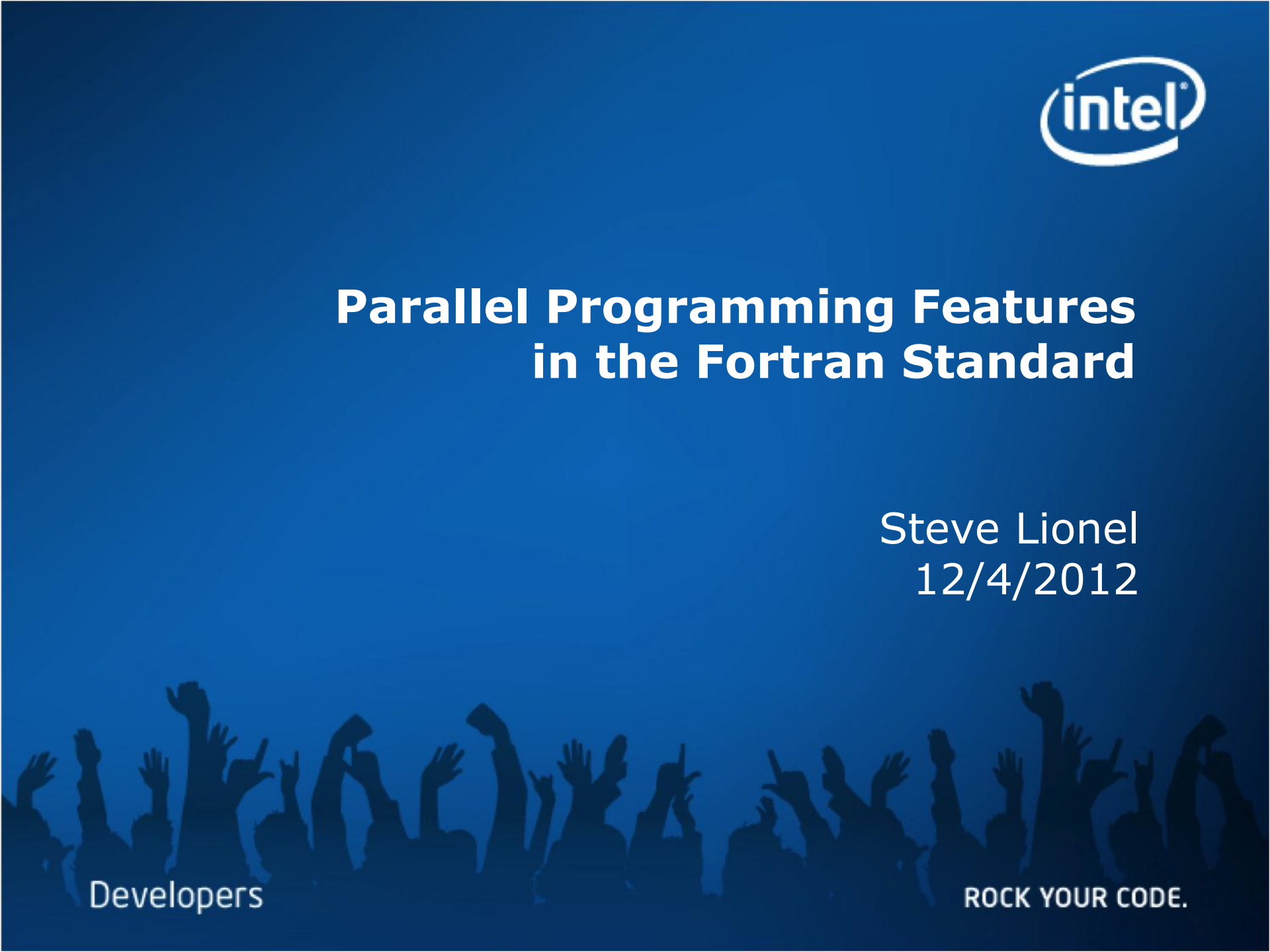
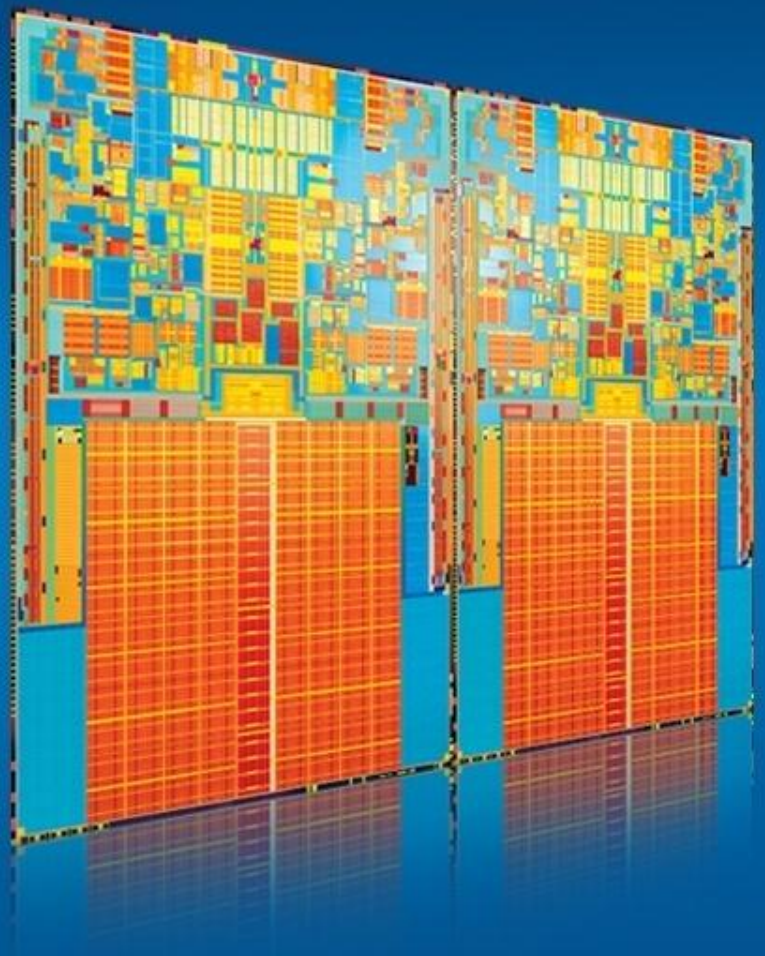

intel
Parallel Programming Features
in the Fortran Standard
Steve Lionel
12/4/2012
Developers
ROCK YOUR CODE.

Agenda

- Overview of popular parallelism methodologies
- FORALL – a look back
- DO CONCURRENT
- Coarrays
- Fortran 2015
- Q+A



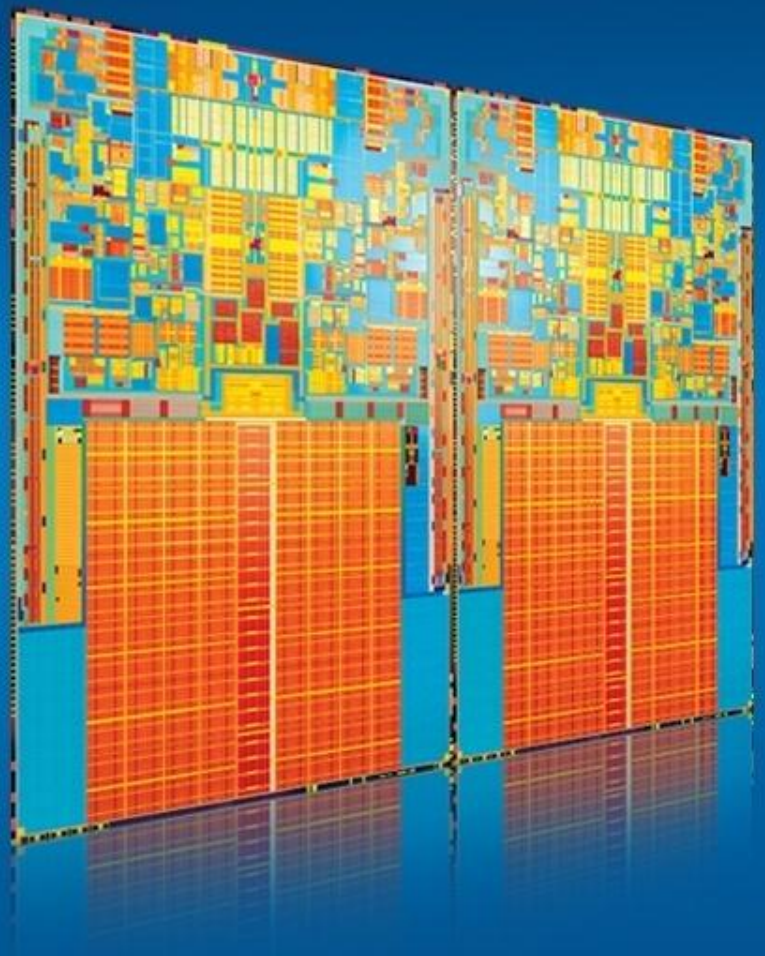
Fortran on the Side

Popular Parallelism Methodologies

- Defined by multi-vendor consortiums
  - OpenMP*
    - Threading on shared-memory systems
    - Directive-based
    - Single program execution, fork-join parallelism
    - Requires compiler support
    - OpenMP Architecture Review Board – OpenMP.org
  - Message Passing Interface (MPI)
    - Shared or distributed memory
    - API (procedure call) based
    - Multiple copies of program run in parallel
    - No explicit compiler support required, but…
    - MPI Forum – mpi-forum.org

Popular Parallelism Methodologies

- Implementation-specific
  - OS threads (Windows threads, pthreads, etc.)
    - Defined by OS vendor
    - API based
    - Single copy of program, typically "worker threads"
    - No explicit compiler support required
  - Auto-Parallel
    - Feature of Intel (and some other) compilers
    - Directives needed for best performance
    - Loops and array operations only
    - Compiler support required



The Fortran Way

FORALL (1/2)

- Provides array assignments controlled by a "triplet-spec" and, optionally, a mask
- Originally part of High-Performance Fortran, a dialect extending Fortran 90
- Adopted in Fortran 95
- Example:

```
FORALL (I=1:10,J=1:10,B(I,J)/=0)
  A(I,J) = REAL(I+J+2)
  B(I,J) = A(I,J) + B(I,J) * REAL(I*J)
END FORALL
```

FORALL (2/2)

- Not a loop construct!
- Each array assignment evaluated completely in turn
- Inefficient to parallelize

DO CONCURRENT (1/3)

- New in Fortran 2008
- Uses FORALL header, but no mask
- Iterations can execute in any order and to any degree of parallelism
  - Programmer is responsible for making sure there are no loop-carried dependencies
- Intel Fortran will attempt to parallelize with auto-parallel enabled
- Helps vectorization
- Fork-join model

DO CONCURRENT (2/3)

- Example:

```
DO CONCURRENT (I=1:N)
   A(I) = T + (B(I) * C(I))
END DO
```

- Limitations
  - Must use BLOCK to create iteration-private variables
    - Intel Fortran doesn't yet support BLOCK 
  - Not suitable for reductions
  - I/O allowed, but no dependence on order

DO CONCURRENT (3/3)

- Example using BLOCK:

```
DO CONCURRENT (I=1:N)
   BLOCK
      REAL :: T
      T = A(I) + B(I)
      C(I) = T + SQRT(I)
   END BLOCK
END DO
```

- Scatter/Gather example:

```
DO CONCURRENT (I=1:M)
   A(IND(I)) = I
END DO
```

Coarrays

- New in Fortran 2008
- Derived from "F--" specification in early 2000s
- Cray implementation on T3E and X1 supercomputers
- Partitioned Global Address Space (PGAS) model
  - PGAS also used by UPC, X10, Chapel
- Multiple copies of program run in parallel
  - Each copy called an "image"

Coarrays

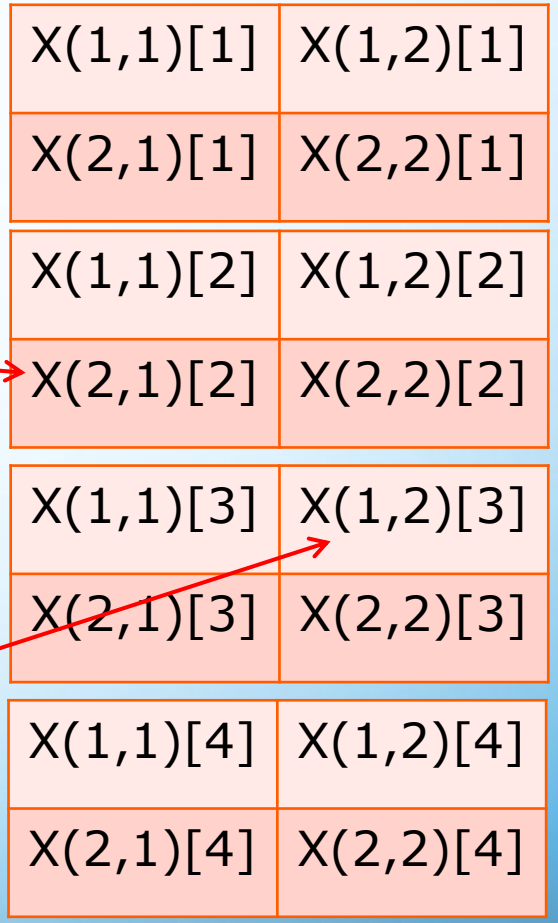
- What is a coarray?
  - Variable declared to have CODIMENSIONs with []
    - REAL, DIMENSION(1000), CODIMENSION [*] :: X
  - Last codimension must be * - computed at runtime from number of images
  - Can be array or scalar
  - Each image has its own piece of the coarray
  - Images can reference other image's pieces by using "coindices" enclosed in []
    - X[3]
  - Images can reference their own piece by omitting the []

Coarrays

REAL :: X(2,2) [*] ! Assume four images

X(2,1) on image 2

X(1,2)[3] on image 4



Coarrays

- Mapping of objects with codimensions: 2D

```
real, codimension[2,*] :: x
```

- Mapping of X if program run with 6 images:

Image 1 X[1,1]	Image 3 X[1,2]	Image 5 X[1,3]
Image 2 X[2,1]	Image 4 X[2,2]	Image 6 X[2,3]

Mapping of X if program run with 9 images

Image 1 X[1,1]	Image 3 X[1,2]	Image 5 X[1,3]	Image 7 X[1,4]	Image 9 X[1,5]
Image 2 X[2,1]	Image 4 X[2,2]	Image 6 X[2,3]	Image 8 X[2,4]	

Coarrays

- Coarrays can be…
  - ALLOCATABLE
  - Polymorphic
  - Used in assignments and expressions
  - Used in READ and WRITE statements
  - Passed as arguments to procedures
  - Used as dummy arguments in procedures (explicit interface required)
- Coarrays can’t be…
  - Allocated differently in different images
  - Interoperable with C

Coarrays and Synchronization

- Implicit synchronization points
  - At image start
  - When a coarray is allocated
  - At image end
- Explicit synchronization
  - SYNC ALL
  - SYNC IMAGES (image-list)
  - SYNC MEMORY (image-list)
  - Critical sections (CRITICAL…END CRITICAL)
  - LOCK and UNLOCK statements
  - ATOMIC_DEFINE and ATOMIC_REF intrinsics
  - ERROR STOP

Intrinsics for coarrays

- NUM_IMAGES()
- IMAGE_INDEX(varname)
- LCOBOUND(varname)
- UCOBOUND(varname)
- THIS_IMAGE()

Input and Output with coarrays

- "Standard output" (unit 6) preconnected in all images
  - Intel Fortran will "merge the streams" and display in image 1
  - Order of output not guaranteed
- "Standard input" (unit 5) preconnected for image 1 only
- For all other units, each image has their own independent set

Example coarray code

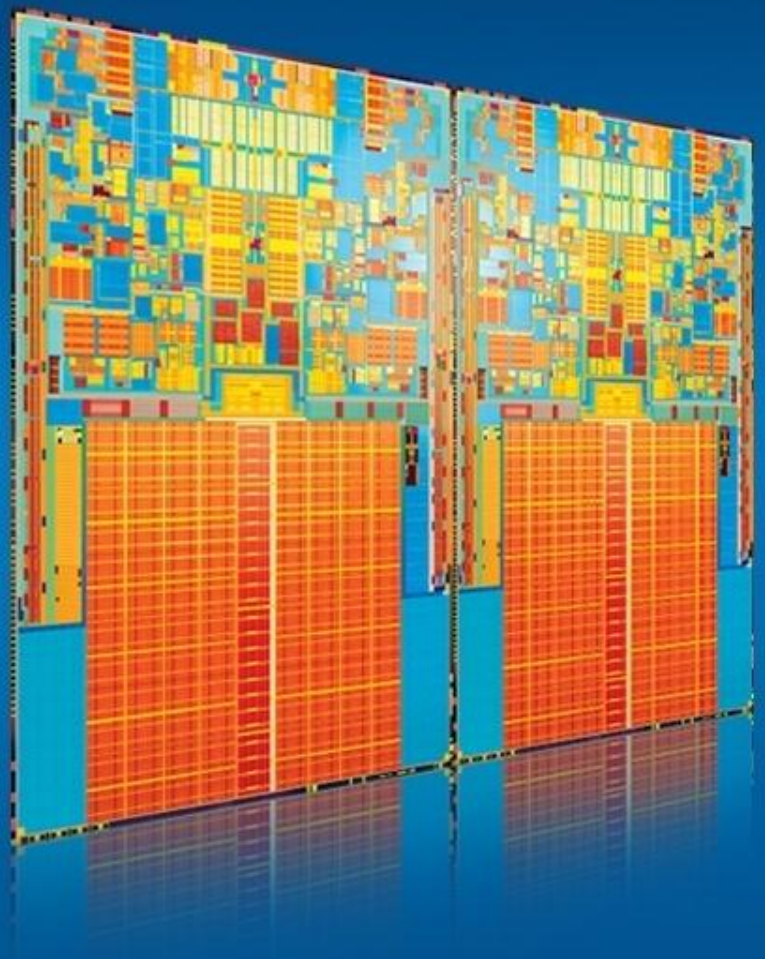
```
my_subgrid(      0, 1:my_M) = my_subgrid(   my_N, 1:my_M)[my_north_P,me_Q]
my_subgrid( my_N+1, 1:my_M) = my_subgrid(      1, 1:my_M)[my_south_P,me_Q]
my_subgrid( 1:my_N, my_M+1) = my_subgrid( 1:my_N, 1     )[me_P, my_east_Q]
my_subgrid( 1:my_N, 0     ) = my_subgrid( 1:my_N, my_M  )[me_P, my_west_Q]


max_global = MAXVAL( ABS( my_subgrid_new_values(1:my_N,1:my_M) - &
  my_subgrid(1:my_N,1:my_M) ) )
SYNC ALL  ! protects both max_global and my_subgrid
IF (me == 1) THEN
  DO I= 2,NUM_IMAGES()
    max_local = max_global[I]
    max_global  = MAX( max_global, max_local )
  END DO
END IF
```

Coarrays in Intel Fortran (1/2)

- New in Intel [Visual] Fortran Composer XE 2011
- Supported on Linux and Windows only
- Underlying transport is Intel MPI
  - Run-time libraries provided
- Shared memory model included with Composer XE
- Distributed memory model (cluster) requires Intel Cluster Studio XE license
- Windows cluster support added in Composer XE 2011 Update 6

Coarrays in Intel Fortran (2/2)

- Compile with –coarray (/Qcoarray) to get coarray syntax and features enabled
- For shared memory, just run executable
- For distributed memory, use –coarray=distributed (/Qcoarray:distributed) and define MPI ring
- Whole programs only – can't create coarray-using library for use by non-coarray programs

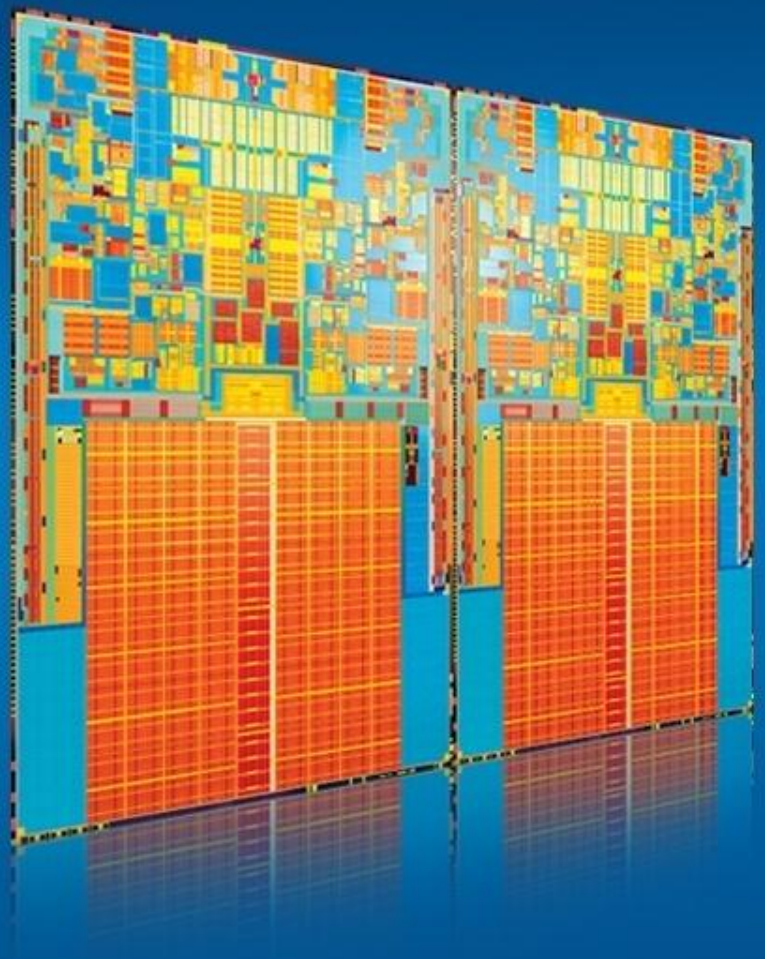


Fortran 2015

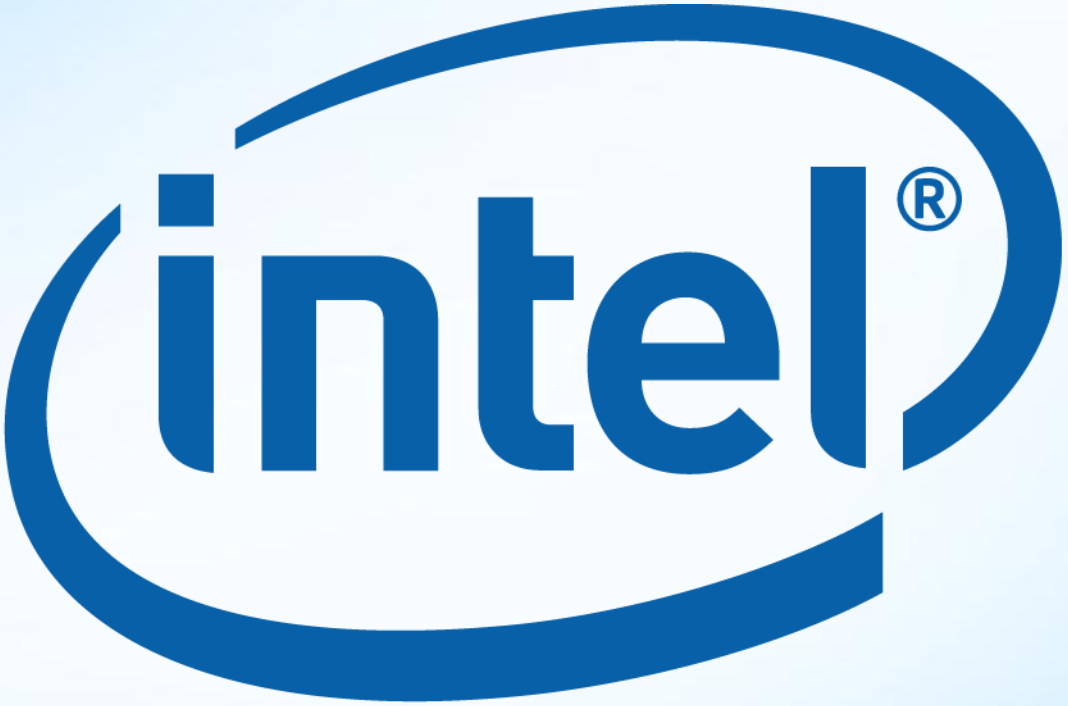
intel Software

Fortran 2015

- Name for next revision of Fortran Standard
- Technical work to be completed in 2014
- Enhancements to C Interoperability (TS29113)
- Enhancements to coarray features
  - Draft under development
    - Teams
    - Events
    - Collective procedures
    - Additional atomic procedures
- Corrections and Clarifications



Q+A



Software

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/software/products.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Atom, Centrino Atom Inside, Centrino Inside, Centrino logo, Cilk, Core Inside, FlashFile, i960, InstantIP, Intel, the Intel logo, Intel386, Intel486, IntelDX2, IntelDX4, IntelSX2, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skoool, Sound Mark, The Journey Inside, Viiv Inside, vPro Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

http://intel.com/software/products

Optimization Notice

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804