OpenMP Accelerator Model for TI's Keystone DSP+ARM Devices

SC13, Denver, CO

Eric Stotzer

Ajay Jayaraj

High Performance Embedded Computing

DVR / NVR & smart camera

Networking

Mission critical systems

Medical imaging

Video and audio infrastructure

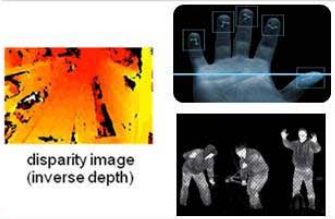
High-performance and cloud computing

Portable mobile radio

Industrial imaging

Home AVR and automotive audio

Analytics



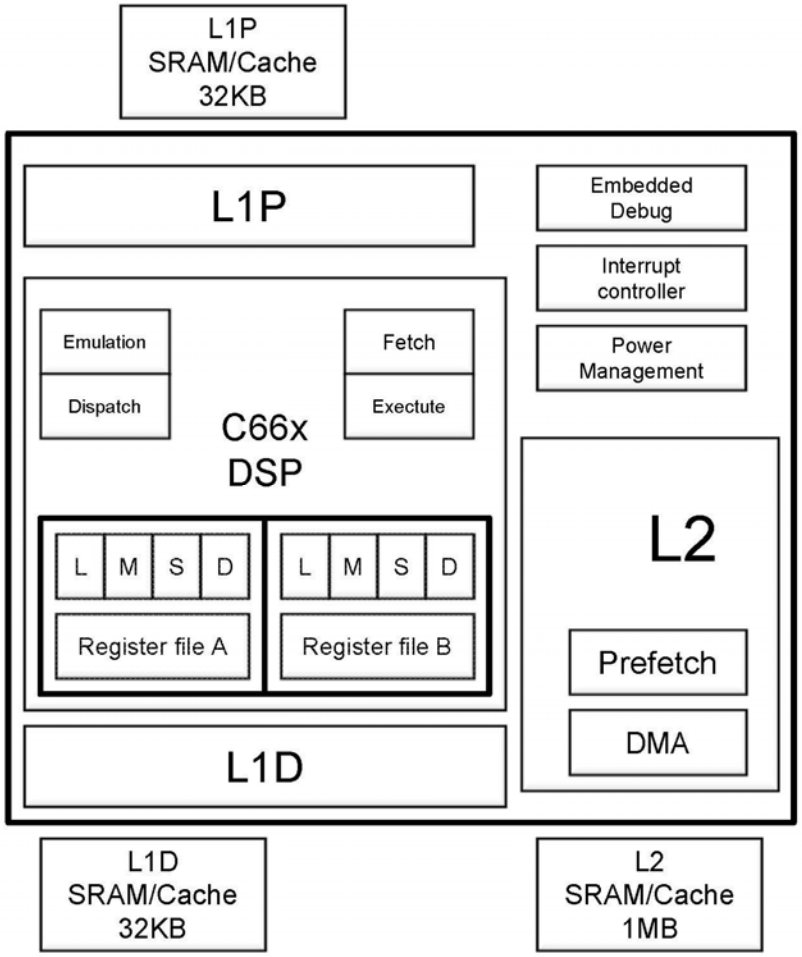
Wireless testers

Industrial control

media processing | *computing* | *radar & communications* | *industrial electronics*

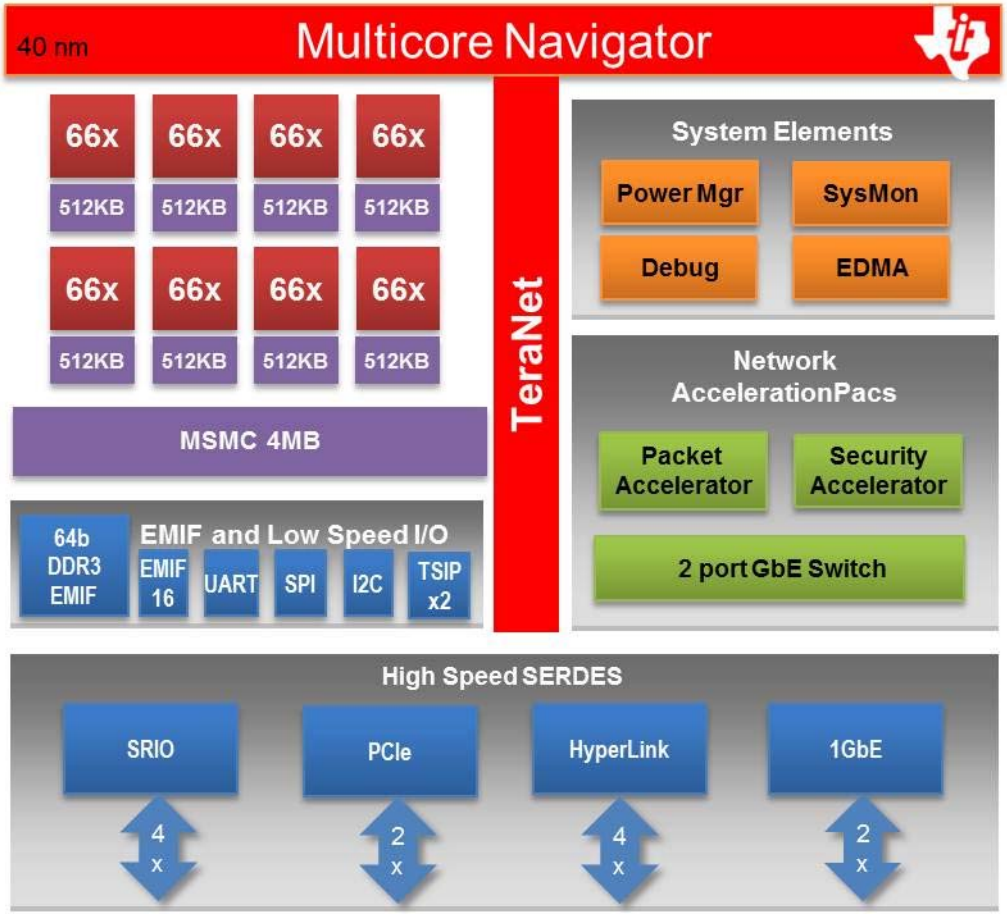
C66x Core Architecture

- 8-way VLIW processor
- 8 functional units in two sides
  - M: multiplication
  - D: load/store
  - L and S: addition and branch
- SIMD up to 128-bit vectors
  - M: 4 SP multiplies/cycle
  - L and S: 2 SP add/cycle
- 8 MAC/cycle
  - 16 GFLOPS (SP) @ 1 Ghz



Keystone I: C6678 SoC

- Eight 8 C66x cores
- Each with 32k L1P, 32k L1D, 512k L2
- 1 to 1.25 GHz
- 320 GMACS
- 160 SP GFLOPS
- 512 KB/Core of local L2
- 4MB Multicore Shared Memory (MSMC)
- Multicore Navigator (8k HW queues) and TeraNet
- Serial-RapidIO, PCIe-II, Ethernet, 1xHyperlink



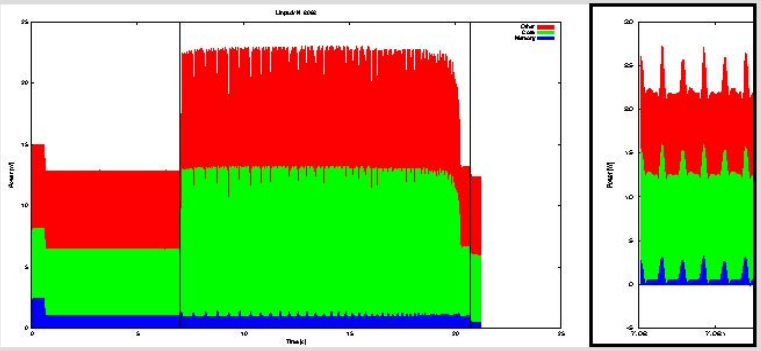
24mm x 24mm package

Energy Efficiency

LINPACK running on C6678 achieves 25.6 Gflops, ~2.1 Gflops/W

PRACE First Implementation Project, Grant RI-261557, Final Report on Prototypes Evaluation. Lennart Johnsson, Gilbert Netzer, SNIC/KTH, 3/29/2013.

Linpack Power Profile



The plot shows the power consumption over time during a single execution of the Linpack benchmark code. Blue shows memory power, green is added power fed to the DSP and red other module consumers stacked atop. The vertical lines denote the timed section of the code. Distinct phases of execution can be seen, for instance the serial back-substitution at the end of the run. A zoom in also reveals the power peaks caused by DMA block copies to and from the main memory.

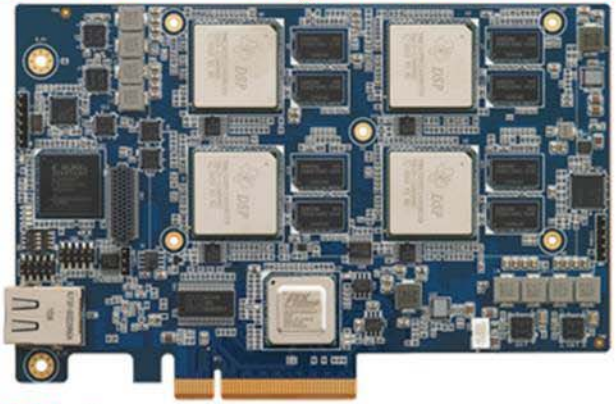
DSP Linpack Energy Efficiency

Size	Perf.	Eff.	Core	Mem.	Other	Total	Core + Mem	Tot
	GF/s	*%*	*W*	*W*	*W*	*W*	*MF/J*	*MF/J*
127	1.3	4	5.95	1.26	6.87	14.08	176	90
255	2.8	9	4.78	0.99	5.17	10.95	493	260
511	6.0	19	6.40	1.12	6.58	14.09	796	425
1023	11.3	35	8.02	1.19	7.65	16.86	1230	672
2047	16.9	53	9.16	1.10	8.13	18.40	1649	920
4095	22.0	69	10.30	1.03	8.70	20.03	1939	1097
8063	25.6	80	11.20	0.99	9.20	21.39	2097	1195

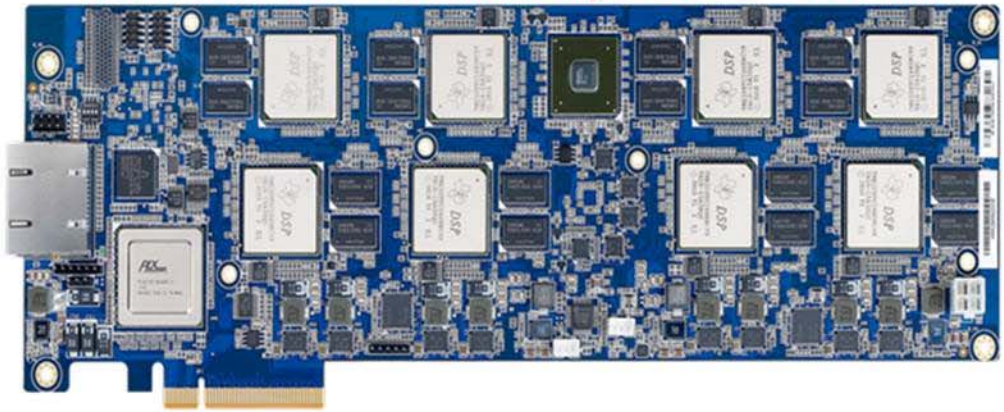
The table shows the power and energy consumption of the major components of the C6678 DSP EVM. Core refers to the C6678 DSP SoC excluding I/O power. Mem is the DDR3 memory subsytem. The 5-9 watts of "Other" power is to a large part, except for about 1.5 W DC converter losses, consumed by debugging and unused hardware features that would not be present in an HPC server node. Therefore the "Core + Mem" power is a good estimate for the energy efficiency of an HPC server node. The values for small problem sizes show deviations due to various parts of the benchmark not being executed. The outmost loop step size is 128 columns, another breakpoint occurs at 1024.

High Density COTS boards

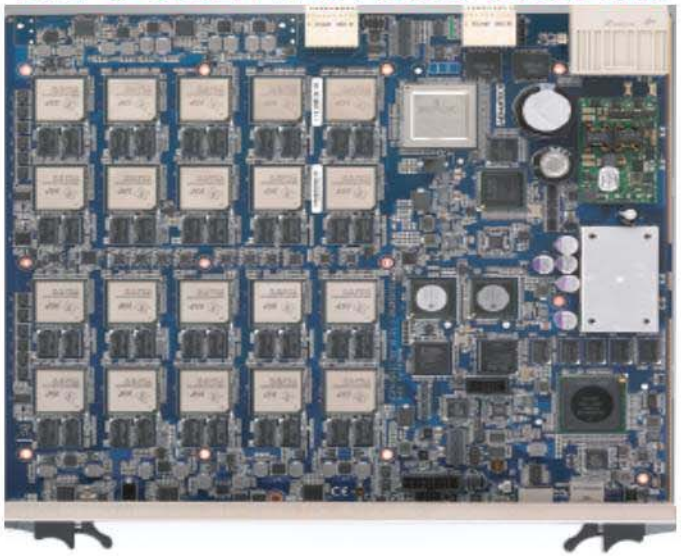
DSPC-8681 ½ length PCIe card - 54Watts



DSPC-8682 PCIe Full-Length Card - 110Watts



DSPC-8682 ATCA blade 350Watts



Keystone II: 66AK2H12/06 SoC

C66x Fixed or Floating Point DSP

- 4x/8x 66x DSP cores up to 1.4GHz
- 2x/4x Cotex ARM A15
- 1MB of local L2 cache RAM per C66 DSP core
- 4MB shared across all ARM

Large on chip and off chip memory

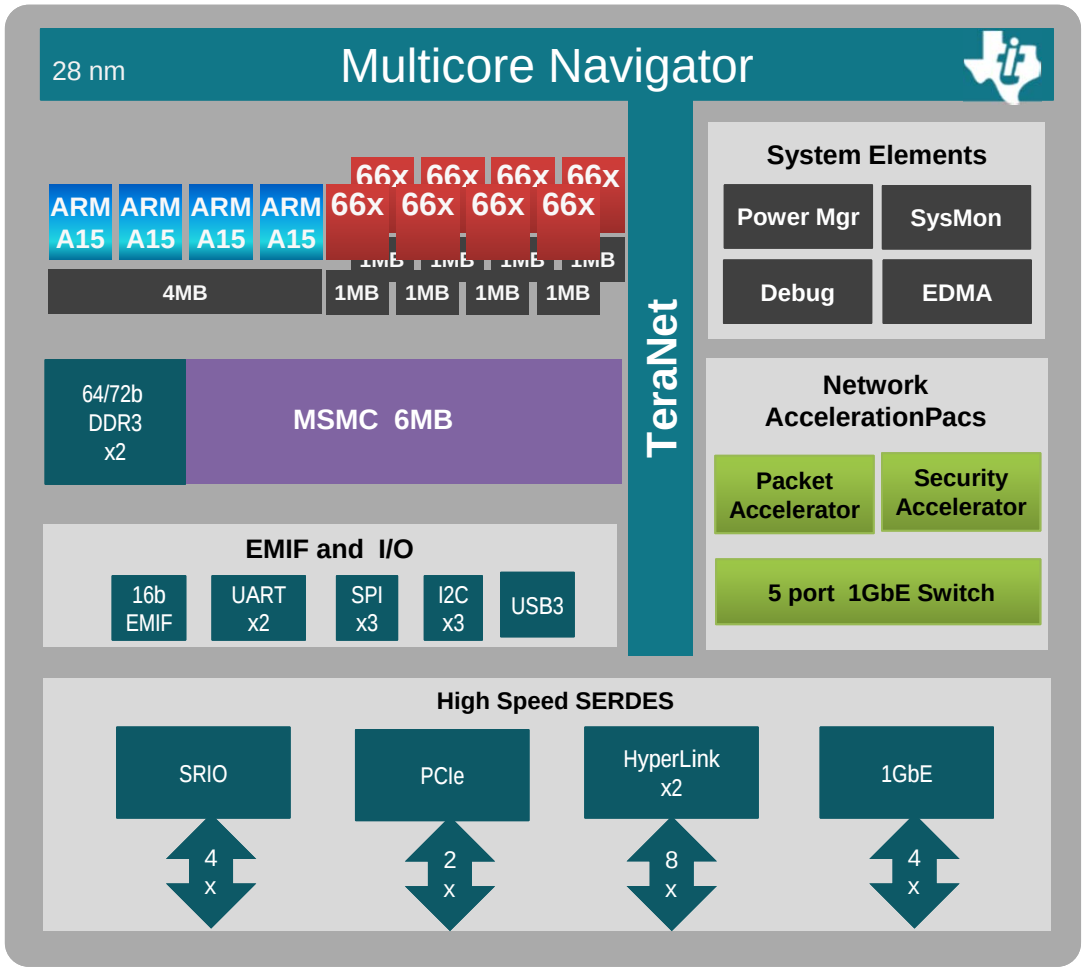
- Multicore Shared Memory Controller provides low latency & high bandwidth memory access
- 6MB Shared L2 on-chip
- 2 x 72 bit DDR3, 72-bit (with ECC), 10GB total addressable, DIMM support (4 ranks total)

KeyStone multicore architecture and acceleration

- Multicore Navigator, TeraNet, HyperLink
- 1GbE Network coprocessor (IPv4/IPv6)
- Crypto Engine (IPSec, SRTP)

Peripherals

- 4 Port 1G Layer 2 Ethernet Switch
- 2x PCIe, 1x4 SRIO 2.1, EMIF16, USB 3.0 UARTx2, SPI, I²C
- 15-25W depending upon DSP cores, speed, temp & other factors



40mm x 40mm package

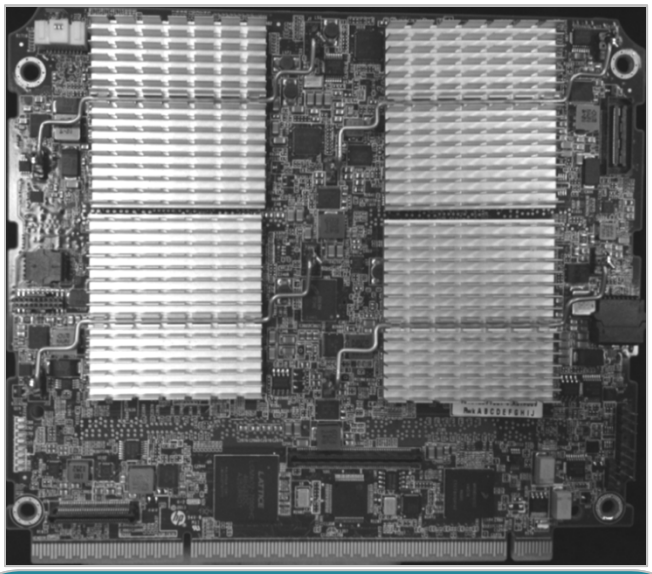
Available HPC Platforms

nCore BrownDwarf

"The BrownDwarf Y-Class system is an incredibly important milestone in HPC system development. -Working in close collaboration with TI, IDT and our hardware partner Prodrive, we have successfully established a new class of energy efficient supercomputers designed to fulfill the demands of a wide range of scientific, technical and commercial applications. We are very excited to be launching the most capable energy efficient supercomputer available. The innovative design of the BrownDwarf Y-Class system has resulted in a network fabric that far exceeds the latency and power efficiencies of traditional supercomputing systems based on x86 and Infiniband or Ethernet systems.- By utilizing existing programming models and toolsets, the BrownDwarf Y-Class supercomputer is a disruptive force in HPC as it leapfrogs a number of the supercomputing incumbents."

-- Ian Lintault, Managing Director, nCore HPC

HP Moonshot



"As a partner in HP's Moonshot ecosystem dedicated to the rapid development of new Moonshot servers, we believe TI's KeyStone design will provide new capabilities across multiple disciplines to accelerate the pace of telecommunication innovations and geological exploration."

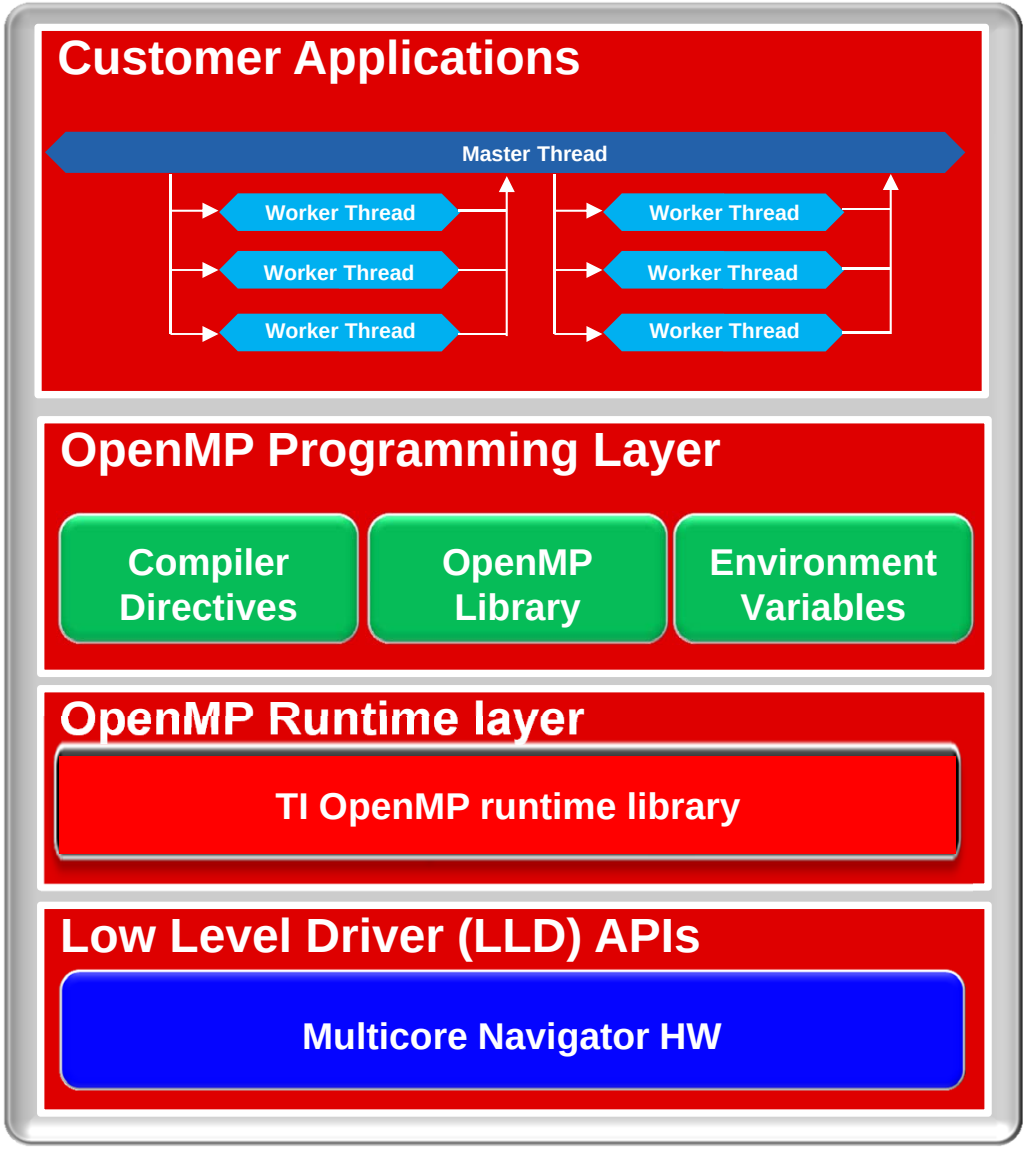
--- Paul Santeler, vice president and general manager, Hyperscale Business, HP

OpenMP on Multicore DSP

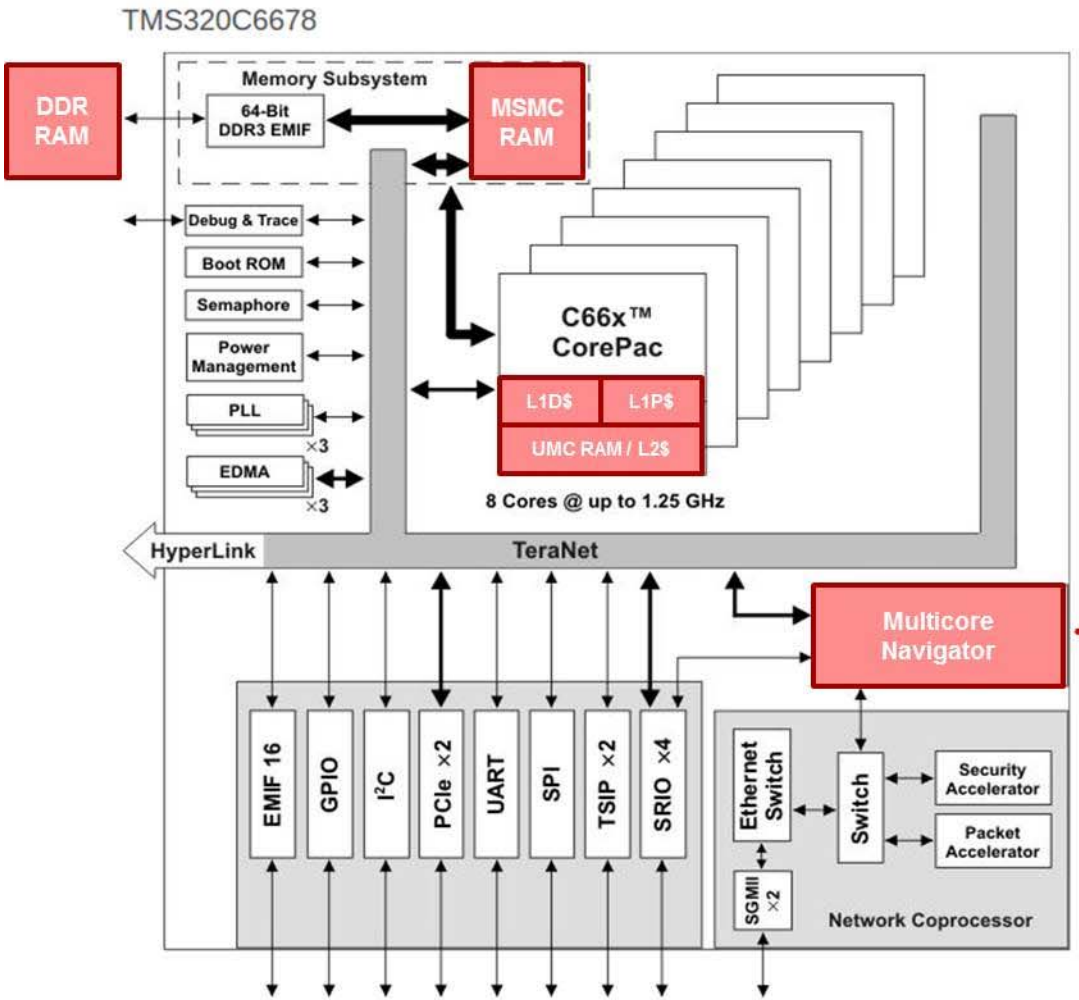
- API for specifying shared-memory parallelism in C, C++, and Fortran
- Consists of compiler directives, library routines, and environment variables
  - **Easy** & **incremental** migration for existing code bases
  - De facto **industry standard** for shared memory parallel programming
- **Portable** across shared-memory architectures
- **Evolving** to support heterogeneous architectures, tasking dependencies etc.

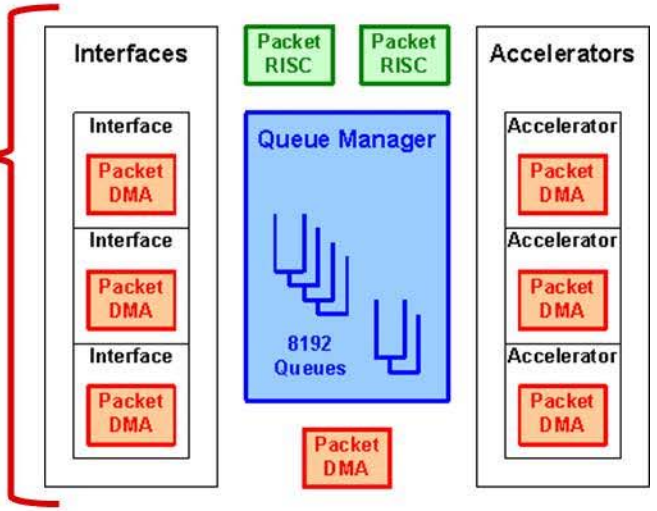
Bare-metal OpenMP Software Stack

- Leverage HW queues from Multicore Navigator
- Thread dispatches events to other threads via HW queues.
- OpenMP runtime built on top of very low-level drivers close to hardware
- OpenMP run-time state and user data is allocated in shared memory



Multicore Navigator Hardware





- 16K HW queues with atomic access
- 2 Packet RISCs
- 1 Packet DMA engine

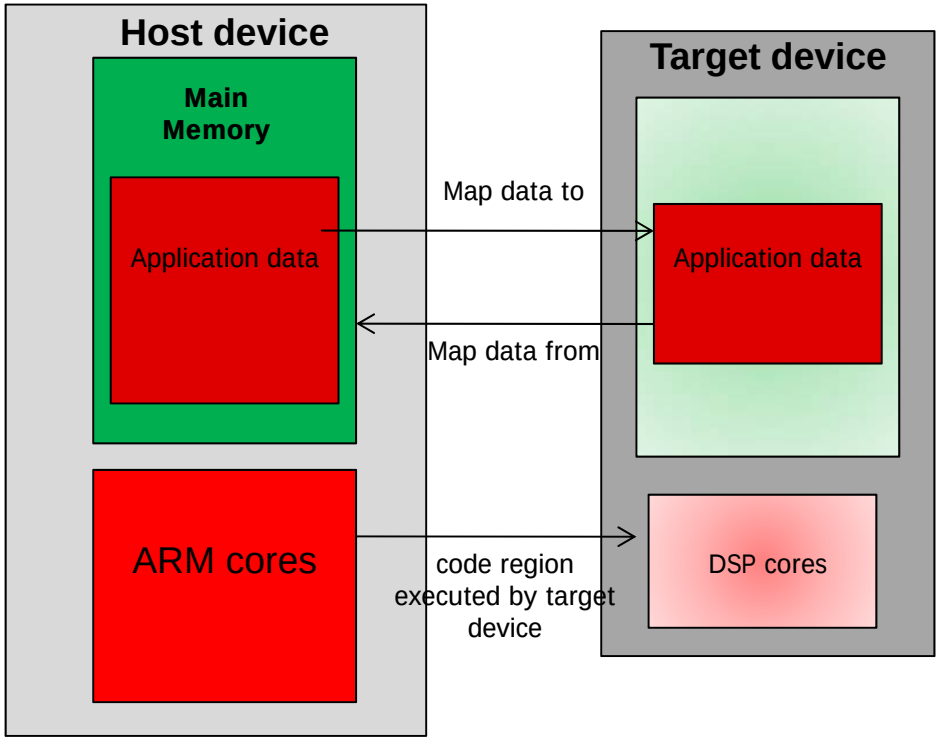
OpenMP 4.0

- Released July 2013
  - http://www.openmp.org/mp-documents/OpenMP4.0.0.pdf
  - Example document released at SC'13
- Changes from 3.1 to 4.0 (Appendix E.1):
  - *Heterogeneous (accelerator) devices: 2.9*
  - *SIMD extensions: 2.8*
  - *Places and thread affinity: 2.5.2, 4.5*
  - *Taskgroup and dependent tasks: 2.12.5, 2.11*
  - *Error handling: 2.13*
  - *User-defined reductions: 2.15*
  - *Sequentially consistent atomics: 2.12.6*
  - *Fortran 2003 support*

Heterogeneous device support

- *target* construct: creates a region of code that is executed by a target device
- Work with wide variety of devices
  - GPGPUs, MIC, DSP, FPGA, etc
  - A target could be a remote node of the same type as the host

```
#pragma omp target
{
      /* This region of code is
       executed on the default
       target device */
{
```



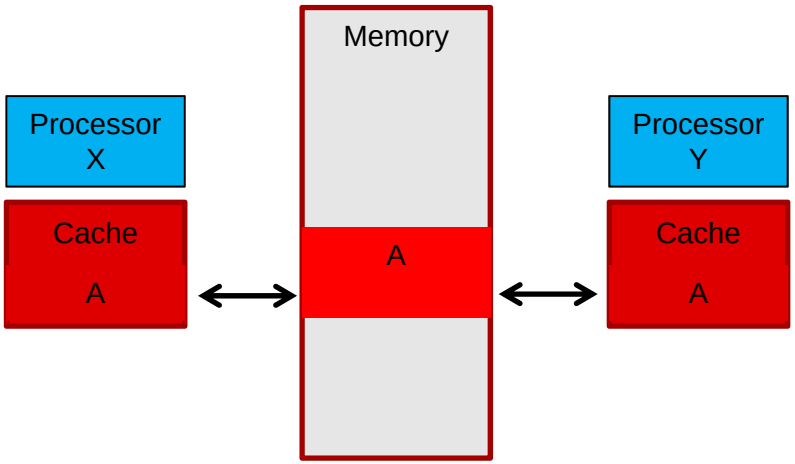
Device constructs: mapping data

- Data is *mapped* from the host device to a target device
  - It might require a copy to/from host memory to device memory
  - OR it might simply be passing a pointer
    - The model supports both, so the user cannot assume a copy

```
long a = 0x858;
long b = 0;
int X[100];

#pragma omp target data map(to:a) map(b, X)
{
    /* a, b and X are mapped to the device */
    b = weighted_average(X, a);
}

/* b and X are flushed back to the host */
```

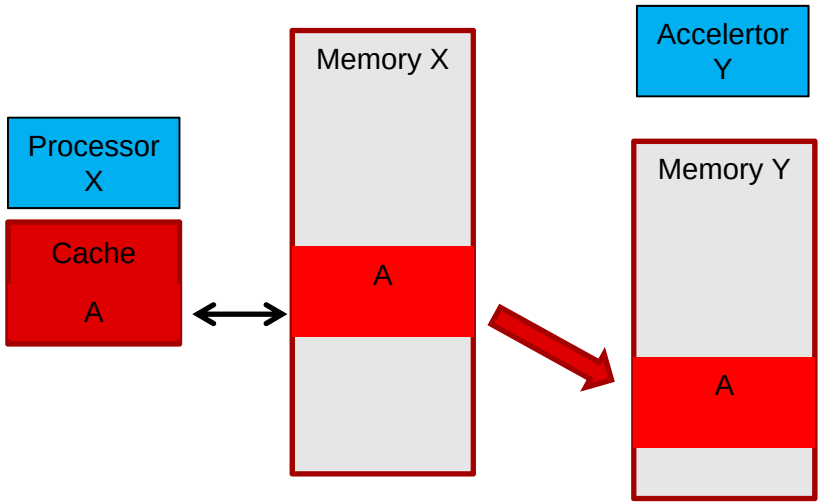
Data mapping: shared or distributed memory

Shared memory



- The corresponding variable in the device data environment *may* share storage with the original variable.
- Writes to the corresponding variable may alter the value of the original variable.

Distributed memory



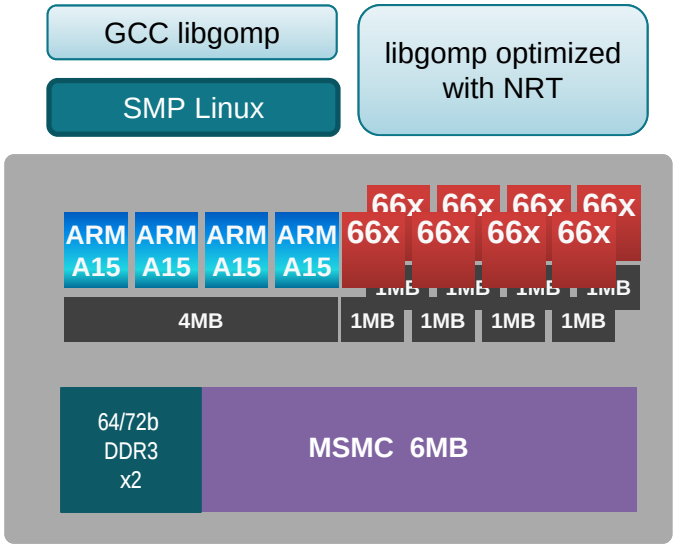
`target` construct and `map` clause

```
extern void init(float*, float*, int);
extern void output(float*, int);

void vec_mult(float *p, float *v1, float *v2, int N)
{
   int i;
   init(v1, v2, N);

   #pragma omp target map(to:v1[0:N],v2[:N]) \\
                      map(from:p[0:N])
   #pragma omp parallel for
   for (i=0; i<N; i++)
     p[i] = v1[i] * v2[i];

   output(p, N);
}
```

- The `target` construct maps the variables v1, v2, and p to the device data environment.
- The variable N is mapped into the device data environment from the encountering task's data environment.

OpenMP on multicore ARM+DSP K2H

- OpenMP-ARM
  - Quad A15 running SMP linux
  - GCC tool chain w/ OpenMP support
  - GCC libgomp runtime
- OpenMP-DSP (C66x)
  - TI C66x tool chain w/ OpenMP support
  - Runtime based on libgomp with optimizations leveraging Multicore Navigator (e.g. hardware queues)



OpenMP on K2H

Dispatching OpenMP-DSP programs from ARM using OpenMP Accelerator Model

Host program with OpenMP target directives

```
void foo(int *in1, int *in2, int *out1, int count)
{
    #pragma omp target   map (to: in1[0:count-1],    \
                              in2[0:count-1], count, \
                              from: out1[0:count-1])
    {
        #pragma omp parallel shared(in1, in2, out1)
        {
            int i;
            #pragma omp for
            for (i = 0; i < count; i++)
                out1[i] = in1[i] + in2[i];
        }
    }
}
```

Source to Source lowering

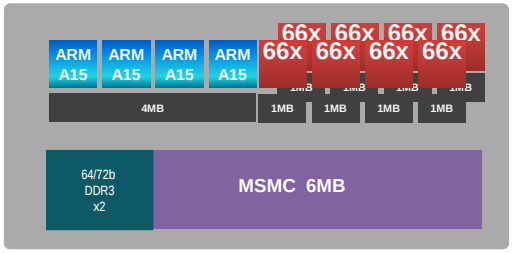
```
#pragma omp parallel shared(in1, in2, out1)
        {
            int i;
            #pragma omp for
            for (i = 0; i < count; i++)
                out1[i] = in1[i] + in2[i];
        }
```

OpenMP Region

OpenMP Accelerator Runtime (ARM)

SMP Linux

OpenMP Accelerator Model Runtime (DSPs)



OpenMP on K2H

Dispatch OpenMP-DSP programs from ARM using standard OpenCL APIs

OpenCL host program

```
program.build(devices, "add_openmp.obj");
Kernel kernel(program, "add_wrapper");
kernel.setArg(0, bufA);
kernel.setArg(1, bufB);
kernel.setArg(2, bufDst);
kernel.setArg(3, NumElements);

Event ev1,ev2,ev3,ev4, ev5,  ev6,ev7,ev8;

CommandQueue InO_Q(context, devices[d], CL_QUEUE_PROFILING_ENABLE);

InO_Q.enqueueWriteBuffer(bufA, CL_FALSE, 0, bufsize, srcA, NULL, &ev1);
InO_Q.enqueueWriteBuffer(bufB, CL_FALSE, 0, bufsize, srcB, NULL, &ev2);

std::vector<Event> vec_ev5(1);
InO_Q.enqueueTask(kernel, NULL, &vec_ev5[0]);

InO_Q.enqueueReadBuffer(bufDst, CL_TRUE, 0, bufsize, dst, &vec_ev5, &ev6);
```

```
__kernel
void add_wrapper(__global const float *a,
                 __global const float *b,
                 __global float *c,
                 int size)
{
  add_openmp(a, b, c, size);
}
```

OpenCL kernel wrapper

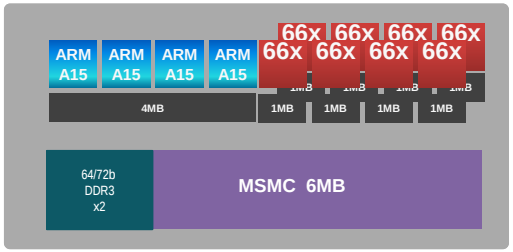
```
void add_openmp(float *a,
                float *b,
                float *c,
                int size)
{
  int i;
  #pragma omp parallel for
  for (i = 0; i < size; i++)
    c[i] = a[i] + b[i];
}
```

OpenMP region

OpenCL Runtime

SMP Linux

OpenCL Monitor + OpenMP Runtime



For more information:

- **Stop** by the TI Booth #3725
- **Watch** the 66AK2H12 EVM board tour
- **Read** “The case for 10G Ethernet”
- **Keep** up with the latest HPC happenings on the Multicore Mix blog