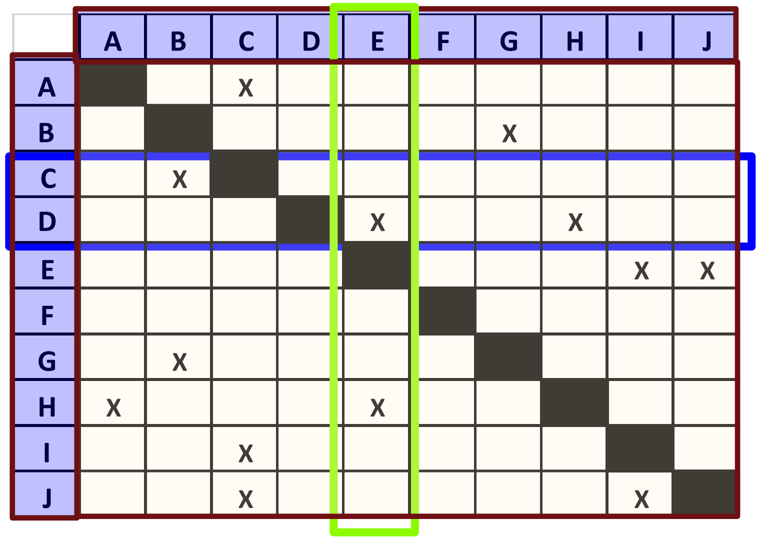
Time Based [1] – “The ordering of the elements in the system represent a flow through time, with upstream activities preceding down stream activities. “Feed-forward” and “feed-back” are used to describe interfaces.

[1] T. Browing, 2001.

DSM Matrix Basics

- The top row and left most column list the elements/tasks/teams to be considered
  - The diagonal is in black because no element is dependent on itself
  - The element sequence currently goes from A to J but will be optimized based on the dependencies defined by the Xs
- Each Row shows the "Needs" (i.e. must occur after the element in the current Row)
  - EX1: Element C "Needs" only B. Since it is currently after B, it is fine
  - EX2: Element D "Needs" E & H which both "Provide" to D. Therefore, Element D must be moved later in the sequence (after E & H).
    - » Note that all X's above the diagonal have this issue and will need to be optimized
- Each Column shows the "Provides" (i.e. must occur before the element in each checked Row)
  - EX3: Element E "Provides" to D & H. It will need to be moved to before D

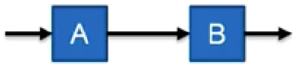
Example DSM



	A	B	C	D	E	F	G	H	I	J
A	■		X							
B		■					X			
C		X	■							
D				■	X			X		
E					■				X	X
F						■				
G		X					■			
H	X				X			■		
I			X						■	
J			X						X	■

Interaction/Interdependency Types

- **Dependent**
  - There is a dependence between the two elements/tasks
  - They must be performed sequentially



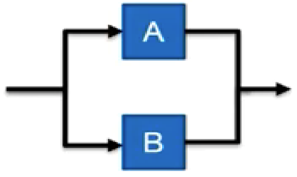
Dependent
-i.e. sequential

Example DSM Appearance

	A	B
A	■	
B	X	■

B needs A before it can start

- **Independent**
  - There is NO dependence between the two elements/tasks
  - They may be performed in parallel

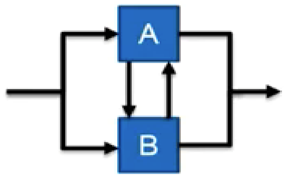


Independent
-i.e. parallel

	A	B
A	■	
B		■

A and B can be started in parallel

- **Interdependent**
  - Each element/task relies on out put from the element/task
  - The two elements/tasks are coupled



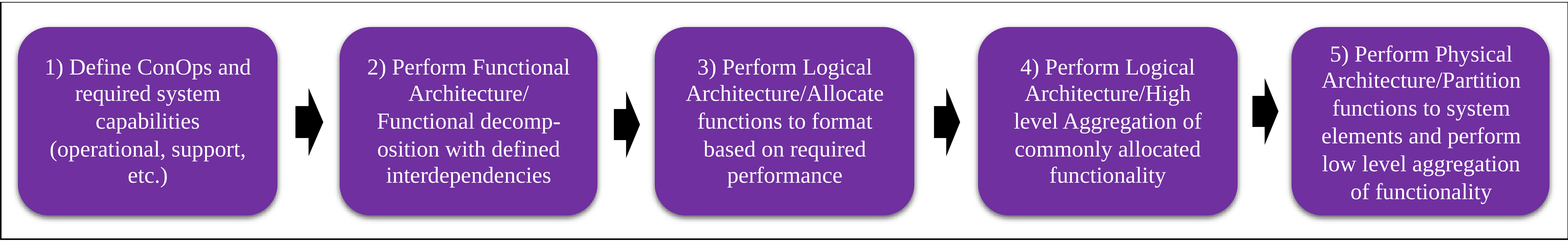
Interdependent
-i.e. coupled

	A	B
A	■	X
B	X	■

A and B must be done simultaneously

Static DSM Example: System Architecture of a Software Defined Radio (SDR) Based System

Derive an Architecture in 5 Easy Steps !



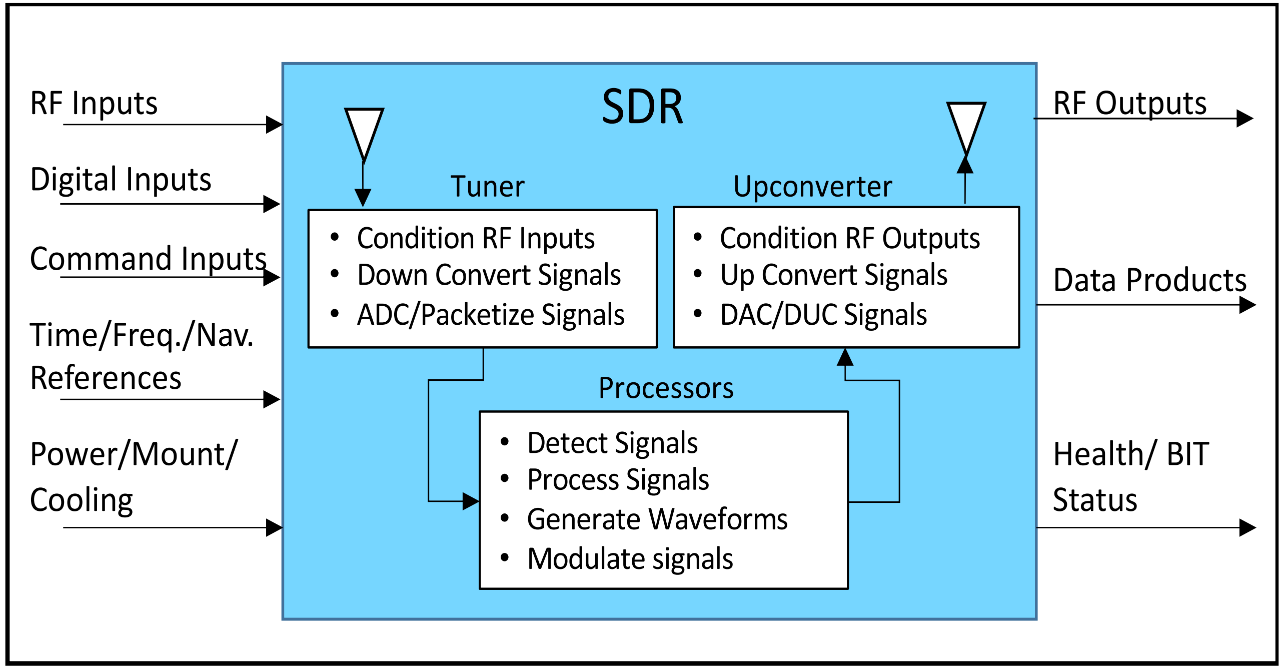
1) Define ConOps and required system capabilities (operational, support, etc.)

2) Perform Functional Architecture/ Functional decomposition with defined interdependencies

3) Perform Logical Architecture/Allocate functions to format based on required performance

4) Perform Logical Architecture/High level Aggregation of commonly allocated functionality

5) Perform Physical Architecture/Partition functions to system elements and perform low level aggregation of functionality



SDR

RF Inputs

Digital Inputs

Command Inputs

Time/Freq./Nav. References

Power/Mount/ Cooling

Tuner

- Condition RF Inputs
- Down Convert Signals
- ADC/Packetize Signals

Upconverter

- Condition RF Outputs
- Up Convert Signals
- DAC/DUC Signals

Processors

- Detect Signals
- Process Signals
- Generate Waveforms
- Modulate signals

RF Outputs

Data Products

Health/ BIT Status

A Software Defined Radio will be used in this example

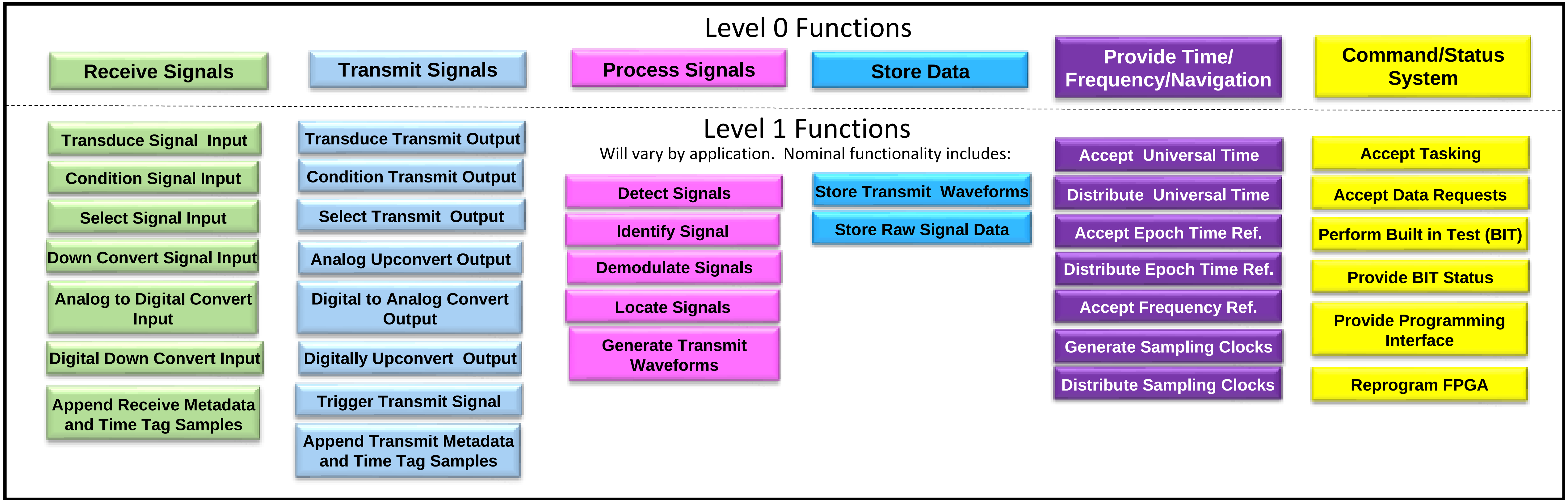
1. Define ConOps and Required System Capabilities

- The architecture process starts with the "hand-in" of the customer specified Concept of Operations (ConOps), Key Performance Parameters (KPPs), Key System Attributes (KSAs) and value statements.
  - The application of option generation techniques is critical during the concept exploration phase and should include holistic systems thinking to find highest level objectives, using analogies to create options, dynamic system modeling and simulation [2], the use of heuristics [3] and proven design patterns for software [4].
- The SDR based system ConOp is to have the capabilities to receive signals, process signals, transmit signals, store data and provide time/frequency/navigation.
  - The ConOps must be further expanded to include supportability, human machine interface, cyber, testability, safety, production, etc. to ensure the full set of ConOps are understood prior to commencing functional decomposition of the system.
  - System architect must ensure a multi-disciplinary solution is obtained [5].

ConOps definition is performed prior to populating the DSM

2. Perform System Functional Decomposition

- The level 0 functionality is defined as part of the functional architecture process
- Level 0 functions are then decomposed into Level 1 sub-functions which will be used in the architecture derivation process [6], [7]



Level 0 Functions

Level 1 Functions

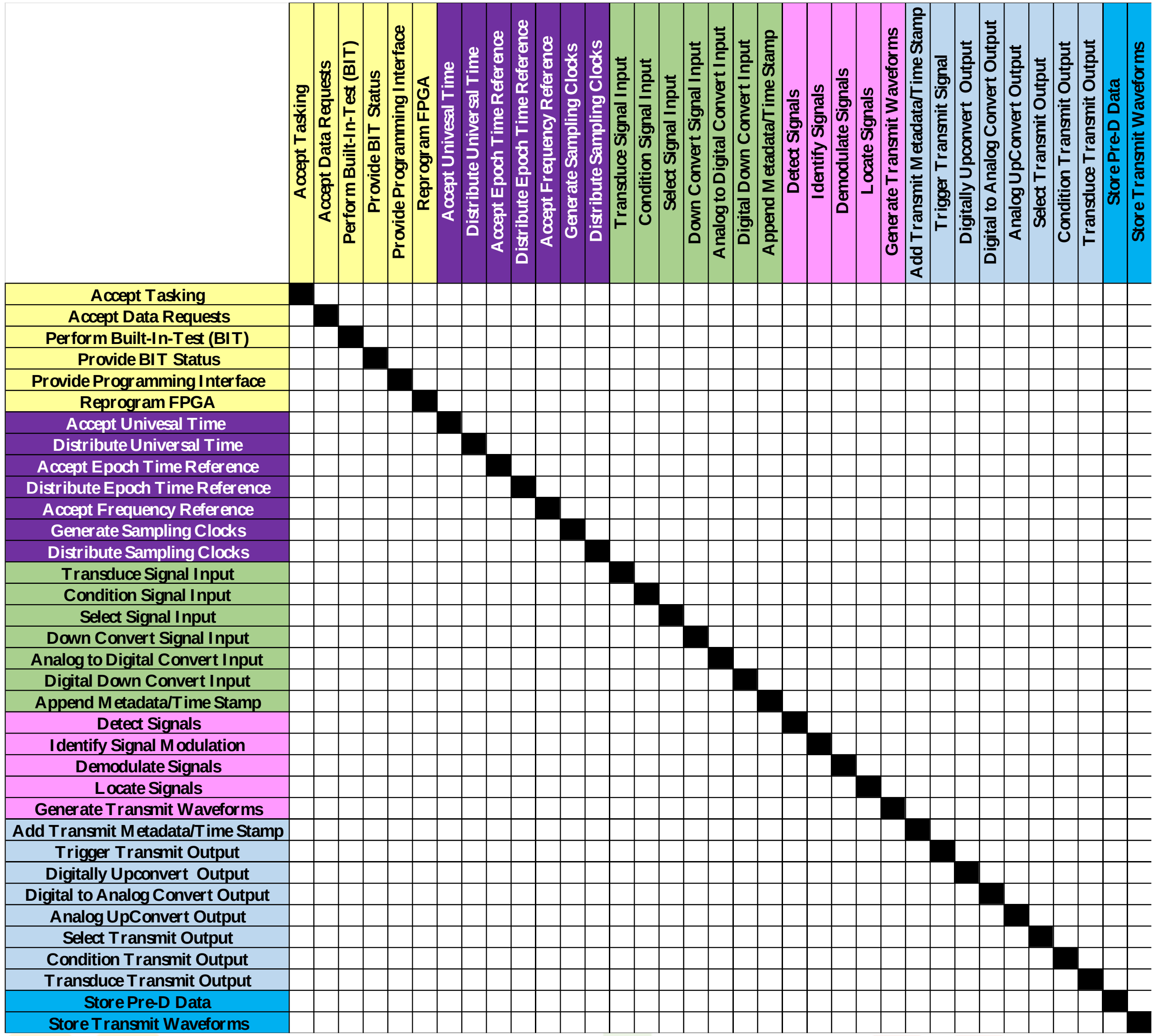
Will vary by application. Nominal functionality includes:

Receive Signals	Transmit Signals	Process Signals	Store Data	Provide Time/ Frequency/Navigation	Command/Status System
Transduce Signal Input	Transduce Transmit Output	Detect Signals	Store Transmit Waveforms	Accept Universal Time	Accept Tasking
Condition Signal Input	Condition Transmit Output	Identify Signal	Store Raw Signal Data	Distribute Universal Time	Accept Data Requests
Select Signal Input	Select Transmit Output	Demodulate Signals		Accept Epoch Time Ref.	Perform Built in Test (BIT)
Down Convert Signal Input	Analog Upconvert Output	Locate Signals		Distribute Epoch Time Ref.	Provide BIT Status
Analog to Digital Convert Input	Digital to Analog Convert Output	Generate Transmit Waveforms		Accept Frequency Ref.	Provide Programming Interface
Digital Down Convert Input	Digitally Upconvert Output			Generate Sampling Clocks	Reprogram FPGA
Append Receive Metadata and Time Tag Samples	Trigger Transmit Signal			Distribute Sampling Clocks	
	Append Transmit Metadata and Time Tag Samples				

Apply Functional Breakdown to DSM Matrix

- Level 1 functionality is placed along the X and Y axises
- Diagonal is blacked out
- Level 1 functionality is usually sufficient to derive the system architecture and define system level modularity

Functional Decomp	
🟨	Command/Status System
🟪	Provide Time/ Frequency Reference
🟩	Receive Signals
🟥	Process/Generate Signals
🟦	Transmit Signals
🟦	Store Data



	Accept Tasking	Accept Data Requests	Perform Built-In-Test (BIT)	Provide BIT Status	Provide Programming Interface	Reprogram FPGA	Accept Univesal Time	Distribute Universal Time	Accept Epoch Time Reference	Distribute Epoch Time Reference	Accept Frequency Reference	Generate Sampling Clocks	Distribute Sampling Clocks	Transduce Signal Input	Condition Signal Input	Select Signal Input	Down Convert Signal Input	Analog to Digital Convert Input	Digital Down Convert Input	Append Metadata/Time Stamp	Detect Signals	Identify Signals	Demodulate Signals	Locate Signals	Generate Transmit Waveforms	Add Transmit Metadata/Time Stamp	Trigger Transmit Signal	Digitally Upconvert Output	Digital to Analog Convert Output	Analog UpConvert Output	Select Transmit Output	Condition Transmit Output	Transduce Transmit Output	Store Pre-D Data	Store Transmit Waveforms
Accept Tasking	■																																		
Accept Data Requests		■																																	
Perform Built-In-Test (BIT)			■																																
Provide BIT Status				■																															
Provide Programming Interface					■																														
Reprogram FPGA						■																													
Accept Univesal Time							■																												
Distribute Universal Time								■																											
Accept Epoch Time Reference									■																										
Distribute Epoch Time Reference										■																									
Accept Frequency Reference											■																								
Generate Sampling Clocks												■																							
Distribute Sampling Clocks													■																						
Transduce Signal Input														■																					
Condition Signal Input															■																				
Select Signal Input																■																			
Down Convert Signal Input																	■																		
Analog to Digital Convert Input																		■																	
Digital Down Convert Input																			■																
Append Metadata/Time Stamp																				■															
Detect Signals																					■														
Identify Signal Modulation																						■													
Demodulate Signals																							■												
Locate Signals																								■											
Generate Transmit Waveforms																									■										
Add Transmit Metadata/Time Stamp																										■									
Trigger Transmit Output																											■								
Digitally Upconvert Output																												■							
Digital to Analog Convert Output																													■						
Analog UpConvert Output																														■					
Select Transmit Output																															■				
Condition Transmit Output																																■			
Transduce Transmit Output																																	■		
Store Pre-D Data																																		■	
Store Transmit Waveforms																																			■

Define Functional Dependencies to Matrix

- X's show interdependencies between the derived Level 1 functions [8], [9].
- This process is focused on modularity (i.e. static DSM) and is not adjusted for order of events (i.e. Time-based DSM)
- EX1: See that metadata/ time stamping relies on:
- EX2: See that metadata/ time stamping relies on:
  - Tasking
  - UTC time message distribution,
  - Epoch Time (1PPS) distribution
  - ADC/DAC clocks distribution

Command & Control

	Accept Tasking	Accept Data Requests	Perform Built-In-Test (BIT)	Provide BIT Status	Provide Programming Interface	Reprogram FPGA	Accept Universal Time	Distribute Universal Time	Accept Epoch Time Reference	Distribute Epoch Time Reference	Accept Frequency Reference	Generate Sampling Clocks	Distribute Sampling Clocks	Transduce Signal Input	Condition Signal Input	Select Signal Input	Down Convert Signal Input	Analog to Digital Convert Input	Digital Down Convert Input	Append Metadata/Time Stamp	Detect Signals	Identify Signals	Demodulate Signals	Locate Signals	Generate Transmit Waveforms	Add Transmit Metadata/Time Stamp	Trigger Transmit Signal	Digitally Upconvert Output	Digital to Analog Convert Output	Analog UpConvert Output	Select Transmit Output	Condition Transmit Output	Transduce Transmit Output	Store Raw Signal Data	Store Transmit Waveforms
Accept Tasking	■																																		
Accept Data Requests	X	■																																	
Perform Built-In-Test (BIT)	X		■																																
Provide BIT Status	X		X	■																															
Provide Programming Interface	X				■																														
Reprogram FPGA					X	■																													
Accept Univesal Time							■																												
Distribute Universal Time							X	■																											
Accept Epoch Time Reference									■																										
Distribute Epoch Time Reference									X	■																									
Accept Frequency Reference											■																								
Generate Sampling Clocks										X	X	■																							
Distribute Sampling Clocks												X	■																						
Transduce Signal Input														■																					
Condition Signal Input	X													X	■																				
Select Signal Input	X														X	■																			
Down Convert Signal Input	X															X	■																		
Analog to Digital Convert Input	X												X				X	■																	
Digital Down Convert Input	X												X					X	■																
Append Metadata/Time Stamp	X							X		X			X						X	■															
Detect Signals	X																		X	X	■														
Identify Signal Modulation	X																				X	■													
Demodulate Signals	X																					X	■												
Locate Signals	X																				X			■											
Generate Transmit Waveforms	X																								■										X
Add Transmit Metadata/Time Stamp	X							X		X			X												X	■									
Trigger Transmit Output	X							X		X			X													X	■								
Digitally Upconvert Output	X												X														X	■							
Digital to Analog Convert Output	X																											X	■						
Analog UpConvert Output	X																												X	■					
Select Transmit Output	X																							X						X	■				
Condition Transmit Output	X																														X	■			
Transduce Transmit Output																																X	■		
Store Raw Signal Data	X																		X	X														■	
Store Transmit Waveforms	X																																		■

3. Perform Allocation of Functionality

- Allocation is done to optimize performance of the various functions
  - Most functions can be performed using multiple allocations
- Allocations mostly driven by required:
  - Latency
  - Throughput
  - Fidelity
  - Cost
  - Leverage

- Allocation Heuristics (lessons learned) and required performance are used to properly allocate functions

	Hardware (H)	FPGAs (F)	GPUs (G)	GPPs/Software (S)
STRENGTH	• Good for generic fixed capabilities • Common open standards and interfaces exists • Relatively low cost	• Highly reconfigurable parallel architecture permits multiple operations to be performed simultaneously • Embedded multipliers and memory enable instantiation of extremely fast filters and synthesizers • Programmable data paths allow for processing a wide variety of data types • Low latency operations	• Excellent for processing intensive algorithms (multi-parallel processing) • Baselined to floating point operations • Fast memory access • Rapid commercial GPU upgrade cycle	• Highly versatile; can implement an almost limitless number of applications • Embedded math logic makes for efficient use of processing resources • Excellent for decision making and branching • Well suited for information management and control
WEAKNESS	• Fixed capabilities, can't be reconfigured • Requires additional components for extended frequencies • Relatively large Size, Weight, and Power (SWaP)	• Relatively inefficient for branching or decision-making operations (these consume large numbers of gates) • Large and fast devices are expensive • High-precision math operations may consume many resources	• High power draw • High heat dissipation • Some must be paired with an interface GPP • Limited vendors	• May be comparatively slow at simple fixed-point math operations, due to inherently serial processing nature • Only able to perform a low number of tasks per clock cycle • Higher latency operations

Allocated Functionality

Allocation Key:

- F – Firmware
- H – Hardware
- S – Software
- A – Application Specific Integrated Circuit (ASIC)
- G - Graphical Processing Unit (GPU)
- O - Other

	Accept Tasking	Accept Data Requests	Perform Built-In-Test (BIT)	Provide BIT Status	Provide Programming Interface	Reprogram FPGA	Accept Universal Time	Distribute Universal Time	Accept Epoch Time Reference	Distribute Epoch Time Reference	Accept Frequency Reference	Generate Sampling Clocks	Distribute Sampling Clocks	Transduce Signal Input	Condition Signal Input	Select Signal Input	Down Convert Signal Input	Analog to Digital Convert Input	Digital Down Convert Input	Append Metadata/Time Stamp	Detect Signals	Identify Signals	Demodulate Signals	Locate Signals	Generate Transmit Waveforms	Add Transmit Metadata/Time Stamp	Trigger Transmit Signal	Digitally Upconvert Output	Digital to Analog Convert Output	Analog UpConvert Output	Select Transmit Output	Condition Transmit Output	Transduce Transmit Output	Store Raw Signal Data	Store Transmit Waveforms
Accept Tasking	S																																		
Accept Data Requests	X	S																																	
Perform Built-In-Test (BIT)	X		S																																
Provide BIT Status	X		X	S																															
Provide Programming Interface	X				S																														
Reprogram FPGA					X	F																													
Accept Univesal Time							S																												
Distribute Universal Time							X	S																											
Accept Epoch Time Reference									H																										
Distribute Epoch Time Reference									X	H																									
Accept Frequency Reference											H																								
Generate Sampling Clocks										X	X	H																							
Distribute Sampling Clocks												X	H																						
Transduce Signal Input														H																					
Condition Signal Input	X													X	H																				
Select Signal Input	X														X	H																			
Down Convert Signal Input	X															X	H																		
Analog to Digital Convert Input	X												X				X	H																	
Digital Down Convert Input	X												X					X	F																
Append Metadata/Time Stamp	X							X		X			X						X	F															
Detect Signals	X																		X	X	F														
Identify Signal Modulation	X																				X	S													
Demodulate Signals	X																					X	S												
Locate Signals	X																				X			S											
Generate Transmit Waveforms	X																								S										X
Add Transmit Metadata/Time Stamp	X							X		X			X												X	F									
Trigger Transmit Output	X							X		X			X													X	F								
Digitally Upconvert Output	X												X														X	F							
Digital to Analog Convert Output	X																											X	H						
Analog UpConvert Output	X																												X	H					
Select Transmit Output	X																							X						X	H				
Condition Transmit Output	X																														X	H			
Transduce Transmit Output																																X	H		
Store Raw Signal Data	X																		X	X														H	
Store Transmit Waveforms	X																																		H

4. Perform Aggregation of Functionality

- Aggregation is done to optimize the grouping of the allocated functions to aid modularity definition in the system and provide [10]:
- **High Cohesiveness** - Large similarity in well-defined functions performed within a module
  - Enables Commonality and Reuse
- **Low Coupling** - Module functionality does not constrain functionality in any other module
  - Reduces complexity, eases testability, catalyst for rapid capability insertion, etc.

	Accept Tasking	Accept Data Requests	Perform BIT	Provide BIT Status	Provide Programming Interface	Accept UTC Time	Distribute UTC Time Ref.	Generate WF (Tx)	Stream Data	Accept 1PPS Ref.	Distribute 1PPS Ref.	Accept Freq Ref.	Generate ADC Clocks	Distribute ADC Clocks	Generate DAC Clocks	Distribute DAC Clocks	Transduce RF Input	Condition RF Input	Select RF Input	Down Convert RF Input	A/D Convert Input	D/A Convert Output	Up Convert IF Output	Select RF Output	Condition RF Output	Transduce RF Output	Store Tx Waveforms	Play WF fm memory (Tx)	Store Pre-D Data	Digital Down Convert (DDC) ADC Output	Time Tag Samples (Rx)	Append Metadata (Rx)	Detect Signals (Rx)	Identify Signals (Rx)	Play Audio (Rx)	Locate Signals (Rx)	Digitally Upconvert (DUC) DAC Input	Time Tag Samples (Tx)	Append Metadata (Tx)	Queue Waveform	Trigger RF Output	Reprogram FPGA
Accept Tasking	S																																									
Accept Data Requests	X	S																																								
Perform BIT	X		S																																							
Provide BIT Status			X	S																																						
Provide Programming Interface	X				S																																					
Accept UTC time						S																																				
Distribute UTC Time Ref.						X	S																																			
Generate WF (Tx)	X							S																																		
Stream Data									S																				X													
Accept 1PPS Ref.										H																																
Distribute 1PPS Ref.										X	H																															
Accept Freq Ref.												H																														
Generate ADC Clocks											X		H																													
Distribute ADC Clocks													X	H																												
Generate DAC Clocks											X				H																											
Distribute DAC Clocks															X	H																										
Transduce RF Input																	H																									
Condition RF Input	X																X	H																								
Select RF Input	X																	X	H																							
Down Convert RF Input	X																		X	H																						
A/D Convert IF Input	X																			X	H																					
D/A Convert Output	X																					H															X					
Up Convert Output	X																					X	H																			
Select RF Output	X																						X	H																		
Condition RF Output	X																							X	H																	
Transduce RF Output																									X	H																
Store Tx Waveforms																											H															
Play WF from Memory (Tx)	X																										X	H														
Store Pre-D Data	X	X																											H													
Digital Down Convert (DDC) ADC Output	X																				X									F												
Time Tag Samples (Rx)							X				X			X																X	F											
Append Metadata (Rx)	X																														X	F										
Detect Signals (Rx)	X																															X	F									
Identify Signal Mods (Rx)	X								X																				X				X	F								
Play Audio (Rx)	X								X																				X				X		F							
Locate Signals (Rx)	X																																X			F						
Digitally Upconvert (DUC) DAC Input	X																																				F				X	
Time Tag Samples (Tx)							X	X			X					X												X										F				
Append Metadata (Tx)	X																																					X	F			
Queue Waveform																																							X	F		
Trigger RF Output	X																																							X	F	
Reprogram FPGA					X																																					F

SW

HW

FW

Functional Decomp

🟨	Command/Status System
🟪	Provide Time/ Frequency Reference
🟩	Receive Signals
🟪	Process/Generate Signals
🟦	Tx fns
🟦	Store Data
X	Identified dependencies between the Functions

Functional Allocation (Based On Performance)

H	Hardware
F	Firmware
S	Software
A	ASIC
G	GPU
O	Other

Software Defined Radio (SDR) Based System
Case 1: Large Platform
Case 2: Small Platform

SDR Characteristics for Functional DSM Methodology Use Cases

Case 1 – Large Platform	Case 2 – Small UAV Platform
• High precision - Geolocation	• Lower Precision – Situational Awareness
• Multiple Receive Array(s)	• Two Receive Antennas
• Multiple Receive Channels	• Two Receive Channels
• Multiple Receive Bands	• One Receive Band
• N-Channel Direction Finding	• 2 Channel Direction Finding
• Multiple Transmit Channels	• One Transmit Channel
• Multiple Transmit Bands	• One Transmit Band
• Multiple Transmit Array(s)	• One TX antenna
• Large Radiated Power – Stand-Off	• Low Radiated Power – Stand-In
• Significant signal distribution	• Direct Signal Cabling

Up to this point both platforms have the same architecture

5. Perform Partitioning of Functionality

- Partitioning groups aggregated functions into a specific system element(s) and ultimately defines the modularity of a system [11], [12]
  - The thinking behind this step is often missed!!!

- Partitioning is determined based on:
  - Heuristics
  - CONOPs
  - Make/buy decisions
  - Top Level Requirements
  - State of COTS technologies/State of internal technologies/Leverage
  - Alignment with Open Standards
  - Architecture Trades/Analysis
  - Modular Open System Approach (MOSA) [13]

The System Architect works with SMEs to ensure an optimal system concept is defined

Partitioned Functionality – Case 1 Large SDR System

- Case 1 – Large SDR system
  - High Rx Sensitivity
  - Rx Arrays
  - Multiple Receive Channels
  - Multiple Receive Bands
  - N-Channel DF
  - Multiple Transmit Channels
  - Multiple Transmit Bands
  - Tx Arrays
  - Large Tx ERP
  - Significant RF distribution, compensation, calibration RF power detection, etc.

Partitioning Key:

🟦	Software Config. Item
🟩	Hardware Config. Item
🟧	Firmware Config. Item

	Accept Tasking	Accept Data Requests	Perform Built-In-Test (BIT)	Provide BIT Status	Provide Programming Interface	Accept Universal Time	Distribute Universal Time	Accept Epoch Time Reference	Distribute Epoch Time Reference	Accept Frequency Reference	Generate Sampling Clocks	Distribute Sampling Clocks	Transduce Signal Input	Condition Signal Input	Select Signal Input	Down Convert Signal Input	Analog to Digital Convert Input	Digital Down Convert Input	Append Metadata/Time Stamp	Detect Signals	Identify Signals	Demodulate Signals	Locate Signals	Generate Transmit Waveforms	Add Transmit Metadata/Time Stamp	Trigger Transmit Signal	Digitally Upconvert Output	Digital to Analog Convert Output	Analog UpConvert Output	Select Transmit Output	Condition Transmit Output	Transduce Transmit Output	Store Raw Signal Data	Store Transmit Waveforms	Reprogram FPGA
Accept Tasking	S																																		
Accept Data Requests	X	S																																	
Perform Built-In-Test (BIT)	X		S																																
Provide BIT Status	X		X	S																															
Provide Programming Interface	X				S																														
Accept Univesal Time						S																													
Distribute Universal Time						X	S																												
Accept Epoch Time Reference								H																											
Distribute Epoch Time Reference								X	H																										
Accept Frequency Reference										H																									
Generate Sampling Clocks									X	X	H																								
Distribute Sampling Clocks											X	H																							
Transduce Signal Input													H																						
Condition Signal Input	X												X	H																					
Select Signal Input	X													X	H																				
Down Convert Signal Input	X														X	H																			
Analog to Digital Convert Input	X											X				X	H																		
Digital Down Convert Input	X											X					X	F																	
Append Metadata/Time Stamp	X						X		X			X						X	F																
Detect Signals	X																	X	X	F															
Identify Signal Modulation	X																			X	S														
Demodulate Signals	X																				X	S													
Locate Signals	X																			X			S												
Generate Transmit Waveforms	X																							S											
Add Transmit Metadata/Time Stamp	X																							X	F										
Trigger Transmit Output	X																								X	F									
Digitally Upconvert Output	X																									X	F								
Digital to Analog Convert Output	X																										X	H							
Analog UpConvert Output	X																											X	H						
Select Transmit Output	X																						X						X	H					
Condition Transmit Output	X																													X	H				
Transduce Transmit Output																															X	H			
Store Raw Signal Data	X																																H		
Store Transmit Waveforms	X																																	H	
Reprogram FPGA					X																														F

All infrastructure software CSCIs functionality partitioned to GPPs

Time/Frequeny Reference functionality partitioned to hardware CCAs

Transduce Signal Input functionality partitioned to hardware Receive arrays

Receive signal condition & distribute functionality partitioned to discrete hardware

Receive signal downconvert and digitize functionality are partitioned to discrete hardaware Tuner CCAs. Internal FPGA assets will host time stamping, etc.

All Signal Processing and Waveform Generation software CSCIs functionality partitioned to GPPs

Transmit signal conditioning & distribution functionality partitioned to discrete hardware

Transmit signal digital upconvert and digital to analog convert functionality are partitioned to discrete Upconverter CCAs. Internal FPGA assets will host time triggering, etc.

Transduce signal output functionality partitioned to hardware transmit arrays

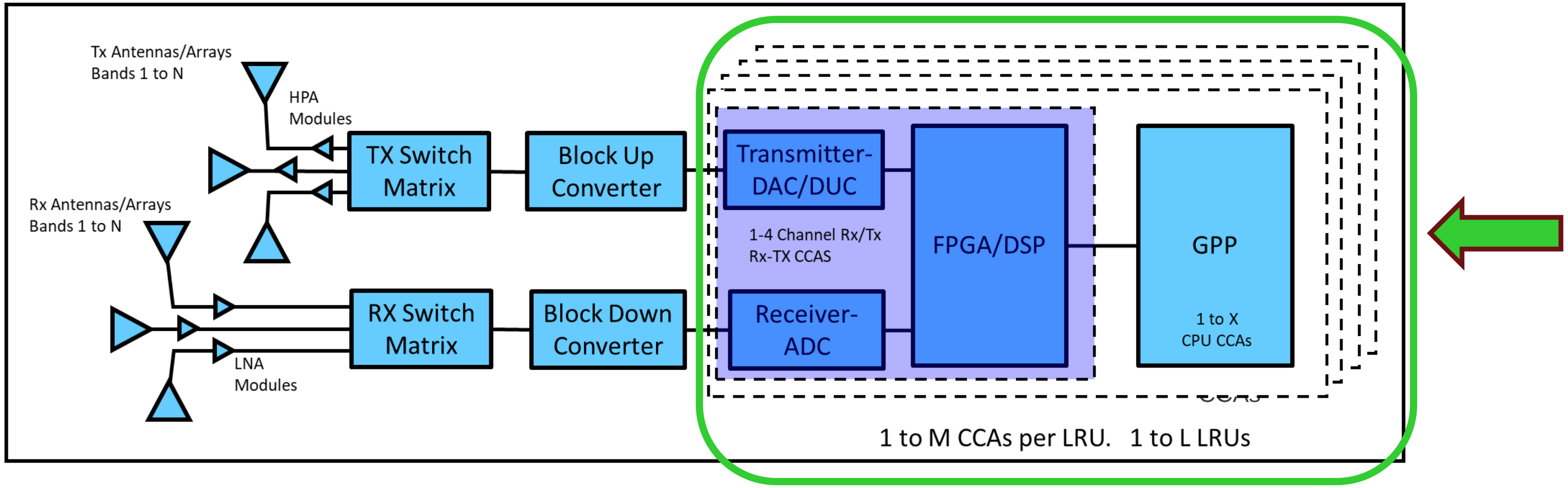
Storage functionality partitioned to hardware mass storage device

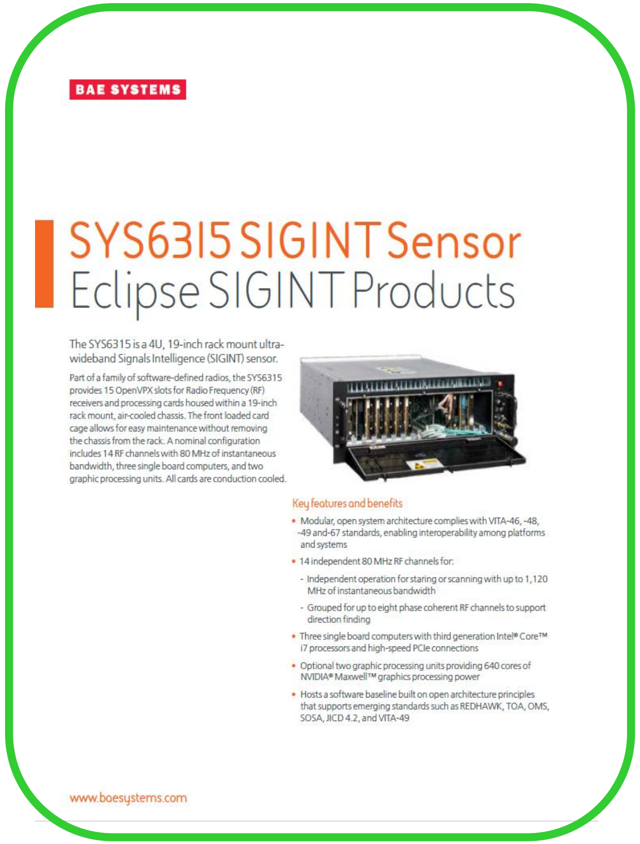
CCA = Circuit Card Assembly
CSCI - Computer Software Configuration Item
GPP = General Purpose Processors
COTS = Commercial Off-The-Shelf

Modularity at the Box and Card Level

Use of several open standards

Case 1 Large SDR System (Cont.)





BAE SYSTEMS

SYS6315 SIGINT Sensor
Eclipse SIGINT Products

The SYS6315 is a 4U, 19-inch rack mount ultra-wideband Signals Intelligence (SIGINT) sensor.

Part of a family of software-defined radios, the SYS6315 provides 15 OpenVPX slots for Radio Frequency (RF) receivers and processing cards housed within a 19-inch rack mount, air-cooled chassis. The front loaded card cage allows for easy maintenance without removing the chassis from the rack. A nominal configuration includes 14 RF channels with 80 MHz of instantaneous bandwidth, three single board computers, and two graphic processing units. All cards are conduction cooled.

Key features and benefits

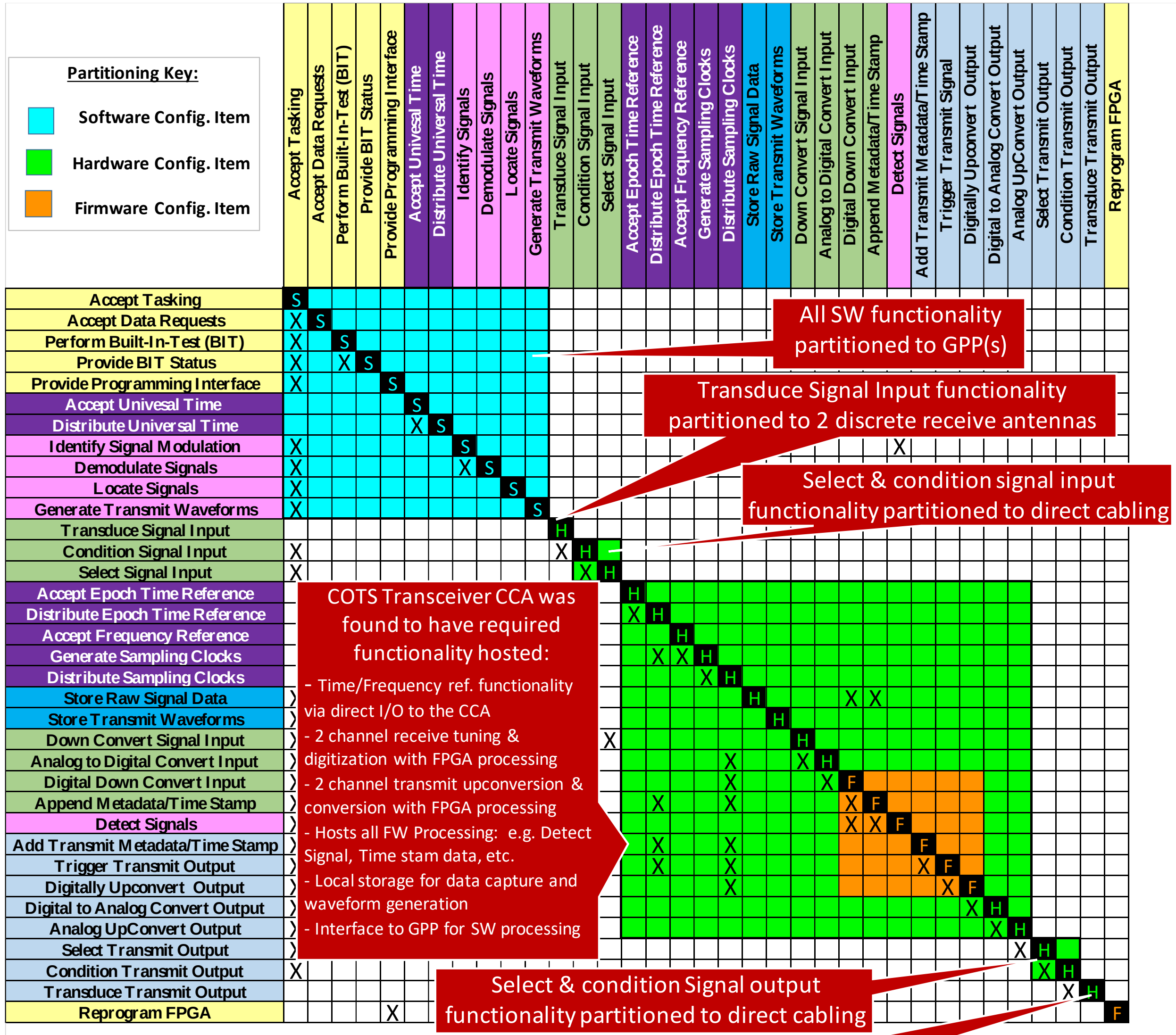
- Modular, open system architecture complies with VITA-46, -48, -49 and-67 standards, enabling interoperability among platforms and systems
- 14 independent 80 MHz RF channels for:
  - Independent operation for staring or scanning with up to 1,120 MHz of instantaneous bandwidth
  - Grouped for up to eight phase coherent RF channels to support direction finding
- Three single board computers with third generation Intel® Core™ i7 processors and high-speed PCIe connections
- Optional two graphic processing units providing 640 cores of NVIDIA® Maxwell™ graphics processing power
- Hosts a software baseline built on open architecture principles that supports emerging standards such as REDHAWK, TOA, OMS, SOSA, JICD 4.2, and VITA-49

www.baesystems.com

Architecture is scalable and can GROW to include the required number of Rx and Tx LRUs

Partitioned Funct. – Case 2 Small (UAV) SDR System

- Case 2 – Small UAV SDR System
  - Less sensitivity
  - One Receive Channel
  - One Rx Antenna
  - One Transmit Channel
  - One TX antenna
  - Less ERP
  - Direct RF Cabling



Partitioning Key:

🟦	Software Config. Item
🟩	Hardware Config. Item
🟧	Firmware Config. Item

	Accept Tasking	Accept Data Requests	Perform Built-In-Test (BIT)	Provide BIT Status	Provide Programming Interface	Accept Universal Time	Distribute Universal Time	Identify Signals	Demodulate Signals	Locate Signals	Generate Transmit Waveforms	Transduce Signal Input	Condition Signal Input	Select Signal Input	Accept Epoch Time Reference	Distribute Epoch Time Reference	Accept Frequency Reference	Generate Sampling Clocks	Distribute Sampling Clocks	Store Raw Signal Data	Store Transmit Waveforms	Down Convert Signal Input	Analog to Digital Convert Input	Digital Down Convert Input	Append Metadata/Time Stamp	Detect Signals	Add Transmit Metadata/Time Stamp	Trigger Transmit Signal	Digitally Upconvert Output	Digital to Analog Convert Output	Analog UpConvert Output	Select Transmit Output	Condition Transmit Output	Transduce Transmit Output	Reprogram FPGA
Accept Tasking	S																																		
Accept Data Requests	X	S																																	
Perform Built-In-Test (BIT)	X		S																																
Provide BIT Status	X		X	S																															
Provide Programming Interface	X				S																														
Accept Univesal Time						S																													
Distribute Universal Time						X	S																												
Identify Signal Modulation	X							S																		X									
Demodulate Signals	X							X	S																										
Locate Signals	X									S																									
Generate Transmit Waveforms	X										S																								
Transduce Signal Input												H																							
Condition Signal Input	X											X	H																						
Select Signal Input	X												X	H																					
Accept Epoch Time Reference															H																				
Distribute Epoch Time Reference															X	H																			
Accept Frequency Reference																	H																		
Generate Sampling Clocks																X	X	H																	
Distribute Sampling Clocks																		X	H																
Store Raw Signal Data	X																			H				X	X										
Store Transmit Waveforms	X																				H														
Down Convert Signal Input	X													X								H													
Analog to Digital Convert Input	X																		X			X	H												
Digital Down Convert Input	X																		X				X	F											
Append Metadata/Time Stamp	X															X			X					X	F										
Detect Signals	X																							X	X	F									
Add Transmit Metadata/Time Stamp	X															X			X								F								
Trigger Transmit Output	X															X			X								X	F							
Digitally Upconvert Output	X																		X									X	F						
Digital to Analog Convert Output	X																												X	H					
Analog UpConvert Output	X																													X	H				
Select Transmit Output	X																														X	H			
Condition Transmit Output	X																															X	H		
Transduce Transmit Output																																	X	H	
Reprogram FPGA					X																														F

All SW functionality partitioned to GPP(s)

Transduce Signal Input functionality partitioned to 2 discrete receive antennas

Select & condition signal input functionality partitioned to direct cabling

COTS Transceiver CCA was found to have required functionality hosted:
- Time/Frequency ref. functionality via direct I/O to the CCA
- 2 channel receive tuning & digitization with FPGA processing
- 2 channel transmit upconversion & conversion with FPGA processing
- Hosts all FW Processing: e.g. Detect Signal, Time stam data, etc.
- Local storage for data capture and waveform generation
- Interface to GPP for SW processing

Select & condition Signal output functionality partitioned to direct cabling

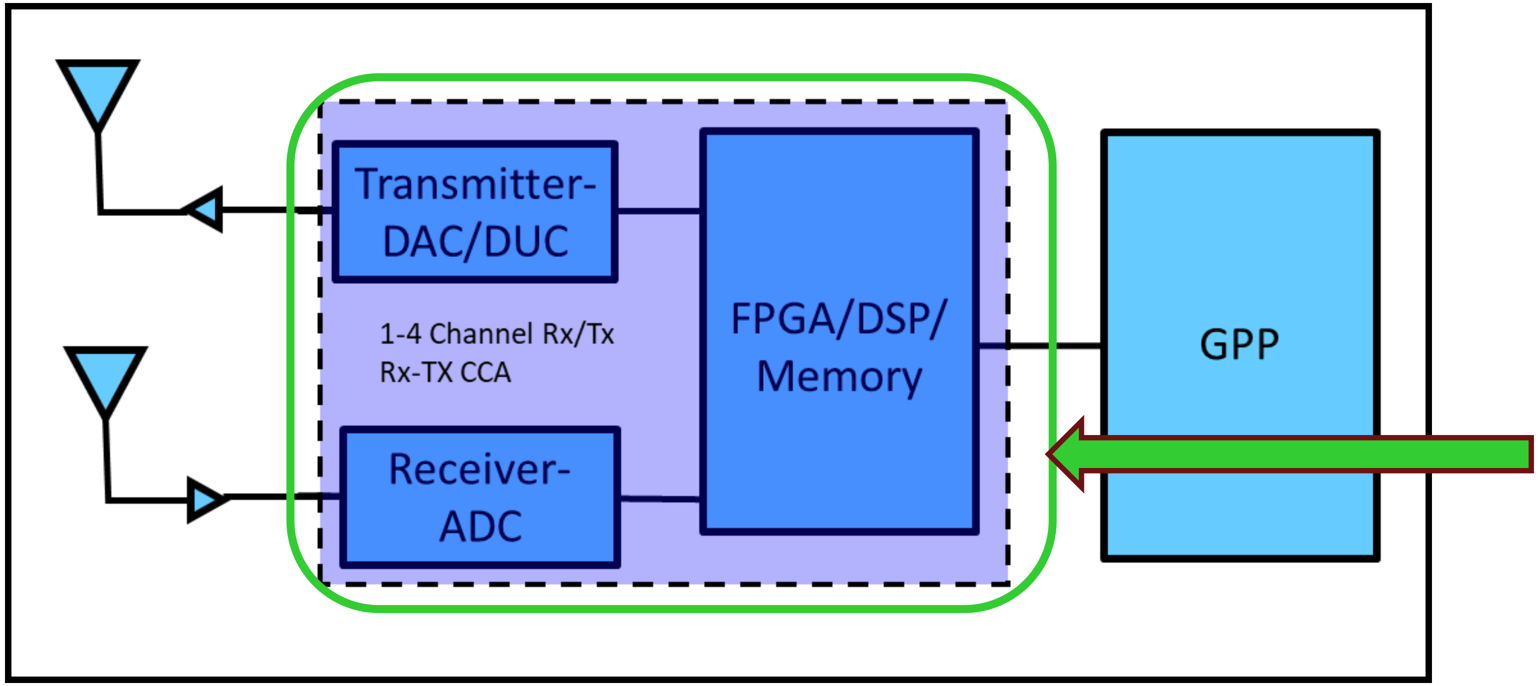
Transduce signal output functionality allocated to discrete transmit antenna

CCA = Circuit Card Assembly
CSCI - Computer Software Configuration Item
GPP = General Purpose Processors
COTS = Commercial Off-The-Shelf

Modularity at the Brick

Use of some open standards

Case 2 Small UAV SDR (Cont.)



Single and dual-channel Picoceptor options.

Architecture is scalable and can SHRINK to a low SWaP Transceiver with all Functionality

DSM Conclusions

- A 5-step architecture development approach was shown using functional (static) DSMs
  - The resulting architecture was optimized for modularity with high cohesion and low coupling between partitioned functionality
- The functional DSM approach had the advantage of:
  - Reinforcing key steps in the architectural process
  - Easily determining the complex interdependencies between functions
  - Performing iterative allocation, aggregation and partitioning (optimizes system modularity)
  - The ability to easily develop and assess alternative architectures/scale during partitioning
- The DSM derived architecture outputs (configuration items and interfaces) can be analyzed using the MOSA Key Open Sub-System (KOSS) tool (see paper) [14]
  - Led to further definition of the modularity required to reduce system cost and facilitate capability insertion over the system lifecycle (iterate with DSM archiitecture.
  - Shows alignment with the 5 MOSA Principles

Static DSMs provide a non-model based approach for performing/teaching system architecture

32nd Annual INCOSE
international symposium

hybrid event

Detroit, MI, USA
June 25 - 30, 2022

www.incose.org/symp2022

References (1 of 2)

[1] Browning, T 2001, ‘Applying the design structure matrix to system decomposition and integration problems: a review and new directions’, *IEEE Transactions on Engineering Management*, Vol. 48, No. 3., pp. 292–306.

[2] Madni, AM 2014, ‘Generating novel options during system architecting: psychological principles, systems thinking, and computer-based aiding’, *Systems Engineering*, vol. 17, no. 1, pp. 1-9.

[3] Maier, M & Rechtin, E 2009, *The art of systems architecting*, *3rd ed.*, CRC Press, New York, NY.

[4] Taylor, R, Medvidovic, N & Dashofy, E 2010, *Software architecture - foundations, theory and practice*, John Wiley and Sons Inc., Hoboken, NJ.

[5] Madni, AM 2018, *Transdisciplinary Systems Engineering*, Springer International Publishing, New York, NY.

[6] Coulston, C & Ford, R 2004 ‘Teaching functional decomposition for the design of electrical and computer systems’, *Proceedings of 34th ASEE/IEEE Frontiers in Education Conference*, pp. F4G-6 – F4G-11.

[7] Kockler, F, Withers, T, Poodiack, J & Gierman, M 1990, *Systems Engineering Management Guide, AD-A223-168*, Defense Systems Management College, Ft. Belvoir, VA.

References (2 of 2)

[8] Arnold, R & Wade, J 2015, ‘A definition of systems thinking: a systems approach’, *Proceedings of the 2015 Conference on Systems Engineering Research*, vol. 44, pp. 669-678.

[9] Monat, J & Gannon, T 2017, *Systems volume 8: using systems thinking to solve real-world problems*, Lawson H, Wade, J & Hofkirchner, W (eds.), College publications, U.K.

[10] Azani, C & Khorramshahgol, R 2006, ‘Modular open systems approach: an effective business strategy for building affordable and adaptable architectures’, *Journal of Management Systems*, Vol. 18, No. 1, pp. 66–76.

[11] Holtta, K, Suh, E & de Weck, O 2005, ‘Tradeoff between modularity and performance for engineered systems and products’, *Proceedings of the International Conference on Engineering Design (ICED)*, pp. 1-13.

[12] Yassine, A 2004, ‘An Introduction to modeling and analyzing complex product development processes using the design structure matrix (DSM) method’, *Quaderni di Management (Italian Management Review*), No.9.

[13] Gillis, M 1999, ‘Open systems joint task force gets the word out’, *PM Magazine*, July-August, pp. 44-47.

[14] Naval Open Architecture Enterprise Team, 2009, *Key Open Sub-System (KOSS) Tool: KOSS Description and Application*, *Public Release SPR-09-674*, NAVAIR, Patuxent River, MD.