CoWare™

The ESL Design Leader

The Use Of Virtual Platforms In MP-SoC Design

Eshel Haritan, VP Engineering CoWare Inc.

MPSoC 2006

MPSoC

- Is MP SoC design happening?

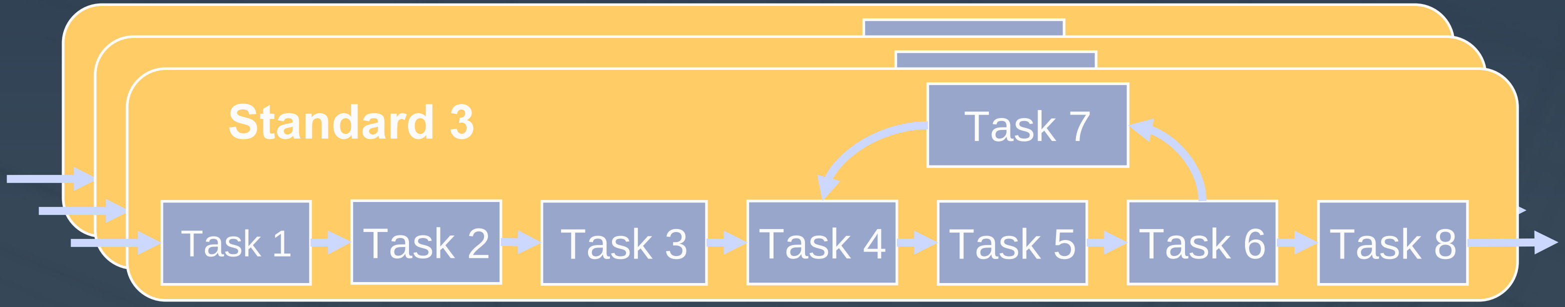
- Why?
  - Consumer Electronics → Complexity
  - Cost of ASIC → Increased SW Content
  - Power consumption → MP SoC

- …..and it comes in many flavors

Agenda

- MPSoC Solution Space
- Virtual HW Platforms
  - What are Virtual Platforms?
  - Virtual Platform usage for Architectural Exploration
  - Virtual Platform usage for SW development
- Requirements from Virtual Platforms
- The ESL Solution Pyramid

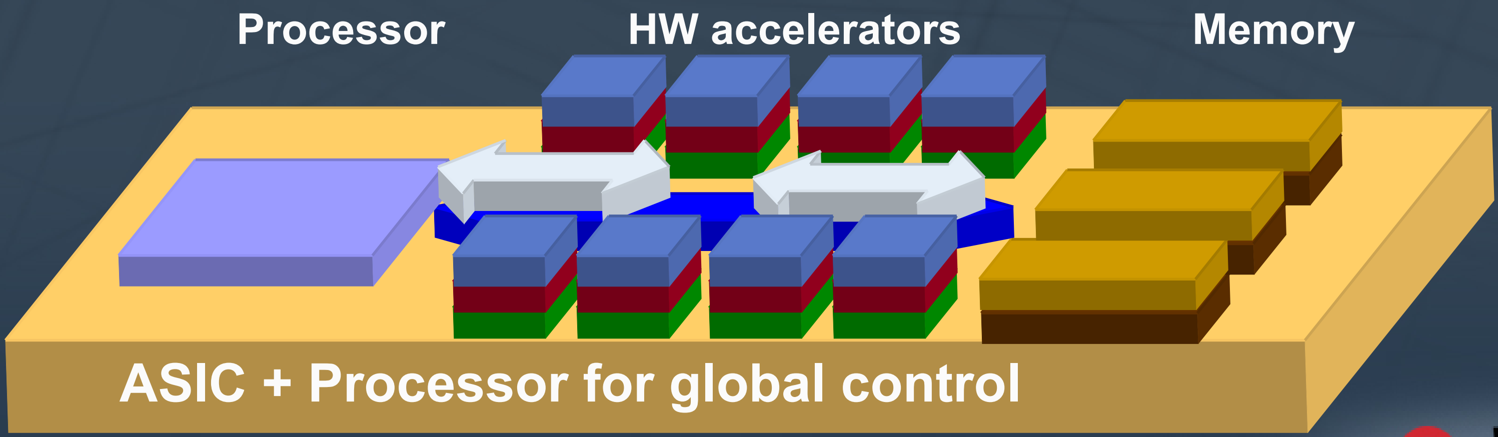
Solution Space: HW Implementation



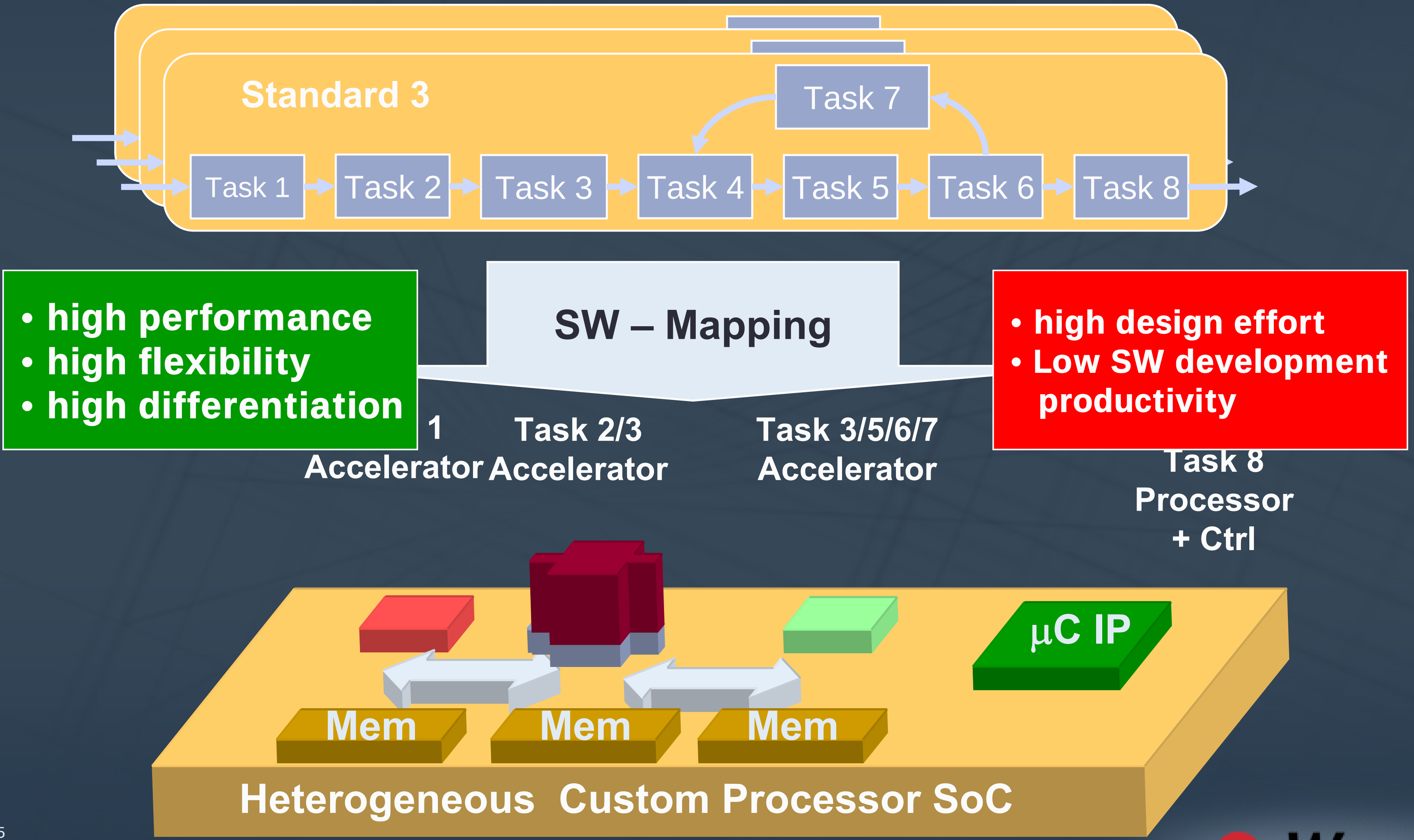
- High performance
- Low power

HW – Mapping

- Limited reuse
- No flexibility
- separate design for each platform



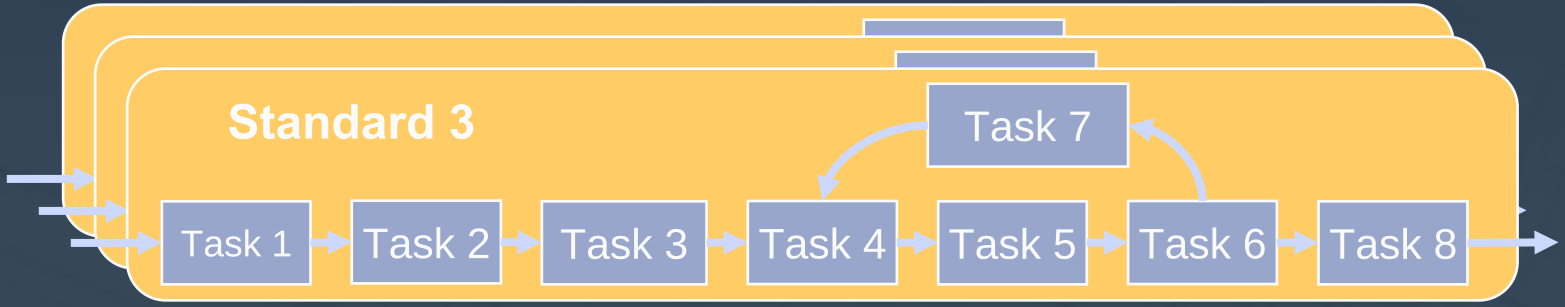
Solution Space: Programmable Accelerators



- high performance
- high flexibility
- high differentiation

- high design effort
- Low SW development productivity

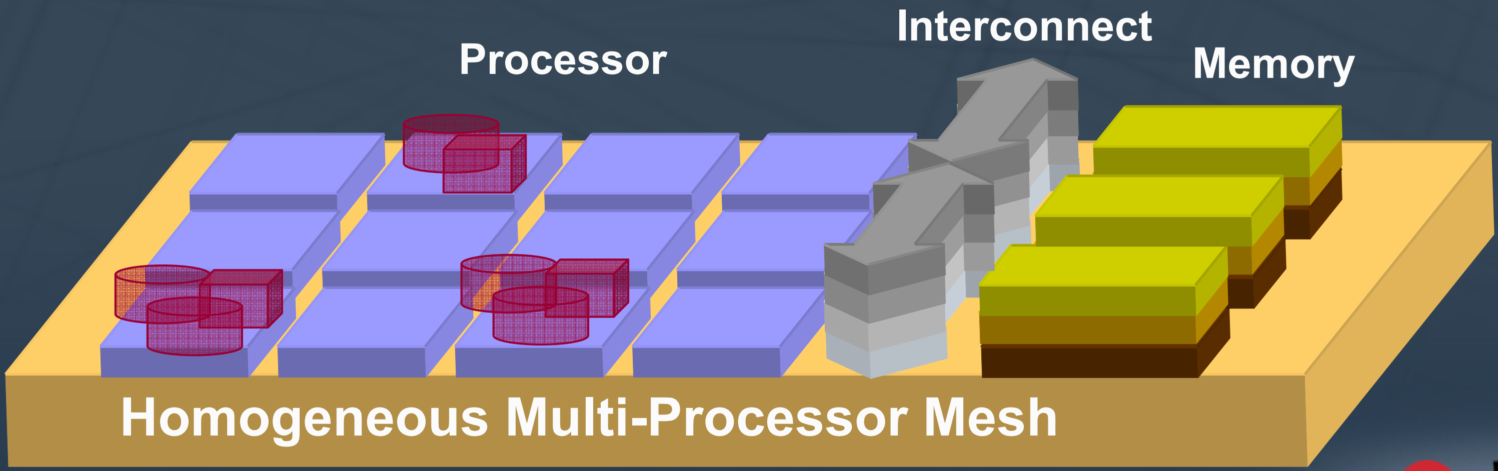
Homogeneous MP-SoC Mesh



- Design flexibility
- Reuse for multiple apps
- Redundancy

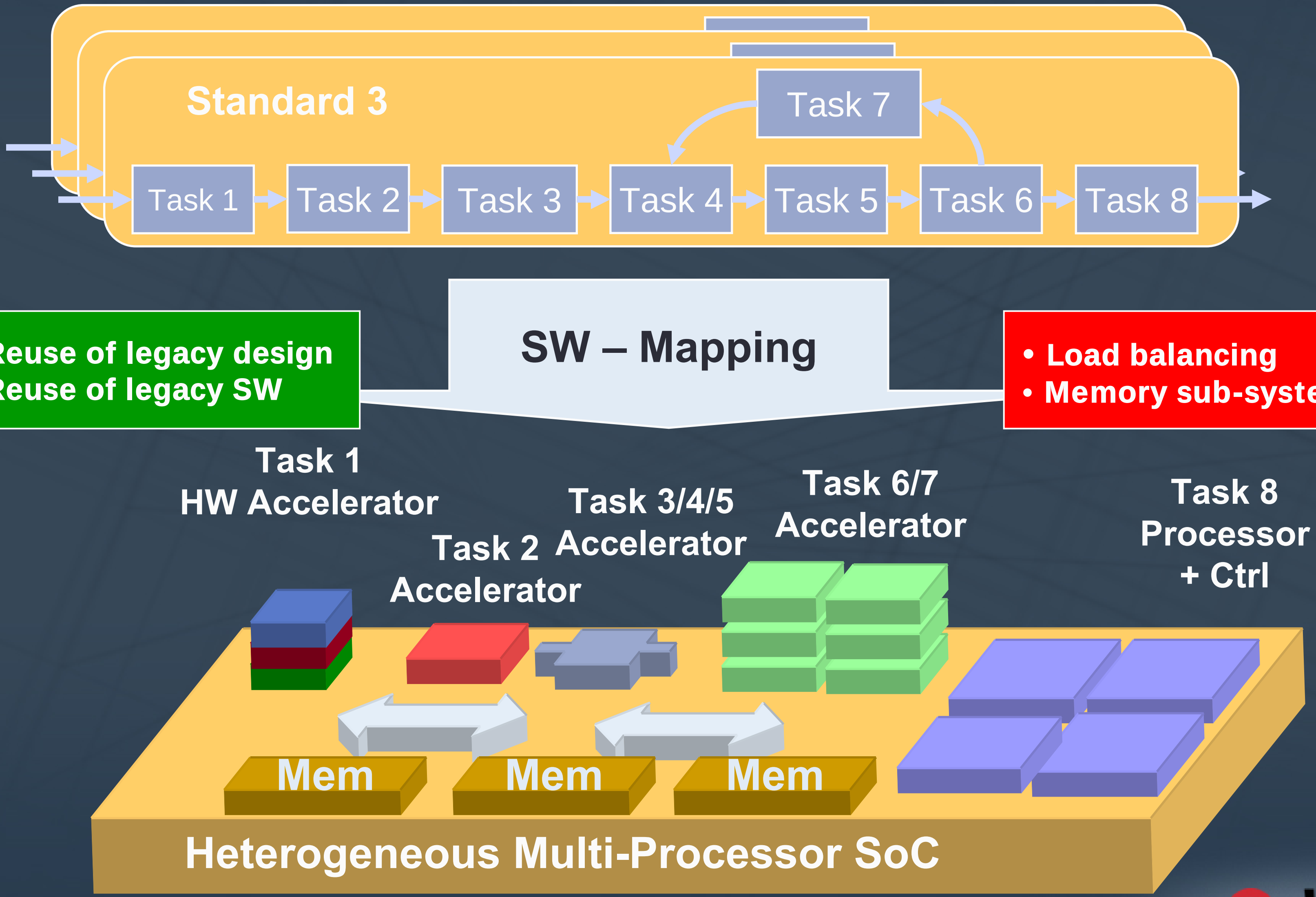
SW – Mapping

- Difficult to program
- Communication becomes bottleneck
- Load Balancing
- High chip / royalty cost



CoWare

Solution Space: Heterogeneous SoC

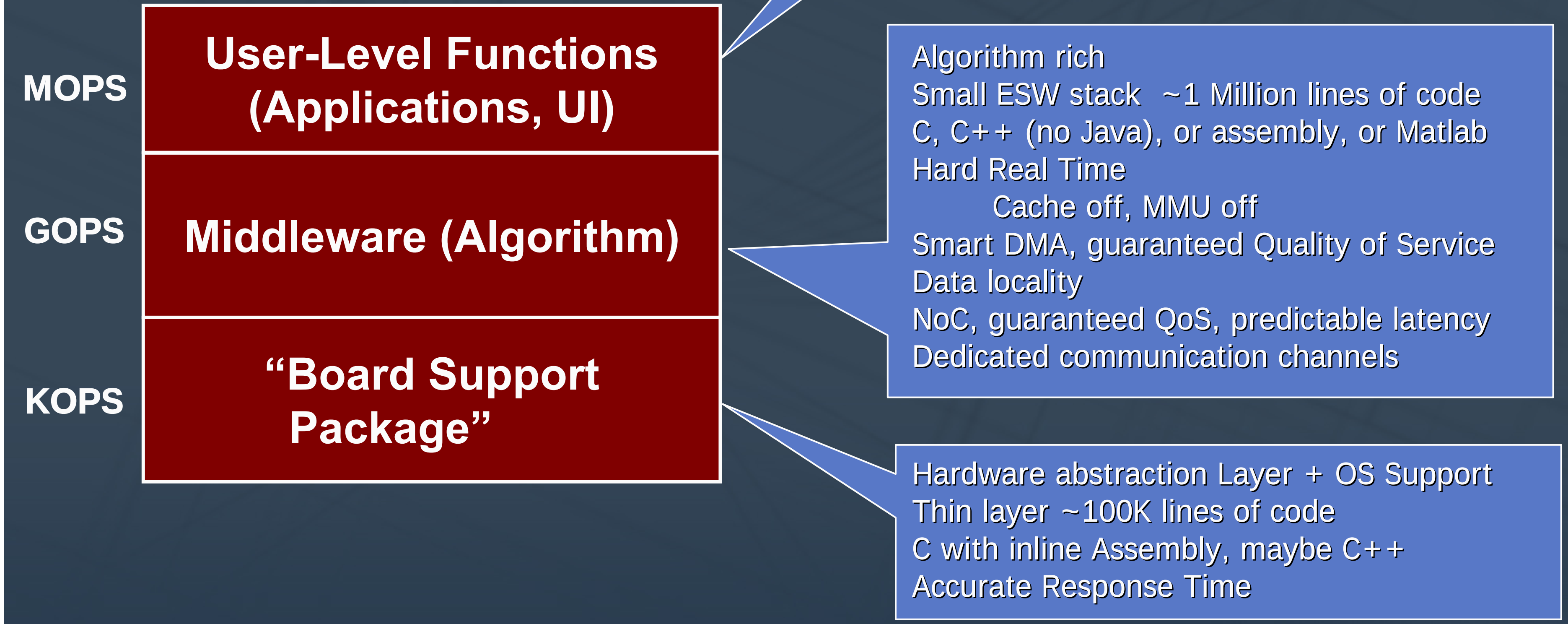


- Reuse of legacy design
- Reuse of legacy SW

- Load balancing
- Memory sub-system design

CoWare

The ESW View



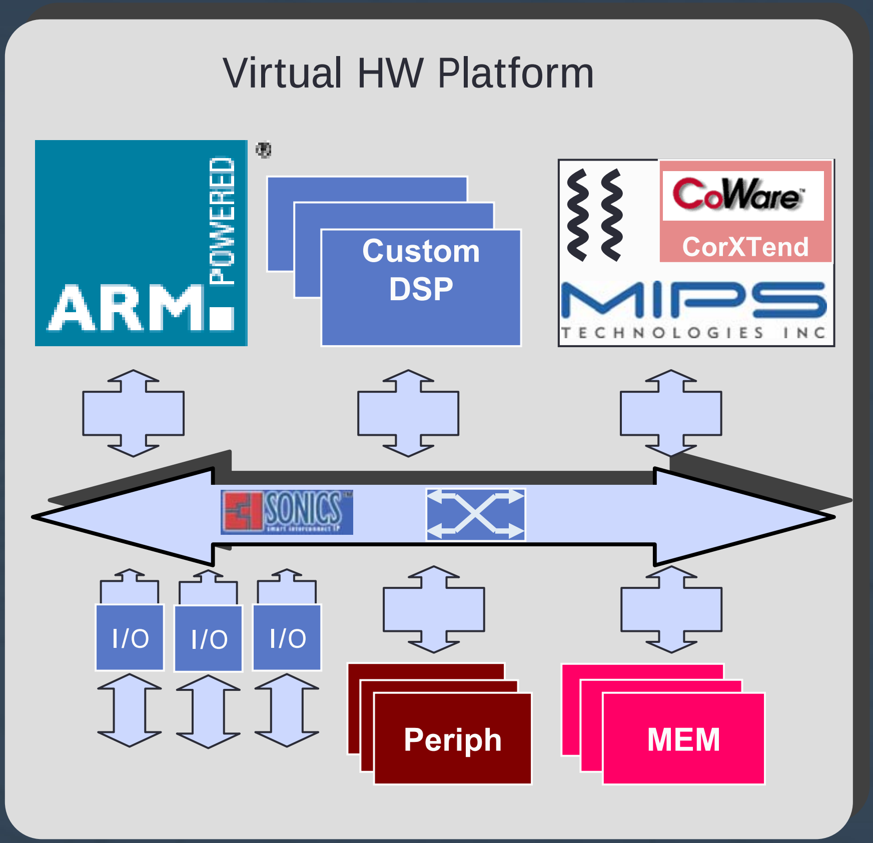
User Interface, User level functions
Large ESW stack >10 Million lines of code
C, C++, Java
Soft real time - Defined by user perception
Cache, MMU
Use generic components for data movement
Generic bus arbitration + peripherals

Algorithm rich
Small ESW stack ~1 Million lines of code
C, C++ (no Java), or assembly, or Matlab
Hard Real Time
    Cache off, MMU off
Smart DMA, guaranteed Quality of Service
Data locality
NoC, guaranteed QoS, predictable latency
Dedicated communication channels

Hardware abstraction Layer + OS Support
Thin layer ~100K lines of code
C with inline Assembly, maybe C++
Accurate Response Time

What is a Virtual HW Platform?

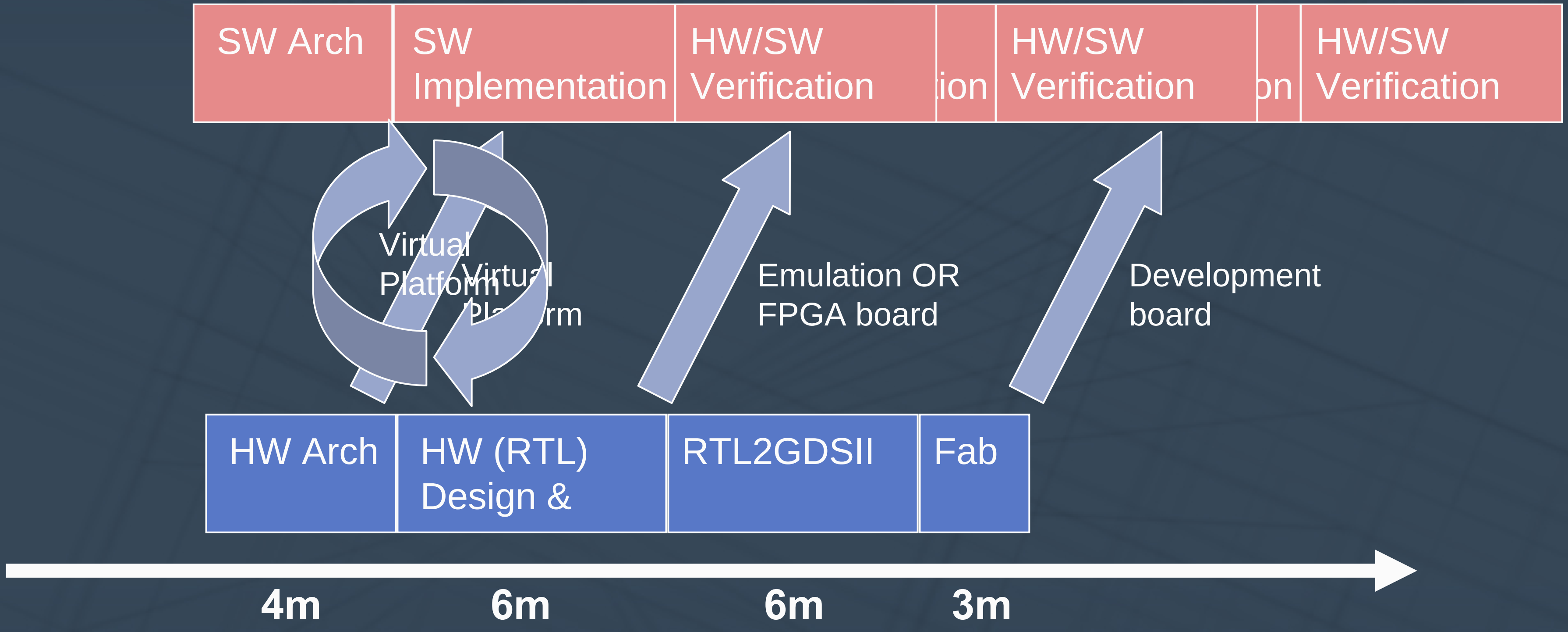
- A SW model of the SoC HW
  - Processors, accelerators, peripherals and interconnect
  - HDS – HAL, Drivers, O.S., I/O
- Enables architecture exploration and optimization
- Enables ESW development, debugging and optimization
- Different abstraction levels for different use models



Virtual HW Platform Value

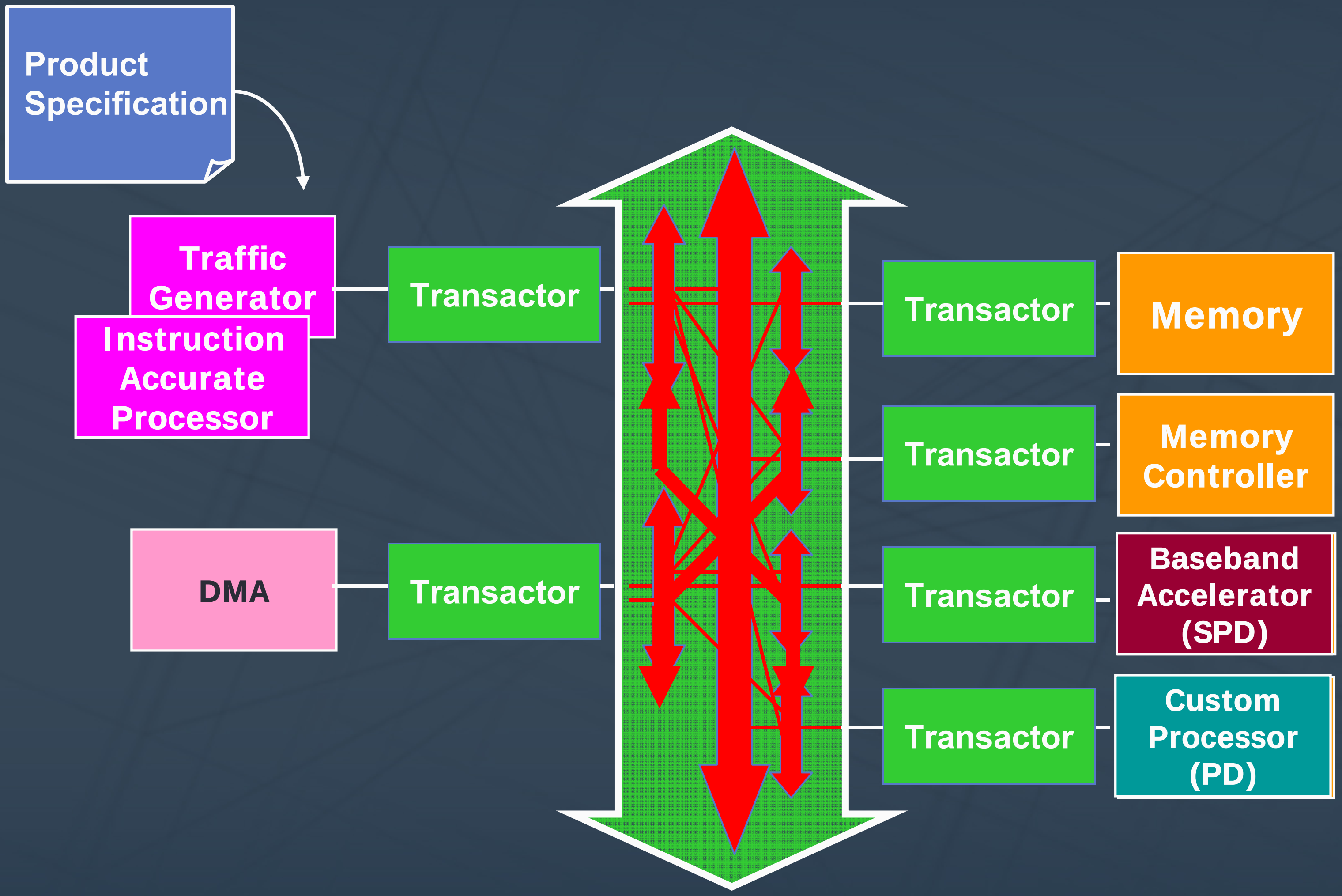
- Fast
- Allows accuracy-speed trade offs
- Flexible
- Cost effective
- Scaleable
- Observable and Controllable
- Available early in the design cycle

Virtual HW Platform Value



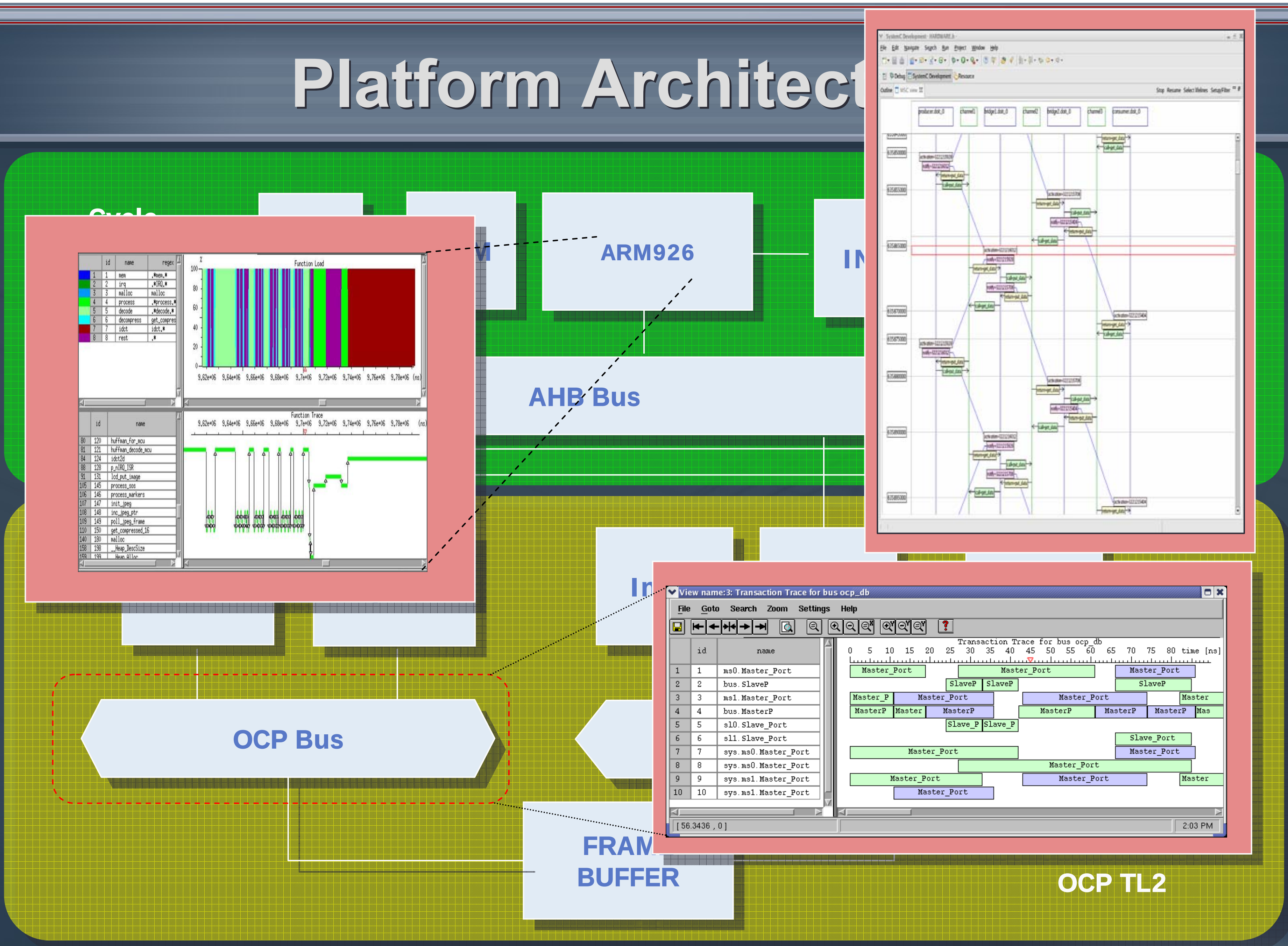
- **Early SW Development**
- **Real HW and SW Co-Design**

Virtual HW Platform for Architecture Exploration



CoWare

Platform Architect

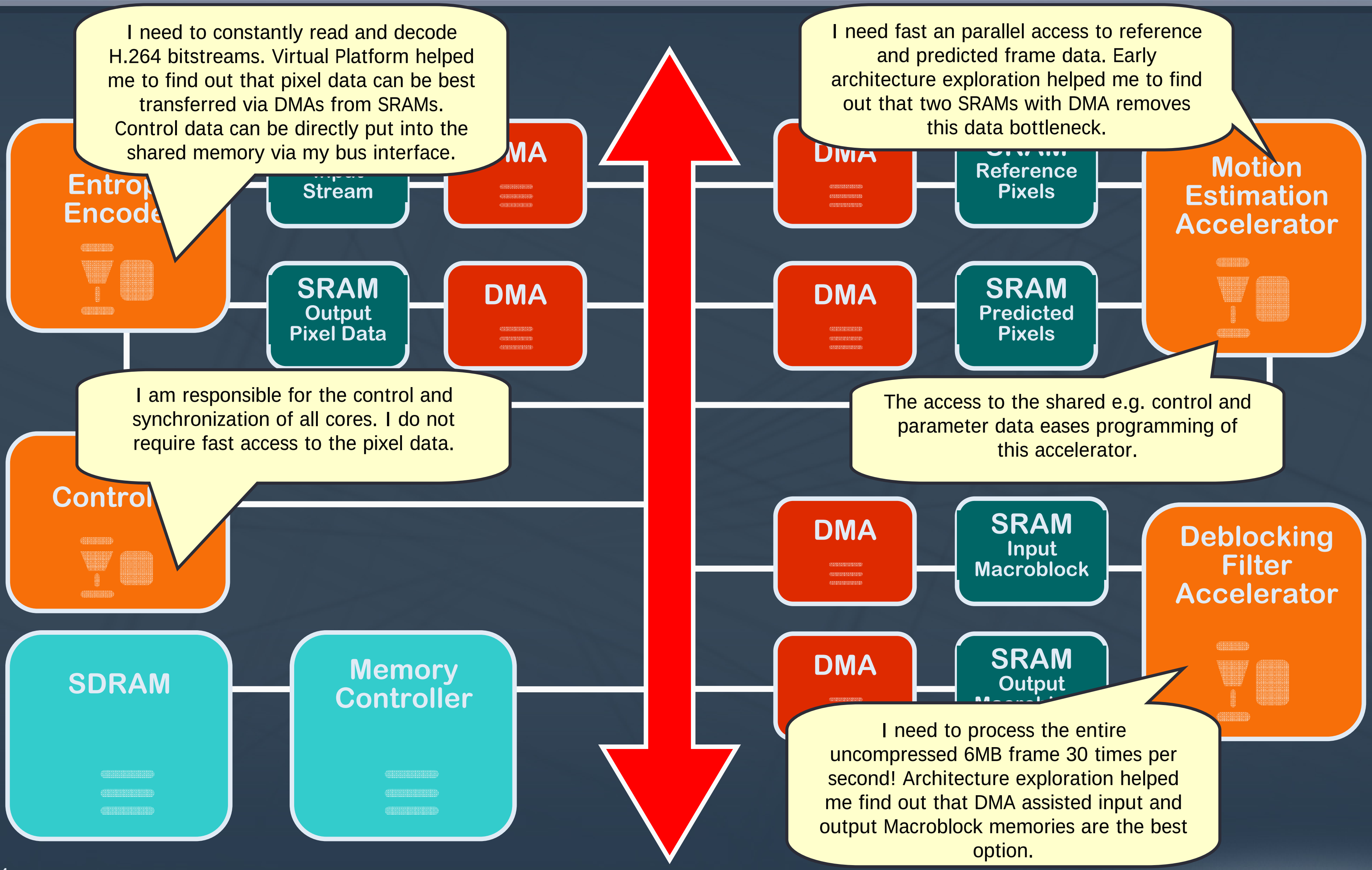


	id	name	regex
🟦 1	1	mem	.*mem.*
🟩 2	2	irq	.*IRQ.*
🟦 3	3	malloc	malloc
🟩 4	4	process	.*process.*
🟩 5	5	decode	.*decode.*
🟦 6	6	decompress	get_compres
🟥 7	7	idct	idct.*
🟪 8	8	rest	.*

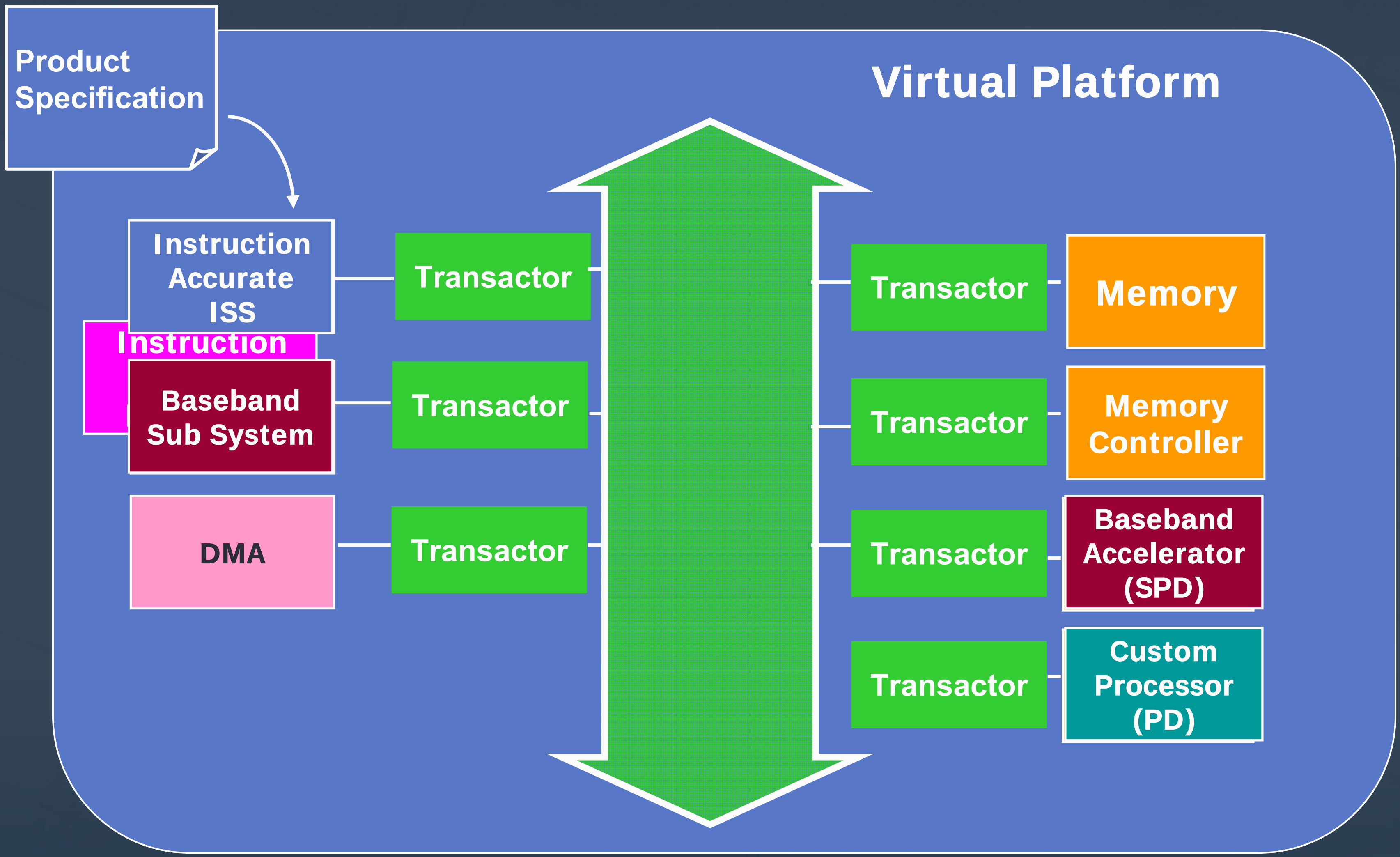
	id	name
80	120	huffman_for_mcu
81	121	huffman_decode_mcu
84	124	idct2d
88	128	p_nIRQ_ISR
91	131	lcd_put_image
105	145	process_sos
106	146	process_markers
107	147	init_jpeg
108	148	inc_jpeg_ptr
109	149	poll_jpeg_frame
110	150	get_compressed_16
140	180	malloc
158	198	__Heap_DescSize
159	199	__Heap_Alloc

	id	name
1	1	ms0.Master_Port
2	2	bus.SlaveP
3	3	ms1.Master_Port
4	4	bus.MasterP
5	5	sl0.Slave_Port
6	6	sl1.Slave_Port
7	7	sys.ms0.Master_Port
8	8	sys.ms0.Master_Port
9	9	sys.ms1.Master_Port
10	10	sys.ms1.Master_Port

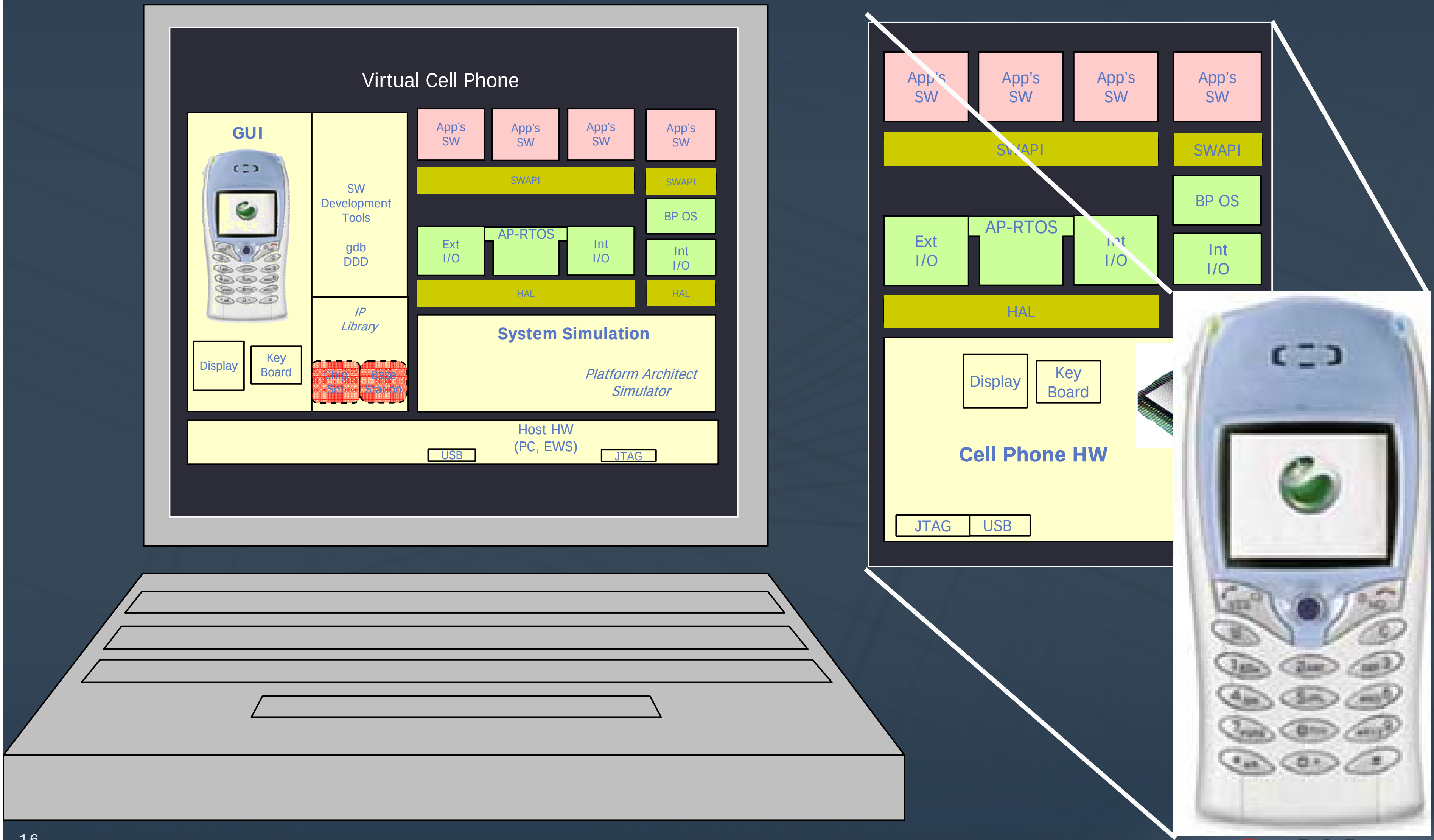
Virtual Platforms for Architecture Exploration


I need to constantly read and decode H.264 bitstreams. Virtual Platform helped me to find out that pixel data can be best transferred via DMAs from SRAMs. Control data can be directly put into the shared memory via my bus interface.
I need fast an parallel access to reference and predicted frame data. Early architecture exploration helped me to find out that two SRAMs with DMA removes this data bottleneck.
Entrop Encode
Stream
MA
DMA
Reference Pixels
Motion Estimation Accelerator
SRAM Output Pixel Data
DMA
DMA
SRAM Predicted Pixels
I am responsible for the control and synchronization of all cores. I do not require fast access to the pixel data.
The access to the shared e.g. control and parameter data eases programming of this accelerator.
Control
DMA
SRAM Input Macroblock
Deblocking Filter Accelerator
SDRAM
Memory Controller
DMA
SRAM Output
I need to process the entire uncompressed 6MB frame 30 times per second! Architecture exploration helped me find out that DMA assisted input and output Macroblock memories are the best option.

Virtual HW Platform for ESW Development

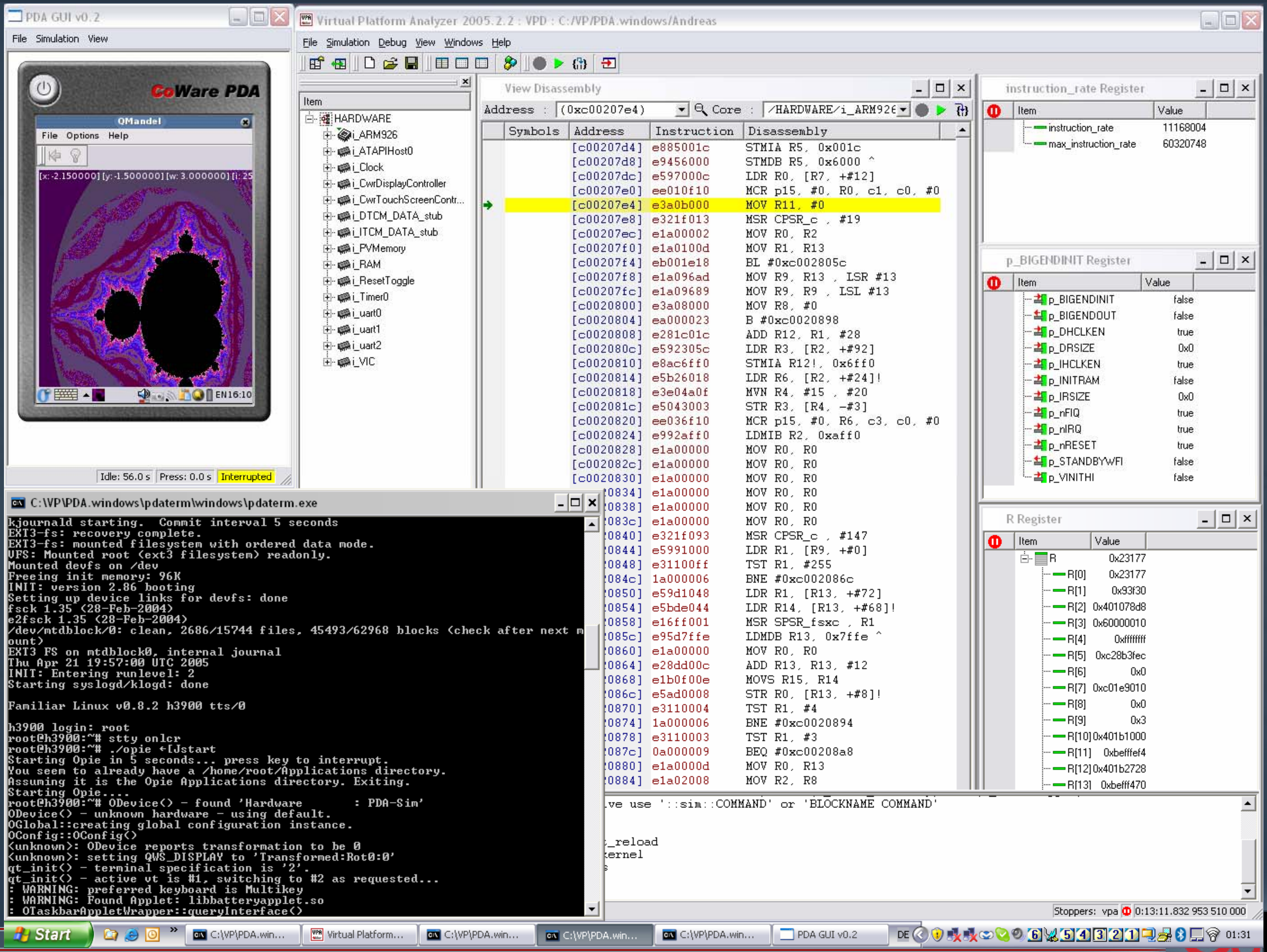


Virtual Platforms for ESW Development



CoWare

Virtual HW Platform for ESW development



```
Symbols  Address     Instruction  Disassembly
         [c00207d4]  e885001c     STMIA R5, 0x001c
         [c00207d8]  e9456000     STMDB R5, 0x6000 ^
         [c00207dc]  e597000c     LDR R0, [R7, +#12]
         [c00207e0]  ee010f10     MCR p15, #0, R0, c1, c0, #0
         [c00207e4]  e3a0b000     MOV R11, #0
         [c00207e8]  e321f013     MSR CPSR_c , #19
         [c00207ec]  e1a00002     MOV R0, R2
         [c00207f0]  e1a0100d     MOV R1, R13
         [c00207f4]  eb001e18     BL #0xc002805c
         [c00207f8]  e1a096ad     MOV R9, R13 , LSR #13
         [c00207fc]  e1a09689     MOV R9, R9 , LSL #13
         [c0020800]  e3a08000     MOV R8, #0
         [c0020804]  ea000023     B #0xc0020898
         [c0020808]  e281c01c     ADD R12, R1, #28
         [c002080c]  e592305c     LDR R3, [R2, +#92]
         [c0020810]  e8ac6ff0     STMIA R12!, 0x6ff0
         [c0020814]  e5b26018     LDR R6, [R2, +#24]!
         [c0020818]  e3e04a0f     MVN R4, #15 , #20
         [c002081c]  e5043003     STR R3, [R4, -#3]
         [c0020820]  ee036f10     MCR p15, #0, R6, c3, c0, #0
         [c0020824]  e992aff0     LDMIB R2, 0xaff0
         [c0020828]  e1a00000     MOV R0, R0
         [c002082c]  e1a00000     MOV R0, R0
         [c0020830]  e1a00000     MOV R0, R0
          0834]      e1a00000     MOV R0, R0
          0838]      e1a00000     MOV R0, R0
          083c]      e1a00000     MOV R0, R0
          0840]      e321f093     MSR CPSR_c , #147
          0844]      e5991000     LDR R1, [R9, +#0]
          0848]      e31100ff     TST R1, #255
          084c]      1a000006     BNE #0xc002086c
          0850]      e59d1048     LDR R1, [R13, +#72]
          0854]      e5bde044     LDR R14, [R13, +#68]!
          0858]      e16ff001     MSR SPSR_fsxc , R1
          085c]      e95d7ffe     LDMDB R13, 0x7ffe ^
          0860]      e1a00000     MOV R0, R0
          0864]      e28dd00c     ADD R13, R13, #12
          0868]      e1b0f00e     MOVS R15, R14
          086c]      e5ad0008     STR R0, [R13, +#8]!
          0870]      e3110004     TST R1, #4
          0874]      1a000006     BNE #0xc0020894
          0878]      e3110003     TST R1, #3
          087c]      0a000009     BEQ #0xc00208a8
          0880]      e1a0000d     MOV R0, R13
          0884]      e1a02008     MOV R2, R8
```

Item	Value
instruction_rate	11168004
max_instruction_rate	60320748

Item	Value
p_BIGENDINIT	false
p_BIGENDOUT	false
p_DHCLKEN	true
p_DRSIZE	0x0
p_IHCLKEN	true
p_INITRAM	false
p_IRSIZE	0x0
p_nFIQ	true
p_nIRQ	true
p_nRESET	true
p_STANDBYWFI	false
p_VINITHI	false

Item	Value
R	0x23177
R[0]	0x23177
R[1]	0x93f30
R[2]	0x401078d8
R[3]	0x60000010
R[4]	0xffffffff
R[5]	0xc28b3fec
R[6]	0x0
R[7]	0xc01e9010
R[8]	0x0
R[9]	0x3
R[10]	0x401b1000
R[11]	0xbefffef4
R[12]	0x401b2728
R[13]	0xbefff470

```
ve use '::sim::COMMAND' or 'BLOCKNAME COMMAND'


_reload
ernel
s
```

```
kjournald starting.  Commit interval 5 seconds
EXT3-fs: recovery complete.
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Mounted devfs on /dev
Freeing init memory: 96K
INIT: version 2.86 booting
Setting up device links for devfs: done
fsck 1.35 (28-Feb-2004)
e2fsck 1.35 (28-Feb-2004)
/dev/mtdblock/0: clean, 2686/15744 files, 45493/62968 blocks (check after next m
ount)
EXT3 FS on mtdblock0, internal journal
Thu Apr 21 19:57:00 UTC 2005
INIT: Entering runlevel: 2
Starting syslogd/klogd: done

Familiar Linux v0.8.2 h3900 tts/0

h3900 login: root
root@h3900:~# stty onlcr
root@h3900:~# ./opie ←[Jstart
Starting Opie in 5 seconds... press key to interrupt.
You seem to already have a /home/root/Applications directory.
Assuming it is the Opie Applications directory. Exiting.
Starting Opie....
root@h3900:~# ODevice() - found 'Hardware       : PDA-Sim'
ODevice() - unknown hardware - using default.
OGlobal::creating global configuration instance.
OConfig::OConfig()
<unknown>: ODevice reports transformation to be 0
<unknown>: setting QWS_DISPLAY to 'Transformed:Rot0:0'
qt_init() - terminal specification is '2'.
qt_init() - active vt is #1, switching to #2 as requested...
: WARNING: preferred keyboard is Multikey
: WARNING: Found Applet: libbatteryapplet.so
: OTaskbarAppletWrapper::queryInterface()
```

Virtual Platform – What is Required?

- Ultra fast processor simulator – ISS
- Accuracy-speed trade-offs
- Library of fast processor models

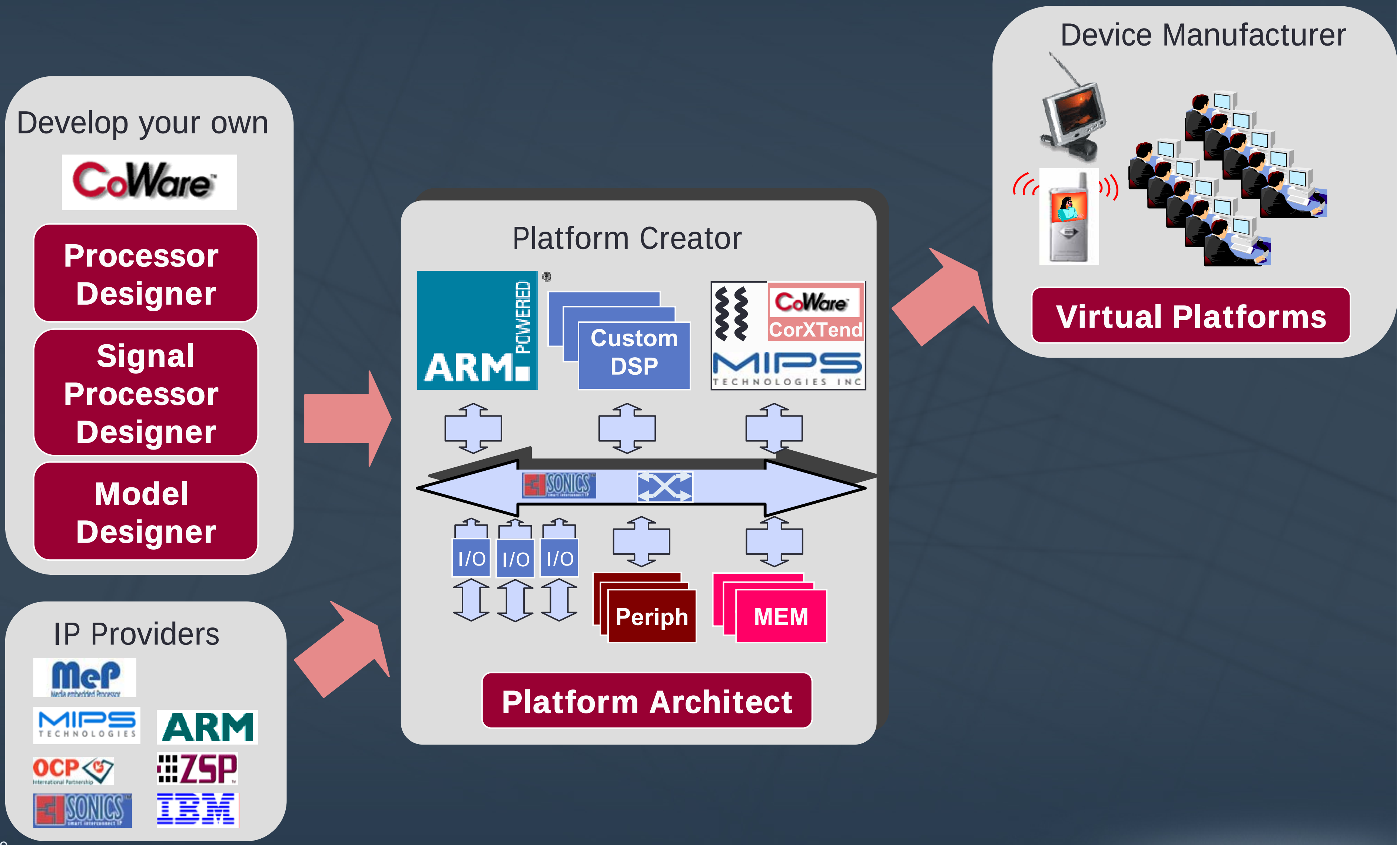
- Explore different communication paradigms
- Strong analysis solutions
- Library of communication protocols

- Integrate DSP algorithm
- Library of popular Wireless and MM standards

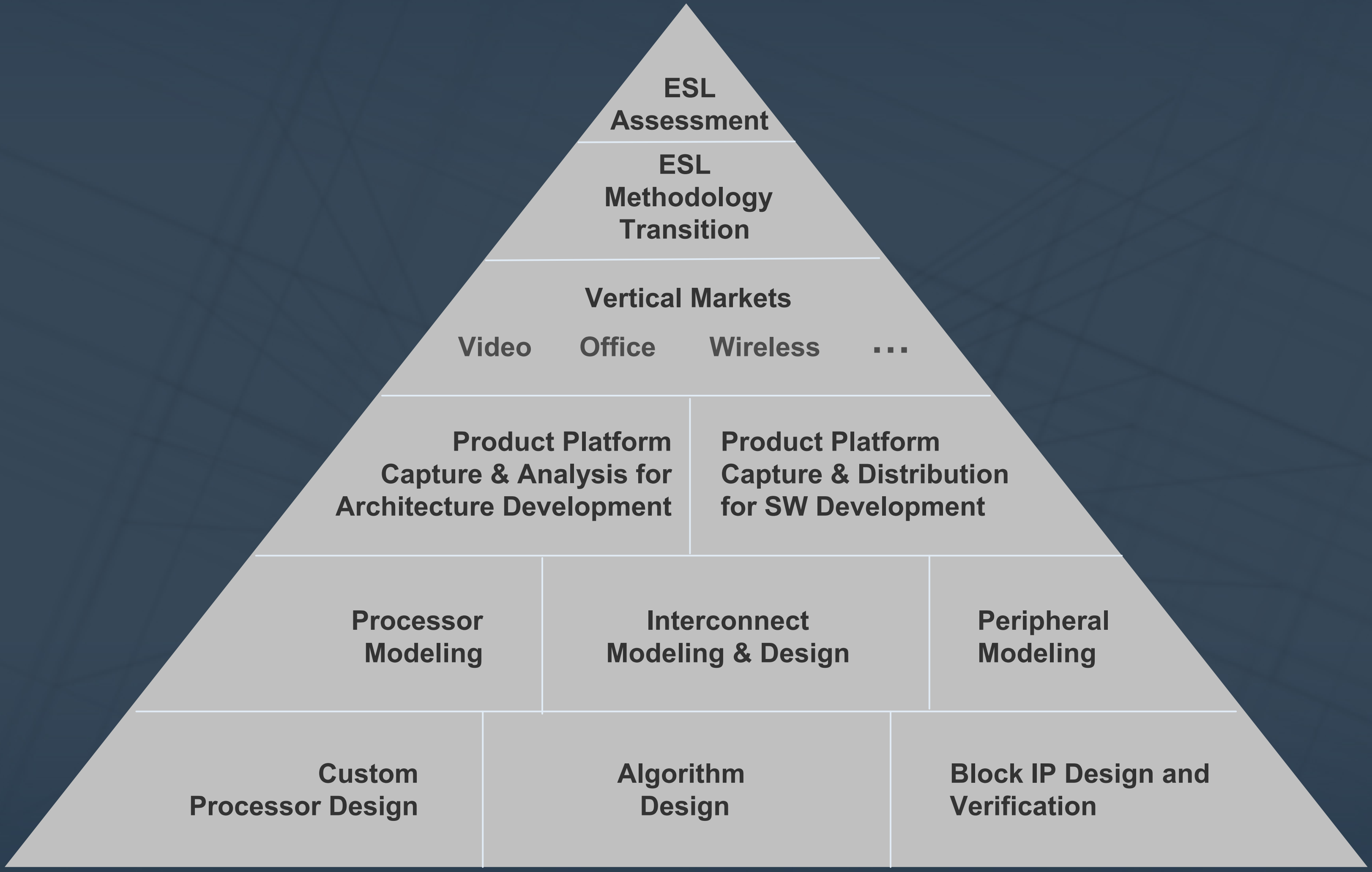
- Ultra fast platform
- Accuracy-speed trade-offs
- Multi core and platform debugging

- Standard based

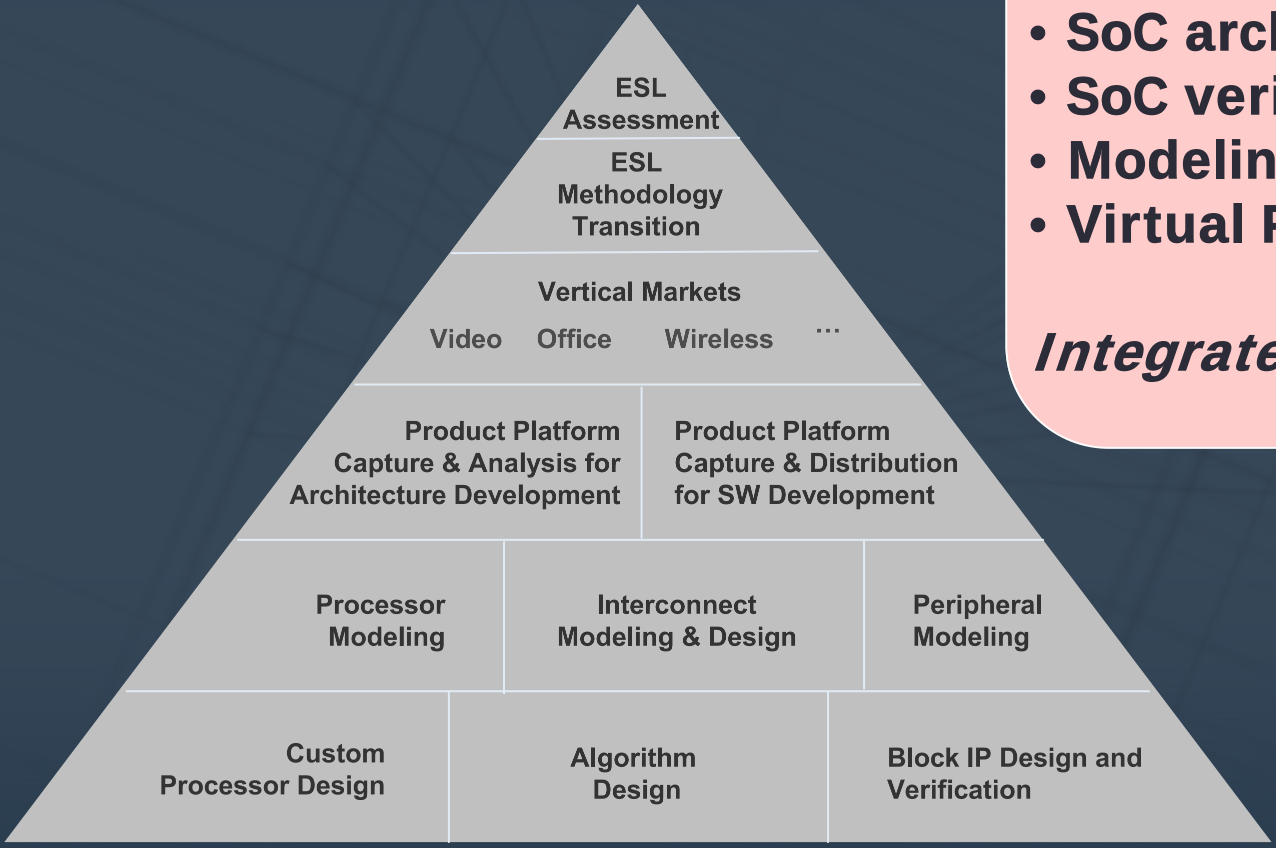
Why Standards Based?



Solutions Pyramid


ESL Assessment
ESL Methodology Transition
Vertical Markets
Video
Office
Wireless
...
Product Platform Capture & Analysis for Architecture Development
Product Platform Capture & Distribution for SW Development
Processor Modeling
Interconnect Modeling & Design
Peripheral Modeling
Custom Processor Design
Algorithm Design
Block IP Design and Verification

Summary

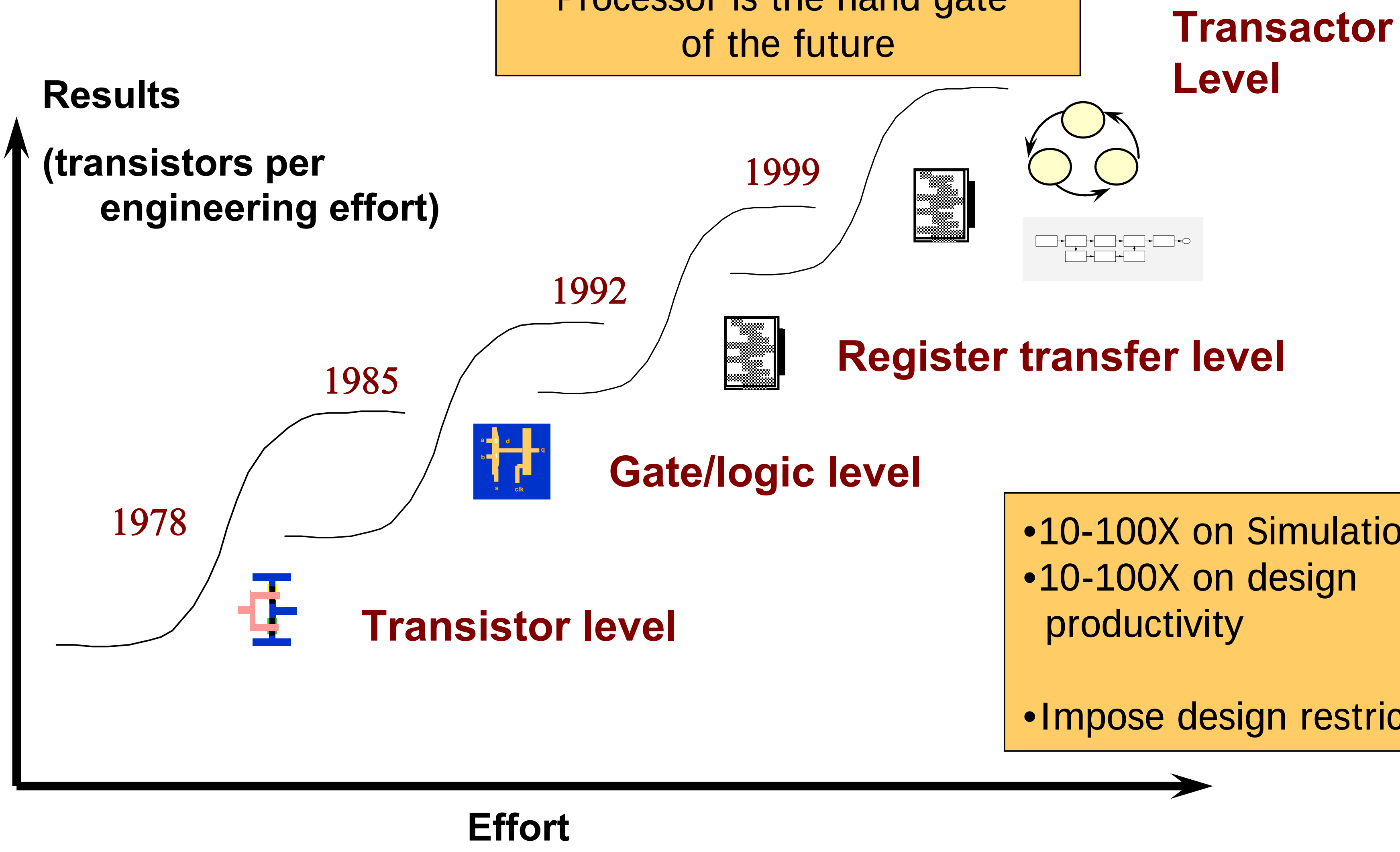

ESL Assessment
ESL Methodology Transition
Vertical Markets
Video
Office
Wireless
...
Product Platform Capture & Analysis for Architecture Development
Product Platform Capture & Distribution for SW Development
Processor Modeling
Interconnect Modeling & Design
Peripheral Modeling
Custom Processor Design
Algorithm Design
Block IP Design and Verification

- Custom processor design
- Complex algorithm design
- SoC architecture exploration
- SoC verification
- Modeling for speed
- Virtual Platforms for ESW

Integrated solution Pyramid

So what’s the next level of abstraction?

Processor is the nand gate of the future



- 10-100X on Simulation
- 10-100X on design productivity

- Impose design restrictions

The Next Level Challenge

- The next level is (maybe):
  - Processors and HW accelerators
  - SW tasks running on these processors
  - Transaction level modeling to model communication
- What are the design restrictions we need to impose to gain 100X productivity?...and what are the benefits to the users?
- Processors
  - Simplify? No Cache? No MMU? Speculative execution? Branch prediction?
- SW tasks
  - Threads? Locks? R/W to/from ports only?
  - Programming models to abstract the SoC (message passing, SMP, streaming)?
  - What do we do with all the legacy code?
- Communication
  - Memory sub system design, NoC design